

# **Semi-Automatic Assessment of Students' Graph-Based Diagrams**

By

**Firat Batmaz, BSc MSc**

**A Doctoral Thesis**

Submitted in partial fulfillment of the requirements  
for the award of  
Doctor of Philosophy of Loughborough University  
September 2010

# ABSTRACT

Diagrams are increasingly used in many design methods, and are being taught in a variety of contexts in higher education such as database conceptual design or software design in computer science. They are an important part of many assessments. Currently computer aided assessments are widely used for multiple choice questions. They lack the ability to assess a student's knowledge in a more comprehensive way, which is required for diagram-type student work. The aim of this research is to develop a semi-automatic assessment framework, which enables the use of computer to support the assessment process of diagrammatic solutions, with the focus of ensuring the consistency of grades and feedback on solutions. A novel trace model, that captures design traces of student solutions, was developed as a part of the framework and was used to provide the matching criteria for grouping the solutions. A new marking style, partial marking, was developed to mark these solution groups manually. The Case-Based Reasoning method is utilised in the framework to mark some of the groups automatically. A guideline for scenario writing was proposed to increase the efficiency of automatic marking. A prototype diagram editor, a marking tool and scenario writing environment were implemented for the proposed framework in order to demonstrate proof of concept. The results of experiments show that the framework is feasible to use in the formative assessment and it provides consistent marking and personalised feedback to the students. The framework also has the potential to significantly reduce the time and effort required by the examiner to mark student diagrams. Although the constructed framework was specifically used for the assessment of database diagrams, the framework is generic enough to be used for other types of graph-based diagram.

**Keywords:** Computer Aided Assessment, Traceability, Self-Explanation Systems, Case-Based Reasoning, Online Assessment, and Graph-Based Diagrams.

*Dedication*

**To My Mother, Samiha  
My Father, Osman  
My Mother in Law, Khairun Nisa  
My Dear Wife, Salima  
and My Son, Siraj**

# ACKNOWLEDGMENTS

I would like to express my full indebtedness to my supervisor *Prof. Dr. Chris Hinde*, who accepted me as his PhD student. Without his care regarding every aspect of my PhD studentship, the conversations that clarified my thinking, his friendship and professional collaboration as well as his kind encouragement, this thesis would not have been possible.

I am particularly grateful to *Dr. Roger Stone, Prof. Dr. Paul Chung* and *Dr. Iain Phillips* for their invaluable help and contributions.

I wish to thank my dear friends *Prof. Dr. Serpil Acar, Dr. Murat Ceren* and *Dr. Volkan Esat* gratefully for their never-ending support and fellowship. I would also like to extend my sincere thanks to my friends and colleagues *Prof. Ray Dawson, Dr. Lin Guan, Dr. Mark Withall, Dr. Sinan Eroglu, Dr. Senay Mihcin, Dr. Ana Salagean, Dr. Eran Edirisinghe, Dr. Waad Yousif, Jeff Pitchers, Satish Bedi, Jo McOuat, Judith Poulton, Patrick Holligan, Sherri Booth, Christine Bagley, Sussi Maguire* and all other friends in Loughborough.

I would also like to thank my dear friends *Rauf Kasapoglu, Fatih Kahraman, Dr. Cem Erbil, Dr. Sirri Demirsoy, Ali Riza Arslan* and all other friends in London.

Last, but not least, I am utmost grateful to my wife, *Salima*, and *my family*, for their invaluable understanding, endless support, patience, and love.

# TABLE OF CONTENTS

<b><u>CHAPTER 1 OVERVIEW.....</u></b>	<b><u>1</u></b>
1.1 INTRODUCTION.....	1
1.2 AIM AND OBJECTIVES .....	1
1.3 APPROACH .....	2
1.4 CONTRIBUTION.....	3
1.5 THESIS ORGANISATION.....	4
<b><u>CHAPTER 2 LITERATURE REVIEW: AUTOMATIC ASSESSMENT FOR DIAGRAMMATIC SOLUTIONS .....</u></b>	<b><u>7</u></b>
2.1 INTRODUCTION.....	7
2.2 COMPUTER AIDED ASSESSMENT .....	8
2.2.1 AUTOMATED ASSESSMENT TYPES.....	11
2.2.2 AUTOMATIC ASSESSMENT OF DIAGRAM-TYPE SOLUTIONS.....	15
2.2.3 SUMMARY OF AUTOMATIC ASSESSMENT .....	16
2.3 ENTITY RELATIONSHIP DIAGRAM .....	18
2.3.1 QUESTION TYPES.....	19
2.3.2 EXTENSION OF ERDS .....	19
2.3.3 COMMON USAGE .....	19
2.3.4 DESIGN SKILLS .....	20
2.4 CURRENT AUTOMATIC ERD ASSESSMENT RESEARCH .....	20
2.4.1 DEAP.....	20
2.4.2 DATSYS .....	21
2.4.3 VLE-ERM .....	21
2.4.4 KERMIT .....	22
2.4.5 SUMMARY OF THE CURRENT RESEARCH.....	23
2.5 SUMMARY .....	24
<b><u>CHAPTER 3 A NEW SEMI-AUTOMATIC DIAGRAM ASSESSMENT FRAMEWORK .....</u></b>	<b><u>26</u></b>
3.1 INTRODUCTION.....	26
3.2 MANUAL DIAGRAM ASSESSMENT PROCESS .....	26

*Table of Contents*

3.2.1 CHALLENGES IN EACH STAGE .....	28
<b>3.3 SEMI-AUTOMATIC DIAGRAM ASSESSMENT .....</b>	<b>33</b>
3.3.1 MARKING PROCESS .....	33
3.3.2 IDENTICAL SEGMENTS .....	37
3.3.3 AUTOMATION .....	41
3.3.4 SCENARIO AUTHORING PROCESS .....	44
<b>3.4 SUMMARY .....</b>	<b>45</b>
<b><u>CHAPTER 4 DESIGN TRACE MODEL .....</u></b>	<b><u>46</u></b>
<b>4.1 INTRODUCTION.....</b>	<b>46</b>
<b>4.2 TRACE DEFINITION .....</b>	<b>46</b>
4.2.1 TRACE ENTITIES .....	47
4.2.2 RELATIONSHIPS TO BE TRACED .....	50
<b>4.3 TRACE PRODUCTION .....</b>	<b>58</b>
4.3.1 OFF-LINE PRODUCTION .....	58
4.3.2 SR RELATIONSHIPS PRODUCTION .....	59
4.3.3 E-RELATIONSHIPS PRODUCTION .....	60
4.3.4 ON-LINE PRODUCTION .....	62
4.3.5 SR RELATIONSHIPS PRODUCTION .....	62
4.3.6 ONLINE E-RELATIONSHIP PRODUCTION.....	65
4.3.7 SELF-EXPLANATIONS .....	67
<b>4.4 DESIGN TRACE MODEL.....</b>	<b>67</b>
<b>4.5 SUMMARY .....</b>	<b>69</b>
<b><u>CHAPTER 5 MARKING PROCESS MODEL.....</u></b>	<b><u>70</u></b>
<b>5.1 INTRODUCTION.....</b>	<b>70</b>
<b>5.2 AUTOMATIC PARTIAL MARKING .....</b>	<b>70</b>
5.2.1 BASIC CASE DEFINITION, CORRESPONDENCE LINKS AND REFERENCE DIAGRAMS.....	71
5.2.2 EXAMPLES OF AUTOMATIC MARKING.....	73
5.2.3 CASE CATEGORIES AND GENERIC CASE DEFINITION .....	76
5.2.4 GENERIC CASE GENERATION .....	78
<b>5.3 SIMILAR SCENARIO TEXT.....</b>	<b>79</b>
5.3.1 STATEMENT TYPES .....	79
5.3.2 WRITING STATEMENTS .....	83
5.3.3 SCENARIO SECTION .....	85

*Table of Contents*

5.3.4 WRITING SIMILAR SCENARIO TEXT .....	87
<b>5.4 A COMPLETE PROCESS MODEL FOR DIAGRAM MARKING.....</b>	<b>89</b>
<b>5.5 SUMMARY .....</b>	<b>91</b>
<b><u>CHAPTER 6 DESIGN AND IMPLEMENTATION .....</u></b>	<b><u>93</u></b>
<b>6.1 INTRODUCTION.....</b>	<b>93</b>
<b>6.2 DIAGRAM EDITOR .....</b>	<b>93</b>
6.2.1 REQUIREMENTS .....	94
6.2.2 COMPONENTS OF THE EDITOR.....	95
6.2.3 A DIAGRAM DRAWING EXAMPLE .....	98
<b>6.3 MARKING TOOL .....</b>	<b>101</b>
6.3.1 REQUIREMENTS .....	101
6.3.2 COMPONENTS OF MARKING ENVIRONMENT .....	103
<b>6.4 AUTHORIZING TOOL .....</b>	<b>105</b>
6.4.1 REQUIREMENTS FOR THE TEACHER DIAGRAM EDITOR .....	106
6.4.2 REQUIREMENTS FOR THE SCENARIO TEXT WRITER.....	106
6.4.3 COMPONENTS OF THE SCENARIO TEXT WRITER.....	108
<b>6.5 COMPLETE SYSTEM OVERVIEW .....</b>	<b>111</b>
<b>6.6 SUMMARY .....</b>	<b>113</b>
<b><u>CHAPTER 7 EMPIRICAL EVALUATION .....</u></b>	<b><u>115</u></b>
<b>7.1 INTRODUCTION.....</b>	<b>115</b>
<b>7.2 SCENARIO WRITING ENVIRONMENT .....</b>	<b>116</b>
7.2.1 PROVISIONS OF THE EXPERIMENT .....	116
7.2.2 RESULTS OF THE EXPERIMENT .....	118
7.2.3 POSSIBLE IMPROVEMENTS FOR THE WRITING ENVIRONMENT .....	119
<b>7.3 DIAGRAM EDITOR.....</b>	<b>119</b>
7.3.1 PROVISIONS OF THE EXPERIMENTS .....	119
7.3.2 THE EXPERIMENTS .....	120
7.3.3 RESULTS OF THE EXPERIMENTS .....	121
7.3.4 ANALYSIS OF ENTITY NAMES IN THE EXPERIMENTS .....	125
<b>7.4 MARKING TOOL .....</b>	<b>126</b>
7.4.1 GROUPING STAGE AND INTERPRETING THE RESULTS .....	127
7.4.2 AUTOMATIC COMPONENT MARKING.....	131
7.4.3 PARTIAL MARKING.....	132

*Table of Contents*

7.4.4 PROVISIONS OF THE EXPERIMENT .....	137
7.4.5 RESULTS OF THE EXPERIMENT .....	137
7.4.5 FULL MARKING .....	138
7.4.6 NAMING AMBIGUITY .....	138
<b>7.5 THE SEMI-AUTOMATIC ASSESSMENT TOOL .....</b>	<b>140</b>
7.5.1 USING THE TOOL AND RESULTS .....	142
<b>7.6 SUMMARY .....</b>	<b>143</b>
<b><u>CHAPTER 8 CONCLUSIONS AND FUTURE WORK .....</u></b>	<b><u>146</u></b>
<b>8.1 INTRODUCTION.....</b>	<b>146</b>
<b>8.2 THESIS REVIEW .....</b>	<b>146</b>
<b>8.3 SUMMARY OF CONTRIBUTIONS .....</b>	<b>149</b>
<b>8.4 LIMITATIONS AND FUTURE DIRECTIONS.....</b>	<b>151</b>
8.4.1 MARKING BASED ON DESIGN TRACES .....	151
8.4.2 HANDLING MULTIPLE GRAPH-BASED DIAGRAM TYPES .....	152
8.4.3 FEEDBACK GENERATION.....	152
8.4.4 USER INTERFACE DESIGN.....	152
8.4.5 DEEP-KNOWLEDGE ASSESSMENT TOOL .....	152
8.4.6 EXTENSIONS TO THE GUIDELINE OF WRITING SCENARIO TEXT.....	153
8.4.7 NEW APPLICATION AREAS OF THE PARTIAL MARKING STYLE .....	153
<b>8.5 OVERALL CONCLUSION.....</b>	<b>153</b>
<b><u>REFERENCES .....</u></b>	<b><u>155</u></b>
<b><u>APPENDIX A .....</u></b>	<b><u>166</u></b>
<b><u>APPENDIX B.....</u></b>	<b><u>175</u></b>
<b><u>APPENDIX C .....</u></b>	<b><u>191</u></b>
<b><u>APPENDIX D .....</u></b>	<b><u>199</u></b>
<b><u>APPENDIX E.....</u></b>	<b><u>203</u></b>
<b><u>PUBLICATIONS.....</u></b>	<b><u>217</u></b>



*Table of Contents*

## LIST OF TABLES

TABLE 2.1 ASSESSMENT QUALITIES ACCORDING TO BROWN ET AL (2002)	10
TABLE 3.1 COMPONENTS FROM STUDENT DIAGRAMS	35
TABLE 5.1 CATEGORY OF GENERIC CASES	76
TABLE 7.1 NUMBER OF PARTICIPANTS IN EACH EXPERIMENT	121
TABLE 7.2 STUDENT ENTITY NAME FOR THE EXPERIMENT 1 OUT OF 20 DIAGRAMS	122
TABLE 7.3 STUDENT DIAGRAM COMPONENTS PRODUCED IN THE EXPERIMENT 2	123
TABLE 7.4 STUDENT ENTITY NAMES FOR THE EXPERIMENT 2	125
TABLE 7.5 COMPONENTS OF SOLUTION DIAGRAMS FOR SCENARIOS IN SECTION 2	128
TABLE 7.6 GROUPS OF STUDENT DIAGRAM COMPONENTS	128
TABLE 7.7 DIVERSITY IN STUDENT SOLUTIONS FOR EACH SCENARIO	130
TABLE 7.8 TEACHER DIAGRAM COMPONENTS	131
TABLE 7.9 AUTOMATICALLY MARKED COMPONENTS	132
TABLE 7.10 COMPONENT GROUPS FOR PARTIAL MARKING	133
TABLE 7.11 SUMMARY OF DIAGRAM MARKING FOR TWO SCENARIOS	142
TABLE B.1 THE LIST OF NOUN PHRASES FOR SCENARIO 1	176
TABLE B.2 THE LIST OF THE NOUN PHRASES FOR SCENARIO 2	180
TABLE B.3 THE LIST OF THE NOUN PHRASES FOR SCENARIO 3	184
TABLE B.4 THE LIST OF THE NOUN PHRASES FOR SCENARIO 4	188
TABLE C.1 DIAGRAM SOLUTIONS FOR SCENARIO 1	191
TABLE C.2 DIAGRAM SOLUTIONS FOR SCENARIO 2	193
TABLE C.3 DIAGRAM SOLUTIONS FOR SCENARIO 3	195
TABLE C.4 DIAGRAM SOLUTIONS FOR SCENARIO 4	197

# LIST OF FIGURES

FIGURE 1.1 THE STRUCTURE OF THE THESIS .....	6
FIGURE 2.1 BLOOM'S TAXONOMY.....	9
FIGURE 3.1 ASSESSMENT PROCESS CYCLE.....	27
FIGURE 3.2 DIRECT AND INDIRECT RELATIONSHIPS BETWEEN STAGES.....	32
FIGURE 3.3 SAMPLE SCENARIO TEXT FOR THE DATABASE DESIGN.....	34
FIGURE 3.4 PARTIAL MARKING PROCESS.....	36
FIGURE 3.5 ENTITY NAME AMBIGUITY.....	38
FIGURE 3.6 ENTITY MATCHING IN KERMIT .....	39
FIGURE 3.7 CONCEPTUAL DATABASE DESIGN IS AN ITERATIVE PROCESS .....	40
FIGURE 3.8 THE CBR CYCLE (ADAPTED FROM AAMODT & PLAZA, 1994) .....	42
FIGURE 3.9 THE PARTIAL MARKING CYCLE .....	43
FIGURE 4.1 GRANULARITY LEVEL EXAMPLES FOR DATABASE DIAGRAMS .....	48
FIGURE 4.2 DIFFERENT GRANULARITY MAPPING.....	49
FIGURE 4.3 REPRESENTATION RELATIONSHIPS IN REQUIREMENTS DOCUMENTS.....	51
FIGURE 4.4 SCENARIO REFERENCE LINK.....	52
FIGURE 4.5 EVOLUTION AND SR LINKS FOR A COMPONENT .....	54
FIGURE 4.6 EVOLUTION AND SR LINKS FOR MULTIPLE COMPONENTS .....	55
FIGURE 4.7 EXTRACT ACTION FOR A COMPONENT .....	56
FIGURE 4.8 TERMINOLOGICAL ITEMS (CERBAH AND EUZENAT, 2001).....	60
FIGURE 4.9 SCENARIO REFERENCE AND COMPONENT CORRESPONDENCE LINK.....	61
FIGURE 4.10 COMPONENTS REFERENCE DIAGRAMS.....	65
FIGURE 4.11 DESIGN TRACE MODEL.....	68
FIGURE 5.1 ASSESSMENT CASE DEFINITION.....	72
FIGURE 5.2 FINDING IDENTICAL REFERENCE DIAGRAMS .....	73
FIGURE 5.3 AUTOMATIC COMPONENT MARKING WITH AN IDEAL REFERENCE DIAGRAM.....	74
FIGURE 5.4 PARTIAL SCENARIO REFERENCE MATCHING.....	75
FIGURE 5.5 USAGE OF GENERIC CASES.....	77
FIGURE 5.6 GENERIC CASE GENERATION.....	78
FIGURE 5.7 EXAMPLES OF STATEMENT TYPE .....	80
FIGURE 5.8 COMPONENT REFERENCE DIAGRAMS .....	81
FIGURE 5.9 LONG COMPONENT REFERENCE DIAGRAM .....	82
FIGURE 5.10 SENTENCE CONSTRUCT EXAMPLE.....	84
FIGURE 5.11 CONVERT ALTERATION EXAMPLE.....	84
FIGURE 5.12 TWO WRITTEN STYLE OF SCENARIO TEXT FOR THE SAME SYSTEM .....	86

FIGURE 5.13 SCENARIO TEMPLATE EXAMPLE .....	88
FIGURE 5.14 STATEMENT TEMPLATES AND SENTENCE CONSTRUCTS .....	89
FIGURE 5.15 MARKING PROCESS MODEL .....	90
FIGURE 6.1 DIAGRAM EDITOR .....	96
FIGURE 6.2 SAMPLE SCENARIO TEXT .....	99
FIGURE 6.3 THE INITIAL DIAGRAM .....	100
FIGURE 6.4 "SPLIT" FUNCTION.....	100
FIGURE 6.5 THE USER INTERFACE OF THE FULL MARKING .....	103
FIGURE 6.6 THE USER INTERFACE OF THE PARTIAL MARKING .....	105
FIGURE 6.7 PLAN PAGE EXAMPLE .....	108
FIGURE 6.8 SECTION PAGE EXAMPLE.....	109
FIGURE 6.9 PRODUCTION PAGE EXAMPLE.....	110
FIGURE 6.10 SYSTEM VIEW OF SEMI-AUTOMATIC ASSESSMENT.....	112
FIGURE 7.1 SPLIT SCENARIO TEMPLATE .....	117
FIGURE 7.2 MAPPING A MERGED ENTITY TO AN ENTITY IN IDEAL DIAGRAM .....	136
FIGURE 7.3 MAPPING A SPLIT ENTITY TO AN ENTITY IN IDEAL DIAGRAM.....	136
FIGURE 7.4 INCONSISTENCY IN MARKING STYLES .....	138
FIGURE 7.5 THE USER INTERFACE OF THE DIAGRAM EDITOR.....	141
FIGURE 7.6 A STUDENT DIAGRAM WITH FEEDBACK (STONE ET AL. 2009).....	141
FIGURE A.1 PLAN PAGE 1 .....	166
FIGURE A.2 SECTION PAGE 2 FOR "EVENT" ENTITY TYPE.....	167
FIGURE A.3 SECTION PAGE 3 FOR "MEMBER" ENTITY TYPE.....	167
FIGURE A.4 SECTION PAGE 4 FOR "HELP" RELATIONSHIP TYPE .....	168
FIGURE A.5 SECTION PAGE 5 FOR "DATE" ATTRIBUTE TYPE.....	168
FIGURE A.6 SECTION PAGE 6 FOR RELATIONSHIP AND ATTRIBUTE TYPES.....	169
FIGURE A.7 SECTION PAGE 7 FOR RELATIONSHIP TYPE.....	169
FIGURE A.8 SECTION PAGE 8 FOR "FEE" ATTRIBUTE TYPE .....	170
FIGURE A.9 SECTION PAGE 9 FOR THE WHOLE OF THE DIAGRAM .....	170
FIGURE A.10 PRODUCTION PAGE 10.....	171
FIGURE B.1 THE SOLUTION OF SCENARIO 1 .....	177
FIGURE B.2 IDEAL REFERENCE DIAGRAMS FOR THE ENTITY COMPONENTS .....	178
FIGURE B.3 THE SOLUTION OF SCENARIO 2 .....	181
FIGURE B.4 IDEAL REFERENCE DIAGRAMS FOR THE ENTITY COMPONENTS .....	182
FIGURE B.5 THE SOLUTION OF SCENARIO 3 .....	185
FIGURE B.6 IDEAL REFERENCE DIAGRAMS FOR THE ENTITY COMPONENTS .....	186
FIGURE B.7 THE SOLUTION OF SCENARIO 4 .....	189
FIGURE B.8 IDEAL REFERENCE DIAGRAMS FOR THE ENTITY COMPONENTS .....	190
FIGURE D.1 THE RESULT OF QUESTION 1 .....	200
FIGURE D.2 THE RESULT OF QUESTION 2 .....	200
FIGURE D.3 THE RESULT OF QUESTION 3 .....	201

FIGURE D.4 THE RESULT OF QUESTION 4 .....	201
FIGURE D.5 THE RESULT OF QUESTION 5 .....	202
FIGURE D.6 THE RESULT OF QUESTION 6 .....	202
FIGURE E.1 ENTITY RELATIONSHIP DIAGRAM OF THE DEVELOPED SYSTEM'S DATABASE.....	204

# ABBREVIATIONS

A-condition	: Alteration Condition
CAA	: Computer Aided Assessment
CBA	: Computer Based Assessment
CBR	: Case-Based Reasoning
C-satisfier	: Condition Satisfier
E-condition	: Existence Condition
ERD	: Entity Relationship Diagram/Database Conceptual diagram
E-Relationship	: Evolution relationship
IRD	: Ideal Reference Diagram
MCQ	: Multiple Choice Questions
RD	: Reference Diagram
RDG	: Reference Diagram Group
SR	: Scenario Reference

# CHAPTER 1

## Overview

### 1.1 Introduction

Diagrams are increasingly used in many design methods, and are being taught in a variety of contexts in higher education such as database conceptual design or software design in computer science. They are an important part of many assessments. Currently computer aided assessments are widely used for multiple choice questions. They lack the ability to assess a student's knowledge in a more comprehensive way, which is required for diagram-type student work. An increasing number of student diagrammatic solutions require computer assistance in order to increase the quality of the assessment process.

This chapter gives an introduction to the thesis. It is organised as follows: Section 2 identifies the problem addressed in the thesis and its objectives; Section 3 discusses the approach adopted to meet these objectives; Section 4 outlines the contributions; and the thesis organisation is given in Section 5.

### 1.2 Aim and Objectives

The research aims to develop a semi-automatic assessment framework which enables the use of a computer to support the assessment process of diagrammatic solutions, with the focus of ensuring consistency of feedback on the solutions.

Automatic assessment of student diagrammatic solutions is a relatively new field with around ten years' history (Thomas et al., 2008). Efforts have been made in automation of marking student diagrams where different techniques are proposed. However, the literature review in Chapter 2 shows that full automation of marking student diagrams has not been successfully achieved yet. This research focuses on a semi-automatic assessment environment which supports human markers. The

outcome of the research may be an intermediate stage for the future fully automatic systems as well as having immediate practical uses.

During the diagram assessment process, much of the examiners' time is occupied with marking student diagrams. They check the student diagrams against ideal diagram solutions. Computer support can ensure the quality of the diagram marking. It may also shorten the assessment time and reduce the assessment cost. Thus, any level of support to this process is useful.

The intention of this work is to provide computer assistance not only to the marking phase but also to other phases of the current manual diagram assessment process. The main objective is to reduce or remove as many of the repetitive tasks in any phase of the process as possible. As the same tasks are performed less (possibly only once) by the examiners, consistency of grades and feedback on the solutions are achieved.

The objectives of this research are:

1. To identify the repetitive tasks in the assessment process.
2. To develop techniques to reduce the repetitive tasks or remove them completely where possible.
3. To develop a novel framework that provides a platform where different intelligent techniques work together to support the assessment process of diagrammatic solutions.

### **1.3 Approach**

In order to meet these objectives, the thesis explores three interrelated threads:

- The computer support that is required for diagram marking
- The specialised diagram drawing editor that is required for automation of marking
- The guidelines that are required for writing questions (scenario text)

Insights achieved from understanding the assessment process of diagrammatic solutions are used as a basis for the proposal of a new assessment framework. The repetitive tasks found especially in the marking phase of the process are utilised for



the automation. The research employs various techniques in order to increase the automation of the marking process.

The proposed solution ensures the consistency of the diagram grades and feedback generated during the marking. It uses a design trace method to match components of student diagrams with each other and against components of the ideal diagram. The design trace method gives contextual information about each diagram component. The usage of design traces for contextual information is studied and discussed in Chapter 4. A special online trace capturing technique is developed for the design trace model for use in the diagram assessment. The research also develops a prototype diagram editor which implements functionalities required for the trace model.

A new concept of partial marking style is proposed in order to mark student diagram components, which are grouped by using the trace model. The shortcoming of the partial marking style is identified and it has been integrated with the full marking style to avoid the limitation. The partial marking enables the full automatic marking of the same types of diagram components. Some automation rules are developed for semi-automatic assessment. The research proposes a prototype marking environment which implements the functionality required for both partial and full marking styles.

The semi-automatic approach allows the addition of new rules to the system for the further automation. The new rules could be gradually extracted from the previous judgements of the examiner on the student diagrams for the similar problem scenarios. To speed up the rule extraction, guidelines are suggested for writing similar text. The similar scenarios also increase the automations since they can use the existing rules during the marking. The guideline for scenario writing is optional for the semi-automation approach but it is beneficial for the examiners since it provides a way to classify the question type.

To assess the proposed approach, several evaluation studies are performed on the implementation of a prototype diagram and marking editors.

## **1.4 Contribution**

The contributions of this thesis are:

- Through the application of assessment in the diagrammatic solution domain, it has contributed to an enhanced understanding of “semi-automatic assessment”.
- The proposed framework gives a platform where a variety of technologies can be used to increase automation of the assessment process of diagrammatic solution.
- A novel trace model is developed, which captures design traces of student solutions and enables construction of contextual information of components.
- A new generic case concept is defined, which enables scalable adaptation rules. It contributes to the case-based reasoning method by defining a new way of indexing natural language text, which is a question text describing the system requirement.
- The novel partial marking style of student diagrams creates a new research direction in computer aided assessment community.
- A novel process model is developed, which integrates full and partial marking styles.
- It contributes to the online assessment area by presenting requirements of a new online diagram marking tool.
- A set of guidelines for writing question text is introduced for diagrammatic solutions.

## 1.5 Thesis Organisation

The research mainly focuses on Entity Relationship Diagrams (ERD) out of the common diagrams taught in Higher Education since it has all the generic properties of graph-based diagrams, which are composed of nodes joined by edges. Wherever student diagrams are mentioned in the thesis, they refer to ERD.

This thesis is organised into three parts. Part 1 introduces the problem and the approach taken. Part 2 describes the development of models required for the semi-

automatic assessment framework. Part 3 describes the development and evaluation of the system.

### **Part 1: introduction**

Chapter 2 gives an introduction to computer aided assessment (CAA) technology, the characteristics of students' diagrammatic answers and a review of CAA applications on diagram-type student works.

Chapter 3 studies the manual diagram assessment process. It identifies the characteristics of the manual process and outlines the requirements of a CAA system that is to succeed in supporting the assessment process. It introduces the proposed framework to address the requirements identified. The components of the framework are presented and discussed.

### **Part 2: Development of Models**

Chapter 4 introduces a new trace model which is used during diagramming. The model supports the proposed framework. The chapter discusses trace production techniques. One of the techniques is adapted to the trace model.

Chapter 5 presents a marking process model which describes the process of the partial and full marking style. It defines the cases used for automatic marking. It also gives guidelines for writing similar scenario texts to support the automation.

### **Part 3: System Development and Evaluation**

Chapter 6 describes the design and the implementation of the proposed framework. The implementation architecture is presented. The new design trace model is used for the diagramming editor and the marking process model is used for the marking environment.

Chapter 7 evaluates the developed system. The system is used, first to write questions, second to assess students online and third to mark students' work. Students' design traces and assessor marks are studied. Marking consistency is highlighted.

Finally, Chapter 8 concludes the thesis with a summary of contributions, limitations and future directions.

Figure 1.1 shows the structure of the thesis.

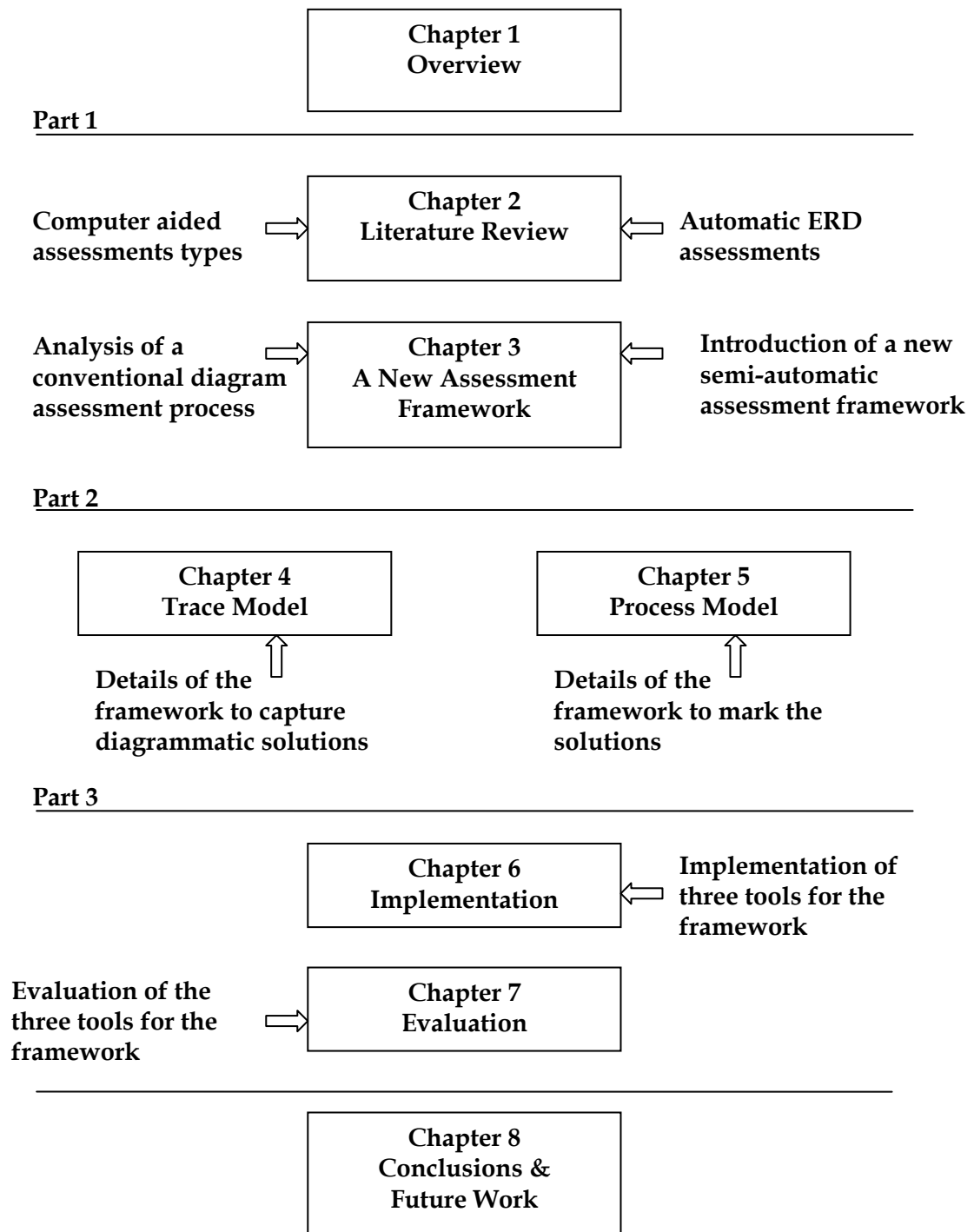


Figure 1.1 The structure of the thesis

## CHAPTER 2

# Literature Review: Automatic Assessment for Diagrammatic Solutions

### 2.1 Introduction

Traditional methods of assessment become difficult to use to undertake effective assessment and provide students with detailed, personalised and speedy feedback as the student numbers in Higher Education increase. Additionally, Laurillard (2002) states there is considerable pressure on higher education institutions to measure learning outcomes more formally. The growing field of computer aided assessment (CAA) is a widely acknowledged solution for these assessment issues. Bull and McKenna (2004) recently defined CAA as ‘the use of computers for assessing student learning’. CAA usually covers the use of computers in marking, administering optical mark reading cards.

CAA is being superseded by Computer Based Assessment (CBA), which refers to the use of computers for the entire assessment process including delivery of the assessment, administration, and management of the assessment and the provisions of feedback (King, 1994). Computer-based assessment involves a computer program marking answers that were entered directly into a computer, whereas optical mark reading uses a computer to mark scripts originally composed on paper (Conole and Warburton, 2005). CBA is employed in various subjects for numerous types of assessment in Higher Education. This chapter presents the background of CBA usage for diagrammatic solutions, particularly database conceptual diagrams (ERD).

This chapter is the literature review of the thesis. It is dedicated to the basic understanding of CAA and the available research applications to diagrammatic solutions. The first section covers the background of CAA and its related issues and discusses types of assessment. The second section focuses on the assessment of diagram-type student work and describes the rationale of the research into automatic

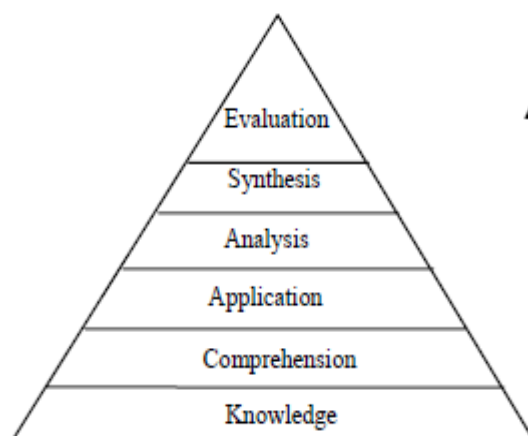
marking of graph diagrams, particularly ER Diagrams. The last section surveys existing CBA systems for ERD, reporting on past and current research.

## **2.2 Computer Aided Assessment**

Computer Aided Assessment has a history nearly as long as computing itself. The earliest documented reference for using computers in assessment dates back to 1959 (Hollingsworth, 1960). Most of the initial systems were built for computer science related subjects. Later on, systems were developed for different fields such as physics, mathematics and chemistry. During the 1990s, automatic assessment was used in academia to assess a wider variety of subjects. The studies in the use of CAA (McKenna 2000; Stephens and Mascia, 1997) report that a significant number of academics in UK HE employ some form of automatic assessment.

The CAA systems have been employed at various levels of assessment process. Their use ranges from management of assessment information to capturing student works and marking them. The marking level is one of the focused areas of the CAA research. Three different approaches for the automation of the marking process have been developed. Firstly there is the computer supported approach, where actual marking is performed by the human and other parts of marking process are automated (for example, Canup and Shackelford, 1998). Secondly there is a hybrid approach, where only some part of the actual marking is done by a human (for example, Mason and Woit, 1999). Lastly there is the fully automatic approach (for example, Arnow and Barshay, 1999).

Deciding an approach for a CAA system depends on what has to be assessed. Assessment activities relate to the learning outcomes. Bloom (1956) formulated a taxonomy of learning outcomes (Figure 2.1). At the lower levels students engage the subject at surface level. They memorise and recall without a deep understanding of the material. At the higher level the students obtain and show a deeper understanding of the material. They can use their knowledge; for example to criticise, compare or evaluate.



**Figure 2.1 Bloom's taxonomy**

McAlpine (2002) states that assessment of the outcomes at the lower end of Bloom's taxonomy traditionally relies on only one 'correct' answer whereas, assessment of higher-order outcomes relies on longer written answers or essays. The automatic marking of the one 'correct' answer can be performed by simple matching algorithms for any CAA systems. However, automatic assessment for other types of student response remains a complex task for researchers to examine. Charman and Elmes (1998) argue that assessment for all types of learning (specifically oral, presentation and interpersonal skills) can't be fully automated by using current technologies. They suggest the employment of hybrid and assisted approaches for the higher levels.

Brown and Race (1996) list the qualities of assessment in their work. Table 2.1 shows these assessment qualities. Automating the assessment process potentially supports most of these qualities. It increases reliability and fairness because the same marking mechanism is employed to mark each piece of work. It removes the possibility of discrimination. It forces both students and educators to respect deadlines when online submission is used. An incremental style of assessment is enabled by providing detailed feedback to the examiner. For example, the examiner can find out averages of the student results for each question and they can improve the questions or teaching methods based on those averages. A variety of methods are used to give feedback to students. Feedback may state only the grade obtained, illustrate the incorrect answers or highlight both the student's strengths and weaknesses and provide meaningful advice on how to improve upon this. The detailed feedback enables students to learn from their mistakes and improve themselves.

**Table 2.1 Assessment qualities according to Brown et al (2002)**

Valid	Accurately assess the delivered material
Reliable	Promote consistency between assessors
Fair	Offer fair opportunity for success
Equitable	Be indiscriminating between students
Formative	Give many opportunities to learn through feedback
Well timed	Provide learning stimulus and be fair
Incremental	Increase reliability and consistency over a period of time
Redeemable	Allow a series of opportunities
Demanding	Challenge students and ensure high standards
Efficient	Be manageable within the constraints of resources

Usage of CAA requires advance planning and some collaboration between teaching, IT support and administration staff as well as the educational requirements. For example IT supports deals with authentication and security issues. They need to be informed and work together which may increase the workload of the examiner. Additionally, authoring of teaching and assessment material for automatic assessment takes more time than for traditional methods. However, once the automatic assessment is set up and fully functioning, it reduces the human resources required.

Assessment can be categorised as either summative or formative. CAA systems can be used for both formative and summative purposes. Summative assessment concentrates on measuring and quantifying the learner's performance. Formative assessment provides helpful feedback to the student to enhance learning and supplies feedback to the educators to adjust the delivery of the material. It enables personalised learning by selecting the material for assessment from an exercise



database according to the student's profile. It supports formulating an accurate judgement about a student's achievement.

The CAA research field focuses on the interests of educators from a wide range of disciplines. Various publications with interdisciplinary topics have been produced. Pedagogical and practical benefits are commonly perceived. The TLTP3 project (Bull, 1999) summarises the opinion and finding of the research in this field by stating that "Computer aided assessment can be used to enhance the student learning experience, expand assessment processes and potentially provide efficiency gains for academic and support staff".

### **2.2.1 Automated assessment types**

Educational assessments use objective or free-response question types. Objective questions have a single correct answer whereas free-response questions have more than one correct answer or more than one way of expressing the correct answer (Brown et al., 1999). There are various types of objective and free-response questions. Examples of objective question types include true/false, multiple choice, multiple-response and matching questions. Free-response question types require essay-type answers, diagrammatic solutions, and program code. This section discusses these question types from an automatic assessment point of view.

#### **2.2.1.1 Objective questions**

The major advantage of objective questions is that the responses to the questions are suitable for automatic marking. However, objective questions have been traditionally viewed as a low-level question type, which lacks the ability to assess higher levels within Bloom's taxonomy (Pritchett, 1999; Davies, 2002). Simas and McBeath (1992) argue that by designing appropriate questions, all the six levels of the taxonomy can be sufficiently tested. Charman and Elmes (1998) support their argument and found that objective assessment is sufficient to cover the main aspects of evaluating student learning. Woodford and Bancroft (2005) address the problem of how multiple choice questions (i.e. MCQ) of the objective assessment can test higher levels of cognition. They provide a practical checklist to write MCQs for this purpose.

A national survey in 1999 into the use of CAA in the UK higher education sector showed that 84% of computer based assessments use objective assessment (Bull and Collins, 2002). Boyle et al (2002) states that multiple choice questions are the most frequently used and the best known question types of objective assessment. The students are required to select the correct answer or answers from several options. Simple text and number matching of objective assessment involves completing a sentence or free entry in response to a question. The response is marked using pre-defined search strings or numbers. The hotspot graphical approach is to have a graphic where the student has to select an area or item in response to a question. The correct response area is defined by the question designer in advance. Bull and McKenna (2004) states that this graphical interaction gives major advantages to CBA over other forms of objective assessment since it gives the ability to assess different levels within Bloom's taxonomy.

Haladyna, (1997) argues that designing objective questions can be time-consuming and requires skill and creativity. Duke-Williams and King (2001) suggests that sufficient care in their construction should be taken. The development cost of questions is justified by reusing them in the assessment and they are stored in item banks for this purpose. Mills et al. (2002) and Sclater (2004) have looked at setting up, maintaining and adapting item banks. Questions in the item banks are classified according to their difficulty levels. Lilley and Barker (2003) developed an adaptive assessment system, which uses the difficulty levels of the questions. The system issues questions of a difficulty level that depends on the student's previous responses. If a question is answered correctly then expectation about the student ability is increased and a more difficult question is given and vice versa.

Many commercial and researched tools have been developed so far, which support various types of objective questions. For example, Perception (Question Mark Computing Ltd, 2004) supports 18 item types and TOIA (2004) supports nine types. Sclater and Howie (2003) highlight the importance of fulfilment of institutional needs as well as questions types the CBA systems should have. Boyle and O'Hare (2003) argue that the lack of an overarching strategy or institutional IT infrastructure is preventing the educators from more widespread usage of CBA systems.

### 2.2.1.2 Free-response questions

Davies (2001) states that there is steady pressure for the use of 'more sophisticated' question types in automatic assessment. Free response questions are suitable when the higher levels of Bloom's taxonomy need to be assessed, specifically: the application of knowledge, analysis, synthesis and evaluation. However, the automation of marking the free responses requires new complex algorithms and approaches which are much harder than the simple matching algorithm for the objective questions. A variety of research has been done for automatic marking of free response questions, such as the computer programs, essays, and diagram type student works.

#### 2.2.1.2.1 Computer Programs

Computer programming is an increasingly popular activity, not only for computer science students but also across a large number of other disciplines. CBA systems for assessment of computer programs are available and used in taught modules. Some of the CBA systems are still in development stage and researchers are working on them. Brusilovsky and Higgins (2005) states two categories of CBA systems for programming: (1) to assess program-tracing skills; (2) to assess program-writing skills. The first group of systems assess students' knowledge of programming language semantics. They present students with a program and students are asked to trace the program and manually produce its output, such as QuizPACK (Brusilovsky and Sosnovsky, 2006). The systems then automatically evaluate the student solutions. They execute the program or the algorithm with the same data and compare that result with the one entered by the student.

The second group of systems evaluates the student's ability to write programs. The systems execute student programs against a set of tests and compare their results with the results of the teacher or model program. These systems form the majority of work on automated assessment of programming assignments such as CourseMarker (Higgins et al. 2003). The marking process of CourseMarker is based on regular expression-matching of source code and test output, and returns a detailed feedback breakdown of the success or failure of the student's work. BOSS (Joy et al. 2005) is another well-known example of a modern automated program evaluation system as it performs automatic tests for correctness and quality, checking for plagiarism, and providing an interface for marking and delivering feedback.

Researchers have also looked into intelligent programming tutors as well as automatic assessment tools. Intelligent tutoring systems analyze student answers by using a range of knowledge-based approaches. They can generate a problem to assess the student's knowledge and they provide detailed feedback. The systems proceed with the next problem that is most relevant to the demonstrated level of knowledge. They diagnose student answers by using embedded knowledge about the domain. An example is SQL-Tutor (Mitrovic, 2003), which diagnoses student errors in solving SQL programming problems by using a set of domain constraints. A knowledge-based analysis typically allows intelligent tutors to achieve a significantly better "understanding" of student answers and to provide extensive feedback for incorrect or suboptimal answers.

#### **2.2.1.2.2 Essay-type work**

CBA systems for assessments of essay-type works are not commonly used in real educational environments, unlike programming assessment. The systems are still in the research stage with few practical applications. Among free response assessment, the automatic marking of essay-type solutions has attracted significant research interest for the last forty years (Tsintsifas, 2002).

Williams (2001) states four conceptual models for automated essay grading described in the literature. The first model mainly relied on linguistic features of the model solution to return a mark (Page, 1966). Linguistic features include attributes such as the number of words, the average sentence length, the amount of punctuation and many other syntactical characteristics. It suggests that the features of an essay could be used to predict the mark that a human examiner would assign to the essay.

The second model uses a hybrid approach of combining linguistic features with structural features of a document (Burstein, 1998). The third model ignores document linguistic and structural features and uses Latent Semantic Analysis and the "bag of words" approach (Landauer et al, 1998). It analyses the textual content to understand the deep structure of each statement. The fourth model uses a combination of modified key words and linguistic features. This model is based on text categorisation techniques (Larkey, 1998). Human judges are needed to grade samples of student essays before the computer systems complete the task.

In contrast to essay-type student work, the automatic marking of diagram-type student solutions has only recently attracted interest among CAA researchers. The automatic marking of diagrammatic solutions is examined in a separate section due to the importance for this thesis.

### **2.2.2 Automatic assessment of diagram-type solutions**

There have been many investigations by researchers into the diagrammatic reasoning areas. They have looked into precise diagrams, such as the use of diagrams in mathematical proof (Jamnik, 1998). However, student diagrams are different from the precise diagrams. Smith et al (2004) state that they are malformed, missing, or have extraneous features. This section briefly gives information about research exclusively into the automatic assessment of student diagrams.

The DEAP Project (Deap, 2007) uses statistical techniques to grade student exam scripts. This work likens imprecise diagrams to free-form text. The associated commercial intelligent free-form text assessor uses latent semantic analysis for marking (Thomas, 2003). In this analysis, to perform a semantic matching between student text and the ideal solution, the semantic of a word is determined from the paragraph in which that word occurs. The DEAP Project looks for suitable keywords in student answers to mark free-form text. It considered “association” in student diagrams equivalent to a word in text and applied the same statistical technique to grade the diagrams. Their initial results show that the automatic grading of simple diagrams is feasible.

DATsys (Tsintsifas,2002) is part of the Ceilidh system and provides a customizable environment to create various kind of diagrams. Model answers and student diagrams are captured by DATsys and then another Ceilidh module marks the diagrams. The Ceilidh system was originally designed for assessing programming. The system marks, for instance, a student flowchart diagram by first converting the diagram into a BASIC program and then checks the program against the test data.

The ABC Project (Tselonis, 2005) and the INFAC system (Fan and Tanimoto 2007) aim to present student designs to the human marker after filtering out diagrams which are identical so that the speed and quality of the marking process can be improved. ABC uses graph isomorphism with some heuristics for local metrics of matching diagrams. It is reported that the approach works well on large, artificial

examples, but tests with real examination data produced some unexpected results. The results have shown some matches which are not actually valid (over-match). In their approach, matching is largely dependant on the component labels.

The automated student diagram assessment system (Hoggarth and Lockyer, 98) provides feedback without specific staff or student attention but does not assess the diagrams with any grading or marking. It compares two diagrams on their internal processing, the processing order, and the connections between the processes. Before submitting the solution diagram, students have to perform a symbol mapping between the components of their diagram and an ideal diagram. Then the differences between the two diagrams are found and guidance and feedback are given to the student.

The DEAP and ABC projects compare visible similarities of student and ideal diagrams. They don't require any changes in the diagramming. Their methods are successful in simple diagrams but need to be improved for complex diagrams. DATsys and Hoggarth's system require additional tasks for the student to do and compare the internal processing. Their methods have been applied to more diagram types successfully.

### **2.2.3 Summary of automatic assessment**

Many parts of the assessment process have been automated in numerous disciplines in higher education for both administration and pedagogical reasons since 1959. Automated systems are employed for formative and summative assessments to mark student fixed (e.g. MCQ) and free responses (e.g. essay). The majority of the commercially available systems automatically mark student fixed responses and they are used mainly for summative purposes. Automatic marking of student free responses is much harder than for the fixed responses.

This research focuses on the semi-automatic assessment of diagrammatic solutions. Instead of the semi-automatic approach, one of the objective question types for diagram assessment could have been chosen to use in order to fully automate the marking process. For example, MCQs can be used to assess the students' design knowledge. Diagrams can represent the design or structure of a system. In that case, students are required to have designing skills. Davies, (2002) argues that objective questions lack the ability to assess higher levels within Bloom's taxonomy likesuch as

designing skills. Although Woodford and Bancroft (2005) address the problem of how the objective questions can test higher levels of cognition, Haladyna, (1997) argues that designing objective questions can be time-consuming and requires skill and creativity and Duke-Williams and King (2001) suggests that sufficient care in their construction should be taken. Therefore, the research has chosen to focus on the free response questions instead of objective questions for diagram assessment.

The developed systems for free responses generally use one of three different approaches. The first approach changes the manual assessment so that the marking process can be easily automated. The second approach automates some parts of the manual marking and leaves the rest to the human marker. The last approach assists the human marker during the process by providing a supportive environment. Among assessment of the free responses, some systems for programming (e.g. CourseMarker, Higgins et al., 2003; BOSS, Joy et al., 2005) are successfully developed and practically employed at universities. The automation for assessment of essay-type solutions has been researched for many years. New techniques have been created but not many practical applications have been developed (Tsintsifas, 2002). Some of the research into the assessment of the program and essay-type work has already looked into adapting their techniques to the diagram assessment (e.g. the DEAP project , 2007; DATsys, Tsintsifas, 2002).

Automating diagram assessment has recently started being researched. Like other free response areas, the research already has three different approaches in its early stage: full automatic (e.g. the DEAP project, 2007), semi-automatic (e.g. the ABC project, Tselonis, 2005) and online assessment (e.g. the INFAC project, Fan and Tanimoto, 2007). This thesis focuses on the semi-automatic approach since it could use the findings of both full automatic and online assessment research as well as contribute to them.

Research in Section 2.2.1.2.3 can be categorized into two types. The first type is like the DEAP and ABC projects, which compare visible similarities of student and ideal diagrams. They don't require any changes in the student diagramming process. The second type is like DATsys and Hoggarth's system, which require additional tasks for the student to do. These systems have been applied to many diagram types successfully which make them attractive to the research in this thesis. During development of the design trace model for this research in Section 4.4, an online trace

production method is chosen, which requires some changes in the student diagramming process.

The next section introduces the conceptual database diagram type and gives the rationale of studying the automatic marking for this diagram type.

## 2.3 Entity Relationship Diagram

James Maxwell defines 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 (Encyclopedia Britannica, 11th edition). This definition is accepted as general enough to subsume detailed definitions given by various fields. Diagrams communicate information like text and pictures. They are used to represent topological maps, geometrical concepts, philosophical ideas, engineering plans and scientific abstractions amongst many other things. Most known types of diagrams obey certain rules of notation.

Research about diagrams relates to numerous and broad areas of science, humanities and the arts. For example, some research in applied psychology analyses how diagrams relate to cognition (Cheng et al, 2001). In education, diagrams are explored as tools that improve learning (Cheng et al, 2001). In computer science, significant studies have been made in areas such as automatic layout and visualisation (Herman et al, 2000), diagrammatic reasoning (Kulpa,94). Diagrams are used in the software development process (e.g. UML) .

Diagrams are broadly used in computer science for visualisation, for solving computational problems and for software specification. For solving theoretical problems well known diagrams include state charts, petri nets and state transition diagrams. There are hundreds of known diagram notations for software specification such as data-flow diagrams, entity relationship diagrams (ERDs), structure diagrams, process diagrams, use case diagrams and class diagrams. Many types of diagrams exist to represent graphs and tree structures.

Among the many known diagram types in computer science, this thesis focuses on graph-based diagrams, which are composed of nodes joined by edges. It particularly looks into entity relationship diagrams (ERD) for automation. Assessment of ER



diagrams has certain qualities, which attract the interest of automatic diagram assessment research. The three important qualities are as follows:

### **2.3.1 Question types**

The ERDs in the assessment context are very rich. They offer scope to investigate a range of assessment issues. A wide range of question styles can be used in the assessment for a basic ERD solution. Questions that require the production of an ERD can have a single correct solution, which is identical to the teacher ideal solution, or alternative equivalent solutions. Like any objective question type, the marking student diagrams of a question with a single correct ERD requires a marking scheme and simple pattern matching with little interpretation. On the other hand, the marking process for a question with alternative correct ER diagrams can require a great deal of domain knowledge and a different style of marking scheme with complex pattern matching. Additionally, correct solutions of the questions can range between small and large or simple and complex.

### **2.3.2 Extension of ERDs**

ER diagrams (Chen, 1976) have a relatively simple notation. A basic ERD notation is extendable to include things such as specialisation and generalisation to draw more complex structure (Elmasri et al., 1985). This quality of ER diagrams allows the research to be developed incrementally. The research later can investigate the way to adapt the findings of the automatic assessment of ERDs to Extended ERDs and various other graph diagrams (e.g. Class Diagrams).

### **2.3.3 Common usage**

The entity relationship model, the product of which is ERDs, and its extensions is widely used in industry. The model is used during database development, which is one of the fastest growing areas of software applications during last decade since databases play a vital role in e-commerce. Consequently, ERM is taught as a component of many university courses with large student numbers. The widespread use of ERM provides access to a wide range of experienced practitioners. The practitioners in the teaching area mark the student solutions. Some of them set the questions and marking schemes. Available questions, marking schemes and student

solutions with feedback can be used in the development of marking algorithms and validation of the research results.

### **2.3.4 Design skills**

ER modelling is a complex task, involving the identification of relevant facts from dissimilar information sources, many of which are text based. Batra and Davis (1992) argue that novices find the ERM task difficult and exhibit systematic errors in their models. Such errors are due both to the lack of understanding of the subject domain and also to unfamiliarity with the task. Increasing the exposure to practical ER modelling would improve students' ability to design databases.

The need of extensive practise in order to gain database design skills and a widespread usage of the ERM make the research into the automation of the ERM assessment relevant and useful. The extendibility of the ERDs increases applicability of the research since it enables the adaptation and generalisation of the research findings to similar modelling areas. The next sections summarise the previous studies made on ERDs exclusively.

## **2.4 Current automatic ERD assessment research**

This section gives brief information about four recent studies, which research exclusively into the automatic assessment of entity relationship diagrams. Additionally, there have been many other studies on computer assisted database design and database schema integration. These could be directed at automatic assessment, but are not addressed here.

### **2.4.1 DEAP**

The DEAP (2007) research is the only project which aims to automatically mark the student ERDs purely based on type and names of components in their diagrams. Like simple diagram marking previously studied, the approach uses latent semantic analysis. The basic scheme is modified in two important ways to be able to apply to ERD marking. First, the diagram comparisons use elements of natural language processing (Manning & Schutze 2002) to identify synonyms and misspellings (Thomas et al. 2005). Second, not only association (relationships in ER diagrams) but also larger structures (namely patterns) are used to identify equivalent sub-structures

in diagrams (Thomas et al. 2006). This new pattern concept is introduced to deal with alternative equivalent solutions of the same question. The tool, called exerciser, is developed to test a pattern for many to many relationships. It is reported that the results are encouraging. The research currently focuses on the development of a pattern library.

The researchers also reported one of the problems they experienced. Two small quite different diagrams can be regarded as equivalent (Thomas, 2004). This problem is a result of using latent semantic analysis. It is known that the analysis doesn't work properly in essay marking if the text size is small (Dessus et al. 2000). Like the essay making, the diagram marking suffers from this behaviour of the analysis. The researchers suggest using their tool (namely Exerciser) in formative assessment. Students submit their solutions many times to improve their design with the help of feedback given. The tools for free text marking are used in the same way in formative assessment since the research has not reached the level where it can be used in the summative assessment. DEAP research is an ongoing project.

### **2.4.2 DATsys**

Bligh (2002) attempts to adapt DATsys for ER diagram marking and has done an initial study. The problems are reported by Higgins and Bligh (2006). The method used in DATsys provides all possible diagram elements as complete, uneditable entities with incorrect entities also included as distracters. In the context of the Entity Relationship diagrams, the report finds this method helps the students too much to get the correct solutions. Another finding is that the method does not deal with several equally valid model solutions with slightly differing, mutually exclusive features. The report also mentions the importance of the diagram appearance during marking and of feedback given. The research currently focuses on resolving the shortcoming of DATsys but has not yet specified the approach followed.

### **2.4.3 VLE-ERM**

The VLE-ERM (Hall and Gordon, 1998) project concentrates on the methodology of ER modelling and provides immediate feedback to students about the quality of their models. The project provides a virtual learning environment in which students create their design. The environment forces the students to enter their design in a methodological way. In order to create a component, the student has to use a phrase

from the scenario which justifies the existence of that component. Unlike DATsys, students are allowed to name their components freely. The approach is very useful since an initial problem for the students was to correctly identify entities, attributes and relationships from the problem description. The system has got only one correct solution for each scenario and students are only allowed to produce that solution. Any wrong attempt by students is trapped and immediate feedback given based on the system solution. This makes the system inflexible for alternative solutions. The research later focuses on collaborative features of the environment.

#### **2.4.4 KERMIT**

KERMIT (Suraweera and Mitrovic, 2002) is an intelligent tutoring system aimed at university-level students learning conceptual database design. KERMIT contains a set of problems and ideal solutions to them. Unlike traditional intelligent tutoring systems, it hasn't got a problem solver. The system compares the student solutions to the ideal solution using domain knowledge represented in the form of constraints, which are classified into syntactic and semantic ones. The semantic constraints enable the system to deal with alternative student correct solutions. Like VLE-ERM, correspondences between the components of the student and the ideal solution are found by forcing the student to highlight the word or phrase in the text whenever a new part is added to the diagram. These correspondences are used to fire the appropriate production rule/s in the semantic constraints. In the case of violation of any of these constraints, feedback is generated.

KERMIT is successfully implemented and commercialized. However, KERMIT's approach requires the problem text written in such a way that the references of all the components must be in the text. This explicit referencing covers only basic types of design issues. Traditional questions in the database module which require application of complex design criteria need to be simplified and adapted to the system. That makes the tool only useful for the initial stage of the learning database design.

The other limitation of the system is that it allows students to predict the correct solutions. The author enters all correct and possible wrong references in advance to the system. Students are restricted to use those references in the text. That may make students see that some of their references are not part of the solutions on which

the system can give feedback. Therefore the question preparation must be done carefully. It requires forward thinking like in multiple choice question preparations. The author should have the experience of student solutions so that they can produce good reference lists for the text.

The approach also allows naming ambiguity. The system depends on the references for the problems in order to assess the diagram automatically. Like VLE-ERM, students can freely name their components after being given their references. A component's name and reference should be matched. The system doesn't check this compatibility. That gives the possibility of marking a correct component wrongly.

Semantic constraints in the system's knowledge base are very important. They are used to deal with alternative correct solutions. Production of these constraints is a difficult and knowledge intensive task. The constraints have to be complete and they have to produce alternative solutions of a given ideal solution in order to have a valid assessment. The research focuses on the production of the constraints in order to apply the approach on various diagram areas.

### **2.4.5 Summary of the current research**

All the research mentioned above addresses the problem of identifying components in student diagrams and proposes various solutions. The DEAP project focuses on marking the diagrams without getting any help from the students. The other research needs student involvement in the identification process before submitting their solution. VLE-ERM's approach justifies the student's involvement educationally. It forces the students to use the database design methodology whilst entering their solution into the system.

Only two research projects address the problem of the alternative solutions. The DEAP project proposes a concept called "pattern" in order to deal with the alternative solutions. For example, a specific pattern is used to deal with different representations of "many-to-many" relationships. KERMIT purposes semantic constraints and produced a complete set of constraints to mark ER diagrams automatically with student involvement.

All the research projects for ERD assessment in Section 2.4 focus on fully automatic systems. These projects are ongoing research. They have developed some practical applications with some restrictions so far. They achieved full automation of marking

ER Diagrams by either over simplifying the questions or restricting the student solutions. For example, the questions with one valid solution are asked and a limited list of component names for student solutions is accepted. A new approach, which covers more types of questions and applies fewer restrictions on student solutions, will be educationally more acceptable.

Full automation requires embedding a complete set of rules and necessary knowledge about the questions into the system. Although this is possible for an individual question, it is only practical by increasing the author's workload unacceptably. For example, the author could be asked to enter all possible solutions for a question. However, this is not practical for the author.

An automatic assessment system for the diagrammatic solutions must be educationally acceptable and its assessment's workload must be practical. In this sense, full automation hasn't been successfully achieved yet. This research focuses on the semi-automatic assessment, which covers more types of ERD questions, applies fewer restrictions on student solutions and has the practical assessment's workload.

KERMIT address the problem of the alternative solutions and marks ER diagrams automatically with student involvement. However, it covers simple question types as mentioned in Section 2.4.4. Like KERMIT, VLE-ERM's approach alters the diagramming process. In addition, VLE-ERM justifies the alteration of the diagramming process. The approach for this research will require student involvement in the identification process, like the KERMIT and VLE-ERM approach. The semi-automatic approach will mainly focus on the consistence of the marking rather than increasing the automation.

## **2.5 Summary**

This chapter gave a presentation of the computer based assessment area, focusing on diagram-type solutions. It first introduced some important assessment terms, concepts and types of automatic assessments. Particularly objective and free response questions were looked into in an automatic assessment context. It has been found that the CAA research community has taken the Bloom taxonomy as a reference point for their discussion about effectiveness of question types. Very brief

background about essay and computer programs among the free response questions was given in Section 2.2.1.2. For both of the response types, the existing research uses one of automatic, semi automatic or online marking approaches.

The chapter also specifically provided discussion on the assessment of diagram type response, from which it was found that recent emerging research about diagram type response has mainly used the fully automatic approach. The current research has seen that the full automation of the assessment is a complex task. A semi-automatic approach is an untouched area to be looked into for the assessment of diagram type responses.

The chapter later proceeded to introduce the rational behind choosing the ERDs for the thesis among all diagram types. Popular usage of ERD, and its extendable notation are main qualities that attracted researchers. The automatic assessment systems for ERDs have been briefly reviewed, and their approaches and limitations are discussed in the context of the automation.

The approach of VLE-ERM and KERMIT systems were successfully implemented and used in taught modules. Section 2.4 has identified the limitations of the approach. Chapter 3 elaborates these limitations and proposes a new semi automatic approach in order to overcome the shortcomings. The approach reduces the repetitive tasks in the question preparation and marking stage of a conventional diagram assessment process. The components of a framework, which use the approach, are introduced in Chapter 3.

# CHAPTER 3

## A New Semi-automatic Diagram Assessment Framework

### 3.1 Introduction

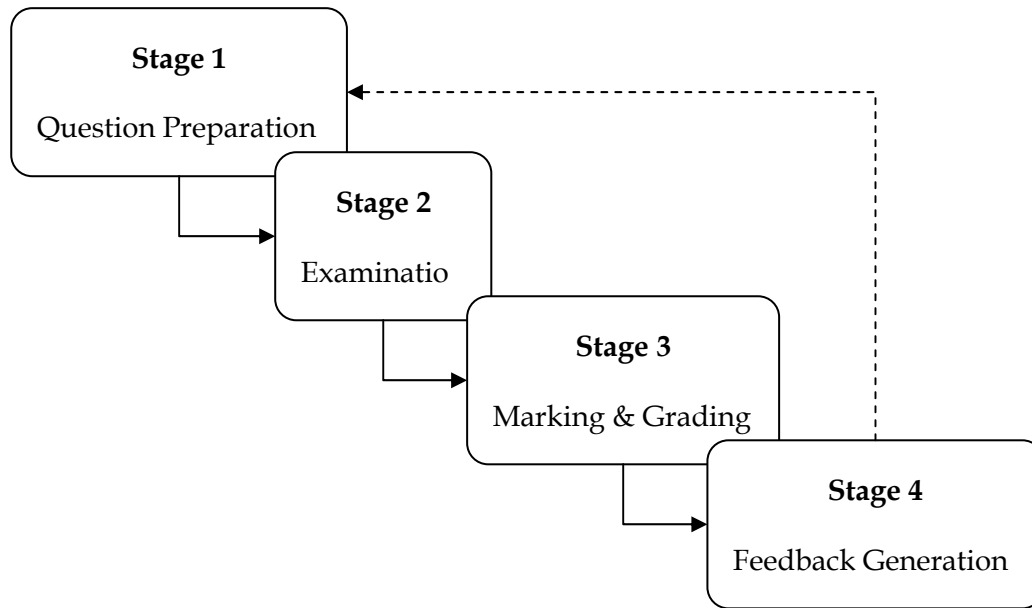
All the research mentioned for the diagram assessment in Chapter 2 works on fully automatic assessment systems. Automatic assessment for student entity relationship diagrams and graph-based diagrams has not been achieved yet. In contrast to those approaches, this research develops a new assessment tool, which mainly helps the examiners during marking. The research analyses the existing manual assessment process in order to computerise it as much as possible. This approach is seen as an intermediate stage for the fully automated assessment and the research results have some immediate practical uses.

This chapter introduces a novel framework for the semi-automatic assessment approach and is organised as follows: The first section describes the manual diagram assessment process. It highlights the problem areas which need to be improved. The second section introduces the semi-automatic assessment concept. It presents the novel framework and describes the functions of its components. It mentions the advantage of this framework against the challenges identified in the first section.

### 3.2 Manual Diagram Assessment Process

The diagram assessment process consists of four stages like other assessment types. Figure 3.1 shows these stages. The assessor has tasks to do in each stage, which are described briefly in this section. Later, some tasks are focused and the areas of improvement are identified. The relationships between the stages in terms of workload are explained.





**Figure 3.1 Assessment process cycle**

In the first stage, the examiner prepares a question, its solution and the marking scheme. In database design context, they write scenario text which is the requirements of a new database model. They prepare an entity relationship diagram as a model solution for the requirements. The examiner uses the generic marking scheme for database modelling questions. The marking scheme is coarse as a general practise. The examiner may adjust the scheme during the marking. This may results in inconsistency in the marking if the examiner doesn't remark the previous marked solutions.

In the second stage, the question is given to students and students produce their solution. Traditionally, in diagram examinations, students draw their diagrams on paper for the given requirements. However, there are also other types of examinations (e.g. Bancroft et al, 2003 and Joy & Luck, 1998) that allow students to produce their solutions and submit online rather than on paper. In both cases, whatever technique is employed to take the students' solutions, the examiner's task and workload is very limited at this stage. However the technique used significantly affects the marking stage.

The third stage compares student solutions with the model solution and gives marks to the student work. The marks on the works are graded by using the marking scheme. The examiner's workload in this stage depends on the number of the student solutions. In the database assessment, if the student numbers are too high

then the scenario text may be prepared which requires a small diagrammatic solution so that the examiner spends less time on each solution. The scenario text given as coursework may require a much larger diagram. In this case, additional tutors may help to mark the papers.

The last stage gives feedback about the student's work alongside their grades. If there are high student numbers, the tutor may prefer to give generic feedback. The feedback mentions common mistakes that the student solutions have. Students are sometimes allowed to see their solution papers with the marks on it. That gives them personalised feedback on their work. In the diagram assessment, the examiner may use cross or check signs to mark components in student diagrams. The examiner sometimes writes a grade next to the signs. Students can see where they gain or lose marks by comparing their solutions with the model solution. If the assessment is formative, the feedback is more important than actual grades. It can be too time consuming for the examiner to leave textual feedback for individual component for the student diagram, in which case they may prefer to use the cross and check signs on the diagram components instead.

Preparing general feedback about the student work helps the examiner to improve the requirement scenario. They find out the possible reasons of common mistakes. The reason could be how the requirements are written in the scenario text. If the requirements are vaguely written then the students might have many alternative solutions as well as common mistakes. In this case, the examiner needs to modify the scenario text to reduce the misunderstanding of the requirements. If common mistakes have been made because of wrong application of design criteria taught then the teaching methods may be improved and the next time a similar scenario text is prepared and asked on the exam. Figure 3.1 shows this relationship between the first and the last stage in the assessment process as a dashed line since the examiner sometimes uses general feedback.

### **3.2.1 Challenges in each stage**

The stages in the diagram assessment process can be improved to increase the quality of the overall assessment and to reduce the assessment load when the student numbers are high. The manual diagram assessment process has been used without any major changes for long time. The quality problems are ignored and not tackled

since the minimum assessment requirements are satisfied. Some educational aspects are sacrificed for the sake of reducing assessment load. This section mentions the problems in the assessment process stage by stage.

### 3.2.1.1 Question Preparation

The main task in the question preparation stage is to write scenario text. The scenario text covers some or all design criteria taught in the module. This means that students have to apply the criteria whilst drawing for the scenario to be able to get the correct diagram. Depending on the criteria, the scenarios can be of various types. The way the scenario is written might make the design task simple or complex. No research has been found which addresses the problem of producing scenario text of various types. The research into text writing has focused on writing requirement specifications of software systems aiming at concise and consistent statements (Miriyala and Harandi, 1991).

Especially in the formative assessment, the scenario type is very important. The assessment is a part of learning process. Students learn from their mistakes. They can practise one type of scenario before moving on the next. If the students make mistakes, a similar type of scenario is given in order to make the students improve their design skills. No research or formal guideline has been found on producing the various types of scenario text. The examiner may write the scenario text and gradually may improve it when necessary and classify it over the years by analysing the students' results.

The tutorial sessions are usually used to make students practice their diagramming skills. The diagrams are drawn interactively together with students in a tutorial session. A small number of scenario texts are required in tutorials. The teacher can adjust the complexity of the scenario by asking additional questions whenever it is necessary. Therefore, the scenario type has not been a crucial problem in the practical sessions. The problem in the tutorials is that some students may hesitate to ask questions (Low et al, 2009). This prevents them from getting the personalised feedback. If students solve problem separately, then the tutors can check their solutions and give detailed feedback to students. The feedback is important for the formative assessment. Students need to improve their skills by doing many exercises. Exercise questions for formative purposes require classification of the scenario text which doesn't currently exist.

### 3.2.1.2 Examination

The main task in this stage is to obtain student solutions. The student solutions can be obtained on paper or in digital form. In a paper-based exam, student diagrammatic solutions are malformed and in addition to the solutions, the papers contain students draft works (Thomas, 2004). Student diagrams for coursework are usually better formed and the papers contain only the final diagrams. This is mainly because there is no time restriction and students often use a diagram editor to draw their final diagrams.

Obtaining student work in digital form is straightforward task and can be used in coursework assignments. However in the case of a test or an examination, the digital form of submission requires additional facilities and help. This may make online submission less preferable option for the examination. Many universities have recently invested in online assessment tools. These tools accept student work in digital format and mark the work or assist the examiner with their online marking. However, no evidence has been found that mainstream tools (e.g. Moodle, Qmark) support diagram assessment. This may cause the examiner to carry on doing paper based assessment. The examiner may prefer the solutions in digital form to the paper form since the solutions are better formed.

The examiners' task is the same and independent from the form of the student solutions obtained in this stage.

### 3.2.1.3 Marking and Grading

The examiner's main task is to mark the student solutions in the marking and grading stage. They match the components in the student diagrams with components in the ideal diagram. The main challenge is the matching process. It is not always straightforward. The components can not be matched directly all the time. The matching could be between same (e.g. entity to entity) or different typed components (e.g. entity to relation/attribute) and the components could be matched in various cardinalities (e.g. one to many). The examiner may need to make the matching approximately in order to simplify the marking process.

The examiner uses a generic marking scheme rather than a detailed one since the marking process may not include all types of matching. That makes the diagram marking subjective. In the case of more than one marker, achieving consistent

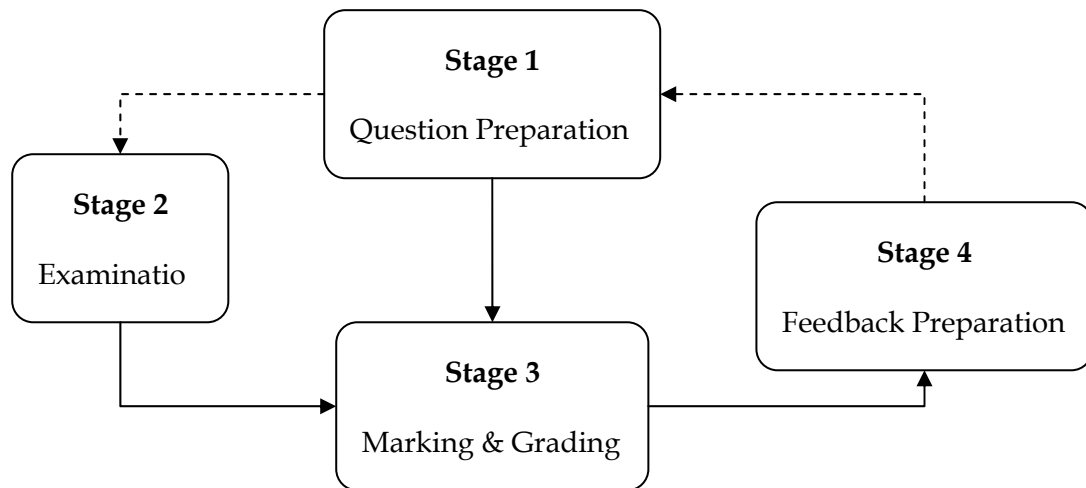
marking is difficult. In the case of a high number of diagrams, the human marker's judgment may vary from the beginning to end of the marking process. The examiner should be extra careful. The grades of the solution should not be very different on average although there may be some inconsistency between marks given to student solutions.

#### **3.2.1.4 Feedback Generation**

The examiner may write a general feedback about the student solutions. This feedback may not be quantitative and may not include the evidence. After completing the marking stage, the examiner has got the understanding of the student work. They may give the reasons for the student mistakes and include their comments in the feedback. They have student work with marks on them as a record. If they don't keep an additional record of the student work during marking, there is an extra task for the examiner to go over all the work and use the marks to write feedback. When the general feedback isn't written straight after marking, only very important mistakes can be remembered. This may make the feedback incomplete. There isn't any guideline for feedback generation. It is subjective and depends on the assessor's experience. When two or more tutors mark the papers, having consistent feedback requires a very detailed marking scheme.

#### **3.2.1.5 Relationships between the stages**

All stages in the marking process are related to each other. Figure 3.2 illustrates these relationships. The feedback stage depends on the marking stage. Marking papers in detail increases the quality of the feedback. Keeping a record about the marks separately eases the feedback preparation task. Storing this record in a digital format enables personalised feedback.



**Figure 3.2 Direct and indirect relationships between stages**

The marking stage depends on the examination stage. Better formed diagrammatic solutions without any duplication ease the marking task. Digital copies of the diagrams enable online marking. Raikes et al (2004) argue that online marking eases the record keeping task. Raikes et al (2004) also highlight the positive impact on learning in their paper. Heinrich and Lawn (2004) developed a tool to mark ER diagrams online. They emphasize the potential benefits of the feedback quality.

The marking stage directly depends on the preparation stage as well. Well prepared scenario text will have clear requirements. The student work will have the expected right and wrong components. The marking scheme prepared in the first stage can be easily used with minimum alteration during marking.

The quality of scenario text depends on the experience of the author. If they have written similar scenarios and have exam feedback then next time they can improve the scenario or write a better but similar type of scenario.

The assessor's workload is different in each stage of the cycle. When the number of students is increased, most of the assessment workload comes in the marking stage. Therefore, most of the research in automatic diagram assessment focuses on the marking stage exclusively as it is seen in Chapter 2. However, as described, the stages aren't independent from each other. The marking task can be eased to a certain degree by improving other stages.

### 3.3 Semi-Automatic Diagram Assessment

The literature review in Chapter 2 shows that available studies for the diagram assessment work on fully automatic assessment systems and they are on-going research. In contrast to those research approaches, this research focuses on a semi-automatic approach so that the research results have some immediate practical uses. This approach could form the foundation for fully automated assessment.

Semi-automation in this thesis means that humans do the novel tasks of a job and the computer does the repetition of those tasks. Human and computer collaborate to complete the overall job. Repetitive parts of a job are important for semi-automation. The preparation and the marking stages of the assessment process have got many repetitive tasks the examiner needs to do. This section focuses on these stages in terms of semi-automation.

This section first identifies the repetitive tasks in the manual marking process and introduces a concept of a new marking style. Then it gives the definition of an identical diagram segment, which is the essential part of the semi-automatic approach. Later it describes the components of the approach. Finally it mentions how the preparation stage could be used to improve the efficiency of the approach.

#### 3.3.1 Marking process

Marking exam scripts takes longer in the beginning than later on. During the marking, the examiner recognizes correct sections as well as identifying wrong parts of the student diagrams by comparing the model solution. They make decisions to grade the sections of the student diagrams. The similarity of the student diagrams reduces the time takes to mark since the examiner simply recalls the decision they made previously. The diagram sections which are not identical to any of the marked segments up to that moment take more time since they require understanding of the diagram segments.

Student diagrammatic solutions for a scenario text are generally very similar but not identical as a whole. However, it is possible to find identical segments among all student diagrams for the same scenario. These identical segments correspond to either the segment of the teacher's ideal solution or previously marked diagrams. The student diagrammatic solutions were analysed to find out the number of

identical segments in 2008. 20 random samples were taken from the student examination scripts of the first year university database module. Table 3.1 is the analysis of the samples for scenario text in Figure 3.3. It shows that 43 percent of the student diagram components are the same and correspond to the teacher's ideal solution, and 55 percent of them are the same as each other without directly corresponding to ideal solutions.

*"The Loughborough library lends books only to its members. On the application form, the details required are name, address and telephone number. Each member is assigned a unique number and issued with a ticket giving this number. Members may borrow many book copies at a time. A record of all book copies borrowed is kept. When the loan is issued, the loan date and due date are recorded. When a book copy is returned, the corresponding loan is updated with the return date.*

*If a member wishes to borrow a book (book title) that is already on loan to another member, that book may be reserved. Each reservation has a reservation date and is given a unique reservation number.*

*The Librarian buys new books for the Library as necessary. Normally, several copies of a particular book are bought. Each copy of each book title is assigned a unique book copy number. A book (book title), on the other hand, is uniquely identified by an ISBN number. The book title, author, date of purchase and price of each book are recorded. Different copies of the same book title can be purchased on different dates at different prices."*

**Figure 3.3 Sample scenario text for the database design**

This observation underlies the basics of the approach. The approach proposes to save the diagram segments, which have been marked by the examiner, and recall the marks whenever a new student diagram has the same segments as the saved segments. If the student diagram has some segments which are not among the saved ones then the examiner interprets the new diagram segment. In other words, the examiner basically accepts or rejects the segment based on the ideal solution. The judgement of the segment is saved for future use.

The approach considers a new diagram segment as a novel task. The human marker needs to do this novel task. Each occurrence of the segment in different student diagrams is seen as a repetition of that task. A Computer needs to do the repetitions.



The approach reduces the number of diagrams marked by the assessor. It also makes the marking process consistent. The assessor doesn't have to repeat their judgement on the same segment for various student diagrams. This repetition may lead to inconsistency in marking.

**Table 3.1 Components from student diagrams**

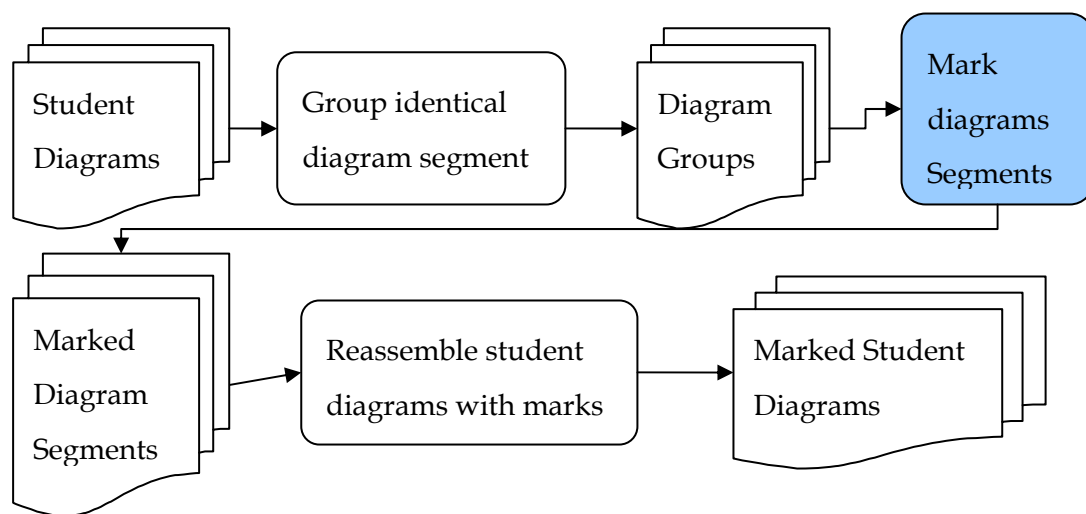
	Student Diagram component	Occurrence (%)	Ideal Solution
Same type One to one Matched	Member (e)	100	Member (e)
	Book Copy (e)	39	Book Copy(e)
	Book Title (e)	39	Book Title(e)
	Reservation (e)	26	Reservation(e)
	Loan (e)	30	Loan(e)
	Has ( r )	26	Has (r)
Same type One to many Matched	Book	61	(Book Copy + Book Title)
Different type One to one matched	Reservation(r)[B-M]	35	Reservation (e)
	Reservation(r) [C-M]	9	
	Loan [B-M](r)	43	Loan (e)
	Loan [C-M](r)	9	
Non-matched Rejected	Library	30	-
	Application form	17	-
Non-matched accepted	Librarian	17	-

The analysis of the student diagrams for the scenario text in Figure 3.3 also gives evidence that the matching process is not always straightforward. Table 3.1 illustrates different types of matching. The matching could be between the same

types. For example, all “Member” Entities in the student diagrams matched with “Member” entity in the teacher’s ideal solution. The matching could be between different typed components. For example 39 percent of student “reserve” relationships matched with “Reservation” entity in the model solution. The matching could be in various cardinalities (e.g. one to one or one to many). For example 61 percent of student “Book” entity matched with the combination of “book copy” and “book title” entities. The approach provides an additional benefit with the complicated matching since it reduced the repetition of the harder task.

The approach may automatically mark the whole of some student diagrams. If they consist of the segments which are the collection of the saved segments marked before then the computer simply recalls the marks and uses them to mark the whole of a new student diagram. The approach depends on the order in which the diagrams are presented to the human marker. The order affects the number of the diagrams marked without any human intervention.

The approach can be modified to make the marking process independent from the order. In the modified version, the computer identifies identical segments in all student diagrams and puts the different segments into separate groups. Then the examiner marks a diagram fragment from each group independently. Later the computer marks the student diagrams by using information in the groups. Since the examiner marks the diagram pieces, human intervention in the marking process can be reduced. Figure 3.4 shows the marking process.



**Figure 3.4** Partial marking process

The human marker sees diagram segments during the marking in the new approach. This research calls this new style of marking “Partial Marking”. The principles of partial marking will be given in the next section. Detailed discussion is in Chapter 4.

In partial marking, the examiner is involved in the marking process only for the number of diagram groups rather than the total number of student diagrams. The computer consistently grades the marked groups by using the marking scheme. Therefore grouping correctly is the key part of the process. The correctness of the grouping depends on the criteria used to match the diagram pieces. The next section discusses these criteria.

### 3.3.2 Identical segments

The semi-automatic approach needs to identify identical segments among student diagrams. Two diagram segments are identical if they have the same number of components with the same properties. An Entity-Relationship diagram consists of two basic components; an entity type and a relationship type. This section defines properties for these types to match the diagram components.

Entities in different diagrams could be considered as matched exactly if they have the same name and the same number of attributes with same name. This initial definition is pretty tight and finding two identical entities among student diagrams may be hard. This definition would increase the number of times the examiner is involved in the marking process. In order to make the definition feasible, the same question should be asked many times over the years in the practical sessions as a formative assessment. This will increase the number of student diagrams for the question. The likelihood of identical diagrams is increased.

The criteria for entity matching are not complete to use in the semi-automatic approach. In some cases, components may be matched wrongly. The diagrams in Figure 3.5 belong to two different students based on the scenario in Figure 3.3. The “Book” entity in the first diagram clearly corresponds to “Book Title” with the missing attributes in the teacher solution. However, the “Book” entity in the second diagram corresponds to the “Book Copy” entity. The approach would not get the examiner to mark the second “Book” entity since it matched with the previously accepted “Book” entity by giving it the wrong meaning. Therefore, even the tight

definition above is not sufficient for correct entity matching. The definition should also include the contextual meaning of an entity.

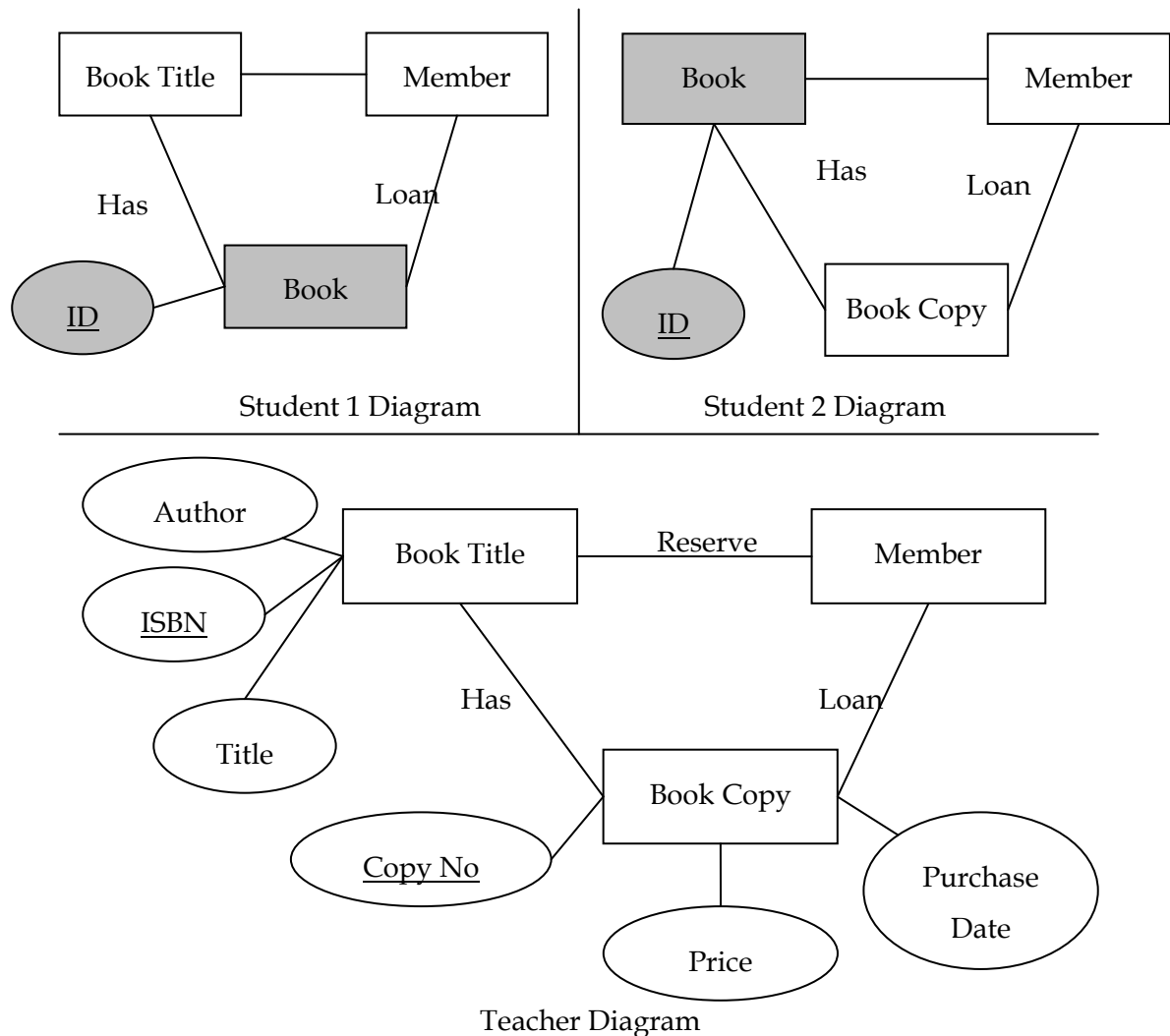


Figure 3.5 Entity name ambiguity

The Kermit approach (Suraweera and Mitrovic, 2002) finds the contextual meaning of an entity by using an additional input obtained from the students. The students need to highlight the related text in the scenario text during diagramming. This approach simplifies finding a semantic match of the two components automatically. Figure 3.6 illustrates the process. Students highlight the “Book Copy” phrase in the scenario text before they draw the “Book” Entity component in their diagram area. The computer matches the student component with the “Book Copy” entity type in the ideal diagram.

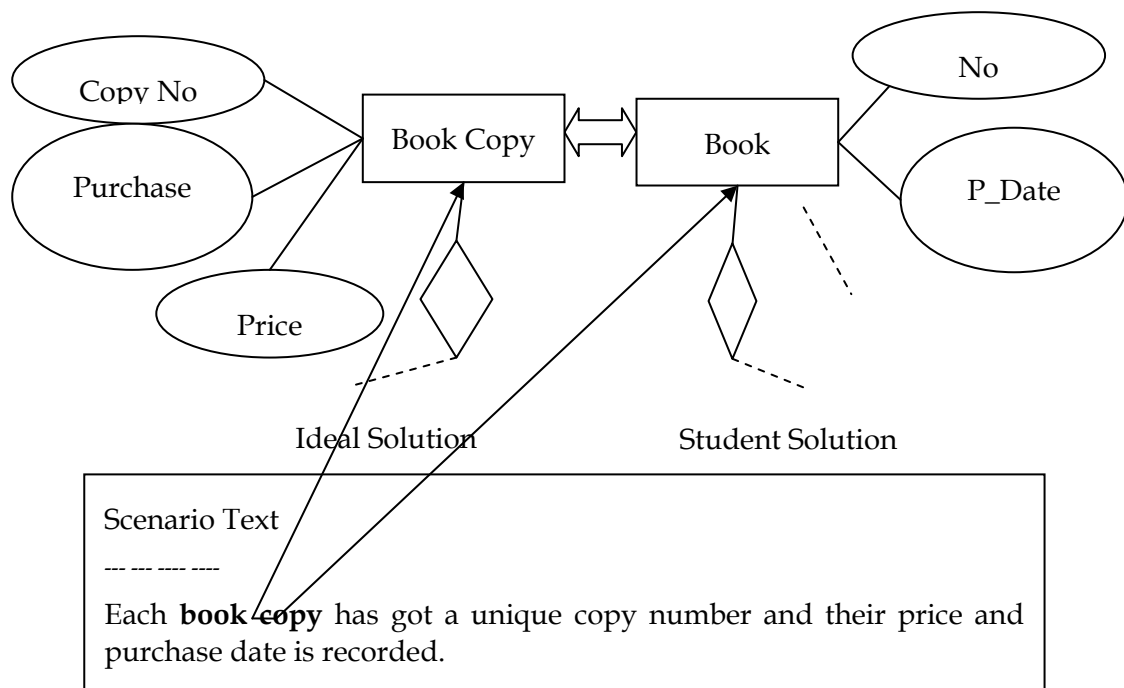
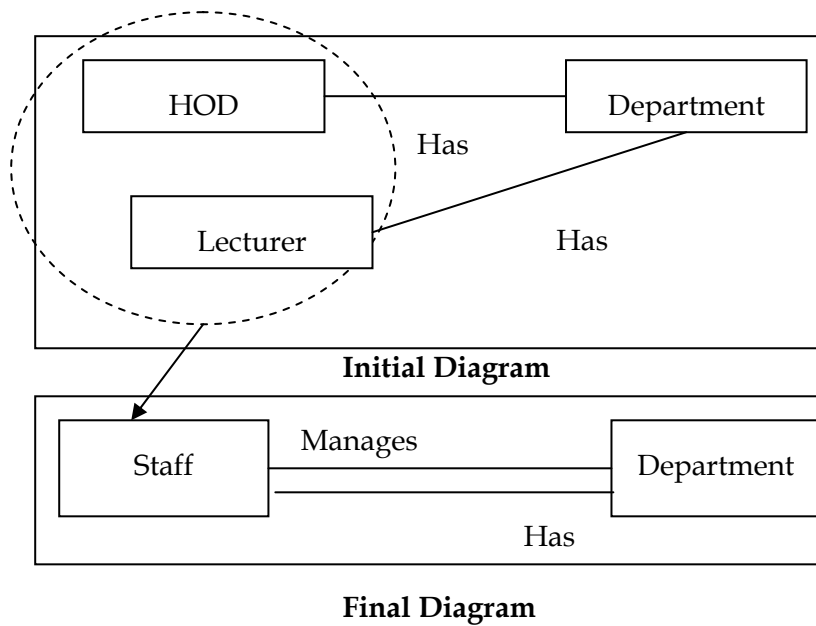


Figure 3.6 Entity matching in KERMIT

Finding a related text to diagram components is not a straightforward task (Suraweera, 2001) and also the direct correspondence sometimes doesn't exist. The main reason is that designing a conceptual database model is an iterative process. Although the initial diagram can have a direct link to the scenario text, afterwards that initial diagram is subject to modification by applying design rules and constraints in the domain. The final diagram might have only implicit links to the scenario text. It is not always possible to show those links explicitly without all the intermediate steps between the initial and the final diagram.

Figure 3.7 shows the limitation of the KERMIT approach. The figure illustrates the development of a student diagram. It shows an initial and a final student diagram. The student initially has "head of department" and "Lecturer" entities. Later they replace them with "staff" entity. It is hard to show the link explicitly between staff component and the related scenario text without using the initial diagram. Their approach needs to be improved to include these types of cases.



**Figure 3.7 Conceptual database design is an iterative process**

This research proposes using not only the reference text but also the intermediate diagrams in order to define the contextual meaning of a component. In partial marking context, the examiners see the diagram segment with its design history. They may understand the students' reasoning from the design history. This enables them to give accurate feedback to students. However extra caution should be taken. The design history may overload the examiner with too much diagram information during marking.

Some researchers in the requirements engineering field focus on the traceability of a design requirements. Ramesh and Jarke (2001) developed a reference model for requirements traceability. Chapter 4 of this thesis develops a new trace model for the partial marking using their reference model. The model mainly deals with capturing the design traces and the method of their representation.

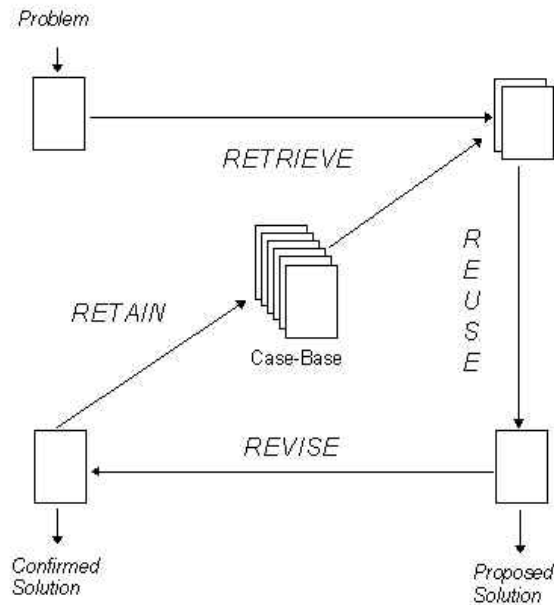
The partial marking approach eliminates the repetitive marking task and enables semi-automation. This section suggested using the student diagramming steps or their design history to determine identical segments for the repetitive tasks. The next section describes using the design history to automate the partial marking process further.

### 3.3.3 Automation

KERMIT is a fully automated system and uses the scenario referencing technique for diagram marking. The system keeps all possible references for a question scenario. The examiner enters the references, which students might make during diagramming, into the system for the automation purpose. The partial marking approach also uses the scenario referencing. However, it extends the technique to support wider question types. It uses the diagramming history of student diagrams. This extension makes the full automation of the partial marking much harder and impractical since all possible diagramming history cannot be foreseen and entered into the system like the KERMIT approach.

The semi-automatic approach aims to decrease human involvement in the assessment. The computer can mark some diagram segments during the partial marking phase when the examiners provide their solutions to the system. They prepare the ideal diagram for the scenario text in the preparation stage. The system can have the examiner diagramming steps for the diagram solution. If any student diagram segment has got the same diagramming history as the ideal solution, then those segments can be marked by the computer. Although this is not enough for full automation, it reduces the examiner involvement during the marking process. In the case of two or more markers, they may enter their own diagramming steps for the ideal solution into the system. This may increase the automation since there may be more different design histories for the solution, which students might have.

The ideal solution for previously marked diagrams can be used to mark a similar student diagram as well as the same student diagrams. Using previously marked diagrams makes the approach very similar to Case-Based reasoning (CBR). CBR is the process of solving new problems based on the solutions of similar past problems (Kolodner,1993). CBR systems keep a library of past cases. Each case typically contains a description of the problem, plus a solution and/or the outcome. To solve a current problem: the problem is matched against the cases in the case base. The current problem might be identical to a problem of some cases. Then the solution of these cases is used for the current problem. Sometimes similar cases are retrieved. The retrieved cases are used to suggest a solution which is reused and tested for success. If necessary, the solution is then revised. Finally the current problem and the final solution are retained as part of a new case. Figure 3.8 illustrates this cycle.



**Figure 3.8 The CBR cycle (adapted from Aamodt & Plaza, 1994)**

The CBR method has been used when records of previously solved problems exist and remembering previous experiences is useful (Watson, 1997). In the CBR context, for the manual diagram assessment, the problems of cases are the student diagrams. The solutions of the problems are the marks or feedback of the diagrams. Student diagrams with marks for a question are the case-base and exist when the examiner (i.e. specialists) has got the ideal solution or experience if the same question has been used previously. The examiner remembers their previous judgment (i.e. the case) during marking. They use the cases with or without any revision.

For the partial marking process, all student diagrammatic solutions for a scenario are limited and available as a batch at the beginning, unlike other domains for which CBR systems are used. The availability of all student solutions enables grouping the diagrams before marking them. During grouping, identical segments from the student solution are gathered into groups. In the CBR context, each group is a problem and the examiner's feedback is the solution. Human marked groups are the cases. The CBR cycle is only observed when the examiner uses the same question again in an assessment. The system can retrieve the identical cases from the case-base and reuse these for marking. If there is not any identical case then the examiner marks the diagram segment. Marked segments are retained in the case-base. The revision stage in the CBR cycle is not in the marking process since it uses only the identical cases.



The CBR method retrieves the cases from the case-base to solve the current problem. It adapts the existing cases to the problem and comes up with solutions. Likewise the marking system could retrieve the previously marked segments which are similar to the current diagram segment in the case where there aren't any identical cases. Then the system may adapt the marks to mark the current segment. The partial marking process will have a revision stage when the similar diagram segments are used to mark a new segment. The partial marking will become a cyclic process like the CBR method.

Figure 3.9 illustrates the new partial marking cycle. The partial marking cycle is the detail of the marking process in Figure 3.4. A new diagram in the figure is one from each diagram group shown in Figure 3.4. Process B in Figure 3.9 represents the reuse stage of the CBR Method. Process B marks the diagram segment by using adaptation rules. If Process B couldn't mark the segment then the examiner marks the segment in Process C which represents the revise stage.

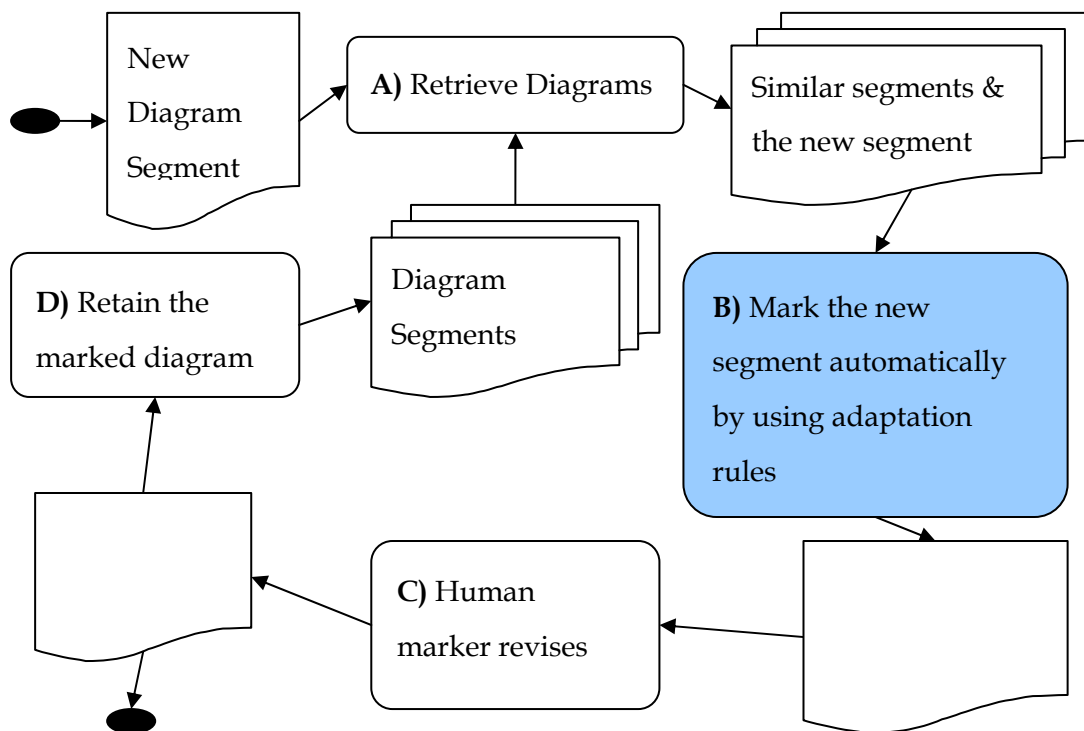


Figure 3.9 The partial marking cycle

The CBR method allows the case-base to be developed incrementally. Similarly, the number of cases is increased gradually during the partial marking. The case-base for the marking initially has some marked diagrams which have been entered by the assessor in the preparation stage. During partial marking, human marked segments

are stored in the case library and the number of the cases is increased. Later domain experts may analyse the case library and generate some adaptation rules. The adaptation rules help the system mark the diagram segments automatically. Auto-marked segments are also stored in the system to use next time. Process B in Figure 3.9 uses these rules. Using the rules increases the system efficiency.

The main issues of the partial marking cycle are defining cases for marking, identifying similarities between segment groups and the adaptation rules. Chapter 5 deals with these issues. The next section focuses on the preparation stage. It describes the link between the adaptation rules and the scenario text.

### **3.3.4 Scenario authoring process**

Student diagrams for a question scenario text are normally similar to each other. They have identical and similar diagram segments. This enables semi-automation. The scenario texts used in the assessment can also be similar to each other. The scenario text covers various design criteria taught in the module. The criteria used and the way the scenario is written may make the scenarios similar. The examiner may find writing similar scenarios straightforward when they have sufficient experience. They recall the previous decisions they have made during writing. Unlike the student diagrams, scenario text, which the examiner writes, rarely has identical text segments. Semi-automation of the writing process is much harder than the marking process. If the computer generates identical text segments to write similar scenarios, they can be too mechanical to use in an assessment. However a computer can support the author to write similar scenarios.

Computer assistance for the scenario preparation is very beneficial for formative assessment. Many similar scenarios are used in formative assessment. If the author has a guideline for writing similar scenarios then computer assistance can be provided to the author to use the guideline. Many guidelines are suggested and tested empirically to produce clear and accurate software system descriptions (CREWS, 1999 and Phalp et al, 2007). However the research doesn't focus on writing similar questions. Therefore a special guideline is needed for the assessment. The guideline can also help to produce various scenario types. The types can be organised in complexity levels as it was mentioned in the first section.

Similar scenarios are also important for the marking stage in the semi-automated system. Student diagrams for similar scenarios may have common right and wrong solutions. These common solutions are useful for the partial marking process cycle. The solutions can be analysed to produce generic adaptation rules for the automatic marking part. For example, if the author writes a new scenario by using guideline text, then the text can be one of the known scenario types. Student diagrams for the scenario may have known diagram segments. The system automatically marks those segments with the rules.

The author may write scenario text without using the guideline. In this case, fewer of the adaptation rules could be used during marking. The examiner marks those segments which decreases the automation. Later on, the domain experts may see the scenario text and create new scenario types.

This section has briefly mentioned the importance of the similar scenarios in terms of semi-automation of the assessment process. A detailed discussion is in Chapter 5. The chapter develops a guideline for writing similar scenarios.

### **3.4 Summary**

This chapter proposed a new semi-automatic diagram assessment framework and presented the basics of the framework. The new framework deals with the challenges of the manual diagram assessment process. The details of the framework are discussed in Chapters 4 and 5.

The chapter introduces a new partial marking process and describes the rationale behind the new marking process. The marking process forms an essential part of the proposed framework. It uses the student design traces to identify the identical segments. The detailed discussion of the design traces is left to Chapter 4.

The partial marking cycle was proposed, which adapts the partial marking process to the case-based reasoning method in order to increase the automation. The partial marking cycle is the initial form of the marking process model described in Chapter 5. The details of the model's components are covered in Section 5.4.

The relationship between the scenario preparation stage and the adaptation task of the partial marking cycle was established in Section 3.3.4. The details are left to Chapter 5 after the case definition for the marking cycle is given.

# CHAPTER 4

## Design Trace Model

### 4.1 Introduction

The previous chapter discussed new semi-automatic diagram assessment approach. The approach uses identical segments of student diagrams, which are identified by using their contextual information. Design traces of the student diagrams are used as their contextual information; Ramesh and Jarke (2001) developed a reference trace model for requirements traceability. This chapter uses the reference model to develop a new design trace model for the semi-automated model described in this thesis. All stages of the diagram assessment process are considered during the development of the design trace model. During model development, only ER diagram examples are given, although the developed model is generic enough to be adapted to support all graph diagrams. This chapter explains the development of the model by discussing alternative approaches. It gives the rationale behind the model.

The design trace model developed in this chapter consists of two main parts: trace definition and trace production. The first section of the chapter explains the trace definition of the model. It discusses alternative trace entities and traces for the model. The second section is for the trace production part and discusses various techniques for the production. It focuses on the student cognitive load and the examiner's workload for the production techniques. During model development, first the production technique is decided among alternatives then the trace definition is done, based on the chosen technique. The final section puts the chosen production technique and defined traces together and gives a design trace model.

### 4.2 Trace Definition

The reference model (Ramesh and Jarke, 2001) defines what trace entities and traces are and which traces should be captured. The model consists of two concepts

"entities" and "relationships". This section discusses the different properties of entities and relationships which should be considered for the semi-automatic assessment process.

### 4.2.1 Trace Entities

The purpose of a trace model is important for the model development. It determines entities that should be traced. A traced entity has got three aspects: The *kind* of the entity, *attributes* of the entity and the *granularity* of the entity. The kind and the attributes of the entity are straightforward to determine for assessment purposes. Conversely the decision concerning the granularity aspect is a complicated task and is significant for the semi-automation. It is discussed separately in the next subsection.

The kind of entity in requirement traceability describes which software documents (e.g. requirements, test cases, or design) should be involved. The assessment process has two kinds of documents: the question text and student solutions. For the assessment of a conceptual design model, the question text provides the database requirements, which is given to students in the exam and the student solutions are the student ER diagrams for the required database.

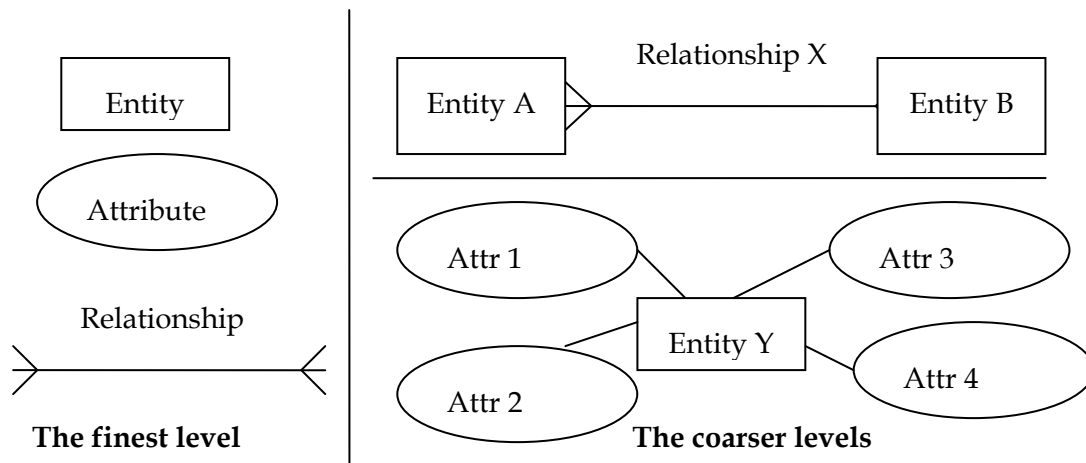
Each documentation entity in requirement traceability is enhanced by attributes, such as the source of the entity or status (e.g., incomplete, complete, or verified). In addition to supporting the planning of changes, some integrate an attribute, such as change probability, which indicates the likelihood of the change occurring. The attributes of document entities are optional for the semi-automatic approach. However they can still be defined to record the overall quality of the assessment process. For example, the complexity level of the scenario text, the feedback about the student diagrams and scenario text are kept as the attributes.

#### 4.2.1.1 Granularity of an entity

The granularity describes the granularity of the entities involved. The granularity is also called "different levels of traceability" (Lindvall, 1994). For instance, classes or attributes and methods of an object-oriented analysis are different levels of traceability. Paragraphs or sentences are the different granularities of a textual requirements document. The coarsest level is the ability to trace from one document

to another (Dellen, 1999). The most fine-grained level is to trace every single statement. The semi-automatic approach uses traceability to find out the contextual information of each design component. The approach prefers the most fine-grained level where possible in order to get the design trace of each component.

Student diagrams are one of the document kinds to be traced for the assessment. For a database exam, they consist of three main component types: entity, relationship and attribute of entity or relationship types. The most fine-grained level is to trace each component in a diagram. Other granularity levels could be to trace the sub-diagrams made from the combination of the different component types. Figure 4.1 shows an example of granularity levels. Relationship X with participant entities A and B, and Entity Y with its attributes are the granularity level of the student diagram document to be traced in the figure. The granularity level will be coarser if the attributes of Entity A and Entity B are included for Relationship X.

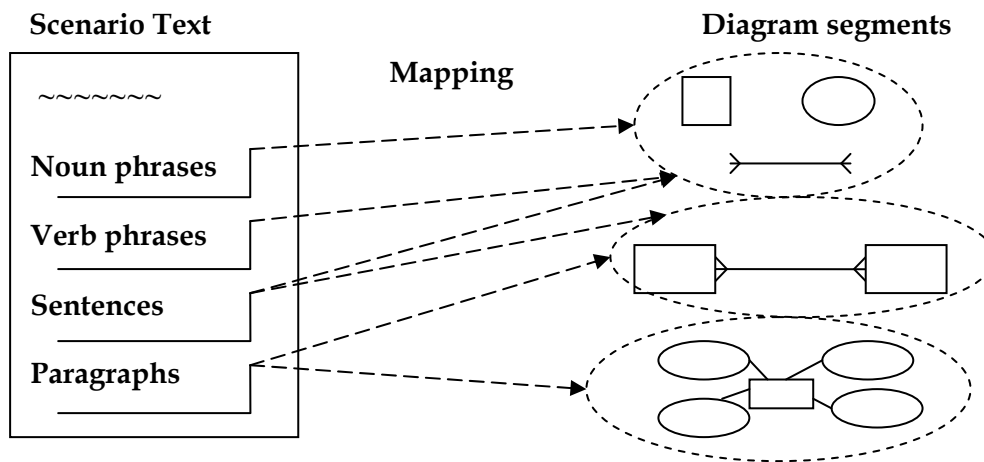


**Figure 4.1 Granularity level examples for database diagrams**

The coarser levels decrease the number of identical diagram fragments, which are found in the student diagrams. The performance of the semi-automatic approach directly depends on the identical diagram number as discussed in the chapter 3. Therefore the coarse levels are not desirable for the contextual information. However, in some cases, the coarser levels can be used. These cases are discussed in Section 4.2.2.

The requirements text is the other document kind to be traced for the assessment. The most fine-grained level for this document is to trace each noun or verb phrase in the requirement text. Other granularity levels could be to trace each sentence clause,

sentence, statement, or paragraph in the text. The granularity level of the requirements text and student diagrams are interrelated. The fine-grained level of the diagram document requires the finest level of the requirement document. Otherwise, the contextual information for each component would have ambiguity. For example, if a component (i.e. finest level) in a diagram maps to a paragraph (coarse level) in a text, this paragraph could correspond to many other components in the diagram. The paragraph fails to identify each component separately but it can identify all components together. So a diagram fragment, which consists of two or more components, can map to a paragraph. The coarse level in one document is correctly mapped to the coarse level of another. A component in a diagram can map to a verb or noun phrase (fine level) in a text without any ambiguity. Kermit and VLE-ERM use this level of granularity. Figure 4.2 illustrates various mappings between two documents at different granularity levels.



**Figure 4.2 Different granularity mapping**

The granularity levels of the requirement text are noun phrase, verb phrase, sentence and paragraph. The granularity levels of the database diagrams are component (entity, relationship and attribute) and different diagram fragments, which are a combination of the components. Figure 4.2 shows some granularity levels without highlighting any of them in order to use in the trace model. The decision on choosing the granularity depends on the relationship definition.

This section only discusses the boundary of the trace entities for assessment purposes. The next section gives the definition of relationships in the traceability context. They will be adapted to the semi-automatic approach.

## 4.2.2 Relationships to be traced

The proposed tracing approach needs a precise definition of the kinds of traceability relationships that must be captured and used for marking purposes. The concept of "relationships" has five aspects: (1) kinds of relationships described, (2) direction, (3) attributes, (4) setting of relationships, and (5) representation. This section discusses these aspects for the relationship definition of the new trace model.

### 4.2.2.1 Kind of relationship

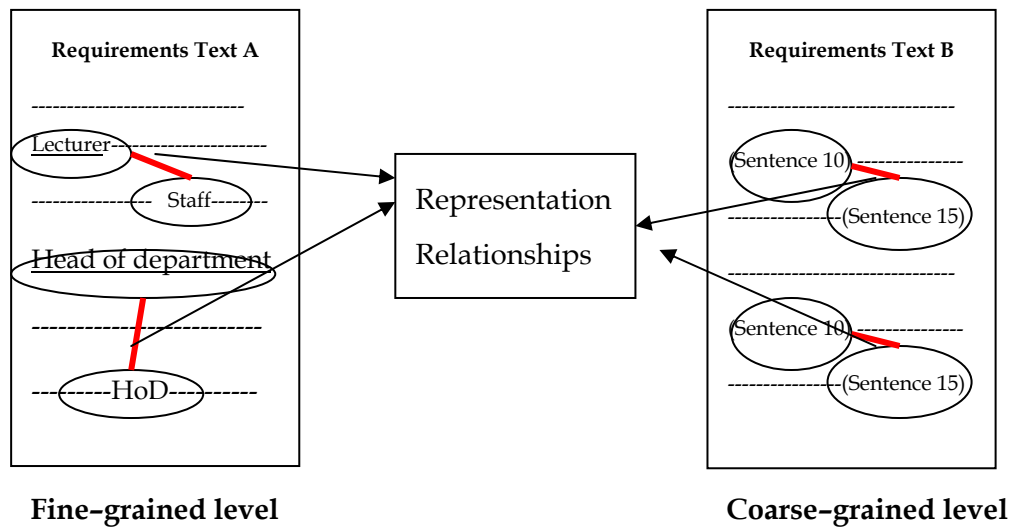
The traceability identifies three general types of relationship. The first type relates entities in the same type of software document. For example, a requirement depends on another requirement. The second type relates entities of different types of software document. For instance, a set of design classes realizes a requirement. The third type relates entities of different versions of the same document type. Bohner (1991) calls the first type vertical relationships and the second type horizontal relationships. The term used depends on how abstractions are arranged in a graphical representation. The third type is called evolutionary relationships by Pohl (1996). All three types of relationships are needed in the trace model for semi-automation. The usage of each type in the model is discussed in the following subsections:

### 4.2.2.2 Vertical relationship

Knethen (2002) defines two types of vertical relationships: representation and dependency relationships. Representation relationships are used for a relationship between documentation entities of different views that represent the same logical entity. Dependency relationships are used for a relationship between two documentation entities that depend on each other and represent different logical entities in an abstraction. A dependency relationship is used to find out the indirectly affected logical entities in the case of a requirement change. In the assessment environment, the requirements document is prepared before the exam and unchanged during the examination. Therefore the dependency relationship is not essential for the model so the research didn't include this type of relationship in the model. However the dependency relationships can be used to record the reason for the change when the requirements document is updated in the future.



Representation relationships relate all text fragments which represent same logical entity from a requirements document. Figure 4.3 shows two representation relationship examples. The representation relationship in the figure between two noun phrases “lecturer” and “staff” indicates that they refer to the same logical entity in the document A. Figure 4.3 also shows a relationship between sentences in the document B. The relationship indicates that the sentences have got the same logical meaning in the document.



**Figure 4.3 Representation relationships in requirements documents**

Representation relationships are desirable but not essential for the semi-automatic approach. They can affect the performance of the approach. They may reduce the horizontal relationships between two software documents. Lindvall and Sandahl (1996) uses two-dimensional (vertical and horizontal) traceability to reduce the number of traceability associations. Decreasing the number of horizontal relationships decreases the amount of human marker involvement in the approach. Discussion of the relation between horizontal and vertical links is left to the horizontal relationship part in the following sub-section.

The examiner can establish representation relationships manually in the preparation stage. A fine-grained level of the requirement text may increase the number of representation relationships. Those relationships increase the examiner’s workload. The coarse level may decrease the number of the relationships and the workload of the examiner. If the representation relationships are produced automatically, the workload will be independent of the document’s granularity level. The fine-grained

level may enable the automatic production of representation relationships. This issue is discussed in details in Section 4.3.

#### 4.2.2.3 Horizontal relationship

Knethen (2002) defines two types of horizontal relationship. The first type is a relationship between documentation entities at different abstractions on a certain abstraction level (e.g. between two system use cases in UML). Student diagrams and scenario text haven't got any refinements. This type relationship is not a requirement of the model. The second is a relationship between documentation entities at different abstractions called "Between-level refinement relationships". The design trace model for the semi-automatic approach, this relationship is between the student diagram and the scenario text. This type of relationship is called a scenario reference link.

Scenario reference (SR) link uniquely identifies a diagram fragment. The fragment contains one component at least. A component could be identified by one or more SR links. Figure 4.4 illustrates that four noun phrases in scenario text separately identify the "staff" entity in the diagram. If all related noun phrases are connected together by the representation links then only one SR link is enough to represent the relationship between textual fragments and the components.

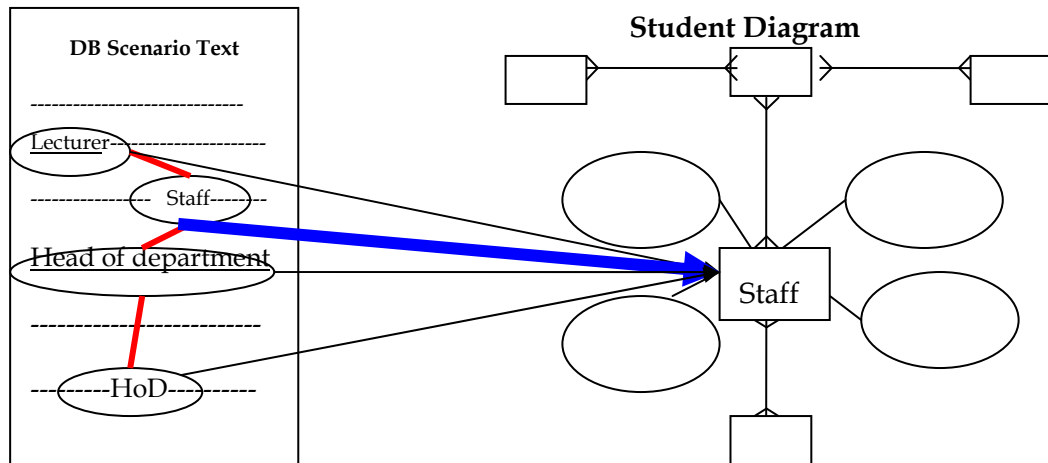


Figure 4.4 Scenario reference link

The number of SRs corresponds to the number of times a human is needed during student diagram marking. The reduction of this number increases the performance of the semi-automatic marking process. The representation relationships can reduce the number of SR links for the same logical concept to one for a component or

diagram segment. This decreases the examiner's workload in the marking stage. However, if the examiner produces the representation relationships manually, then this causes additional workload in the preparation stage. The automation of the production or, at least using a computer assistant, will be very beneficial for the semi-automatic approach.

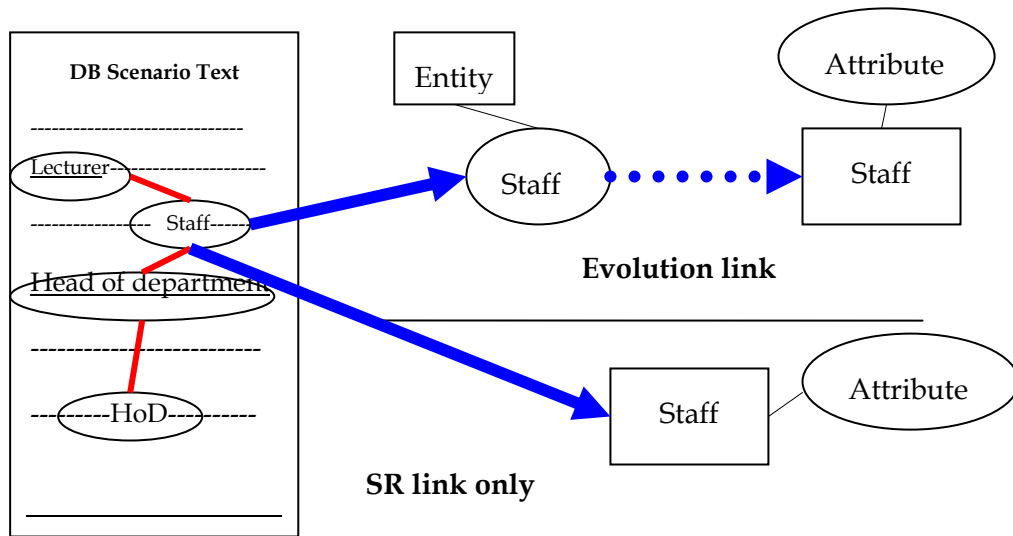
#### 4.2.2.4 Evolutionary Relationships

In the traceability literature, evolutionary relationships trace all previous versions of a particular documentation entity to recover its development history. Ramamoorthy et al (1990) called them historical links. Evolution relationships are essential in the assessment system since SR links are not always sufficient to directly identify a diagram fragment as demonstrated in chapter 3. Although the evolutionary links are to be used for this technical reason, they have an educational value as well. The evolution and SR relationship together represents the design processes of the students. The educational aspect of these relationships is discussed in Chapter 6.

The design trace model has two document entities: the requirements text and the student diagrams. In the assessment process, unlike in a software development process, the system requirements are not subject to change, whereas student diagrams are developed gradually with many alterations during the modelling. Evolutionary relationships represent some of these alterations in the student diagrams.

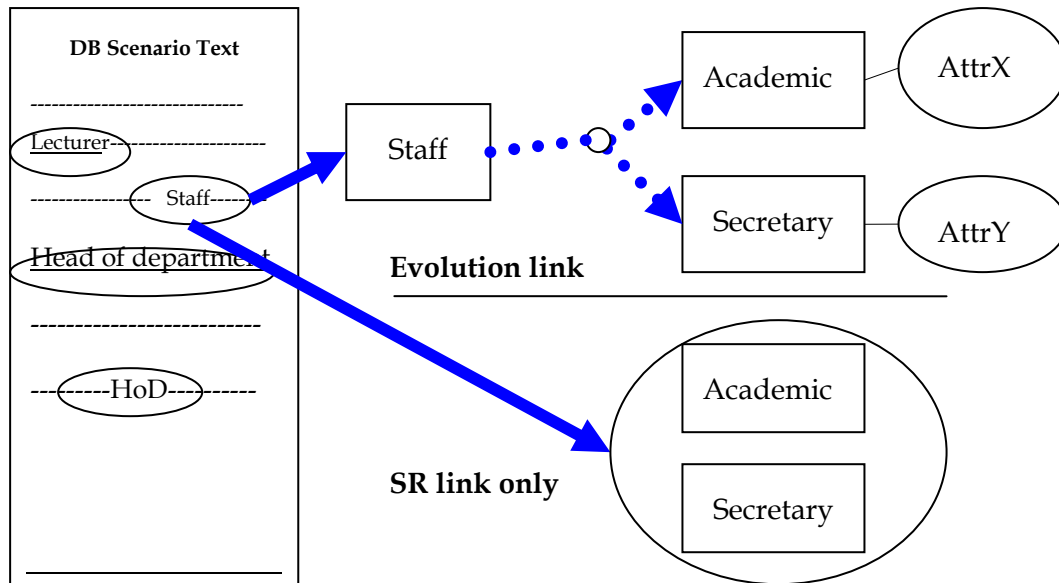
The human marker reads SR and evolutionary links during marking. Links should be represented in an easily readable form. SR links points at components directly and are simple and easier to read than evolutionary links. Evolutionary links represent the design process. They can sometimes be complex and harder to read. Therefore SR links are preferable to evolutionary links where possible. Figure 4.5 shows the evolution and SR links together for a component. In the first part of the figure, the noun phrase "staff" in the scenario text was firstly designated to be an attribute of an entity in a diagram and then changed into an entity. This intermediate stage could be saved to represent exactly the design process by using an evolutionary link. However, that might increase the cognitive load of the human marker. Instead of using an evolutionary link, the intermediate stage can be ignored. Figure 4.5 represents the contextual information of the "staff" entity by using a SR link. The SR link is used to uniquely identify the component. Using the SR link only

will decrease the cognitive load of the marking although it doesn't give the full history of the student's design.



**Figure 4.5 Evolution and SR links for a component**

The coarse granularity level can enable the use of SR links instead of evolutionary links, but coarse granularity decreases the performance of the systems as discussed in Section 4.2.1. For some cases, a SR link alone in the fine granularity level can not uniquely identify a component, it needs evolutionary links as well. Figure 4.6 shows the use of the evolution and SR links together for multiple components. The figure illustrates that the noun phrase "staff" in the scenario text was firstly designated to be a single entity in a diagram and then split into two entities. The SR link in the figure identifies the initial "staff" entity and then the evolution link represents the logical link between the "staff", "academic" and "secretary" entities. If only the SR link is used, then the SR link can only identify the sub-diagram which consists of "academic" and "secretary" entities together. The evolutionary links enables identification of these entities individually. The example shows that although using SR links alone are preferable, in some cases, a design history forces evolutionary links to be used.



**Figure 4.6 Evolution and SR links for multiple components**

The cardinality of the evolution relationships defines the number of participant entities on either end of the relationship line. The cardinality could be “many to many” or “one to many”. The evolutionary links don’t use “one to one” cardinality since SR links can be used instead. Figure 4.5 depicts this case. The “Many to many” cardinality for evolution links can be represented with two “one to many” relationships. Since the trace production of “many to many” relationships is a complex task, only “one to many” cardinality is used for the evolutionary link. The trace production section will discuss the “many to many” cardinality in detail.

Each evolution relationship is needed as a result of a certain action during the design. “Many to one” relationships are created after merging action and “One to many” evolution links are created after a splitting action. Figure 4.6 depicts this case. The merging actions define the event of merging the same or different types of components. Splitting action defines the event of splitting a component into two or more components. Splitting actions can result in either the same or different types of components. In both cases, each component should be uniquely identifiable for the semi-automatic approach.

Components created after a split action can be distinguished by using their labels, and types. For example, after a split action in Figure 4.6, an entity is replaced by two entities. They can be distinguished by using their labels and attributes. The semi-automatic approach can use the labels during the grouping stage. However, using

labels for distinguishing new entities may affect the system performance negatively. There might be too many different names for the entities. To increase the performance, the split action for an entity can be redefined as an extract action. Figure 4.7 shows the extraction action. The extraction creates one additional entity and keeps the existing entity. The new entity will have some attributes of the existing entity.

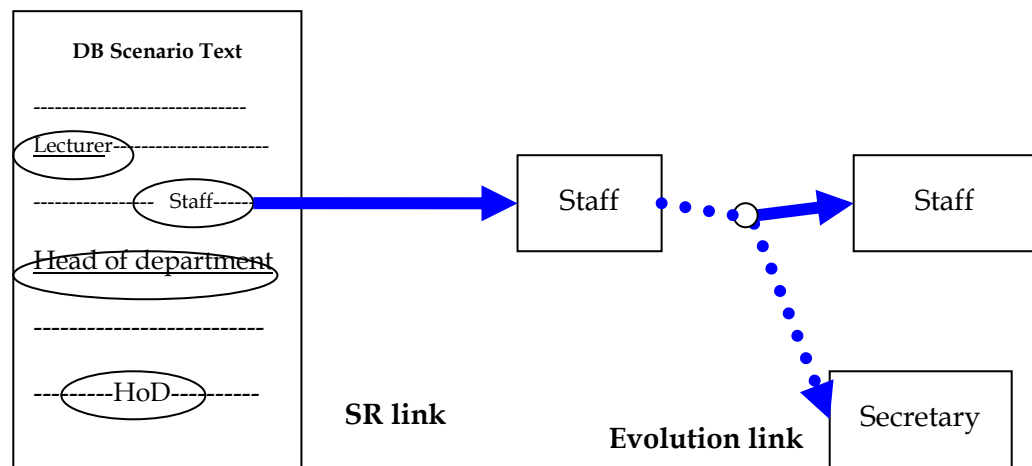


Figure 4.7 Extract action for a component.

This section defined the evolutionary link for the design trace model and focused on the diagramming actions and cardinality of the evolutionary links. The next section focuses on the general properties of the trace relationships.

#### 4.2.2.5 Direction

The term traceability is commonly used for tracing requirements. Gotel and Finkelstein (1994) stated that requirements traceability has two main parts: Pre-Requirement Specification (RS) and post-RS traceability. Post-RS traceability is about tracing how each requirement is implemented. Whereas pre-RS traceability is about tracing back from requirements to their underlying user needs. The semi-automatic approach is only interested in the post-RS traceability. For the assessment purpose, the scenario doesn't need to be traced back to the user requirements.

Post - RS traceability can be forward or backward. Forward traceability is to trace documentation entities to realisation documentation entities on the next abstraction levels. Backward traceability is to trace documentation entities to source documentation entities on previous abstraction levels. Gotel and Finkelstein state that traceability should be in both directions. For the assessment system, only

backward traceability is essential. Students' final designs are traced back to the requirement for contextual information of each component. However, forward traceability can be implemented if the assessment tool is used as a learning tool. Students can follow the examiner's rationale from the scenario text to the final diagram.

#### **4.2.2.6 Attributes of relationships**

Traceability approaches improve relationships by adding different attributes, for different purposes such as the weighting attribute for the impact of a required change. Additional relationships' attributes could be defined to describe the decision history as well. Fischer et al (1995) define the decision history or the design rationale, as statements of reasoning underlying the design process that explain, derive, and justify a design decision. In the requirement traceability context, Conklin (1989) claimed that the design rationale increases the system's maintainability. In the assessment context, it would enhance the teachers' understandings about students' work. An approach to capture and to use design rationales for assessment purposes is discussed in the self-explanation section.

#### **4.2.2.7 Setting relationships**

The relationships between documentation entities are set implicitly or explicitly. Implicit relationships do not require manual setting, such as name tracing. In name tracing, the same names and abbreviations denote the same concepts in two documents. That allows tracing a documentation entity in one document to its correspondence in another document without the need for manual setting. Implicit relationships are preferable in traceability since they can be automated.

Explicit relationships need to be manually implemented and increase workload. Traceability approaches use explicit relationships where implicit relationships are not possible. In the assessment context, both purely implicit and explicit relationships are possible. However using only implicit relationships may cause too many restrictions on assessment environment. For example, a student can be forced to use limited component names during the design process. This will ensure the usage of the same names in different student diagrams. Although that restriction allows using implicit relationships, it might not be suitable educationally. Therefore the proposed design trace model uses explicit relationships where their use is educationally justified.

#### 4.2.2.8 Representation of relationships

Both implicit and explicit relationships between documentation entities need to be represented to support different purposes of traceability. During the marking process, the semi-automatic approach also requires the representation of the relationships. Wieringa (1995) summarizes three ways used to represent links: matrices, graphical models, and cross references for requirements engineering. The design trace model uses the graphical model so that design history can be represented flexibly. Details of the representation are discussed in Chapter 5.

### 4.3 Trace production

Trace production is a process model that documents when the identified information should be captured and by whom (von Knethen and Paech, 2002). Pinheiro (1996) describes two kinds of trace productions: Off-line and on-line. The off-line production performs the capturing process after the act of activity. For example, students submit their work and then the work is analysed automatically or manually by the human marker to produce relationships. Whereas in online production, the relationships are captured as a result of performing the design activities, this section discusses usage of offline and online production in student diagram assessment.

The traces to be produced are vertical or horizontal relationships. Horizontal relationships are essential part of the assessment system. The marking process can only be performed by using them. They can be SR relationships or combination of SR and E-relationships. The production of SR and E-relationships is investigated in the next sub-section. As for vertical relationships, they are optional. The system can work without them. They improve the performance of the assessment system if their production is done automatically. Automatic production of vertical relationships is also considered for each production type in the next section.

#### 4.3.1 Off-line production

The thesis assumes that in off-line production for the semi-automatic approach, students use an available diagram editor to draw their diagrams. An external program records the design activities. The record consists of all components created and deleted during the design process. The following two sections investigate the



use of these design records for the automatic trace production of both SR and E relationships.

### 4.3.2 SR relationships production

SR relationships are links between phrases in scenario text and a component in a diagram if they are used at the most fine-grained level. This section discusses briefly the possibility of using the information retrieval techniques to produce these relationships automatically for the design trace model.

The goal of the information retrieval systems is to retrieve all documents which are relevant to user keywords whilst retrieving as few non-relevant documents as possible. The systems extract semantic and syntactic information from the document text and use this information to match user information need (Yates and Neto, 1999). Incorporating the domain knowledge improves the success rate. The success rate is affected by keywords which convey the semantics of information need and index terms which is a logical view of documents.

Component names in student diagrams may be used as keywords and the information retrieval techniques can search the keywords in the scenario text for the trace productions. In order to perform the production correctly, the techniques incorporate domain knowledge. Domain knowledge consists of linguistic, scenario and subject domain and general world knowledge (Bohner, 1991). All this knowledge should be complete for correctness of the trace production. Preparing such domain knowledge could be expensive for assessment purposes. It increases the set-up cost of the questions.

Apart from the domain knowledge, the component names are also very important for the production. Misnaming components decreases the success rate and could make the system unreliable. Smith et al (2004) found student diagram solutions are imprecise where required features are either malformed or missing. If the techniques are used for the production then the students should approve the traces or the examiner needs to filter and validate the traces. This would increase marking workload much more than the traditional method.

Information retrieval systems use information extraction techniques and automatically build index terms for the documents. Cerbah and Euzenat (2001) employ the technical terms (terminologies) for a specific domain for the trace

production. They automatically extract terminologies and represents as term hierarchies. These hierarchies (the taxonomy of classes) are at an intermediate level between the text in documents and the formal models. Figure 4.8 illustrates the terminological items for the trace production. In the technique, the users exclude some of links/terms or add more links/terms during the production if required.

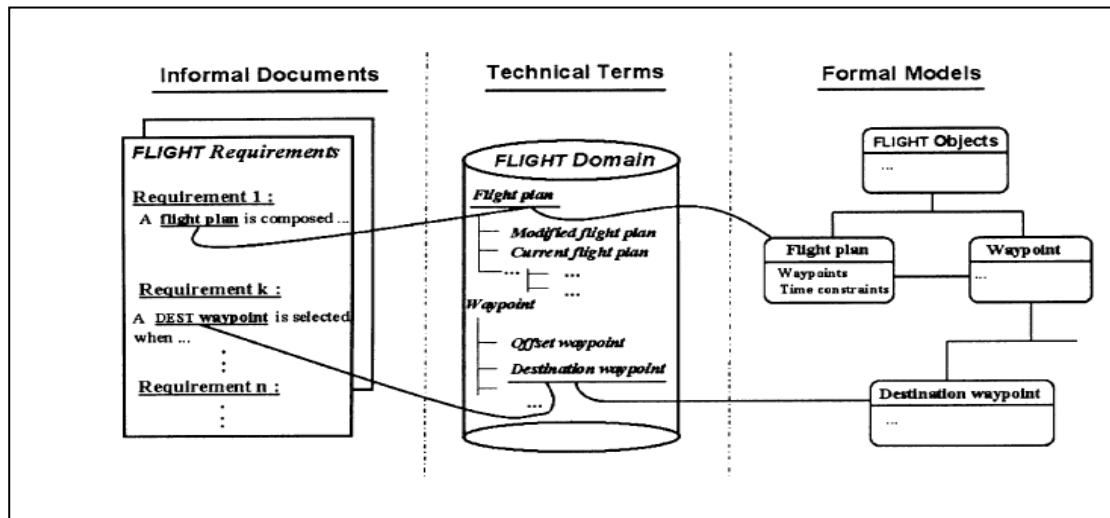


Figure 4.8 Terminological items (Cerbah and Euzenat, 2001)

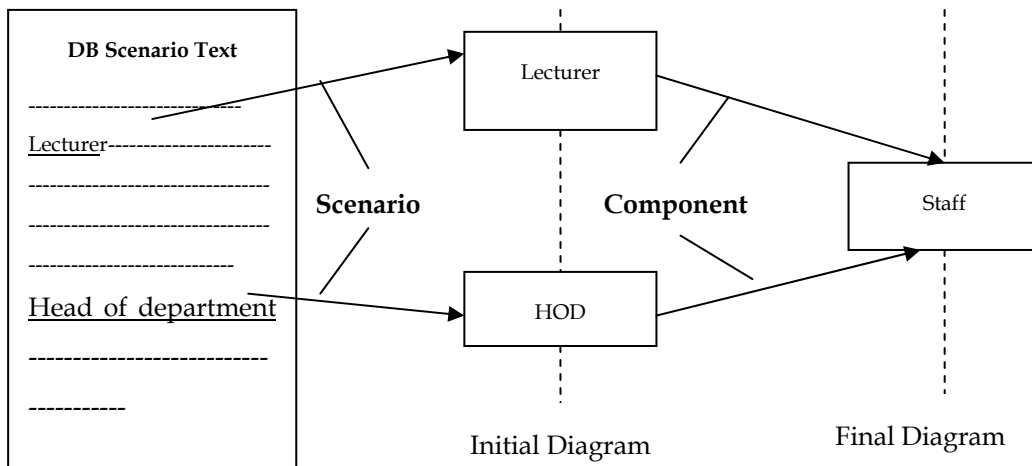
The technique may be applied to the diagram assessment. The phrases in the scenario text can be extracted automatically and a reference phrase list can be built automatically. Later, the examiner modifies the list by adding and deleting terms during the question preparation. To automate the trace production, the students are only allowed to use the phrases in the list to name their components. Wrong phrases should be added into the list in order to prevent students from predicting the answers.

This section discussed that the information extraction techniques can be employed for the SR relationships production if the some restrictions are applied to the diagramming. The next section will discuss the production of the evolution relationships.

### 4.3.3 E-relationships production

Evolution relationships are links between components in the design history. They have got “one to many” cardinality. This section discusses briefly the possibility of using the phrase list to produce the relationships automatically for the design trace model.

E-relationships are needed after some design activities (e.g. a split or merge action). Students may create new components to replace the existing components. Figure 4.9 shows an example action of merging two entities. This action may consist of three events. Students may do two “delete a component” events (e.g. lecturer and HOD) and one “add an entity” (e.g. Staff). These three events are interrelated to each other. They need to be interpreted to be able recognise the compound action. Recognised actions can be used to produce traces for the model.



**Figure 4.9 Scenario reference and component correspondence link**

Students may draw their diagrams in an irregular order. Unrelated events in the design history may interleave one another. For example, “delete a component” events may be about a correction of a drawing mistake rather than a part of a merge action. This kind of event causes the misrecognition of actions. As well as the event type, component names need to be used to correctly relate events of an action.

Students need to use the phrase list to name their components for E-relationships production like SR relationships. However the list consists of supplementary phrases as well as all phrases in the scenario text. The list should be a taxonomy of phrases for the scenario text as in Figure 4.8. For the scenario text in Figure 4.9, the list will include the “head of department” and “lecturer” phrases. The list should also include the “staff” phrase even though it is not in the scenario text. The staff term should be represented as a parent of other two terms in the taxonomy in order to be used for trace production.

The list may also include wrong phrases to prevent students from predicting right answers. Wrong answers are called distracters in multiple choice questions. As in

multiple choice question types, the examiner has to foresee students' reasoning in advance and add the possible wrong terms into the list. Building such a phrase taxonomy may be an expensive task and may make the setting up of a question complicated and the subsequent assessment process infeasible.

Students may use the terms in the list wrongly. This causes an invalid production of traces. For example, the "book" noun phrase in the list may have the meaning of "book title". Students may use the "book" noun phrase to name the "book copy" component. During the trace production, the student's component will have the meaning of "book title". For this reason, students need to see the interpretation of their diagram and approve it before they submit their solution. As students see the interpretation, they may rename the component to correct it. If not, students accept the grade and feedback for their work.

The off-line trace production requires both the naming restriction and student approval of produced traces in order to be able to give a valid grade for the student work. Conventional diagramming editors need additional features to implement these requirements. Since off-line production uses student design activities without altering them, checking whether the offline production can be acceptable by student is not required.

#### **4.3.4 On-line production**

In on-line production, the relationships are produced as a result of performing design activities. To automate online productions, the conventional design activities are changed and new activities are added. These new design activities require educational justification. Section 4.3.7 gives this justification by explaining the self-explanation concept. The following two sections discuss the online production of SR and E relationships.

#### **4.3.5 SR relationships production**

This section considers first the SR relationships in the finest granularity level, which are links from a phrase to a component, then later considers the SR relationships in the coarse granularity level for some component types. It emphasizes the problems of KERMIT's approach (Suraweera and Mitrovic, 2002) and proposes a new approach.

KERMIT forces students to first highlight the phrase and then create a component to produce the SR Relationships. Students name the components as they wish. The examiner grades the student diagrams and gives feedback based on the SR relationships rather than the components' names.

This approach allows students to reference components wrongly. For example, the "Student" entity can be referenced to the "department" noun phrase in the scenario. As a result of this, the "Student" entity is interpreted as department entity during marking. The thesis calls this problem a "naming discrepancy". This potential problem may be solved if components are named by using the highlighted phrases as a label of the component.

The highlighted phrases may cause a "naming discrepancy" as well. For example, students might highlight the "worker" part of the "permanent worker" phrase to name a component. The "worker" entity is interpreted as a "permanent worker" entity even if the entity has got a different meaning in the diagram context. Students should be aware of this marking process. They should be extra cautious during referencing components since incorrect references make the correct component wrong.

Referencing components can be more reliable if component names are determined earlier and a phrase list is created. Students use the lists instead of the free-highlighting. For example, a student can pick the "Permanent Worker" phrase from the list to name a component. Then the phrase will be highlighted in the scenario. The picking technique prevents the phrase from partial selection. This technique requires building the phrase list. The phrase list can be automatically constructed by using information extraction techniques as discussed in the previous section. The examiner reviews the list to correct the wrong items. Then the list is made visible to students during the design process.

The noun phrases in the scenario text are usually used to name attribute or entity types for database diagrams. The verb phrases in the scenario text may name a relationship type. The same verb phrases could occur in different sentences and refers to different relationship types in a diagram. For example, the "has" verb phrase could be a reference for both a relationship between "Student" and "address" and between the "hospital" and "ward" entity types. The "has" verb phrase needs to occur again in the verb phrase list each time it refers to different relationships. This

ambiguity between the “has” phrases can be resolved if scenario sentences, in which the phrases are used, are listed as references instead of phrases only. Both noun phrases and sentences in the scenario text are used for diagram components. This makes the granularity level coarse and fine at the same time for the requirements document.

Some sentences in the sentence list might refer to a same relationship type in student diagrams. Representation links show these related sentences. Production of these relationships during the question set-up time improves the marking performance. Creating representation links automatically is a more complex task in sentence level granularity than in the phrase level. The manual trace production can be preferred due to the small number of sentences. Representation links in sentence level granularity can even be ignored to decrease the set-up cost.

Some complex sentences may cause referencing ambiguity. They may refer to two or more relationship components semantically at the same time. This makes the cardinality of SR relationships “one to many” and prevents the components from being uniquely identified. For example, the sentence, A, below can be a reference to a relationship between the “student” and “optional module” entity types. The same sentence can also be a reference for another relationship between “programme” and “optional module” entity types. This ambiguity may increase the cognitive load of marking since the examiner needs to understand what the student means from this reference.

- A) *Students choose optional modules of their programme.*
- B) *Each programme has many optional modules.*

A solution to referencing ambiguity is to impose a restriction on sentence referencing. Students can refer to a sentence for only one component and the component name should be the verb of the sentence. In this case, the scenario writer should make sure that sentence B or a semantically similar sentence is written for the second meaning of the sentence A.

This section discussed the student’s involvement of the SR relationships production. It suggested using noun phrases and sentences together. The next section extends the student’s activities for E-relationship production.

### 4.3.6 Online E-relationship production

This section discusses the students' involvement of E-relationship production. It suggests using special functions for different design activities and focuses on diversity in student reasoning.

Evolution relationships are needed after certain design activities. Special functions are defined for each action in order to recognise them without interpreting low level events like delete and add component. Students use the functions to perform an action and modify existing diagrams. For example, for the merge action, the function gets Entity A and Entity B in a diagram and replaces them with a new Entity C with the attributes of both entities. Figure 4.10 shows this action in diagrammatic form. For online trace production, these functions need to be implemented into a diagram editor and conventional diagramming process needs to be changed.

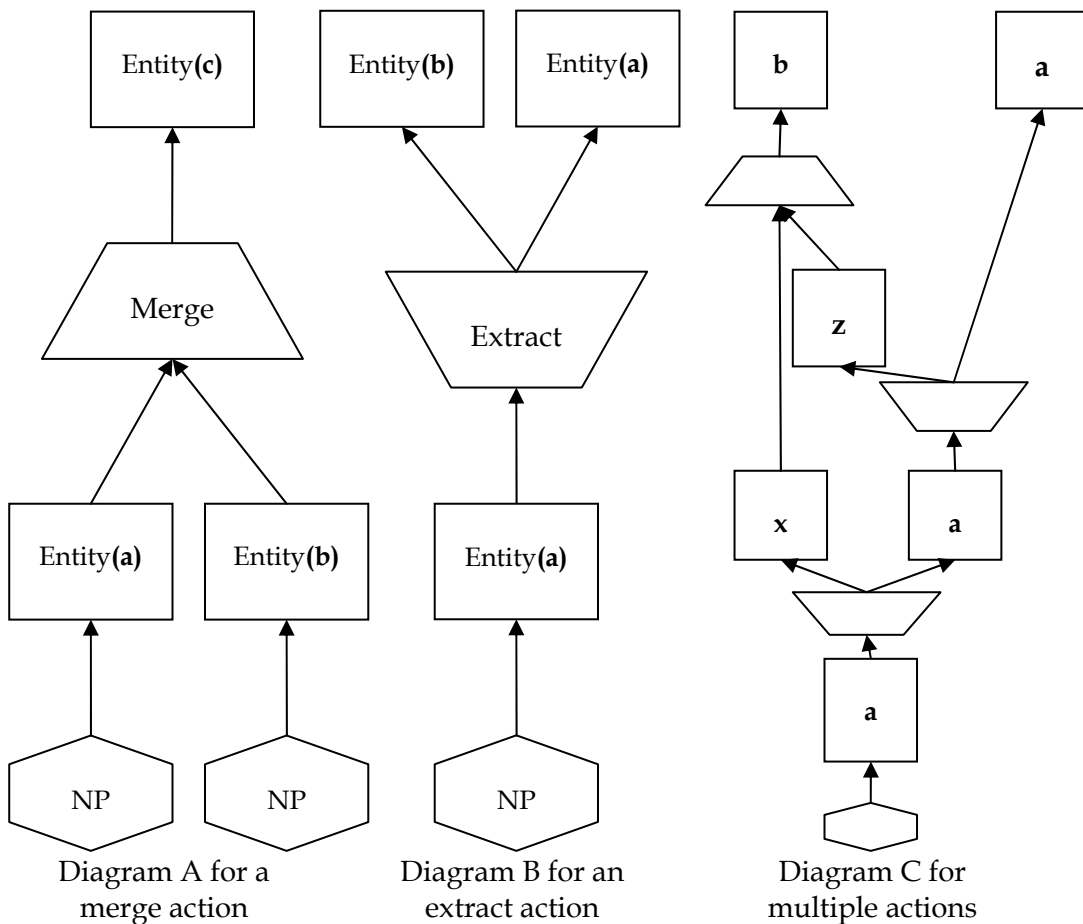


Figure 4.10 Components reference diagrams

Students use the provided functions in a special editor to draw their diagrams. With these functions, the same diagrams can be drawn in the many different ways.

Diagram B in Figure 4.10 shows an application of the extract function. This function extracts Entity B from Entity A. Diagram C in the same figure shows different combination of merge and extract functions which produce the same Entity A and Entity B. These functions allow students to create different design traces for the same component. The different traces for a component give the different contextual information. The examiner marks each context separately, therefore a high number of these traces decreases the system performances.

A number of traces for a component shows the diversity of drawing the component. This diversity should be at an acceptable level. It depends on the order in which students consider the requirements. The order may be loosely controlled by the scenario scaffolding technique. The scaffolding (Bunt et al, 2004) reveals the scenario text, section by section. The assumption is that the students may consider the sections in the same order they are revealed. However, students might still draw the diagram in a different order. Effectiveness of scaffolding depends on the contents of each section. How the sections should be written is discussed in Chapter 5.

The usage of the functions may cause the “naming discrepancy” problem. The functions create one or more new components and students name these components. The names of these components are not important in terms of the diagram marking since only traces are used for grading. However the components’ names should be compatible with their traces. For example, “temporary worker” and “permanent worker” are merged and students are allowed to name the new entity “department” instead of “worker”. Students should see the reference of this component so that misnaming can be spotted before they submit their solutions. Alternatively a naming convention can be prepared and checked by the system during diagramming. Chapter 7 discusses this problem in detail.

This section discussed that the online production of E relationships requires student supports. They use the functions for their design activities. The diversity in students’ activities is controlled by scaffolding techniques. Students are responsible for solving the “naming discrepancy” of a component. Online trace production alter conventional diagramming process. The next section justifies this alteration.



### 4.3.7 Self-explanations

In industry, there are people with many years experience of trace production. They create traces explicitly between requirement documents and their design or products (Ramesh, 1998). In education, the people who produce traces are students and novices. They are expected to create traces explicitly between the scenario text and their solutions for online trace production. In other words, students are forced to explain their actions. This is called self-explanation in the literature. Psychological studies (Bunt et al 2004) show that self-explanation is a very effective learning strategy resulting in deep knowledge.

Many self-explanation systems (Conati, VanLehn, 2000) have been developed to support students' self-explanations. Self-explanation systems may support students while they study solved examples or are asking for an explanation while solving a problem. The main problem of self-explanation whilst solving the problem is the high cognitive load (Chi et al, 1989). For online trace production, a new diagram editor needs to be designed. The editor should support students' self-explanation and reduce the cognitive load of the self-explanation. Chapter 6 looks into the components of the required diagram editor.

## 4.4 Design Trace Model

Both offline and online production can be adapted to the new design trace model. Online trace production expects students' self-explanation, whereas offline production applies some restriction on component naming during diagramming. This section first discusses both the production techniques and then adopts the online production technique. It gives the details of the new model.

The off-line trace production requires the examiner to do many additional tasks during the preparation stage and/or marking stages. They need to build the detailed phrase list in the preparation stage and to validate the produced traces in the marking stage. The benefit of the offline production is that there aren't any changes in diagramming activities. Additionally it is very close the full-automatic assessment. On the other hand, the on-line production requires fewer additional tasks. It shifts the load of the assessment to the students. Students need to do the

additional diagramming activities. Although this load on the students has been justified educationally, students need to learn the new activities for diagramming.

The on-line production requires less initial preparation and it can use incomplete knowledge. The production technique can be used in formative assessment. Later the student solutions may be analysed to build the complete knowledge required for the offline production. Therefore this thesis adopts the online production to the design trace model. Chapters 5 and 6 will discuss the implementation details of online trace production and the offline production is out of the thesis's scope.

Online trace production uses scenario text and student diagram documents. Figure 4.11 illustrates the trace model developed, based on the online production. Scenario text in the figure consists of text fragments, which can be noun phrases or sentences. Each text fragment may link to another. These links are representation relationships as discussed previously. The student diagram in the figure consists of components which can be different types. Component types are entity, relationship and attribute types for the database diagram. Component types can be different depending on the diagram type. There is a "one-to-one" scenario reference relationship between one of the related text fragments and a component. Related components in a design history are connected by a "many-to-many" evolutionary relationship. This relationship has an attribute which keeps the information about the function used for the design action.

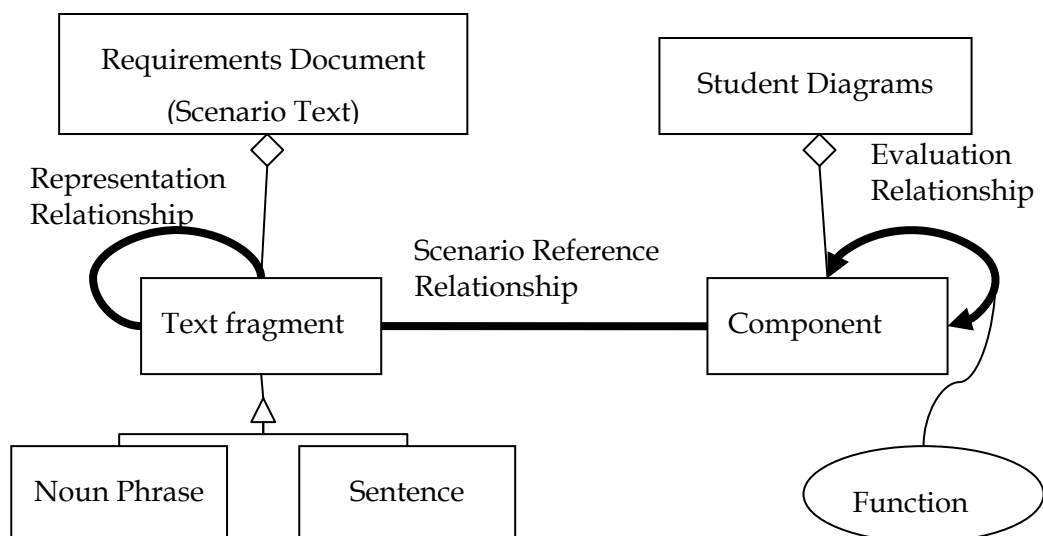


Figure 4.11 Design trace model

The design trace model in Figure 4.11 shows the entities and relationships to be traced for the diagram assessment. The model is designed for online production but it doesn't show the production process model. This is discussed in Chapter 6.

## 4.5 Summary

This chapter presented the design trace model required for the semi-automatic diagram assessment. It adapted vertical and horizontal relationships from the reference model. Three relationships types are introduced for the model. Two trace production approaches are discussed. The online production techniques are adopted for the model. For online production "function" concept is developed which eases the production of evolutionary relationships.

The design trace model is developed for ER diagrams specifically. However the model can be used for many similar graph diagrams as well. The "Function" attribute of the model is generic. For the new diagram type, new action functions can be defined. The model currently has "noun phrase" and "sentence" entities. For the diagram, a new entity can be added if it is required.

This chapter explained the development of the design trace model. Similar development steps can be followed to develop a new trace model for other graph diagrams.

The next chapter develops a new marking process model, which uses the design traces during marking. The chapter focuses on the automation of the marking. It defines a new generic case concept and explains the use of the generic cases for the automation. The chapter also develops a set of guidelines for writing question text which contributes to the automation.

# CHAPTER 5

## Marking Process Model

### 5.1 Introduction

Semi-automatic diagram assessment aims to remove the repetitive tasks of the assessment as much as possible. Chapter 3 developed the partial marking process for semi-automation and discussed automation of the partial marking process to increase the performance of the approach.

The partial marking process groups the student diagrams based on their design history. To save the design history, the design trace model is developed in chapter 4. The assessor marks one diagram segment from each group during partial marking. To increase the performance of the partial marking process, the case-based reasoning (CBR) method is adapted in chapter 3.

This chapter presents the adaptation details of the CBR method for the semi-automatic diagram marking process, focusing on partial marking. The first section develops the case definition for automatic partial marking and establishes a relationship between the case definition and the writing style of a requirements document. It also shows that scenarios with the same writing style increase the automation of partial marking. The second section develops a guideline for writing similar scenarios. The last section introduces the full diagram marking process. It combines full and partial marking in order to have a new marking process model.

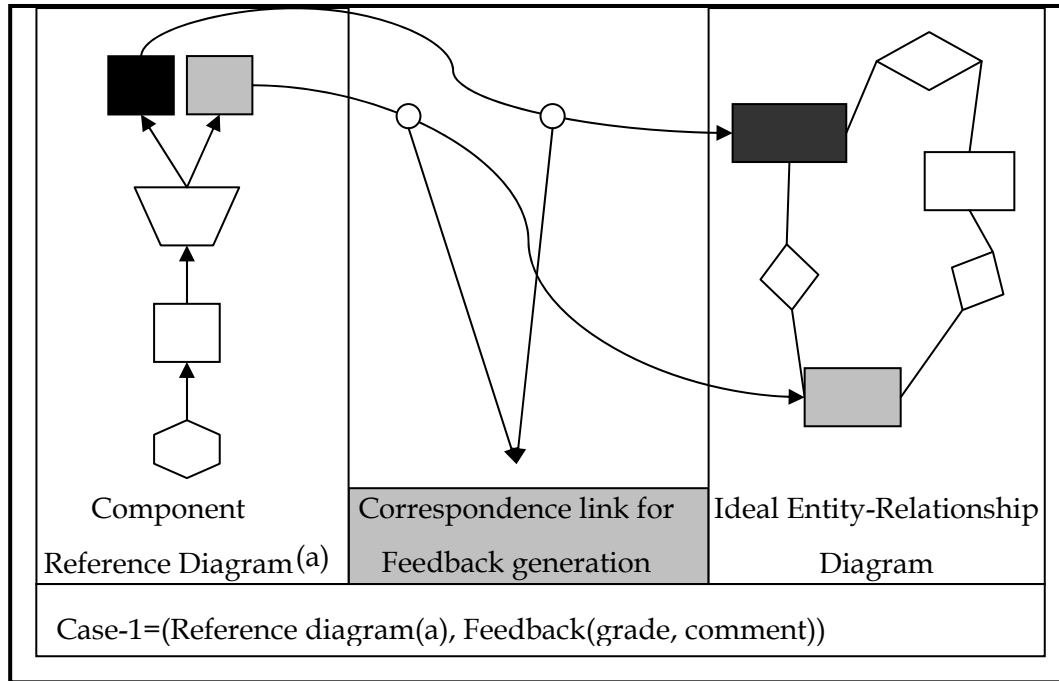
### 5.2 Automatic partial marking

This section first gives the basic definition of **cases** used for the partial marking. Then it exemplifies the usage of the cases to automatically mark some of the diagram components. The examples only demonstrate the possibility of automation without providing the full coverage of the area, which is not within the scope of this research. A separate and extensive study will be planned in the future. The section later enhances the first case definition in order to improve the automation and creates

generic cases. Lastly it focuses on the generation of generic cases and highlights the importance of the scenario writing.

### **5.2.1 Basic case definition, correspondence links and reference diagrams**

The proposed marking process starts when student diagrammatic solutions are received and ends when feedback for the all solutions is produced. The feedback consists of a grade and the examiner's comment for each component in the diagrams. A component and its feedback together make up a *case* and are stored to be reused later for subsequent cases whenever it is possible. Components are identified and indexed by their design traces. Each trace for a component is represented as a diagram called the reference diagram. The reference diagrams are read by the human marker. The marker matches a component with the ideal diagram component based on the reference diagram. The correspondence link is created between two components. These links are very important. They are used later for standard feedback generation. Figure 5.1 illustrates two correspondence links. The links show that two entities from a student solution are matched with two entities in the ideal solution. Both links in the figure have a one-to-one cardinality. The cardinality of correspondence links could be one-to-many in other cases. Figure 5.1 also shows an abstract type case, Case-1, which consists of a reference diagram(a) and its feedback.



**Figure 5.1 Assessment case definition.**

Correspondence links are also used for schema integration which is the activity of integrating the schemas of existing or proposed databases into a global, unified schema (Batini and Lenzerini, 1989). Schema integration is a basic problem in many database application domains, such as data integration, E-business, data warehousing, and semantic query processing. Schema matching is typically performed manually. There are many different types of correspondence links defined for this purpose. This research uses a basic type of correspondence link for database diagram. The basic type could be extended for different graph based diagrams if it is needed.

Reference diagrams represent student design activities for a component. They are an instance of the design trace model. A reference diagram can have three component types which are called Ref-components: (1) scenario references (SRs) which can be noun phrases or sentences, (2) design actions and (3) intermediate components. Figure 5.2 illustrates these three types of Ref-components. In the figure, the scenario references are represented by hexagons which are noun phrases. The intermediate components are represented by rectangles which are entity types in student diagrams. The design actions are represented by trapeziums which are merge actions. The figure shows the reference diagrams of Component A and Component B.

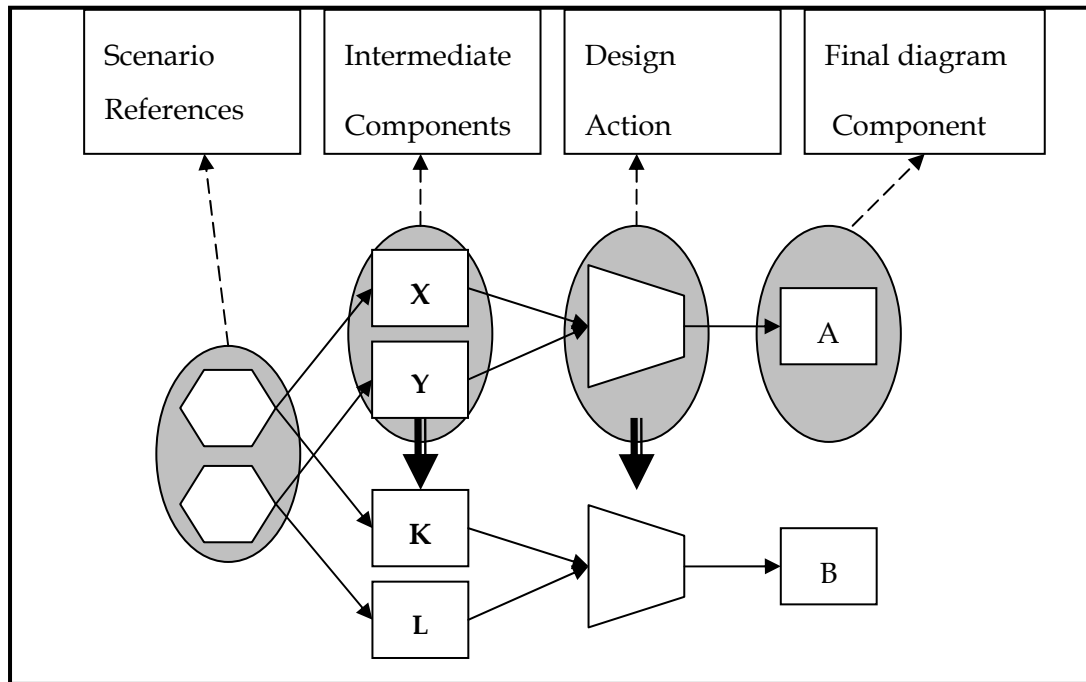


Figure 5.2 Finding identical reference diagrams

The marking system groups reference diagrams based on their ref-components. During grouping, firstly, reference diagrams which have got the same scenario references are put together into temporary groups. Next, within these groups, intermediate components and diagram actions are matched. If all ref-components are completely matched in two reference diagrams, they become a member of the same group. Figure 5.2 shows an example of grouping two reference diagrams. In the figure, both reference diagrams have got common scenario references. Later Entity X is matched with entity K and entity Y is matched with entity L. Lastly, the merge design action of both reference diagrams are matched. Since all ref-components are matched, Entity A and Entity B are placed in the same group. This is called the reference diagram group. The human marker marks only one reference diagram in each group.

### 5.2.2 Examples of automatic marking

Some of the reference diagram groups can be marked automatically. This increases the efficiency of the marking process. Basic automation can be done by using the RDs of correct components which is called the ideal reference diagram. All ideal reference diagrams for an ideal solution can be entered into the system. If a reference

diagram group contains an ideal reference diagram then the reference diagram group is marked correct and correspondence links for reference diagram are generated automatically. For example, Entity A in Figure 5.2 is a component in an ideal solution. The reference diagram of Entity A is an ideal reference diagram. Since the reference diagram of Entity B is matched with the reference diagram of Entity A, Entity B is automatically marked correct. If there is more than one ideal reference diagram for a component, entering these reference diagrams into the system enables the marking of more diagrams automatically.

Some reference diagrams are partially matched rather than complete matching, unlike the reference diagrams in Figure 5.2 during the grouping process. The partial matching can be used to help further diagrams to be marked automatically. Two reference diagrams could have the same scenario references but the rest of the diagrams' parts wouldn't match completely each others. For example, Figure 5.3-B illustrates an entity and an attribute component, which have the same scenario reference. If the entity component is correct, the attribute will be wrong. Therefore standard feedback for this component is given to students without human intervention. In the entity relationship diagram domain, the generic case in Figure 5.3-B is stored to be used for the automatic marking.

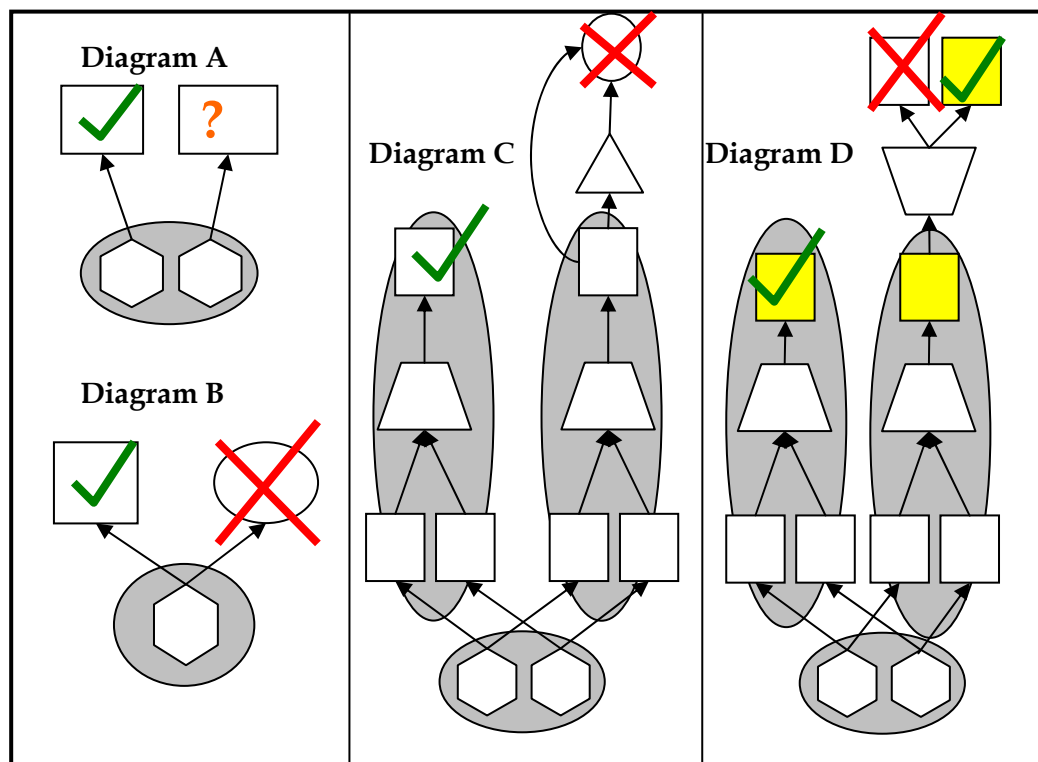


Figure 5.3 Automatic component marking with an ideal reference diagram



Another partial matching example is a subset of a correct reference diagram. In this case, the final components are automatically marked based on the rest of the diagram by using rules. Figure 5.3C shows this type of marking. The shaded area in the figure shows the matched part of the diagrams. The right diagram shows that a student converted a correct entity type to an attribute type which is represented by triangle shape in the diagram. Figure 5.3D shows a slightly different version of Figure 5.3C. The right diagram shows that the student created a new entity from the correct one by using an extract action. As a result of this action, the new entity becomes wrong and the other entity remains correct. This specific case can be generated and entered into the system so that more diagrams are marked without human intervention.

A reference diagram could have scenario references which is the subset of scenario references of an ideal reference diagram. Standard feedback for these diagrams can be generated automatically. Figure 5.4A shows two wrong entity components with respect to an entity component in the ideal diagram. When the system grades these components, half of the marks for the correct component can be given to them. This kind of marking scheme distinguishes a completely wrong component from the half correct one.

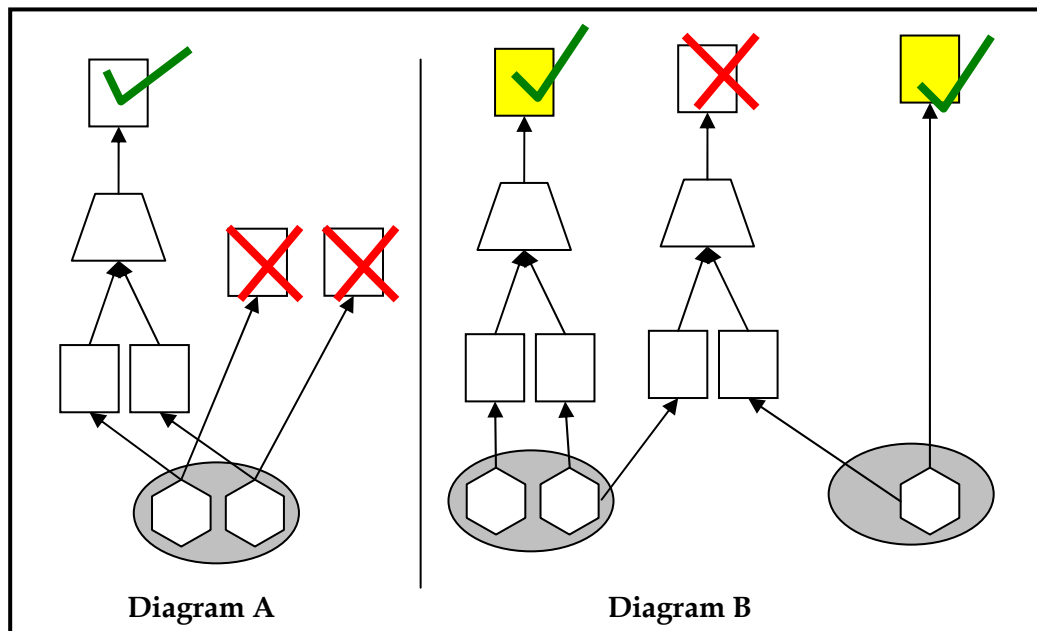


Figure 5.4 Partial scenario reference matching

The combination of two correct reference diagrams can also be used to mark additional components automatically. Figure 5.4B shows two correct reference

diagrams with the tick signs. The reference diagram in the middle of Figure 5.4B shares its scenario references with both of the correct ones. In this case, this entity component will be marked wrong. This type of complex generic case can be built and used for the marking until it is proved that is invalid for some situation.

The examples given in this section is for the entity relationship diagram domain. The same generic rules may not be applicable to other graph diagrams. Student solutions and examiner comments can be analyzed for each diagram type separately to create special generic cases for them.

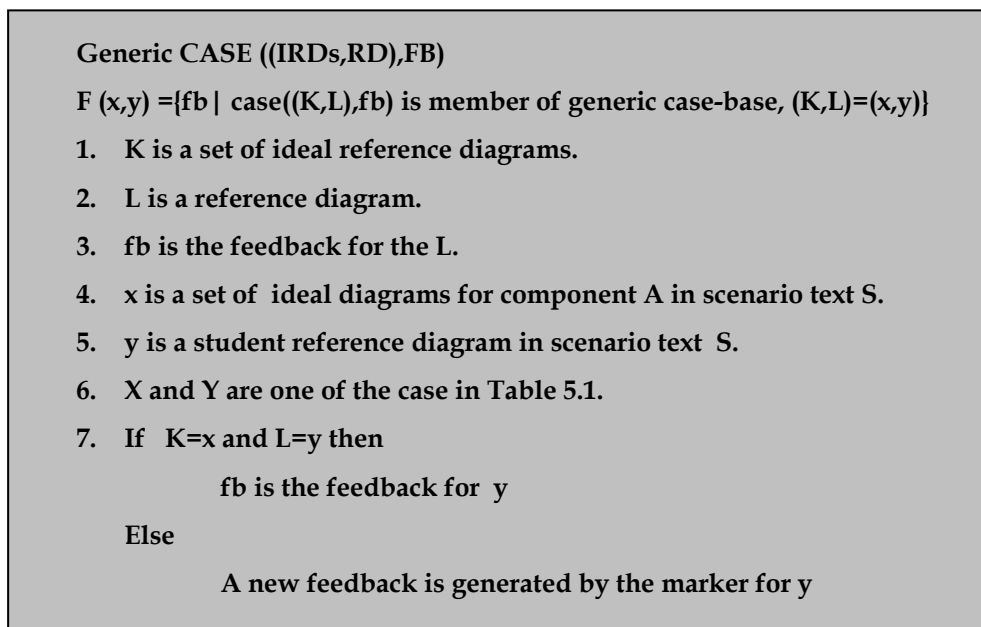
### 5.2.3 Case categories and generic case definition

Scenario references of the reference diagrams are the main components used for matching reference diagrams discussed in the previous section. Generic diagram cases can be categorized based on the scenario reference matching. Table 5.1 lists two main categories of generic cases. The scenario references (SRs) of a reference diagram (RD) and an ideal reference diagram (IRD) can be completely or partially matched or they can be completely separate from each other, in which case the diagrams aren't generalised for automation (see Figure 5.3A). If  $SR_{IRD}$  and  $SR_{RD}$  are the same, the rest of the diagram parts could be the sub-diagram of another one or partially matched. Only the sub-diagram situation is considered for automatic marking. The latter situation is not suitable for generalisation. If  $SR_{IRD}$  and  $SR_{RD}$  are partially matched then  $SR_{RD}$  can be the subset of one  $SR_{IRD}$  or many  $SR_{IRD}$ . This type of partial matching can be considered for automatic marking. If  $SR_{IRD}$  and  $SR_{RD}$  are partially matched and the unmatched part of the  $SR_{RD}$  is not the part of other  $SR_{IRD}$ , then these reference diagrams aren't considered for generic case generation.

Table 5.1 Category of generic cases

1. Complete matched ( $SR_{IRD} = SR_{RD}$ ) <ol style="list-style-type: none"> <li>a. <math>IRD \subset RD</math> (e.g. Figure 5.3C)</li> <li>b. <math>RD \subset IRD</math> (e.g. Figure 5.3D)</li> </ol>
2. Partial matched ( $SR_{RD} \subset SR_{IRDs}$ ) <ol style="list-style-type: none"> <li>a. One IRD for a component (e.g. Figure 5.4A)</li> <li>b. Many IRDs for a component (e.g. Figure 5.4B)</li> </ol>

There can be many generic cases in each category listed in Table 5.1. These generic cases are used to produce feedback for components. The generic case is the enhanced version of the basic case definition. Each case consists of a reference diagram, a set of ideal reference diagrams and some feedback. Figure 5.5 shows the usage of the generic cases. It shows a formal function and its explanation. The function  $F$  generates feedback for a reference diagram and takes  $x$  and  $y$  parameters as an input.  $X$  is a reference diagram and  $y$  is a set of the suitable ideal reference diagrams for the reference diagram. The suitable ideal reference diagrams are found based on categories in Table 5.1. If there is no suitable ideal reference diagram then function  $F$  cannot be used. The reference diagram and the ideal reference diagrams together are mapped to one of the generic cases. If there is no generic case applicable for the input, then the human marker marks the reference diagrams.



**Figure 5.5 Usage of generic cases**

The generic cases should be used for any scenario text. They are independent from the domain of the scenario text. For example, scenario text can be database requirements of a rent-a-car system or a library system. For both domains, the same generic case can be used. Scenario text  $S$  in Table 5.1 indicates that  $S$  can be any scenario.

The generic cases are desirable but not essential for semi-automatic marking. The assessment system can work without the generic case-base library. This allows

adding new generic cases into system gradually. The next section describes the process of generating new cases.

### 5.2.4 Generic case generation

Some of the generic cases in each category can be foreseen in advance and can be embedded into the generic case-base. The unanticipated generic cases can be detected and added into the system later on. Figure 5.6 shows the process of generic case generation. The existing marked diagrams of all scenario text are analysed component by component. Abstract IRDs, the related abstract RDs and feedback is detected. Abstract RDs or IRD are the reference diagrams without any labels of components. If the detected cases repeat in the solution set more than a certain number then the same cases are highlighted. If the highlighted cases are approved by a human, a generic case is generated and added into the case library.

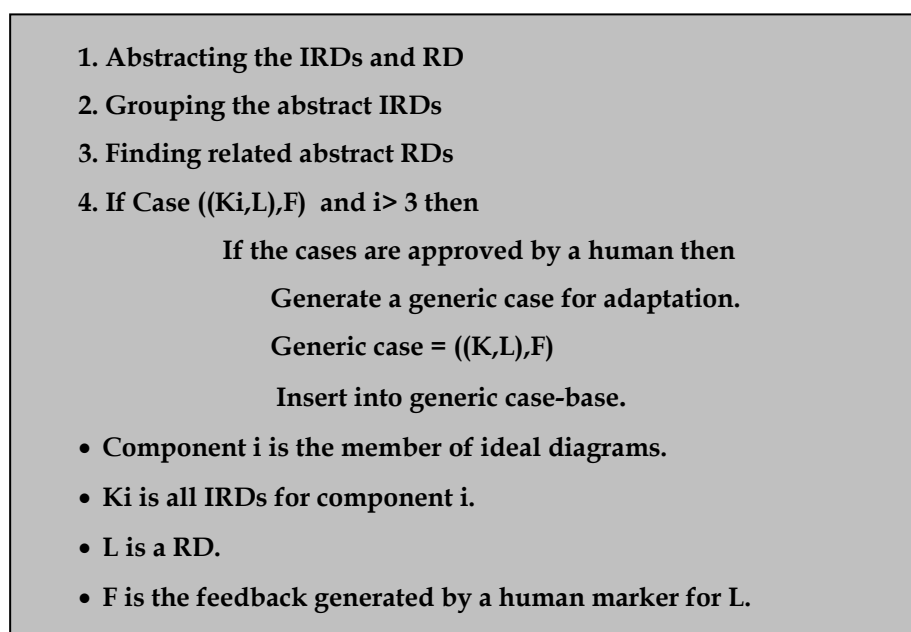


Figure 5.6 Generic case generation

Increasing the number of the same cases (e.g. case  $((\text{IRDs}, \text{RD}), \text{F})$ ) is desirable for generic case generation. The number of the same cases mainly depends on cases that have the same ideal reference diagrams. This is because the number of ideal reference diagrams is generally less than reference diagrams since ideal reference diagrams can only be the correct solutions whereas reference diagrams can be either correct or wrong student solutions.

Initial ideal reference diagrams are captured while the teachers are solving the problem. They depend on how the scenario is written to explain the required diagram components and the written style is independent from the content of the scenarios. If the requirements of two components are written in the same way, these requirement texts are defined as similar in this research.

The numbers of same cases can be increased by given a same question every year to different students or similar questions to same students. Similar questions are also desirable for pedagogical reasons as explained in Chapter 3. The next section describes a guideline for writing similar questions.

### **5.3 Similar Scenario Text**

This section is about the question part of the assessment process. It focuses on the requirements text for the database conceptual model. It explains how to write similar scenario texts.

Similar scenarios help the production of generic cases as discussed in section 5.2. The scenarios also affect the students' reasoning while modelling. They may involve few or many student reasoning steps, which increases the number of reference diagram groups, which in turn will increase the number of reference diagrams to be marked. The system efficiency can be improved if this number is reduced. The scenario can be written in a way that the students reasoning diversity is controlled to a certain degree in order to improve the efficiency.


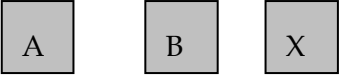
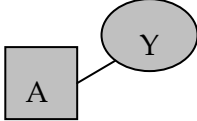

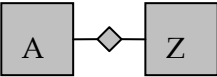
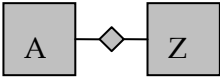
The section first introduces the concept of scenario statements and defines statement types. Then it explains the importance of the scenario sections and how it may affect the students' reasoning process. Lastly it describes how to use statement types and sections to write a scenario text.

#### **5.3.1 Statement Types**

The scenario text consists of statements. Statements can be one or many sentences. They can provide the database system requirements a question in a database modelling exam. They have an effect on the students reasoning. It may change the diagram students are developing. This research classifies the statements into the three types based on their effect: extension, alteration and support types. Statement

types assume that students or readers understand the statements and act on them correctly or in the expected way. Although this is not always the case, it could increase the number of students using the same reasoning. It could even reduce the student reasoning diversity which is good for the semi-automatic assessment.

An example of each statement type is given in Figure 5.7. Example 1 is an extension statement type. The statement mentions system requirements which require a new component in the existing diagram. Students read the extension statement and create Entity X. Example 2 is an alteration statement and requires a change in the diagram. Students replace Attribute Y with Entity Y. Example 3 is a support statement. It makes no change to the diagram but provides additional support to the existing diagram components. For example it mentions an existing relationship between two entities and therefore supports the existence of the relationship and the entities.

<u>Scenario Text</u>	<u>Student Diagram</u>	
	Existing Diagram	After reading the statement
<div style="border: 1px solid black; padding: 5px; background-color: #e0e0e0;">Extension Statement</div> Example 1		
<div style="border: 1px solid black; padding: 5px; background-color: #e0e0e0;">Alteration Statement</div> Example 2		
<div style="border: 1px solid black; padding: 5px; background-color: #e0e0e0;">Support Statement</div> Example 3		

**Figure 5.7 Examples of statement type**

The statement types in Figure 5.7 can work if students consider them in an expected order. For example, a support statement in the example 3 may have an alteration affect on one student diagram and extension effect on another one. This is because students may have read the support statement first before reading any other related statements. The research suggests that different statement types for the same concept are written in separate sections. Then the scenarios are presented section by

section to students. Only if students consider one section at a time during the design can the statement type have the required affect on the student diagrams.

This segmentation of the scenario or scaffolding technique can't force the students to follow scenario sections in the specified order during design. Students may still read the whole scenario text and then create the required component in a diagram. Figure 5.8B shows a reference diagram of two components. In this case the students read the expansion statement of "staff" entity in the first section and then read the alteration statement of the same entity in the second section, which requires "part time staff" and "full time staff" entities as a replacement of the staff entity. The reference diagram shows that a student has created these entities by using the extract function on the "staff" entity. However, some students may create two entities straight away. Figure 5.8A shows this case. The students didn't create a "Staff" entity after reading the first statement. They read the second statement and create the two components. The statements may be read in a different order. If more students follow the sections in order, it reduces the reasoning diversity.

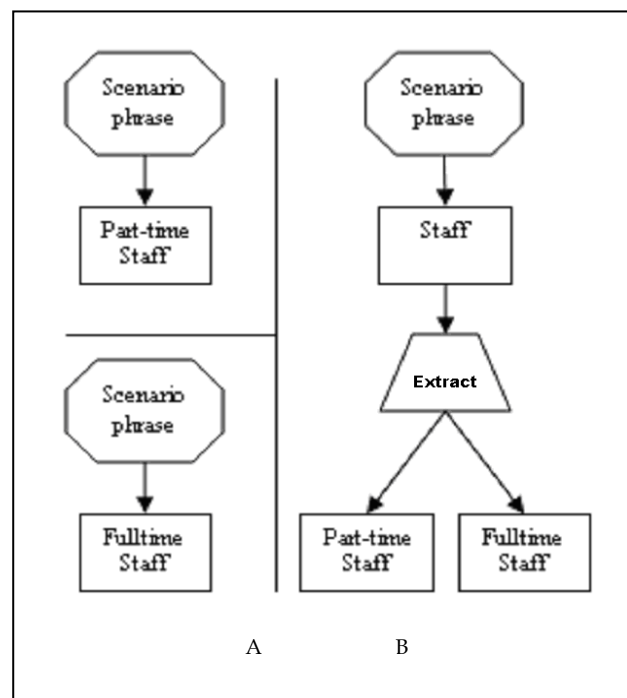


Figure 5.8 Component reference diagrams

Alteration statements can be written in such a way that students have to consider the sections in an order. In the previous example, if the alteration statement didn't mention the part-time and full-time staff explicitly then the student would have to use the extract function. Students would not be able to create components directly.

As a result of this, the entities would have only one ideal reference diagram in Figure 5.8. The number of ideal reference diagrams for a component has an effect on the diversity of the students' reasoning. Controlling the number of ideal reference diagrams could help the semi-automatic approach deal with the reasoning diversity.

The size of the ideal reference diagrams has an effect on the reasoning diversity as well as the number of the ideal reference diagrams. The size depends on the number of the related alteration statements in the scenario text. Each alteration statement causes creation of new component/s in the ideal reference diagrams. The ideal reference diagram becomes longer to reflect these alterations. Figure 5.9 shows a long reference diagram. It shows that there are two alteration statements in the scenario text. Students need to come up with the module entity first and then apply the two extract functions to the module in order to have the correct diagram. If they miss any of the alteration, they may have potentially wrong diagrams. A long reference diagram is a harder scenario than a short one and increases the reasoning diversity.

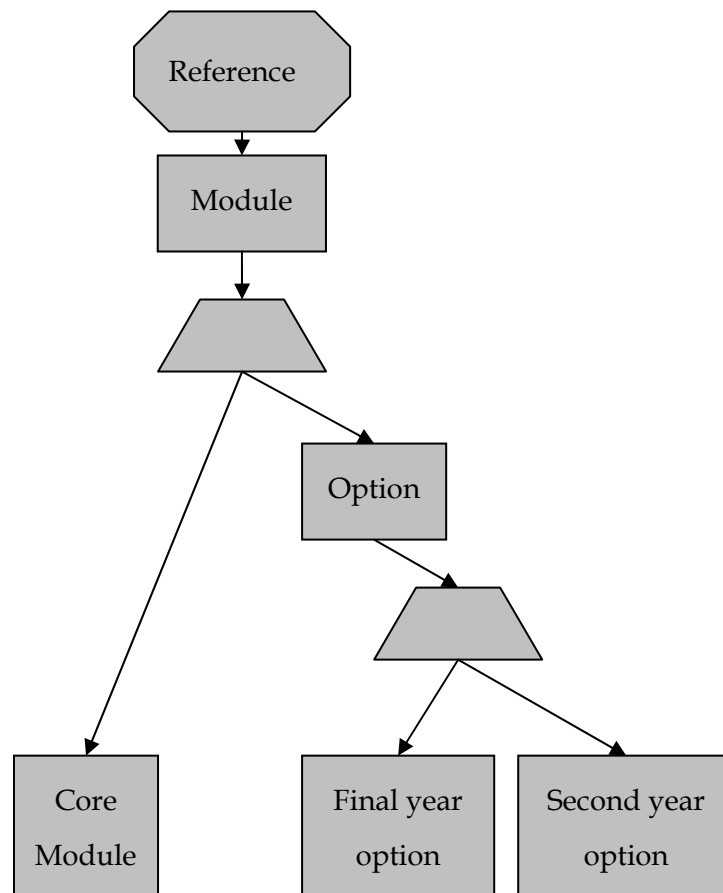


Figure 5.9 Long component reference diagram



Alteration statements can be different types. Each type corresponds to the action functions defined in chapter 4. Students read the alteration statement and apply the related action function to their diagram. For example, a merge type alteration statement requires the usage of the merge function. The same type alteration should have the same affect on the student reasoning. Scenario text can be written in various ways to get same alteration affect. Section 5.3.2 mainly focuses on writing the same type of alteration statements.

### 5.3.2 Writing statements

This subsection introduces e-condition, a-condition and c-satisfier concepts. They are used to create different statement types. This section gives the example sentence construct for explicit c-satisfiers. At the end it highlights the relationships between scenario sections and c-satisfiers.

Students are taught to come up with their diagram components when certain conditions are satisfied for them. The research calls these conditions “existence conditions” or, in short, e-conditions of a component. Each component type has various e-conditions. Some of them are common and some are different from each other. For example, an entity type in ER diagrams has got four e-conditions: uniquely identifiable, many occurrences, minimum one attribute and interest of business. Facts given in the scenario text and known as general knowledge about the system satisfy e-conditions of a component. The research calls these facts “condition satisfiers” or, in short, c-satisfiers. Students should know the e-conditions of each component type and find the c-satisfiers of all components to get the correct design.

Some c-satisfiers of a component are mentioned explicitly in scenario text. Others are assumed to be known by the student as subject domain knowledge (Fulford, 2001). If all c-satisfiers were explicit in the scenario text, it would make the design very predictable. On the other hand, if too many c-satisfiers are implicit then students make assumptions about the system. This causes different interpretations of the text and increases the reasoning diversity.

C-satisfiers of a component can be written in various ways. The c-satisfier can be a sentence or phrase in the scenario text. If a sentence is a c-satisfier of a component, the same sentence can implicitly be a c-satisfier of another component. For example, a sentence can mention a relationship between two entities. This sentence supports

the existence of the relationship explicitly and those entities implicitly. An explicit c-satisfier should be written as a sentence with the consideration of implicit meaning so that the scenario text can be divided into sections easily. Otherwise segmentation of the scenario text can cause unexpected effects.

A c-satisfier can be written as a sentence in different ways. For example; multiple instances is an e-condition of an entity type. The first sentence in Figure 5.10 mentions the existence of the instances generally. The second sentence mentions all instances of an entity type specifically. These sentences are different sentence constructs for the same c-satisfier. Scenario text can be written naturally by using sentence constructs with no limited grammar and vocabulary. Sentence constructs can be defined as guidelines for scenario writing.

**Sentence A:** *“Our company is divided into departments.”*

**Sentence B:** *“The company has got accounting, sales and purchase departments.”*

**Figure 5.10 Sentence construct example**

Some e-conditions of component types overlap. This might cause a different interpretation of a c-satisfier. For example; “multiple instances” is an e-condition of both an entity type and multi-valued attribute type. A unique identifier is an e-condition of only the entity type. If a c-satisfier of multiple instances for an entity is written in a section and a unique identifier of the entity is mentioned later on in the next section, then some students may create first a multi-valued attribute then convert it to an entity after reading the second section. Figure 5.11 exemplifies this case. After reading Section A in the figure, students may create a “programming language” attribute for a employee entity or a new entity with a relationship to “Employee” entity. When the students read section B they all need to have a new “programming language” entity. These same e-conditions of two component types make an alteration effect during design process. Some students consider the alteration statement and some not. This results in the reasoning diversity.

**Section A:** *“The manager wants to keep the record of which programming languages each employee knows.”*

**Section B:** *“Each programming language has been given a unique number.”*

**Figure 5.11 Convert alteration example**

Students alter the diagram components as well as creating one during the design process when certain conditions are satisfied. These are called the “alteration conditions” or, in short, a-conditions. For example, merging two entities may be performed if three a-conditions are satisfied: (1) Two entities have got the same number of attributes and (2) the same types of attributes, (3) the usage of data from both the entities together. The c-satisfiers of e-conditions can be used to satisfy first two a-conditions. Students create the required two entities after reading the c-satisfiers. An explicit c-satisfier can be written in the next section for the last a-condition. Students, who consider this c-satisfier, should merge the existing entities.

The c-satisfier for “the usage of entities together” condition can be kept implicit if this is general knowledge. In this case, students need to merge the entities whenever they are present in their diagrams. If each a-condition is satisfied in a different section, the c-satisfier in the last section will be an alteration statement.

A statement consists of c-satisfiers. Their size can be one sentence or several sentences. The types of statements depend on their c-satisfiers and related c-satisfiers revealed in previous sections. The scenario writer uses c-satisfiers to construct expand, alter or support statements by using sections. The writers use all c-satisfiers of a component in one section to create an expansion statement. Alternatively, they could use c-satisfiers of a component in a different section to construct the support and alteration statements. Section 5.3.3 discusses the usage of the sections for the statement types.

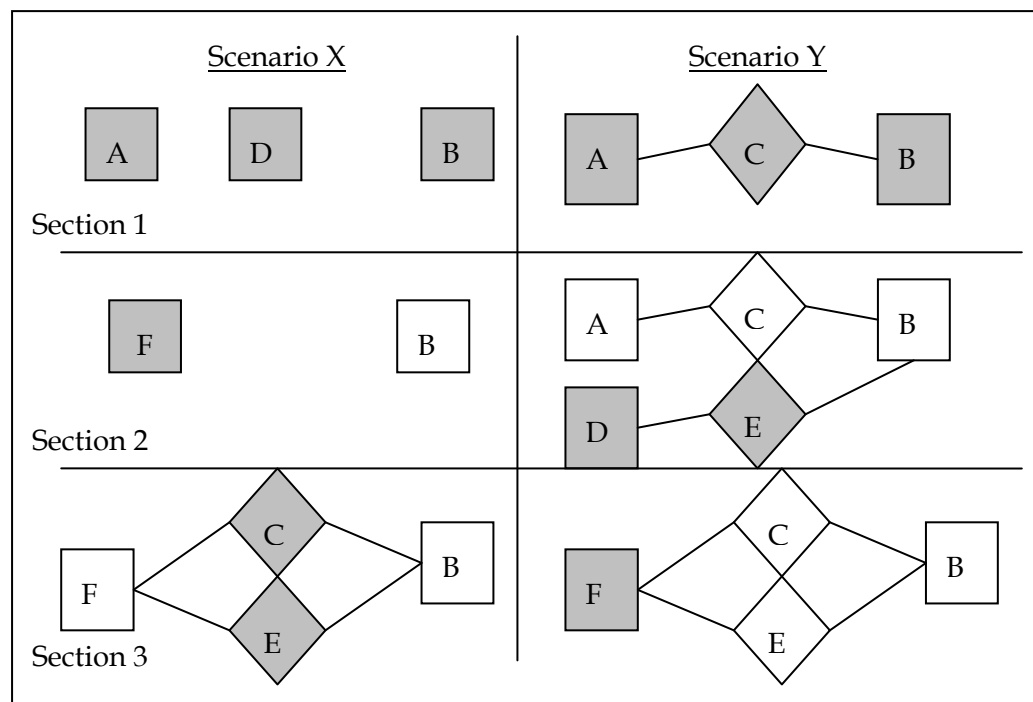
### **5.3.3 Scenario Section**

This subsection suggests a guideline about the writing of scenario sections in order to get the required effect from the statement type.

The scenario section is a part of the requirements text, which consists of one or several statements. Statements in a section can be any type. It is suggested that writers should only use one statement for a concept to avoid mixed typed statement. Thus the sections don't have both expansion and alteration statements of a component. Otherwise, students can consider these statements in the different order. Then statement types may affect the students' reasoning unexpectedly.

Section size is a number of the statements in the section. Some statements in the same section can be logically related. Related statements may cause implicit

alterations of a diagram. For example, a section contains three expansion statements for two entities and a relationship between them. Deleting one of the entities causes the deletion of the relationship or modification of the relationship. Related statements in a section may increase reasoning diversity. Figure 5.12 shows two design histories for the same system. Scenario Y mentions relationship C and E in section 2. After merge action, these relationships are changed and linked to entity F. Scenario X mentions these relationships in section 3 after the merge action. Some students may wrongly modify the relationships C and E after considering section 3 of scenario Y. The style of writing scenario Y has potentially two more wrong components than the style of scenario X has. Sections 1 and 2 in scenario Y have two related statements, which mention components C and E. It is suggested that sections shouldn't have too many related statements. The author can decide this number depending on the complexity of the scenario text.



**Figure 5.12 Two written style of scenario text for the same system**

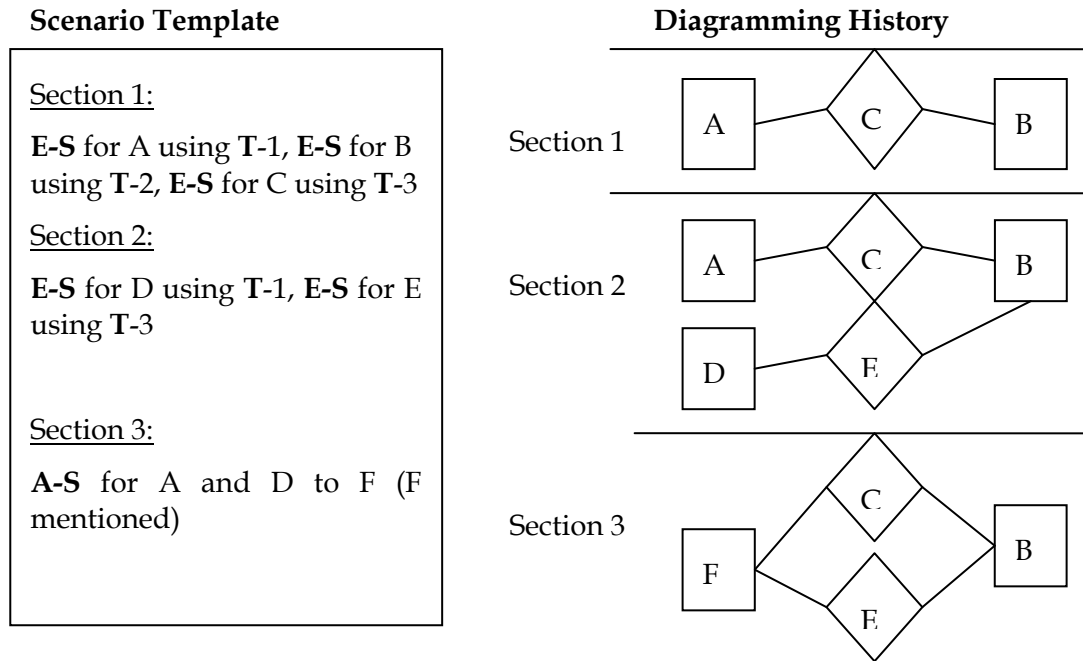
The similarity of the scenario text is based on the reference diagrams. The reference diagrams don't represent the related statements. The ideal reference diagrams of component F in both diagrams in Figure 5.12 are the same, although their design histories are different. Related statements might have indirect effects on the reference diagrams. Some students might have the same wrong or right reference diagrams for different scenarios if the scenarios have used related statements in the

same way. In this case, any association between ideal reference diagram, Students' reference diagrams and related statements are investigated. If there are any relationships, a generic case can be generated. For example, feedback for relationship E between entity D and B can be automatically generated since the correct E relationship is between entities B and F. Even if any generic case can't be generated, the information about related statements can be used to determine the complexity of the scenario text. For example, Scenario Y is more complex than Scenario X since it allows more reasoning diversity.

### 5.3.4 Writing similar scenario text

No research has been found which focuses on producing similar scenarios. The research into text writing has focused on writing requirement specifications of software systems aiming at concise and consistent statements (Miriayala and Harandi, 1991). Examiners write scenarios based on their experience. They may gradually improve the scenario after getting feedback from the student solutions. Some of these scenarios can have common statement types and sentence constructs. The similarity of scenarios can be detected during marking to prepare generic cases and improve the automation. On the other hand, the assessor can also write the scenario by using predetermined statement types and sentence constructs which are mentioned in previous sections. These scenarios naturally become similar to each other. The assessment process is automated more when similar scenarios are written deliberately rather than detecting the similar scenarios. However this increases the workload of the preparation stage of the assessment process. The research adopts the latter method in this section. The section introduces concepts of scenario template, statement template and sentence construct.

The assessor can be provided with a template scenario to ease the preparation of scenario text, Figure 5.13 shows a scenario template. The template consists of three sections. Section 1 consists of three expansion statements. The first two statements are for entity A and entity B. The last one is for relationship C. The right side of the figure shows the diagramming history of the template. Section 2 consists of two expansion statements. The diagram in section 2 has two additional components as a result of section 2 in the scenario template. Section 3 has one alteration statement, which merges entity A and D, creating entity F.



**E-S** : Expansion statement , **A-S** : Alteration Statement, **T** : Statement Template

**Figure 5.13 Scenario template example**

The scenario template also has the statement templates for each statement. In Figure 5.13, section 2 has two expansion statements for different entities. Statement template 1 is used for entity A and statement template 2 is used for entity B. Figure 5.14 shows these templates. Template 1 consists of four sentences. Each sentence is for one c-satisfier of an entity. Template 2 consists of two sentences. The second sentence is for two c-satisfiers of an entity. Figure 5.14 also shows two sentence constructs for the “important” c-satisfier. Template 1 uses construct 1 and template 2 uses the construct 2. However, the sentence construct is optional. The assessor can consider the example sentence and write their own. This can make the scenario text more natural.

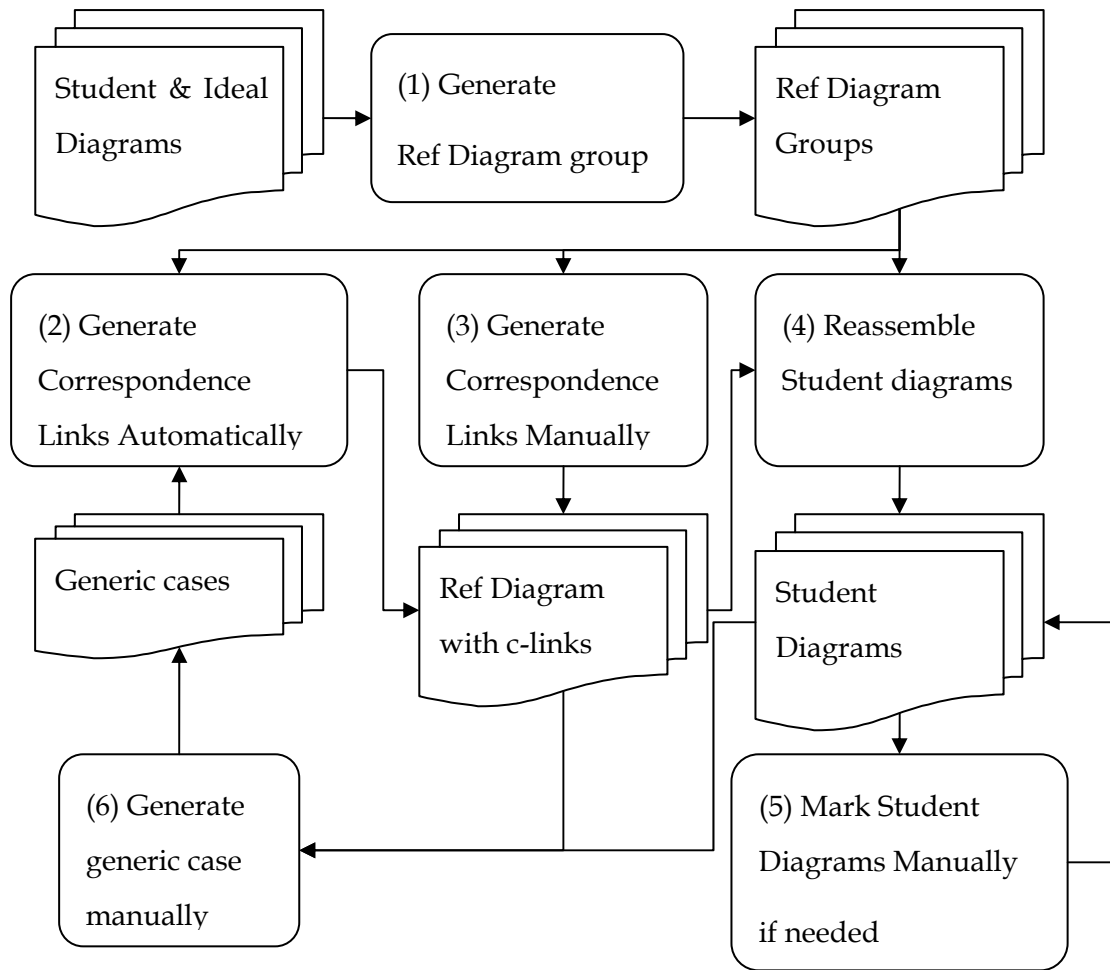
Statement Template (T)	Sentence Construct (S C)
<b>T- 1:</b> Entity Expansion <ol style="list-style-type: none"> <li>1. Important : S C -1</li> <li>2. Multiple -attribute</li> <li>3. Multiple-occurrence</li> <li>4. Uniquely identifiable</li> </ol>	<b>SC- 1:</b> <Subject> wants to keep track of <object> Example: A school wants to keep track of students
<b>T- 2:</b> Entity Expansion <ol style="list-style-type: none"> <li>1. Uniquely identifiable</li> <li>2. Important : S C -2</li> </ol>	<b>SC- 2:</b> It is important to know <object>

**Figure 5.14 Statement templates and sentence constructs**

The assessor is provided a procedure alongside the template to follow. The assessor first decides a domain. Then they enter a label for each component in the diagramming history for the scenario template. After that sentences are written by using those labels. Computer assistance can be given to the assessor for the writing process. However, it is hard to check whether the scenario follows the template. The scenario will be verified only after getting student feedback.

## 5.4 A complete process model for diagram marking

The partial marking method in Chapter 4 illustrates only the framework of the method. The first sections in this chapter showed how to use reference diagrams to automate the marking more. Figure 5.15 gives the details of the method by including new concepts seen in the first section. The marking includes 6 processes. The first process generates reference diagrams from the student and the ideal diagrams. The second process automatically marks some of the reference diagrams by using generic cases. Basically it creates correspondence links between student diagram components and the ideal diagram if it can. The third process lets the assessor mark some of student diagram manually. Finally the students diagrams are reconstructed with their marks.



**Figure 5.15 Marking process model**

Generic cases in Figure 5.15 are generated by Process 6 on the diagram. It uses the reference diagrams with correspondence links to ideal reference diagrams. It uses student diagrams of all examinations which the system keeps. The details of the process are discussed in the first section. Process 6 is too much to be done by the assessor. It is not part of the diagram marking process. The process has to be carried out by experts. Initially generic cases are prepared and built into the system. The experts can add more cases later when the system has more reference diagrams in order to improve the performance or to tune the process.

Manual partial marking is done after automatic partial marking in the model (see Figure 5.15). If some reference diagrams can not be marked automatically then the assessor needs to mark them. However marking some reference diagrams manually can be impractical. They can be too big and time consuming to be understood and interpreted by the assessor. In this research, these are called “ malformed reference



diagrams". The marking of malformed reference diagrams is postponed until after process 4. Process 4 reassembles the student diagrams. As a result of this, some of the student diagrams will have unmarked components. Those student diagrams are presented to the assessor one by one. They mark the unmarked components in the student diagram. This is referred to as the full marking style as opposed to a partial marking style. Since the assessor doesn't have to interpret the reference diagrams of components, marking the malformed reference diagram is avoided in the full marking style. Full marking is placed in the model as a Process 5 in Figure 5.15.

The components with malformed reference diagrams could be correct but have wrong reference diagrams since the assessor doesn't verify those reference diagrams in the full marking style. The case generation process (6) considers malformed reference diagrams separately from other type of reference diagrams. The process needs to validate the reference diagrams of the marked components. However it doesn't need to do any validation of reference diagrams when partial marking is used since they have been seen by the assessor. Detailed discussions of this are made in Chapter 7.

## 5.5 Summary

This chapter presented the marking process model, which is the extended version of the partial marking process in chapter 4. Automatic partial marking is developed to improve the performance of the partial marking. A new generic case definition is developed for the automation. A manual full marking style is added to the model in order to overcome shortcomings of partial marking. The model combines automatic and manual partial marking, and full marking together. The model uses a generic case base, which eliminates the necessity of having complete rules. The case base grows gradually when the number of examinations entered into the system is increased.

This chapter made a definition of "similar scenarios". It highlighted the relationship between similar scenarios and the case base. It also introduced statement type and sentence construct concepts in order to write similar scenarios. A sample scenario template is prepared which uses a combination of different statement types. The procedure of using the template is defined.

The next chapter focuses on development of the components needed for the proposed framework in Chapter 3. It identifies the requirements for the components and gives the implementation details. Since the components are used to evaluate the framework, only the main parts of the components are implemented and user interface issues for components are not touched in the chapter.

# CHAPTER 6

## Design and Implementation

### 6.1 Introduction

The proposed semi-automatic approach consists of two essential components: the diagram editor and the marking tool. The main part in the marking environment is the partial marking. To support partial marking, a design trace model is developed in Chapter 4. Additionally, the marking process model is enriched by automatic partial marking and manual full marking to improve the performance of the process, as described in Chapter 5. The scenario writing guideline is also introduced in Chapter 5 to increase the automation of partial marking.

This chapter discusses the design and implementation details of the diagramming, the marking and authoring components of the semi-automatic assessment environment. The developed prototype system in this chapter is used to evaluate the semi-automatic approach in Chapter 7. The implementation details of the system are given in Appendix E.

The first section discusses the requirements of a diagram editor, which captures the design history of the student diagrams. It describes the components of the editor and how the design trace model is implemented. The second section discusses the requirements of the marking environment in which the examiners mark the student diagrams captured by the diagram editor. The section describes its components and explains how the marking process model is supported by the environment. The third section introduces the components of the computer assisted authoring tool which enables the usage of scenario templates in order to prepare scenario text complied with the scenario writing guideline.

### 6.2 Diagram Editor

The design trace model which is developed in Chapter 4 uses the online trace production approach. This approach requires additional activities from the students

during diagramming. These activities are the students' self-explanation about their actions. The main problem of self-explanation whilst diagramming the problem is the high cognitive load. A new diagram editor is designed to deal with the cognitive load of self-explanation. This section looks at components of the diagram editor and examines how cognitive load may be reduced.

### 6.2.1 Requirements

Some of editor's requirements have already been determined in Chapter 4. The chapter suggests that the editor needs to support scaffolding, "reference listing", and "special functions" techniques to be able to do the online production. Additionally, the editor needs to have functionalities that conventional editors have, such as adding and deleting diagram components. During the implementation of these functionalities, user interface issues need to be considered as well.

Conventional commercial diagram editors use a simple drag-and-drop user interface (e.g. MS Visio, IBM Rational rose, Visual paradigm) and some editors additionally provide commands to users in order to draw complex engineering diagrams (e.g. AutoCAD). All research, except ERM-VLE mentioned in Chapter 2 only uses a drag-and-drop interface for their editors. ERM-VLE uses a simple command-based user interface. The users can interact with the editor only via the commands. It draws the diagram automatically based on the user commands. It gives immediate feedback about the user actions.

ERM-VLE's diagram editor, which uses automatic diagram drawing may have some advantages over the conventional drawing tools in the assessment. For example, the analysis of database exam scripts (Thomas, 2004) reveals that students often redraw their diagram during the design. Moreover, some of them redraw their final diagram to have a better layout of their designs. The automatic drawing could save the student's time during the examination in this case since the editor can redraw the diagram instead. The students can then focus more on designing than drawing the diagram.

In the automatic diagram drawing environment, additional commands of the editor can be added to undertake some design tasks. This can provide some higher level commands to modify the diagram. These commands can enable the implementation

of the special functions defined in chapter 4 (e.g. Merge and Split). Therefore this research adapts the automatic diagram drawing for the diagram editor.

A user interface for the commands can be developed in various ways. The simplest implementation would be command buttons and input boxes for the parameters for the functions. Alternatively, clickable diagram components can be used to eliminate input boxes. Moreover, new technologies can be used to make commands user friendly. For example multi-touch screen technology (Pennock & Tabrizi, 2008) can be used for special functions. The user holds two entities by using their two hands and drag them together to merge them, or pull out some attributes of an entity with one hand whilst holding the entity in place in order to split the entity. However, this thesis doesn't focus on creating a best diagram editor for the semi-automation. For the purpose of this research, the editor only needs to be good enough to test out the semi-automatic approach. However, the user interface of a new editor shouldn't create any additional difficulty, which prevents students from entering their design into the system.

### **6.2.2 Components of the editor**

The prototype diagram editor is designed to fulfil the requirements of the diagramming for semi-automation. The editor is based on automatic graph drawing (Ellson et al. 2002). This section explains the components of the editor.

The prototype editor consists of three panes (see Figure 6.1): a scenario text pane, a diagram pane and a command pane. The scenario text pane is for displaying a scenario. It displays the scenario section by section. This method is called scaffolding in the self-explanation literature (Chi, 1989). Chapter 4 discussed the reason for using scaffolding technique for the trace production. It potentially reduces diversity in student reasoning. Students are expected to consider the information in that section only. The scenario text pane also has a feature, which is used to highlight the reference text. Noun phrase or sentences are highlighted when a component from the reference list in the command pane is selected. This feature is implemented to reduce the naming ambiguity and accountability purpose. For example if the student selects the "name" noun phrase in the list, the phrase will be highlighted in the scenario text and the student will see that the "name" phrase refers to "Department name" not "Employee name" in the context. The

misunderstanding will be prevented and students become accountable for their choices.

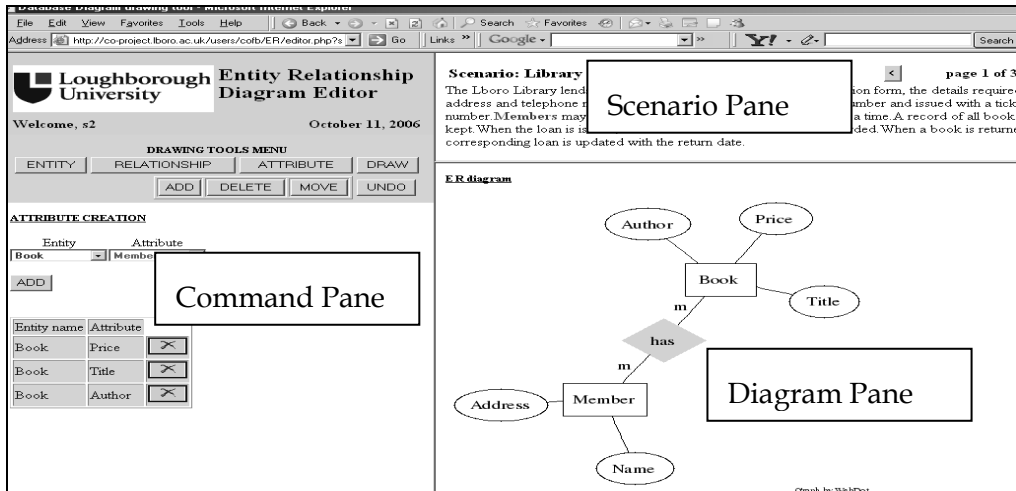


Figure 6.1 Diagram editor

The diagram pane displays the automatically drawn ER-diagram of the student design. In this pane, the database diagram is drawn or refreshed after the “Draw” button is pressed. This is a limitation of the current automatic diagram production rather than the satisfaction of a requirement. The diagram is reshaped to have a best presentation after each modification command is entered into the system. This sometimes makes a huge difference between the pre-modified diagram and the current one. The students might find this unacceptable. Therefore the “Draw” button is added into the pane to avoid any unwanted effect of automatic drawing. However, a new automatic drawing algorithm could be written which aims to minimise changes rather than achieve the best presentation. The algorithm could be developed later as another research programme to improve the usability of the editor.

The command pane is the main part of the editor. In this pane, students can add new components or modify current diagram components. The pane has one tab for each command needed for diagramming. Each tab has some elements required by the commands. Three tabs are created in the pane to add a new diagram component for an entity relationship diagram. To create a new entity component the students choose the tab and pick a name for the entity from the list box. The list box element has all different noun phrases or sentences appearing in the current section of the scenario text, which is called a “reference list” in this thesis. In this way, a direct

reference of the component is captured. Unlike KERMIT, the editor does not allow the student to name the DR-component. As discussed in Chapter 4, the explicit naming allows name ambiguity which allows inconsistencies between the label of the component and the reference phrase.

The methodology of the entity relationship modelling requires creating entity components first. The editor enforces the students to follow this method. The relationship or attribute components can be added to the diagram only if the related entities exist on the diagram. To create an attribute component, the students need to choose a name from the reference list and entity from entity list which holds entity names in the current diagram. To create a relationship component, the students need to choose the participant entities from the entity list. The reference list in this tab includes sentences in the current section. Students need to choose a sentence as a reference for the relationship. As, unlike the entities and attributes, students need to name the relationships. This is because generating a list for relationships from the sentence is less obvious than a list for entities and attributes.

The students can also modify their current diagram. Three main commands are created for this purpose: Split, Merge and Convert. The split command extracts a new entity from a current entity. The merge command combines two entities into one. The convert command creates a new entity out of many-to-many relationships. Each command is implemented with the use of buttons which are called “function buttons”. The function buttons are applied on diagram components, for example, to split an entity into two entities, the students goes to the related tab, chooses an entity, fills the required fields and presses the “split function” button. The diagram in the diagram pane is redrawn with the change as a result of split function. The entity in the diagram is modified and a new entity is created out of some of the entity’s attributes. The editor records the use of the function buttons. The record gives the history of the modification.

The function buttons are designed to reduce the cognitive load of self-explanation. The students implicitly explain their activities whilst modifying the diagram. Detailed discussion of implicit explanation can be found in Chapter 4. Function buttons can be enhanced to make more self-explanation orientated assessment. In this case, the students need to explicitly give a reason why the function button is used. This could be a rule of the modelling and a reference from the scenario. The

students get an additional mark according to the reason they give. The current version of the editor supports only the implicit self explanation.

The number of function buttons can be increased to modify the diagram in various ways. For example an entity convert button is created to transform an entity to a composite attribute or an attribute convert button is created to transform a composite attribute to an entity. These types of additional buttons aren't a requirement of partial marking, although it may make the editor more user-friendly for specific scenarios. On the other hand, there should not be too many function buttons since it may increase the cognitive load (Miller, 1956). The editor only implements the main three commands.

The delete command is a requirement of any diagramming tool. Deleting an attribute of an entity and a relationship removes these components from the diagram. The effect of deleting an entity is quite different in the proposed editor from a traditional editor. A delete command not only removes an entity from the diagram but also remove the attributes of the entity and related relationships from the diagram. This may increase the number of activities the students have to do. They may have to recreate all those attributes and relationships for a new entity again. They may even have to recreate some entities if the entity is a merged or is an extracted entity. The editor provides an undo command to avoid the use of the undesirable delete command. Function buttons of an entity or attribute convert can reduce the use of the delete command for some cases.

The implemented tool is a prototype editor, which fulfils the minimum requirement of the semi-automatic approach for testing purposes. The scenario pane of the editor mainly satisfies the scaffolding requirement of the editor. The diagram pane satisfies the automatic diagramming requirement. The command pane satisfies the self-explanation requirement. All these panes can be improved and different diagramming approach could be developed to find the optimum user-interface for a semi-automatic interface. However, this is outside of this thesis's scope.

### **6.2.3 A diagram drawing example**

This section gives a simple diagram drawing example of the use of the developed new editor in order to clarify the functionalities of the components described in the previous section. The editor forces the students to design their database models



systematically and iteratively. The students first produce an initial diagram which has the direct referenced components. Then they apply the design rules and system constraints to build their final model which has indirect referenced components until it satisfies all the system's requirements.

Figure 6.2 shows a scenario text used for the diagram drawing example. The scenario text has two sections, which are represented with bullets. The editor displays each bulleted section in the scenario pane one at a time. The students start drawing their initial diagrams by using the requirements in the first section. Then they modify their initial diagram until it satisfies the system's requirements in the second section.

- The Lboro Library lends **books** only to its **members**. On the application form, the details required are name, address and telephone number. Each **member** is assigned a unique number and issued with a ticket giving this number. *Members may borrow a maximum of six different books at a time.* A record of all **books** borrowed is kept. When the loan is issued, the date of loan and due date are recorded. When a **book** is returned, the corresponding loan is updated with the return date.
- The Librarian buys new books for the Library as necessary. Normally, several copies of a particular book are bought. Each copy of each book title is assigned a unique book copy number. A book title, on the other hand, is uniquely identified by an ISBN number. The title, author, date of purchase and price of each book are recorded. Different copies of the same book title can be purchased on different dates at different prices.

**Figure 6.2 Sample scenario text**

The user sees the list of noun phrases in the current section of the scenario text. They select one of them to create an entity or an attribute of an entity. For example "Member" is a noun phrase which appears three times in the first section. When the phrase is selected for an entity creation, the "member" text is highlighted with a red colour which is seen in Figure 6.2. "Book" is another phrase in the reference list which appears four times in the section. When this phrase is selected from the list, the "member" text is de-highlighted and the "book" text is highlighted. The users select a sentence from the section when they create a relationship. The first section consists of seven sentences. When one of them is chosen, it is highlighted so that students can keep track of the reference. Figure 6.2 shows the selected sentence in an italic font style. Figure 6.3 shows the diagram pane which has an initial diagram of a user, created for first section. Each component in the figure is directly referenced. For each directly referenced component there is a corresponding reference phrase in the scenario text.

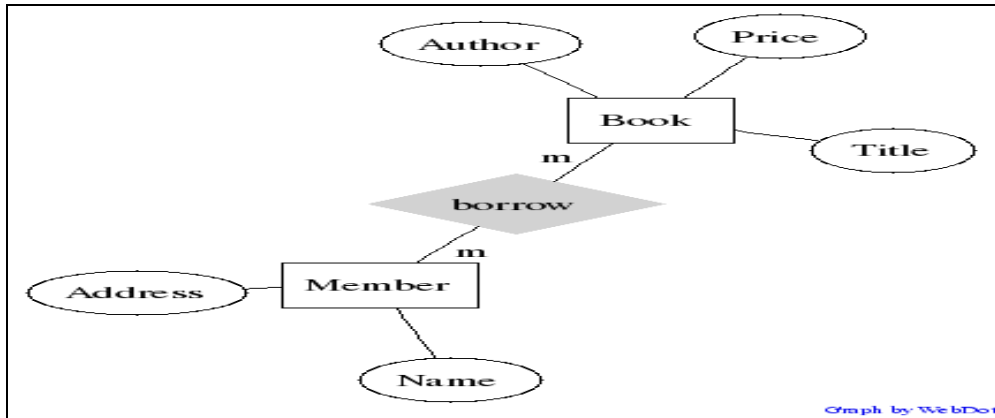


Figure 6.3 The initial diagram

The scenario text requires using the “split” function button during the design for the second section. Users, who consider the second section of the scenario after the first one, may modify their diagrams. For example, they split the book entity and create a new entity called “copy”. Figure 6.4A shows the split tab with input elements for the “split” command. The users pick the entity from the list to split and move its attribute to the new entity’s attribute box. The entity is named “copy” in the example. There isn’t any restriction on naming. Any name can be given to the entity. Users can create a relationship between “book” and “copy” entities to get the final ER-Diagram. See Figure 6.4B.

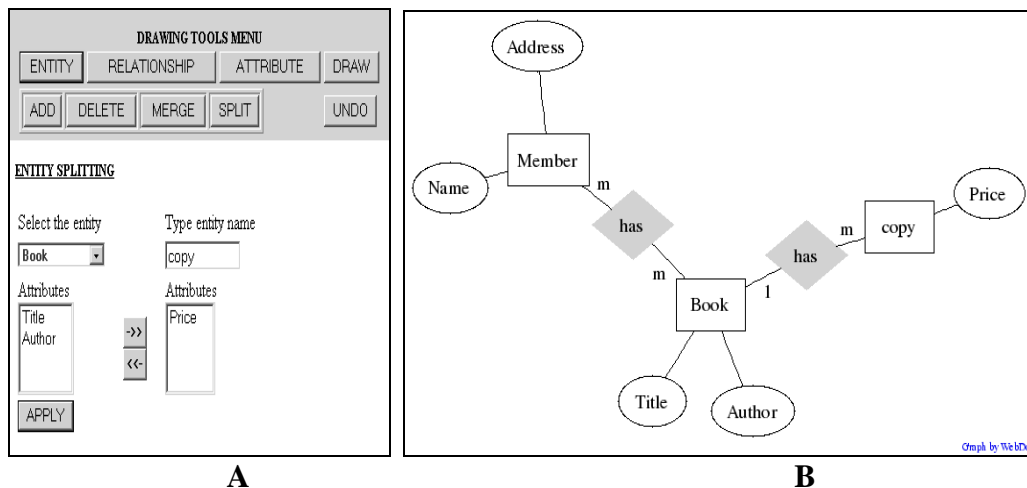


Figure 6.4 "Split" function

The students may read the first and second section before drawing the initial diagram. They might come up with the initial diagram, which is the same as the final diagram in this example. They can see the “copy” noun phrase in the reference list in the second section and create a copy entity. They later create the book entity.

The example scenario doesn't enforce the usage of the function button to allow the correct diagram to be drawn. Depending on the given scenario text, some functions of the editor are more useful than others, as discussed in Chapter 5.

## 6.3 Marking Tool

The marking tool provides an online marking environment. Online marking of assignments can lead to improved marking consistency and integrates well with online feedback (Mason and Woit, 1999). Online marking tools (Heinrich and Lawn, 2004), (Heinrich and Wang, 2003) have been developed to mark student diagram and essay-type work. The tools allow the teacher to leave some annotation and numeric marks on the electronic version of the student work. Unlike this research marking tool, it doesn't focus on removing the repetition of the marking task. Nevertheless, Plimmer and Mason (2006) suggest that online marking is a viable alternative to both paper and existing paperless environments.

The marking tool implements the main parts of the marking process model which is developed in Chapter 5. It uses the design traces captured by the diagram editor. The manual marking part of the model requires a user interface so that the human marker can mark some of the student diagrams online. The user interface consists of two interface types. One is for partial marking and the other is for full marking. The interface for full marking imitates the paper-based manual marking. The partial marking is a new marking style. It doesn't resemble the manual marking. The partial marking requires a very different interface to the full marking one. A new marking environment is designed to combine these two interfaces. This section examines the components of this marking environment and describes how the tool is used to do the main marking tasks.

### 6.3.1 Requirements

The two marking styles in the marking environment have some common requirements. For both marking styles the teacher's ideal solution and student work need to be presented to the examiner and the students' diagrams' components the can be selected to be matched to each other. Matching the diagram components is the main functionality of the environment.

There are four types of matched and two types of unmatched cases. The matched cases are “one-to-one”, “one-to-many”, “many-to-one” and “many-to-many” matches. One command is used for all these cases. The human marker selects components from both diagrams and uses the command to match them. The unmatched cases are for rejected and accepted components. Student diagrams might have some extra components which are not part of the ideal diagram. These components might be accepted which means the human marker doesn’t give any marks to the components. The components might be rejected, as they shouldn’t be in the solutions. Unlike the accepted ones, the rejected components need feedback to be given to the students. Even a negative mark can be given to the component. Two separate commands need to be given for these unmatched cases.

In the full marking style, student diagrams are represented as a whole. The main problem in this marking style is the room needed on the screen to display the diagrams. The environment needs to display the complete diagrams of both students and teachers. If the diagram solutions are large, the tool needs to have an auto-scale function. This function changes the size of the diagrams to fit them on the screen. Another option is to increase the size of the screen by using a large screen or multiple screens.

In the partial marking style, the student diagrams are displayed partially. The human marker sees some components of the diagrams with the design traces of the components during matching. Design traces are represented as a graph, called a reference diagram. The scenario reference part of the graph needs to be highlighted in the scenario text. This requires the scenario text to be displayed with the student diagrams in the environment. To reduce the cognitive load of reading design traces the reference diagram can be drawn gradually from the scenario references to the final component. The human marker can follow the student steps for that component while it is being redrawn. However, this animating of design traces is not an essential requirement of the marking.

The marker part of the environment is used to create a correspondence link between student and teacher diagrams. Before the system starts being used actively the environment needs to also have grade and feedback generators which give numeric grades and textual feedback by using correspondence links. The requirements of

these generators haven't been identified in the thesis since to evaluate the partial marking style, only the user interface parts of the online marking tool are needed.

### 6.3.2 Components of marking environment

The prototype marking tool is designed to satisfy the requirements of the manual marking for the semi-automation. The tool consists of four main components. This section explains those components. The section illustrates and explains user interfaces for both marking styles. As an example, the user interfaces shows a sample student and teacher solutions for the scenario text in Figure 6.2.

The marking tool has got an environment for matching diagram components. It displays the pictures of the two diagrams simultaneously. The pictures are clickable so that the marker can select the diagrams' components during matching. Figure 6.5 shows the user interface of the full marking.

The teacher pane has got the picture of the teacher's ideal solution. The diagram in the picture is dynamically drawn by the tool. Any selected components are colour coded.

The student pane of the interface has got a student solution. The selected components of the student diagram are coloured the same as the components of the teacher diagram.

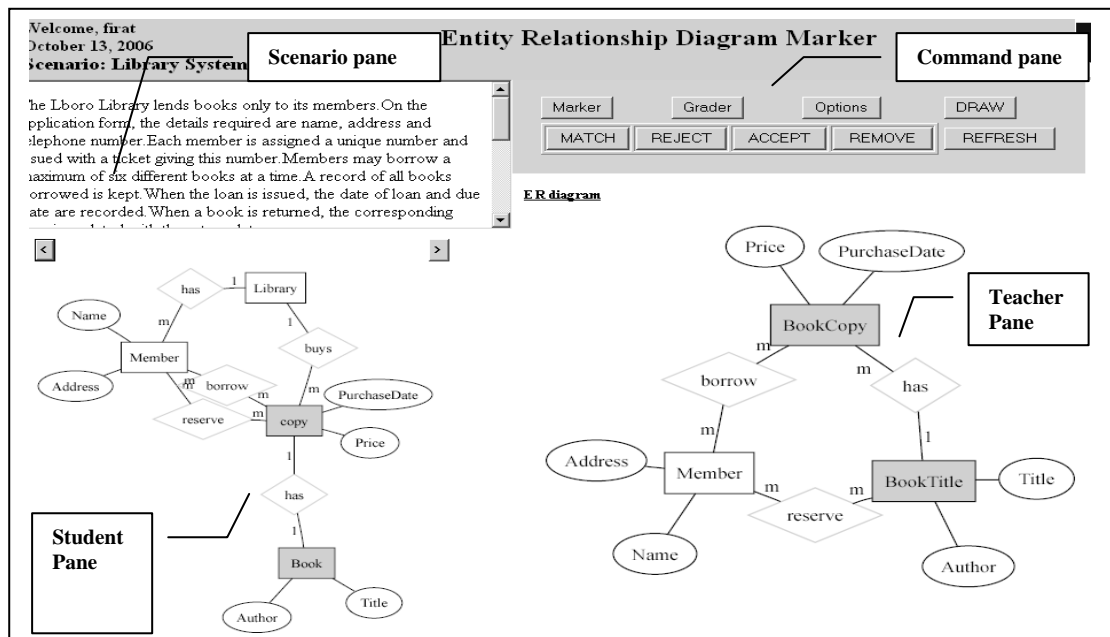


Figure 6.5 The user interface of the full marking

The student pane also has navigation buttons so that the human marker can move from one student solution to another in order to do the matching with the ideal diagram. The user interface of the match commands are implemented as command buttons. The human marker selects the components from both teacher and student diagrams and presses the match, reject or accept button. To deselect the components, the selected components are clicked again. To remove the correspondence created between the components, the remove button is used.

The scenario pane displays the scenario text for the student solutions. There is one scenario text and ideal diagram per marking session. References in the scenario are highlighted for each selected components in the student pane. This allows the human marker to check whether the component name and its reference are consistent. However, this is not a requirement of the full marking since the correspondence is used only for the current student solution, but it is a requirement of partial marking style. Further discussion about this can be found in Chapter 5.

Figure 6.6 shows the user interface for partial marking. The only difference to the user interface for full marking is the student pane. The picture in the student pane has the reference diagram of a diagram component. In the figure, the human marker sees that the book entity is composition of two entities and reference text of these entities is in the scenario text. The reference diagram helps the human marker to find the corresponding components in the ideal diagrams. In this case, the book entity matches with book title and book copy entities and a one-to-many link is created. Navigation buttons in this interface are used to move from one reference diagram to another.

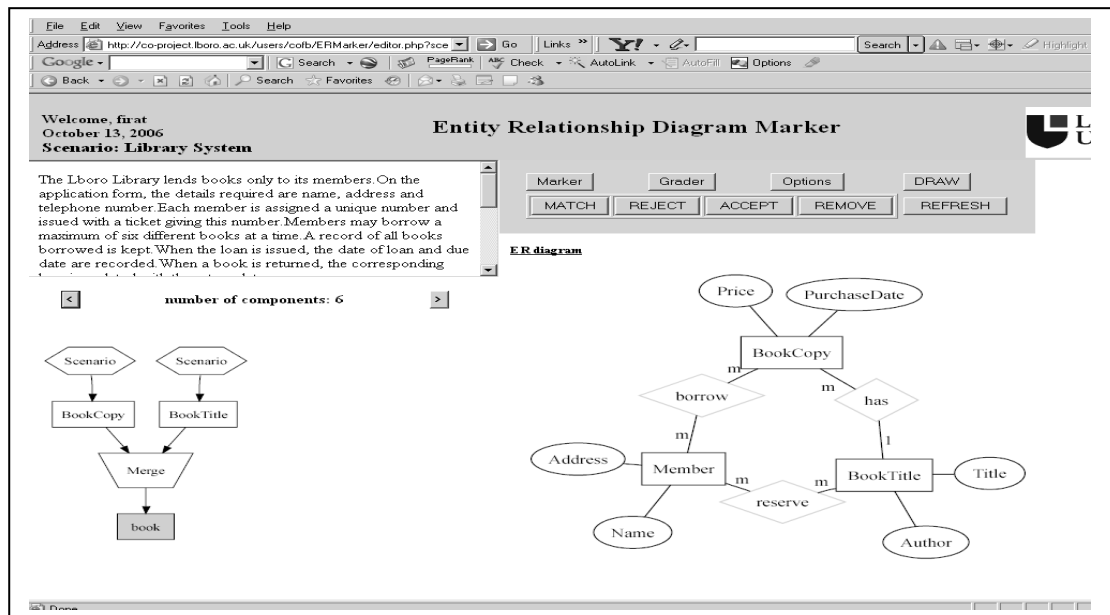


Figure 6.6 The user interface of the partial marking

The partial marking interface first displays the reference diagram of the entity components and then the attributes of those entities. Lastly, relationships between entities are displayed. This order of the partial marking allows further automatic marking. For example, the borrow relationship between book and member entities in a student diagram will be able marked automatically after book entity is marked.

The marking process model describes the how these two interfaces are used during marking. The marking tool firstly groups the submitted diagrams and marks some of the diagrams automatically. Then it displays the partial marking interface. This interface is used to mark the diagram components which couldn't be marked automatically. Finally the full marking interface is used to mark unresolved cases. That occurs when reference diagrams have become too long or they are malformed, which is described in Chapter 5. In the full marking style, the diagram's components that matched with the teacher's ideal solution are colour coded automatically to save time. The examiner can focus on only non-matched components.

## 6.4 Authoring Tool

The proposed semi-automatic system needs an authoring tool for the question preparation stage of the assessment process. It consists of two separate applications: the teacher diagram editor and the scenario text writer. The teacher diagram editor is very similar to the student diagram editor with some extra features. The scenario

text writer is optional for the semi-automation since the approach can use either a specially written scenario text for the semi-automation or the ordinary one. The scenario writer is designed to help semi-automation process in the diagram marking. This section describes the essential requirement of both applications and explains the user interface components of the scenario writer.

### **6.4.1 Requirements for the teacher diagram editor**

The teacher diagram editor of the authoring tool is used once the scenario text is available. The editor helps the examiner prepare the reference list for the scenario editor and enter the ideal solution for the scenario text. There are two types of reference lists used in the diagram editor: a sentence and a noun phrase. A complete sentence reference list can be automatically generated without human involvement. However, only part of the noun phrase reference list can be automatically generated. Noun phrase reference lists need to be edited by the examiner after they are generated. The examiner adds a new item to or removes an item from the list in order to have the complete noun phrase list.

The examiner enters an ideal diagram solution by drawing the diagram in the same way as the students do. This part of the teacher editor is the same as the student editor. The examiner could enter many solutions since each alternative solution increases the semi-automation. The detailed discussion about the ideal solution and automation are given in Chapter 5. In the student editor, each student can enter only one solution for each scenario.

### **6.4.2 Requirements for the scenario text writer**

The teacher diagram editor is easy to implement with the minor modification of the student editor. However, the scenario text writer of the authoring tool is complex system to implement. A complete solution for the automatic or semi-automatic scenario writer is out of this research scope, as this research focuses on the computer environment, which helps the examiner follow the guidelines given in Chapter 5. The environment will be used to write similar scenarios for different domains.

The CREWS (Achour, 1998) and The CP (Cox and Phalp, 2000) projects provide a set of guidelines for writing requirements text. Part of the guidelines suggests the language to use, including present tense subject-verb-object like sentences with no



adverb or adjectives. Christiansen and Have (2007) adapted the language guideline and developed a tool, which accepts the subset of natural English and represents them in the formal design diagrams. The guidelines for the scenario text writer are different to the CREWS (Achour, 1998) and CP (Cox and Phalp, 2000) guidelines. However, the research might use their suggested language to generate the sample sentence for the sentence construct described in Chapter 5.

In the environment, the examiner decides a domain for the scenario text (e.g. cinema ticket booking system) and a scenario template before they start writing the text. The examiner follows the template to write the scenario text. The representation of the template is important. It should be intuitive for the author to follow. For this reason, a diagrammatic representation is chosen. The teacher can follow the diagrammatic representation of the template to write a scenario, as opposed to students, who follow the scenario text to draw the diagrammatic solution. Diagram components can be labelled by using the ontology in the domain chosen by the examiner.

The environment can write initial mechanical text for the template in the chosen domain to give an example usage of the template for the domain. Additionally it displays the example text previously written by a human in a different domain for the same template. This natural language text provides extra help to the examiner when writing the scenario.

The scenario text writing in the environment described above requires scenario template preparation, labelling components in the template diagrams for the chosen domain and writing sample formal text for the template diagram. It is possible to design a computer assistant tool for the template preparation, component labelling and short text writing. This tool will reduce the workload of scenario preparation and they will increase the acceptability of the new system. However, implementing the tool is not essential to test the scenario writing environment. Implementing only a user interface for the environment is adequate for testing purposes. The user interface for the prototype tool displays the information needed for the scenario writing, which has been manually entered rather than being prepared by another tool.

### 6.4.3 Components of the scenario text writer

The writing environment consists of three types of pages: Plan, Section and Production pages. The plan page is used to select a domain and a scenario template. Section pages are used to get the input text from the examiner. The production page is used to generate a complete scenario text by using the input text. This section explains the user interfaces of these pages.

#### 6.4.3.1 The Plan Page

The plan page is the first page of the environment. It is used to select a scenario type and a domain for the new scenario. Depending on the selected scenario type, the page displays a sample scenario text and the related ideal diagram. The page also shows a whole diagram template. Components in the diagram template are labelled by using the chosen domain. Figure 6.7 shows an example of this page type. In the example, the examiner decided to use an “event organiser” domain and chose the “split” scenario type. The diagram template uses “event” and “member” names in the event domain. A related example scenario text and its diagram in the “computer training school” domain are displayed in the page.

The screenshot displays the 'The Scenario Planner' interface. It features three dropdown menus: 'Chose A Domain' (set to 'Event Organising'), 'Chose A Scenario Template' (set to 'Split Type'), and 'Chose A Diagram' (set to '1'). Below these is a section titled 'Scenario text for the example diagram' containing three paragraphs of text. At the bottom left, it shows 'Page 1/5' and a 'Next' button. On the right side, there are two diagram templates. The top one, 'The diagram template', shows entities 'Event' and 'Member' with their attributes and relationships. The bottom one, 'The example diagram', shows entities 'Course' and 'C offering' with their attributes and relationships.

Figure 6.7 Plan page example

Many possible diagram templates may exist for the same domain. One of them can be used for the chosen scenario type. For example, one of the diagrams can have a “sponsor” entity and other can have a “Member” entity instead. This enables the examiner to write a structurally similar but different scenario text in Figure 6.7. The plan page provides options to choose a different diagram.

Some components in the diagram template might not have a label in the chosen scenario type. In that case, components are named as X ,Y, Z etc. These names indicate that the examiner doesn't need to mention a component's name explicitly in the scenario text. As a result of this, students need to use their own names to label components. The diagram template in Figure 6.7 uses the "Entity-X" label as a component name for this reason.

The diagram template in the plan page is the final diagram or ideal solution for a system, which the examiner is going to write about. The diagram templates in the section pages are the intermediate diagrams based on which the examiner writes the scenario sections. Seeing the final diagram initially provides the examiner the overview of the required system.

#### 6.4.3.2 The Section Page

Section pages are the main part of the environment. There is one page for each section of a scenario template. For example, if there are three sections in the template, the environment will have three section pages. Figure 6.8 shows an example section page. The section page consists of four areas: the diagram template, example diagram, and example natural text and new text. The diagram template area displays a part of a diagram template. Components of the diagram are labelled if the examiner has chosen a domain in the plan page. The example diagram area is similar to the diagram template area and shows the diagram previously used by a user to write a scenario. The components of this diagram fragment are labelled. These labels are different from the labels in the diagram template when the example scenario is written in a different domain.

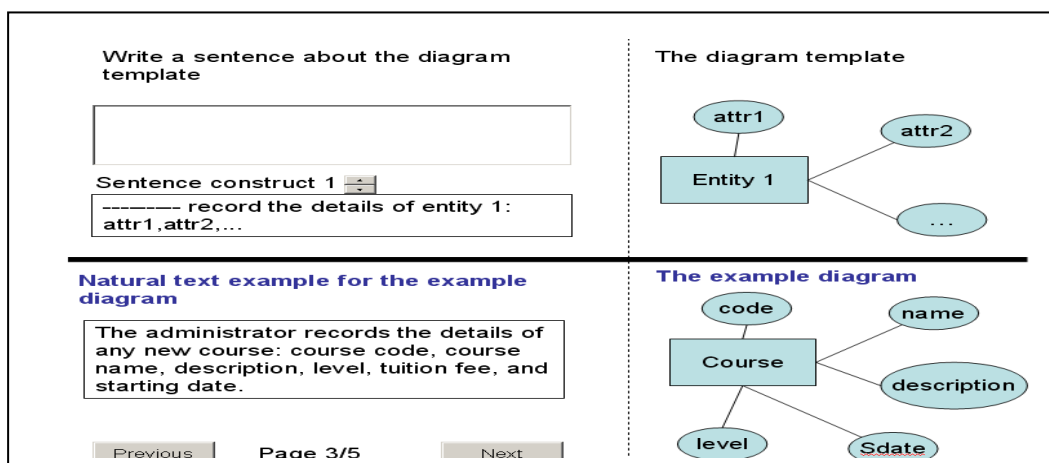


Figure 6.8 Section page example

An example natural text area in the section page is used to display the text previously written for the sample diagram. The sample diagram and the natural text are not an essential requirement of the environment. They are implemented to help the novice user follow the scenario template. A new text area is the main part of the page. It is used by the examiner to input their scenario text for the given diagram template. This area also displays computer generated text for the diagram. The text is automatically generated by using a predefined sentence construct. The examiner may choose a construct from the list and rewrite the mechanical text in order to make the text more natural. It is optional for the examiner to use the sentence constructs. If the constructs are used, it increases the similarities between scenario texts. That may improve the performance of the semi-automation.

### 6.4.3.3 The Production Page

The final page type of the scenario writer is the production page. This page combines the scenario sections written by the examiner in the section pages and displays them together as a new scenario text. The examiner edits the new scenario in order to make it more like natural text. Like the plan page, it displays the final diagram template. The examiner checks the compatibility of the scenario text with the diagram. The environment also displays the sample scenario text and its related diagram to the examiner. After editing, the scenario text is submitted to the teacher diagram editor. Figure 6.9 shows the production page with an example scenario.

**The Scenario Producer**

The True Colour (TC) is a charitable trust. It organizes events to increase the awareness of animal abuse for children. TC's manager gives you the following description of the business:

Events are planned by the trustees. They decide the details like: event name, description, purpose, date. Executive management board (EMB) calculates the cost and set the fee of the event. The charity secretary records event information with a unique event number to the event file.

TC keeps the record of its members. Member's name, address and phone number are taken on the membership form. At the end of the membership process, each member is given a member number.

The charity gets the members' help for organization of an event. The members who contribute an event are recorded. A special thanks-card is sent to their addresses.

Popular events are repeated several times a year. If necessary, the fee of an event is adjusted whenever it repeats.

**The diagram template**

ER Diagram showing entities: Event, Member, Entity X. Attributes: Event (description, purpose, No, name), Member (name, address, Tel, No), Entity X (fee, Edate).

**Scenario text for the example diagram**

The Blue Computer Training School (BCTS) provides a wide range of computer training short courses. BCTS's manager gives you the following description of the business:

The administrator records the details of any new course: course code, course name, description, level, tuition fee, and starting date. The details of new students are kept into the student file. The school needs to know their name, address and qualification. Each student is assigned a unique student id. A student may enrol on several courses. At the end of a course, the student is assessed and the grade achieved is recorded.

Same course is offered several times a year. Students select a suitable starting date of the course during the enrolment. If necessary, the tuition fee of the same course is adjusted whenever it is offered.

**The example diagram**

ER Diagram showing entities: Course, Student, C offering. Attributes: Course (description, level, code, name), Student (name, address, qualification, StudentID), C offering (fee, Sdate).

Previous Page 5/5 Submit

Figure 6.9 Production page example

The environment doesn't validate the new scenario text to see whether the examiner follows the scenario template during the writing. It only facilitates the scenario writing process. However, teaching assistants can be used to produce a diagram for this scenario and their diagrams can be analysed as an initial validation. In the long run, students' diagrams can be analysed to validate the text each time it is used in the exam and improved before it is used again.

## 6.5 Complete system overview

The experimental semi-automatic assessment system has been implemented as three components: the authoring tool, the diagram editor and the marking tool. If all three components are used in the assessment process, they enable the semi-automation of the assessment process.

Figure 6.10 illustrates the complete system view. The examiner uses the authoring tool to prepare the scenario texts and enter their diagrammatic solutions. These semantically controlled scenarios and their solutions are important for the automation as discussed in Chapter 5. Students choose one of the scenario texts in the system and use the diagram editor to enter their solutions. The editor doesn't only get the student solution but also captures the design histories of the students, which enable partial marking. After all students submit their solutions, the marking tool automatically marks the student work as much as possible by using the case-based reasoning method. The examiner uses the marking tool to mark unmarked components partially and can comment on the individual components. Finally, student diagrams with long reference diagrams may be marked by using a full marking style as discussed in Chapter 5.

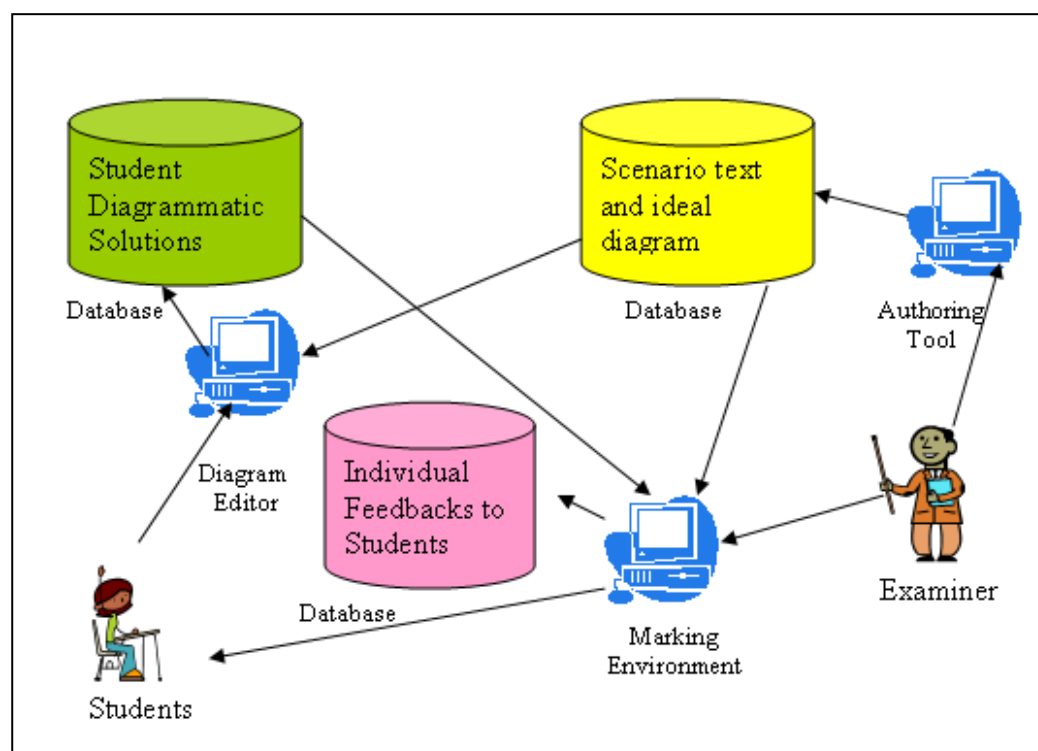


Figure 6.10 System view of semi-automatic assessment

The marker tool produces the personalised detailed feedback by using the examiner marks and comments after the marking completes. Students can analyse their feedback and may seek further clarification from the examiner. If students choose a scenario that the examiner has already marked then the student may get immediate feedback from the system. In this case, some components may be unmarked if they are different from the available marked components. Later, the examiner can comment on these unmarked components. For example, students get immediate feedback for their common mistakes during a tutorial session. The examiner marks the rarer mistakes in the student diagrams and then the fully marked diagrams can be made available the next tutorial sessions to discuss. This thesis calls this examiner feedback, “incremental feedback”. Separate, pedagogical research is needed to investigate the benefits of the incremental feedback concept for formative assessment.

Feedback is an important part of formative assessment. It makes learners aware of any gaps that exist between their desired goal and their current knowledge, understanding, or skill (Sadler, 1989). Feedback on tests and homework is most useful when it provides specific suggestions for improvement (Elawar and Corno, 1985). Nicol and Macfarlane in their paper (2004) presented a conceptual model of

formative assessment, feedback and the seven principles of good feedback practice. Although feedback is widely accepted as a crucial part in the learning for assessment, in higher education, Gibbs and Simpson (2004) argue that feedback to individual students in a class must have declined significantly as class sizes have increased. However, the semi-automatic assessment approach is independent of the class size, as discussed in Chapter 3.

Each component of the experimental assessment system can be used separately as well as all together for the conventional assessment process. The authoring tool assists the examiner to write scenario text for various domains by providing sample scenarios. The full marking style of the marking tool enables the examiner to mark diagrams online, even if the student diagrams haven't been captured by the diagram editor. The diagram editor assists students to follow the design methodology.

The marking tool can be potentially used in peer assessment as well. Students can mark student diagrams and the tool will highlight the marking inconsistencies between students. The examiner can resolve the inconsistency. The marking tool can also be used as an e-learning tool. In this case, students remark the marked components. Students' marks can be compared with the examiner's marks. The marking discrepancies are shown to students.

The feedback generated for the previous year student diagrams for a scenario can be used as a teaching tool. Students can explore this feedback. A toolset (Cooper and Macrae, 2003) has been developed to support the teaching of databases. It permits the student to select components of an ER diagram and to see the equivalent set of relational tables highlighted. The paper by Cooper and Macrae (2003) argues that allowing the student to observe the transformations being made facilitates understanding of those transformations. The feedback generated from the marking tool could show the link between the scenario text and the diagram components. Students could see the process of transforming scenario text into an ER diagram. They can learn from other students' mistakes.

## 6.6 Summary

This chapter describes the requirements of the semi-automatic assessment system and its components. It has mainly discussed the user-interface of the main system

components. Finding out the best user interface design is out of this thesis scope. The suggested user interfaces of the components are implemented in order to be able to test the semi-automatic approach. Chapter 7 explains how the components are used in the experiments. It summarises the findings.

All three components need to be improved and extended in order to increase the acceptability of the system by universities so they would adopt the system in a taught module. For example, the student diagram editor should be improved to be used as the teacher diagram editor so that the examiners can prepare the noun phrase list for the scenarios. At the moment, the noun phrase lists are hard coded into the scenarios.

A light version of the system has been implemented and used in taught modules as a part of funded project (Hinde et al., 2008). The light version supports only direct reference components. This version has a feedback mechanism. It gives colour coded feedback to students in the student diagram editor. The findings about the use of the tool in a class are in Chapter 7.



# CHAPTER 7

## Empirical Evaluation

### 7.1 Introduction

This chapter argues that the proposed semi-automatic assessment system of diagrammatic solutions is feasible, can be practically used in a taught module, and is useful. Following the introduction of the system's main components in the previous chapter, this chapter provides an evaluation of these components. Experiments are performed for each component independently and each result is discussed in a separate section.

The authoring tool is an important component for the system but it is not essential for the evaluation of the semi-automatic approach. It is included in this research in order to increase the automation of diagram marking. Writing similar scenario texts is the part of the authoring tool, which helps the automation. Section 7.2 provides the details of the required experiment and interpretation of the results. To evaluate the feasibility of the tool, this research measures the number of the volunteer teachers who successfully managed to use the tool to produce the scenario text.

The essential part of the system is the specialised diagramming editor. The system's reliability depends on this editor. Students should be able to use the editor correctly and enter their solutions into the system. Section 7.3 summarises the provisions that have been made for the experiments. In the experiments, students enter their diagrammatic solutions for the various scenario texts by using the editor. The student solutions are analysed and the findings are given in the same section. To evaluate the feasibility of the diagram editor, the research measured the number of the students who successfully used the editor to produce their solutions.

The other part of the system is the marking tool. It supports the semi-automatic marking process. The marking tool for the system can be fully functional if the student solutions are entered by the diagrammatic editor. Otherwise, the tool is used only as an online marking tool without any automatic feature. Semi-automation of

diagram marking relies on the human's ability to undertake partial-marking. Section 7.4 discusses the experiments performed for partial-marking. To evaluate the feasibility of the marking tool, the research measured the number of the users who successfully used the tool to mark diagrammatic solutions. To evaluate the usefulness of the tool, the research measured the number of components marked.

The last section of this chapter introduces the basic implementation of the semi-automatic assessment system, focusing on the feedback mechanism of the system and summarising the findings of the tool usage in a taught module.

## 7.2 Scenario Writing Environment

The scenario writing environment is introduced in Chapter 6. In the environment, the examiner uses scenario templates to write scenario text. Scenario texts written with the same template are semantically similar to each other and the use of these scenarios potentially increases the automation. How similar scenarios help the automation of the marking process is discussed in Chapter 5. This section provides the details of the experiment performed for the scenario writing environment.

Writing scenarios with a scenario template needs to be intuitive for the examiners. The developed tool aims to reduce the cognitive load on the examiner due to the template usage. The usability of its interface is not the prime concern of the research and is not explored in this experiment, however, an initial experiment is performed to see whether the examiners can use the tool with a brief introduction.

### 7.2.1 Provisions of the experiment

The participants chosen for the experiment were three lecturers who have taught the database design module in the past at university level. They had some experience of writing scenario text, but the experiment only needed to introduce the environment to the lecturer rather than going over the whole scenario writing process.

They were given an example scenario text about a "**course registration system**" written with a split scenario template. The template is used to write complex scenarios, which require the use of function buttons to get the correct diagram. If the examiner can write complex scenarios then they can also write simple scenarios. Thus only one of complex scenario templates (i.e. the split scenario template) was

used in the experiment. Since the experiment was designed as a proof of concept, not all types of complex scenario templates were tried out.

The participants were asked to write scenario text about an “**event management system**” with the same split template used for the “**course registration system**”. Figure 7.1 illustrates diagrams for each step of the split scenario template. The participants were expected to write a section for each step. Step 1 and Step 2 in the figure need text entry about the “**Event**” and “**Member**” entities. Step 3 then needs a text description about the relationship between these two entities. Step 4 and Step 5 expect a text entry about the “**Date**” multi-valued attribute from the participants. Step 6 expects the entered text to cause creation of a relationship between the unnamed entity and “**Member**” entity and deletion of a relationship between “**Event**” and “**Member**” entities. Step 7 expects a description about the “**Fee**” attribute of the unnamed entity. Step 8 shows the complete diagram. The diagram shows the unnamed entity as an entity X as well as all other named components so that the author can revise their scenario texts.

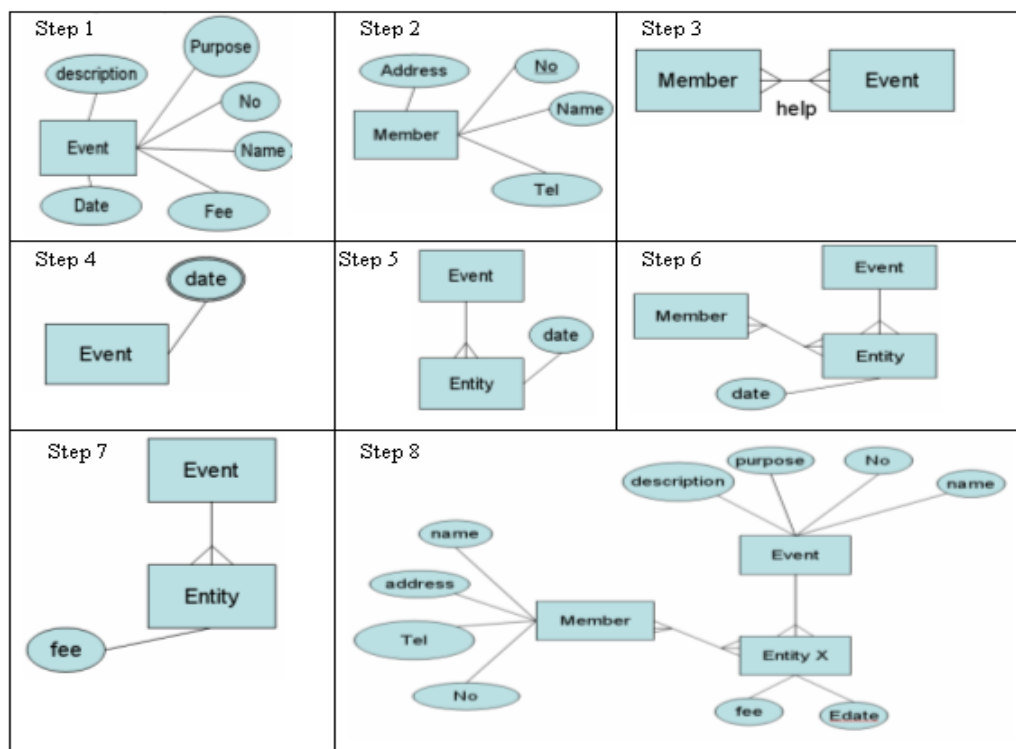


Figure 7.1 Split scenario template

The type of scenario template and the kind of participants used in the experiment were the main decisions made before the experiment. The experiment could be

repeated with different template types and with different kinds of participant, but this is unnecessary for the proof of concept. For example, the merge scenario template can be used instead of the split type and participants can then be asked to write the scenario text with the new template. Their feedback about the usability of the tool in slightly different circumstances could then be used to improve the presentation of the templates.

### 7.2.2 Results of the experiment

The participants in the experiment followed diagrammatic representation of the scenario template by using the tool and they wrote scenario text for each diagram in Figure 7.1. The text entry for each diagram is gathered and displayed as the whole scenario text to the participants at the end of the experiment so that they can modify their text.

All the participants wrote appropriate texts in each section page. At the end of the text entry, they all modified their complete scenario text in the production page, which is introduced in Chapter 6 alongside with the structure of the scenario editor. The tool allows this modification so that the authors can write more natural scenario text since separately written text pieces may be disjoint when they are put together. Although the tool doesn't have compliance check mechanism after modification, the participants' final texts are still compliant with the template and are natural text as opposed to text which is produced automatically.

The given scenario template expects the authors to implicitly mention one of the entities which is shown in Figure 7.1 as an entity *x*. The students who read the scenario text written with the template are expected to produce the entity *x* with their own label. All the participants successfully wrote the required text segment for entity *x*.

Only one of the participants' scenario texts, which complied with the template, is enough to prove that the use of the tool is possible. The experiment has shown that all the scenario texts are appropriate for the template. A paper by the author , (Batmaz and Hinde, 2008) briefly describes this finding as well as the scenario writing method and writing environment. The details of the scenario writing environment used for this experiment and the actual scenario text produced by the participant can be found in Appendix A.

### **7.2.3 Possible improvements for the writing environment**

The scenario texts produced by the help of the tool may not be compliant with the template used since the users can modify the text freely at the end of the scenario writing process. However the scenario texts can still be used in the semi-automatic assessment. They neither contribute to nor detract from the automation process. After these scenario texts are used in the real assessment, students' results can be analysed to modify the text. The author tool could be developed further to help the author analyse these results during editing of the scenario text which has been used previously.

The user interface of the scenario writing environment can be improved in order to decrease workload of the question preparation task and increase the quality of the scenario text. New experiments can be designed to evaluate the alternative user interfaces for this purpose. The thesis only shows the new research field of the computer aided assessment area. Computer aided assessment researchers may look into the requirements of the authoring tools for free response questions.

## **7.3 Diagram editor**

The diagram editor has two aspects. The first aspect is to capture contextual meaning of diagram components, which helps the examiner during marking. The second aspect is to provide an environment for the students to enter their design. To be able to implement these aspects, the editor has a very different environment from those of traditional diagram drawing tools. Experiments are designed to see whether the users can produce their design solutions by using the editor without being negatively affected.

### **7.3.1 Provisions of the experiments**

This section briefly explains the decisions made about the type of scenario text and the kind of participants used in the experiment.

The design solution can be affected by the user's poor design knowledge and skills as well as the editor's environment. In order to eliminate these negative effects, the experiments are designed to separate student design abilities and the editor usage skills. Two types of users are identified for the experiments: novice and non-novice

users. **Novice users** are students who are studying a database module at university level. They have basic design knowledge and very few design skills. **Non-novice users** are PhD students who have a computer science degree and took the database design module during their study. They have good design knowledge and sufficient design skills.

The scenario texts used in the experiments requires the use of the function buttons in order to get the correct diagrams. The function buttons in the environment are important for the approach. They make the diagramming process different from the traditional editor. Correct diagrams can be drawn for some scenarios without using any function buttons. This type of scenario is called a **simple scenario** and a scenario that requires the use of the function buttons is called a **complex scenario** in this section. The editors for KERMIT (Suraweera and Mitrovic, 2002) and VLE-ERM (Hall and Gordon, 1998) are designed for simple scenarios. Their editors are successfully used by novice users. Since simple scenarios are used in the experiments of their research, this establishes that simple scenarios are tractable, only complex scenarios are used in the experiments of this research.

### 7.3.2 The experiments

The complex scenarios could be hard for some novice users initially. They might need to practice more in order to gain the required design skills for the scenarios. The effect of the editor usage is measured by checking the user solutions. The number of wrong solutions is counted as the negative influence of the editor. If the user cannot get the solution correct because of the complex scenario, the environment of the editor will be seen as the reason. To separate the practical design skills of the novice users from the usage of the editor, two experiments were conducted. Since non-novice users are more comfortable with the complex scenarios, only one experiment was carried out with them.

#### 7.3.2.1 Experiment 1 for novice users

Firstly, twenty participants were given introduction sessions and shown how to use the editor on two example database scenarios. The given scenarios required the use of the split and merge function buttons in order to get a correct diagram. They were then asked to reproduce the solution for the same scenarios. Since the participants

know the correct solutions and their design process, they just needed to enter a design by using the editor. This experiment evaluates the use of the editor only.

### 7.3.2.2 Experiment 2 for novice users

The same novice users in the first experiment participated in the second experiment. This time they were given two new scenarios, which were similar to the ones used in the first experiment. The new scenarios required the same functions buttons used in the previous experiment to produce correct solutions. However, the new scenarios were written in different domains. This experiment was used to discover how practical design skills affect the editor usage.

### 7.3.2.3 Experiment 3 for non-novice users

Only the non-novice users participated in the third experiment. The experiment was similar to the first experiment. The participants were given a brief introduction session and shown how to use the editor using example database scenarios. Then the participants are asked to design two conceptual database diagrams for the scenarios used in the first experiment. Unlike experiment 1, participants weren't given the solutions for the scenarios. This experiment evaluated the use of the editor identically to the first experiment with novice users but with the assumption that the subjects have got good practical design skills.

## 7.3.3 Results of the experiments

Four database scenarios were used in the experiments. They are named as scenario 1, scenario 2, scenario 3 and scenario 4 in this section. The actual scenario text and their teacher solutions can be found in Appendix B.

Table 7.1 shows the number of participants in each experiments. Twenty people participated in experiment 1 and 2. Seven people participated in experiment 3. This section first interprets the result of experiment 1 and experiment 3 since they use the same scenarios. Then the section explains the results of experiment 2. All diagrams produced by the participants in experiment 1 and 2 can be found in Appendix C.

**Table 7.1 Number of participants in each experiment**

Experiment 1	20 Participants
Experiment 2	20 Participants
Experiment 3	7 Participants

### 7.3.3.1 Experiment 1 and 3

Experiment 3 was conducted before Experiment 1. In the experiment, the solutions for scenarios 3 and 4 weren't given to the non-novice users. All managed to draw the appropriate diagrams for both Scenario 1 and 2. Since the result was successful, the experiment was repeated with the novice users in Experiment 1. All the novice users also managed to produce acceptable diagrams for the same scenarios. They successfully used the editor to redraw the solution diagrams given to them.

The produced diagrams are not identical to the teacher solution for the scenarios. The diagrams have some missing or additional components. However, all the participants used the required functions buttons appropriately. They managed to apply both "split" and "merge" buttons to modify the initial diagram to get the correct components during the design. This result shows that the use of the editor's unique environments and the function buttons is possible.

In both experiments 1 and 3, the participants have to enter an entity name for the resulting entities after applying the required function buttons for the scenarios.

Table 7.2 shows the results for the Experiment 1. Scenario 1 in the experiment requires split button usage. Scenario 2 requires the use of the merge button. Ninety percent of the participants gave the same entity name for scenario 1 as the entity name in the teacher solution, which this research called the "ideal names". Seventy percent gave the ideal entity name for scenario 2. Some participants gave different names to their resulting entity for both scenarios as expected.

**Table 7.2 Student entity name for the experiment 1 out of 20 diagrams**

Scenario 1 requiring the split function button		Scenario 2 requiring the merge function button	
Entity name	Frequency	Entity name	Frequency
course offerings	90%	staff	70%
offerings	5%	acsupport	5%
coff	5%	staffmember	5%
		merge attributes	5%
		member of staff	5%
		people	10%



Entity names, which are different to the ideal names, are not a problem for the proposed approach since contextual information about the component is the main criterion for the entity match and this context is provided by the use of the function button. As the participants use the function buttons correctly, entities listed in Table 7.2 are accepted for marking. However, in some cases, names become important for the consistent marking purpose, which is discussed in section 7.4.6 on naming ambiguity.

### 7.3.3.2 Experiment 2

The participants in Experiment 1 also participated in Experiment 2. Although they didn't know the solutions for scenarios 3 and 4, the majority of them managed to draw the main part of the diagrams correctly. Table 7.3 shows the frequency of the correct and wrong diagram components for the scenarios. The table only focuses on entity and relationship components for simplicity. The teacher solution for scenario 3 has three entities and two relationships. One of the entities is a split entity created after using of the split button. 95 percent of the participants managed to have all three entities, implying that they decided to have a split entity and managed to use the function button correctly. 5 percent failed to have the split entity.

**Table 7.3 Student diagram components produced in the experiment 2**

Marking results	Frequency
<b>Scenario 3 requiring split function button</b>	
All three correct entities & both correct relationships	35%
All three correct entities & one correct relationship	55%
All three correct entities & no correct relationship	5%
two correct entities & no correct relationship	5%
<b>Scenario 4 requiring merge function button</b>	
Both correct entities & one correct relationship	90%
One correct entity & no correct relationship	10%

As for Scenario 4 in Table 7.3 the teacher solution has two entities and one relationship. One of the entities is a merge entity created after the use of the merge

button. 90 percent were successful and 10 percent were unsuccessful in creating the merge entity.

Participants, who failed to have the required entities for both scenarios, commented that they failed to design the system rather than failing to use function buttons. They did not wish to split or merge any entity. Therefore they didn't have any cognitive stress due to being unable to use the function buttons. In addition to the student comments, a paired t-test was performed to determine if the difference between the results of Experiment 1 and Experiment 2 is significant. In experiment 1, all students manage to have correct entities, whereas, in Experiment 2, 19 students for scenario 3 and 18 students for scenario 4 have correct entities. One-tail p is 0.102 at  $\alpha = 0.05$ . Therefore, there isn't enough evidence to support the claim that the use of function buttons prevents students from producing the correct diagrams.

For scenario 3, one of the relationships needs to be created between the split entity and the other entity in the solution. Table 7.3 shows that only 35 percent of participants had this relationship in their solutions. However, 90 percent managed to create other relationships between two entities for both scenario 3 and scenario 4. These results show that the participants know how to draw a relationship and the editor environment didn't stop them having the required relationship in their results. So the environment is not responsible for poor results, it seems that the scenario texts and the problems embodied in them are the reason. From this we can deduce that the participants found scenario 3 harder than scenario 4 for relationship creation.

### **7.3.3.3 The issue about the participants**

Participants in the experiments were student volunteers from the Databases module at Loughborough University. They are a self-selected group. After the module examination, the results were analysed to check how well the participant students represent all students in the module. The mean of students' exam results in experiments is 71 and the standard deviation is 14. The mean of all students' exam results is 51 and the standard deviation is 18. This shows that the student volunteers represent those near the top end of the class. This may have affected the results of the experiments since it doesn't represent the lower end of the class.

### 7.3.4 Analysis of entity names in the experiments

Like experiments 1 and 3, in experiment 2, the participants had to name a split entity for scenario 3 and a merge entity for scenario 4. Table 7.4 shows the names used in percentages for both split and merge entities. 75 percent of participants named the split entity differently from the others which means 15 different names since 20 students participated (i.e.  $15=0.75*20$ ). The rest of the participants gave two different names to the entity. Therefore scenario 3 has 17 different names all together.  $17/20$  gives a naming diversity percentage of the split entity which is 85%.

**Table 7.4 Student entity names for the experiment 2**

Scenario 3 requiring split function button		Scenario 4 requiring merge function button	
Entity name	Frequency	Entity name	Frequency
Event offerings	15%	Module	40%
Event repeats	10%	Option	10%
All different	75%	All different	50%
<b>Diversity</b>	<b>85%</b>	<b>Diversity</b>	<b>60%</b>

Table 7.4 shows that 50 percent of the participants (10 students) named the merge entity differently from each other. Scenario 4 has 12 different names for the entity. The naming diversity is  $12/20$  which is 60%. The average of the diversity for the experiment 2 is 72.5%. The average of the first experiment's diversity is 22.5% (i.e. the average of  $[3 \text{ different names for scenario 1}]/20 + [6 \text{ different names for scenario 2}]/20$ ). The naming diversity in the second experiment is much higher than in the first experiment.

There are many reasons for naming diversity. Some of the diversity can be controlled by the scenario text. For example scenario 4 mentions an optional module and a core module. The examiner uses the "option" noun phrase instead of "optional module" in the text. It is easier to see that the new entity is called "module" after merging the entities "optional module" and "core module". However, it is much harder to name the entity if the "option" noun phrase is used rather than "optional module". If the examiner uses compound noun phrases with a common noun in them, the user may find naming a merge entity easier. The proposed environment can be used for experiments in the future to check how compound noun phrases affect the students' results. Findings can be used to improve the guidelines for scenario writing.

Scenario 1 mentions that some of the courses are offered more than once. The split entity is named “course offering”. This name was suggested by the examiner during experiment 1. Scenario 3 mentions that some of the popular events are repeated. Some of the users named the split entity “event offering” and some of them “event repeat”. This shows that the users may have carried their previous experience into experiment 2. Students can improve their naming skills by doing exercises which use the same type of scenarios. Naming convention can be developed and students can be taught the conventions.

The percentage of the split entity and the merge entity names are also important in Table 7.4. It shows that 40 percent of the names chosen by the participants corresponded to the ideal name for the merge entity and 15 percent matched the ideal name for the split entity. This may suggest that the user found naming the split entity harder than the merge entity. More experiments for the use of functions buttons could reveal more interesting results. In particular the findings could be used to discover difficulty levels of scenario texts.

All the experiments were successful. The results of both experiment 1 and experiment 3 show that the users can produce design solutions using the proposed editor without being negatively affected. The results of experiment 2 show that a few users haven't the design skills required for the given scenarios. Additionally, student solutions were analysed to see how the users name the entities. The analysis discovered that students have some difficulties in naming the entities when there aren't any directly related noun phrases in the scenario text. The discovery suggested that scenario text can be categorised depending on the naming requirements and naming entities can be taught to students. Student design solutions from experiment 3 will be used in the next section for the marking tool experiments. More research is required on the difficulties of naming entities. Jayal and Shepperd (2008) have recently started looking into this area in order to use their findings in automatic diagram assessment.

## 7.4 Marking Tool

This section gives the details of the experiments done for the marking tool and the analysis of their results. Experiments were carried out to see whether the examiners can manage to use the tool for marking student diagrams. The environment of the

marking tool is used for two styles of marking: partial and complete marking. The complete marking is the online version of the manual marking process. The examiners are familiar with this marking style. Therefore an experiment for complete marking is not significant for the evaluation of the semi-automatic approach. However the usability test of its interface can be beneficial before it is used actively in a taught module. Since the usability of the system's user interface is out of the scope of the research, there were no experiments carried out for complete marking.

Partial marking is a new marking style and requires a very different interface from the complete marking one. It doesn't resemble manual marking. Experiments for partial marking are essential for the evaluation of the approach.

The section first explains the steps in detail, which the marking tool takes before the partial marking on the example data collected from the experiments in the previous section. Then it describes the experiments for the partial marking.

#### **7.4.1 Grouping stage and interpreting the results**

The semi-automation processes the student diagram before an examiner partially marks the diagrams. It groups the diagrams as described in Chapter 5. Table 7.5 shows the total number of components in both student and teacher diagrams produced for each scenario in Section 3. The main components of these student diagrams can be found in Appendix C. There are three component types and four scenarios in the table: entity type, relationship type and attribute type. Total entity numbers in each scenario are determined and shown as directly referenced and indirectly referenced entities. The table shows that teacher diagrams have one indirect referenced entity for each scenario and student diagrams together have twenty indirect components. Since twenty students participated in experiment 1 and 2, there aren't any redundant indirect referenced entities. The direct referenced entity column of the table illustrates that the student solutions have redundant entities. In the same way, the number of redundant attributes and relationships can be predicted in the early stages of the marking process. The later stages examine the acceptability of these redundant entities.

**Table 7.5 Components of solution diagrams for scenarios in section 2**

Scenario no	Total # of Comps	Entity		Attribute	Relationship
		Indirect	Direct		
<b>Student Diagrams</b>					
1	312	20	42	209	41
2	337	20	47	226	44
3	260	20	20	199	21
4	265	20	35	180	30
<b>Teacher Diagrams</b>					
1	16	1	2	11	2
2	15	1	2	10	2
3	12	1	1	9	1
4	12	1	1	9	1

The marking tool groups the student diagrams first by using their scenario references. Table 7.6 shows numbers of groups for the example data. The total column in the table shows the total number of component groups. For example, for scenario 1, there are 28 groups and for scenario 2 there are 55 groups. The group numbers shows how many components the examiner needs to mark using the tool.

**Table 7.6 Groups of student diagram components**

Scenario no	Total	Entity		Attribute		Relationship	
		Indirect	Direct	Indirect	Direct	D-to-D	D-to-I
1	28	1	4	2	18	1	2
2	55	2	8	10	23	4	8
3	60	4	3	38	9	6	0
4	107	5	7	42	34	13	6

The marking tool presents each type of diagram component to the examiner differently. First, it displays the entity type with their scenario reference and the

examiner marks them. Next the attribute type is shown. Attributes are shown with their entities, which have already been identified by the examiner. Since attribute marking requires displaying their entities, attributes are grouped by using their entities as well as their scenario references. The attribute column in Table 7.6 shows the number of attribute groups for each scenario. There are two types of attribute grouping depending on their entity types. If the attributes' entities are indirect referenced entities, then the numbers of groups are written under the "indirect" column heading. The attributes of direct referenced entities are written under the "direct" column heading.

The relationship groups for each scenario in Table 7.6 have been shown in two columns like the attribute groups. If the relationship groups are between two directly referenced entities then the numbers of groups are written under the "D-to-D" column. Relationship groups between direct and indirect referenced entities are written under the "D-to-I" column. Like the attribute type components, relationships are represented to the examiner with their participant entities after entities have been identified.

Table 7.7 shows the "total number of components" column in Table 7.5 and the "total number of component groups" in Table 7.6 together. These columns can be compared with each other to find out how effective the grouping is during the marking. For example, for scenario 1, the examiner would have marked 312 components during the manual marking whereas they will mark a maximum of 28 components by using the tool. A paired t-test was performed to determine if the difference between these columns is significantly high. One-tail  $p = 0.003$  shows that the grouping can significantly reduce the number of components marked by the examiner.

**Table 7.7 Diversity in student solutions for each scenario**

Scenario no	Number of components in the student solutions	Number of different components	Diversity in student solutions (%)
1	312	28	9
2	337	55	16
3	260	60	23
4	265	107	40

The diversity column in Table 7.7 shows how much the student solutions are different from each other for a particular scenario. The values in the diversity column are the number of components divided by the number of different components in the student solutions for each scenario. For example, for scenario 1, the number of components “312” is divided by the number of different components 28 in the table. The diversity is 9 percent for scenario 1. The diversity is 40 percent for scenario 4 which is highest among all the other scenarios in the table.

Detailed analysis of the teacher and student solutions reveals the reasons for the diversity. Table 7.8 shows the components of the teacher solution for the example scenarios. The teacher solution has 2, 2, 6 and 6 attributes of an indirect referenced entity for scenario 1, 2, 3 and 4. Table 7.6 shows that the student solutions have 2, 10, 38 and 42 different “indirect” attributes for the same scenarios. The value of the correlation coefficient for these data is 0.983. This suggests that there is a strong relationship between “indirect” attributes of the teacher solution and “indirect” attributes of the student solutions. Therefore the number of “indirect” attributes in the teacher solution is one of the reasons for the diversity.



**Table 7.8 Teacher diagram components**

Scenario no	Entity		Attribute		Relationship	
	Indirect	Direct	Indirect	Direct	D-to-D	D-to-I
1	1	2	2	9	0	2
2	1	2	2	8	0	2
3	1	1	6	3	0	1
4	1	1	6	3	0	1

The “indirect- attribute” column of Table 7.8 and Table 7.6 could be compared for two types of scenarios and the result may yield another reason for the diversity. For example, scenario 1 and 3 are “split” type scenarios and scenario 2 and 4 are “merge” type scenarios. The average ratio for the “merge” type is 4:20 (i.e.  $(2+6:2+38)/2$ ). The average ratio for the “split” type is 4:26 (i.e.  $(2+6:10+42)/2$ ). The comparison of these ratios shows the split action caused more reasoning diversity than the merge action. More student solutions for similar scenarios could be analysed in the future in order to check whether this finding can be generalised. The diversity ratios of the columns in Table 7.7 can be compared to assess the scenario text. According to the finding of that comparison, the scenario text can be altered in order to control the diversity.

### 7.4.2 Automatic component marking

Component matching operation follows after component grouping. The marking tool matches the component groups of students with the teacher components. Matched groups are considered to be automatically marked. At this stage, the number of automatically marked components is usually the same as the number of available teacher components in the system. Table 7.9 shows the numbers of matching components for each scenario. The system has only one teacher solution for each question. More teacher solutions in the system increase the numbers in “Total auto marked” column of the table.

**Table 7.9 Automatically marked components**

Scenario no	Student Comp Groups	Total Auto marked	Entity		Attribute		Relationship
			Indirect	Direct	Indirect	Direct	
1	28	15	1	2	1	9	2
2	55	15	1	2	2	8	2
3	60	11	1	1	6	3	0
4	107	12	1	1	6	3	1

Interpreting data in Table 7.9 may highlight the areas, which should be checked to improve the quality of the examination process. In the table, if the number of each component type is less than the number of the same teacher component type, it indicates that none of the students reasoned in the way the teacher did. For example, the indirect part of attribute column in Table 7.9 for scenario 1 shows only one attribute matched with the teacher's one in Table 7.8. However the teacher diagram has got two attributes, so the teacher needs to look into two areas. The examiner might revise their solutions. Although theoretically the teacher solution is ideal and correct, in practice it may not be correct. The other area at fault could be the scenario text. Some part of the text could have an ambiguous meaning which the scenario author didn't intend. In this case, the scenario should be modified.

Full automatic marking assumes that questions and their solution are prepared well and marking rules are complete, whereas the semi-automatic marking does not make these assumptions. The marking tool still works with the imperfect scenario and solution. The tool enables the examiners to improve their examination process.

### 7.4.3 Partial marking

The system could include more automation rules apart from the rule for the first matching operation at this stage. Chapter 5 discusses some rules and the possibility of generating new rules for automation. After this automatic marking stage or component matching stage, the system moves to a partial marking stage. In this stage, all unmarked component groups are presented to the examiner. Table 7.10 shows the number of these component groups to be marked partially by the human marker.

**Table 7.10 Component groups for partial marking**

Scenario no	Total		Entity		Attribute		Relationship	
	Short	Well-form	Indirect	Direct	Indirect	Direct	D-to-D	D-to-I
1	13	0	0	2	1	9	1	0
2	50	1	1	6	8	26	4	6
3	46	3	3	2	32	6	6	0
4	91	4	4	6	36	31	13	5

Component groups are presented to the human marker with their reference diagrams. Reference diagrams are either short or long. Short diagrams are for the direct referenced components. Long ones are for the components with indirect references. If the reference diagrams have got only one diagram action, then this research calls them well-formed reference diagram otherwise it calls them malformed reference diagrams. Malformed reference diagrams can have two or more diagram actions. Table 7.10 shows the total number of short and well-formed reference diagrams for each scenario. The table hasn't got any data for the malformed reference diagram. This means that all participants use, at most, one diagram action to get their components.

The marking process model developed in Chapter 5 marks the components with the malformed reference diagrams by using the complete marking interface since reading and interpreting malformed reference diagrams could be impractical and time-consuming. The model marks components with short or well-formed reference diagrams in the partial marking environment of the tool. The human marker should be able to mark short and well-formed reference diagrams partially in order to make the proposed system acceptable. The process of marking a reference diagram is slightly different for each component type.

#### **7.4.3.1 Marking a simple entity type**

Marking short reference diagrams for the entity type is a straightforward process. The reference diagrams have only one noun phrase reference. The examiner has three options for these short reference diagrams: reject, accept and map. The options are the same as in traditional marking. In the reject case, the noun phrase wasn't supposed to be an entity. For example students wrongly identified an attribute as an

entity. In the accept case, the noun phrase is an entity but it is not significant for the solution. In the last case, students had found a new reference for an entity of the solution. During marking, the examiner first decides whether the phrase is an entity or not. If it is an entity type then they check whether an entity is mapped to an entity of the teacher solution. In traditional marking, the examiner makes the decisions based on the whole student diagram. In the partial marking approach, the decisions are made, based on noun phrase references in the scenario text.

#### **7.4.3.2 Marking a simple attribute type**

Short reference diagrams for attribute components refer to only one noun phrase reference, like entity types. They are represented with the related entity to the human marker. The examiner not only uses the reference but also uses the related entity whilst marking. The human marker rejects the attribute if the noun phrase should be a different component type other than an attribute type or if the noun phrase should not be used at all. The human marker accepts the attribute if it is part of the teacher solution. Then they map the attribute to the attribute in the teacher solution. If related entities of both attributes are the same, the attribute is completely accepted. Otherwise, the attribute is partially accepted. Feedback is generated in this case. The human marker may accept the attribute even if it is not mentioned in the teacher solution. This could be the case if the examiner had forgotten to include the attribute in the solution or the attribute is insignificant for the solution. In both cases, Feedback is given to the examiner and students.

#### **7.4.3.3 Marking a simple relationship type**

Short reference diagrams for the relationship component have a sentence reference. Like attribute components, it is presented to the examiner with related entities. Relationships can have one or many participant entities. The examiner maps each relationship from the component group to a relationship in the teacher solution based on the sentence reference. After manual mapping, the related entities of the relationships are compared. If all are matched, the relationship is completely accepted. Otherwise it is partially accepted. If the human marker cannot map the relationship to any component in the solution, the relationship is rejected. However the relationship might be an unpredicted but acceptable one. Then the human marker adds the relationship into the solution. In both cases, feedback is given to the students and the examiner, as it is in the attribute marking process.

#### 7.4.3.4 Marking a relationship type

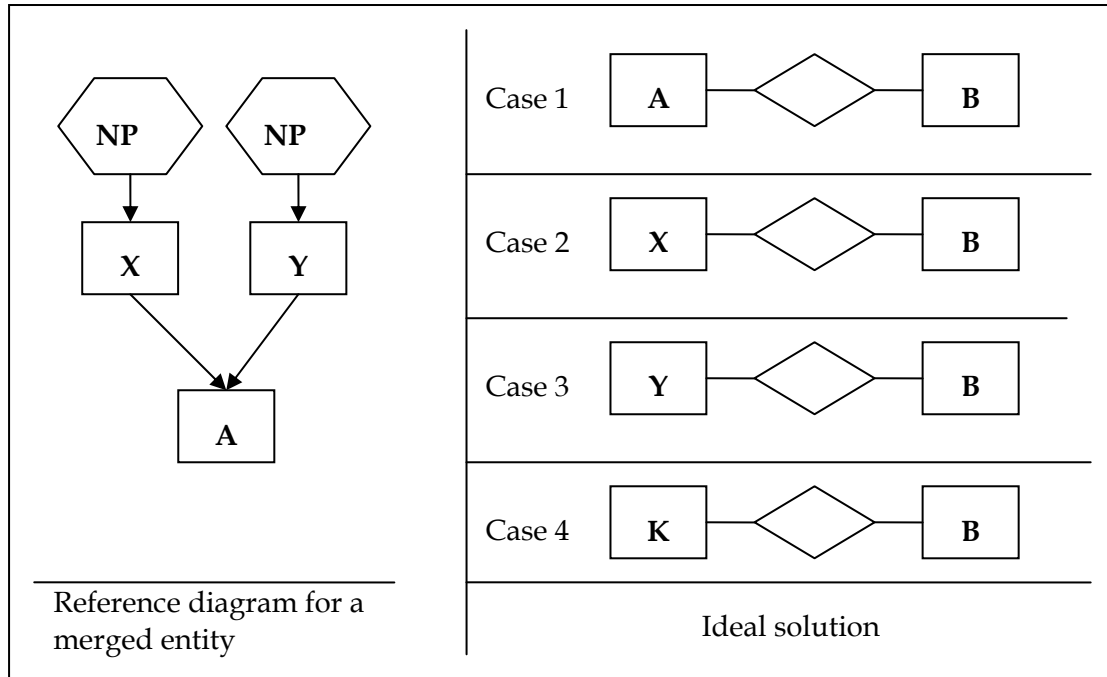
Relationship components may have well formed diagrams. Marking their diagrams is similar to marking the relationships with short reference diagrams. The only addition is that a relationship could be mapped to an entity which is created out of a many to many relationship. Both the entity and the relationship have sentence references. The examiner uses these references for mapping. Then participant entities of the relationship are automatically matched to the entities which have relationships to the entity. If the matching cannot be done, appropriate feedback is produced for this case.

#### 7.4.3.5 Marking an entity type with the well-formed reference diagram

Marking well-formed diagrams for entity type is an unusual marking process for the examiners. During the process, the examiners map an entity with the reference diagram to an entity of the ideal diagram. The mapping approves the references and it is added to the system as an alternative diagram. If the mapping cannot be done, then depending on the reference diagram, the examiners do different actions. Figure 7.2 shows possible mapping cases for a merged entity. Case 1 in the figure shows an ideal teacher solution. The examiner maps teacher entity A to student entity A. The entity A in the ideal solution may have a short or well-formed reference diagram. The examiner doesn't see the reference diagram of the teacher entity A. The examiner makes their decision based on the ideal solution.

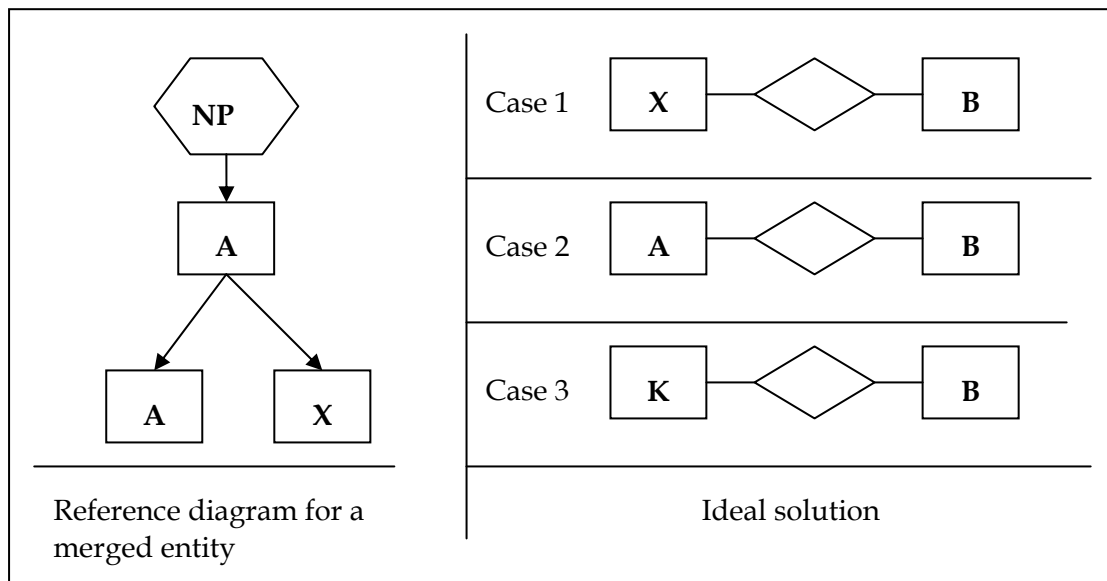
Case 2 and Case 3 show that Student entity A doesn't have a corresponding entity in the ideal solution but the entity X and the entity Y of the reference diagram have corresponding entities in the diagram solution. This happens when the students wrongly merge the entity X and Y which are supposed to be separate entities.

Case 4 shows that none of the entities X, Y and A in the reference diagram is mapped to an entity in the ideal solution. This case happens when students have found new entities. The examiner accepts them or rejects them without doing any mapping. Feedback is prepared for the results of this mapping.



**Figure 7.2 Mapping a merged entity to an entity in ideal diagram**

A well-formed reference diagram for a split entity is shown in Figure 7.3. Case 1 shows that students correctly extracted the entity X from entity A. Case 2 shows that entity X isn't part of the solution. The examiner accepts entity X if it doesn't violate any requirements, otherwise Entity X is rejected. In case 2, Entity A is valid and mapped to Entity A in the solution. Attributes of Entity A are handled during attribute mapping.



**Figure 7.3 Mapping a split entity to an entity in ideal diagram**

Case 3 happens when neither Entity A nor Entity X are part of the solution. Then the examiner decides first on the acceptance of entity A and later on entity X. If Entity A is rejected then Entity X is rejected automatically.

#### **7.4.4 Provisions of the experiment**

Both split entities and merged entities require an unusual mapping process. Apart from these entity types, the mapping processes of other components, which have short reference diagrams, is straightforward and similar to traditional marking. However in the traditional case, examiners are familiar with only the mapping components without using any references. This difference is not significant for components with a short reference diagram. Therefore the experiment was carried out for components with well-formed reference diagrams only.

The seven participants chosen for the experiment were people who have studied database design at university level rather than people who have some experience of marking. They were given an introduction session and shown how to mark components partially. The given components have indirect scenario references which cannot be automatically marked. Then the participants are asked to mark components with similar references.

#### **7.4.5 Results of the experiment**

All the participants but one managed to mark all components correctly. The failed person used a component label rather than the component reference to mark one of components. Although some components are purposely labelled indistinctly, which is the case in student diagrams, the participants managed to distinguish them without seeing the whole diagram.

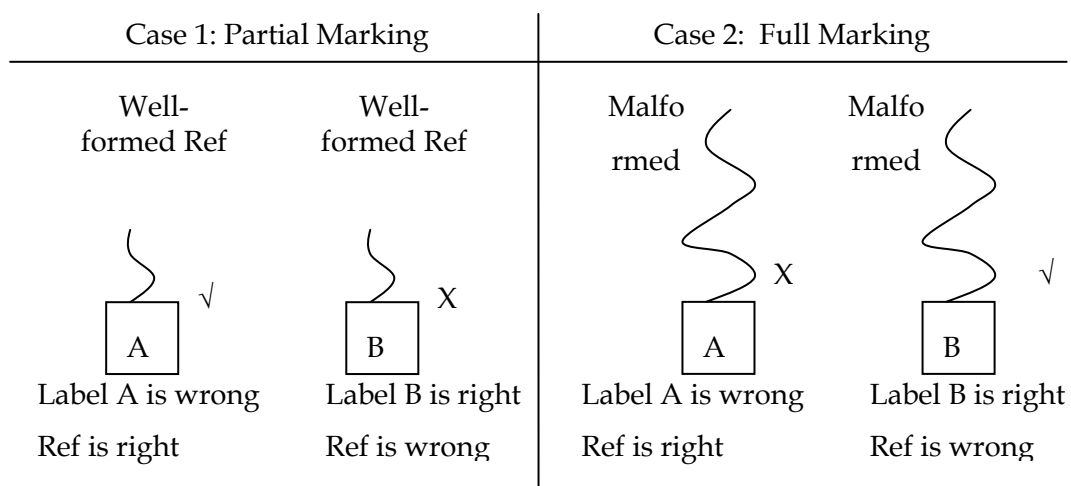
The system cannot prevent the examiner from marking wrongly. However, wrong marks can be spotted easily by students for correction since the system can provide mark-ups to students for their diagrams. The web-based environment enables more than one examiner to mark the same diagrams. Where there are many examiners, the system could detect any inconsistency between markers. This feature allows the marker tool to be used in collaborative assessment also.

### 7.4.5 Full marking

Diagram components, which have malformed reference diagrams, are marked during the full marking process straight after partial marking. Although the malformed references could be marked partially, examiners might find interpreting the references too complex and this negatively affects the acceptability of the semi-automatic marking system. No experiment was carried out for full marking of student diagrams since complete marking is an online version of traditional marking. The examiners are familiar with this type of marking. The experiment may be required for usability of the interface, however, designing the best user interface is beyond the scope of this research. Alternative user interfaces may be designed and tested in the future.

### 7.4.6 Naming ambiguity

The semi automatic approach requires two marking styles used together. This usage may occasionally cause some marking inconsistency in student marks because of the naming ambiguity discussed in chapter 4. There aren't any examples for this in the student solutions collected during the experiments. Figure 7.4 shows artificial examples. A student gets a correct reference but a wrong label for Component A and wrong reference but correct label for Component B.



**Figure 7.4 Inconsistency in marking styles**

During partial marking, the examiner marks student components based on the components' references. In Case 1, Component A is given a full mark and Component B is given no mark. During full marking, the examiner marks student



components without considering the references. This case makes Component A wrong and Component B right. These two marking styles give opposite marks for the same components.

There are two strategies to solve the marking inconsistency. The first strategy is that student feedback is given based on the partial marking. This implies that students' reasoning is more important than their final diagrams. According to partial marking, component A and B in Case 2 of Figure 7.4 are wrongly marked. Wrong marking of Component B is ignored since the mark favours the students. If it is necessary, the problem can be solved. The examiner is able to see the malformed references during full marking. However, this solution isn't adopted since it may increase the cognitive load of the marking. The wrong marking of Component B is corrected with the help of student feedback. After marking, the system gives personalised detailed feedback to students. Students see the malformed reference of Component B. If they spot their reference is correct then they can ask the examiner to mark this component manually again. The same solution is applicable to marking of component B in Case 1. Although Component B is marked correctly based on partial marking, students will consider the mark of component B to be wrong. Students are allowed to request their component to be marked again.

The second strategy to solve the marking inconsistency is that student feedback is given based on the full marking which is same as in traditional marking. Correctness of a student solution depends on their final diagrams. Hence Component A and B in Case 2 of Figure 7.4 are marked correctly. However Component B is marked wrongly. In this case, students will request that their components are marked again. Component A is marked wrong in the favour of students. This can be ignored since the student reasoning is correct. However student feedback may be confusing since they will see their diagram wrongly marked. To avoid this confusion, Component A can be relabelled by the system and given a matching name. This strategy requires less human involvement than the first one. Both strategies can be implemented in the system and the examiner can choose the first strategy to assess students reasoning ability or the second one to assess their final diagram only.

Semi-automatic marking focuses on the common, correct and wrong components in the solutions. The approach does not deal with the odd cases. Odd cases are marked by a human marker. Marking of some odd components can be time consuming for

the human marker and in this case, the system lets the students handle their own case by giving an opportunity to object, based on their feedback. This section explained how student solutions are marked with a real data example. It also described the experiments done for partial marking of two reference diagram type.

## **7.5 The semi-automatic assessment tool.**

The semi-automatic assessment research, reported by the author in a paper (Batmaz et. al., (2009) has developed a new complete assessment tool. The tool is the light version of the prototype tool described in the thesis. This section briefly explains the new tool and then gives the findings of the tool's use in a taught module.

The assessment tool has a new feedback component as well as the marking and the diagramming components. The feedback component is integrated with the diagram editor so that the students can see and analysis their feedback by using the diagram editor.

The diagram editor supports only simple scenarios. Students can only create a diagram component which has a direct reference. Students are allowed to name the component by using only noun or verb phrases in the scenario text. This prevents any naming ambiguity as discussed in section 5. The diagram editor has a robust, simple, drag-and-drop user interface. Figure 7.5 shows a snapshot of the user interface.

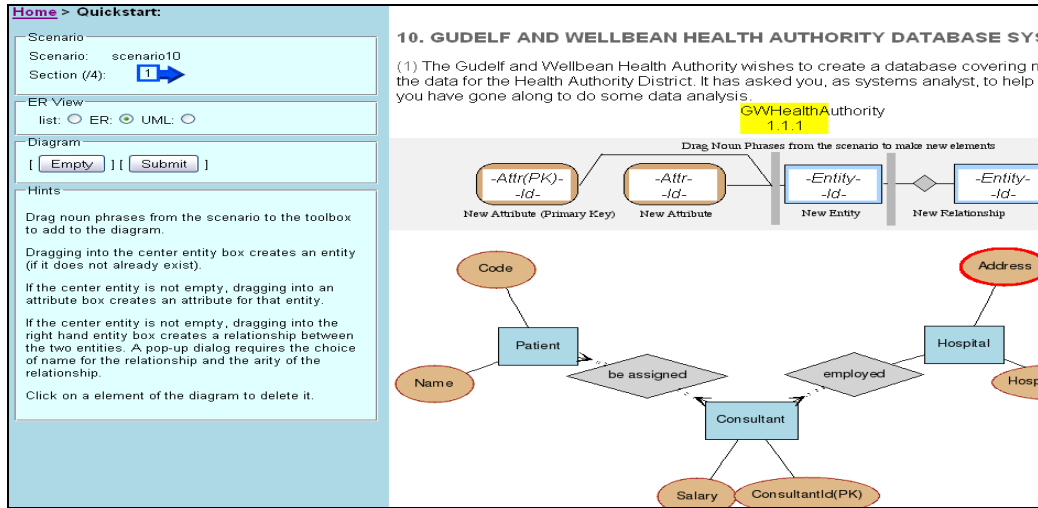


Figure 7.5 The user interface of the diagram editor

Students draw diagrams without using the keyboard since they don't type the component names. This feature allows the editor to be used on a touch screen easily without changing the user interface. Further discussion of the interface can be found in the paper by Stone et al (2009).

The editor additionally shows the colour coded feedback about their diagrams if the student diagrams are marked. Figure 7.6 shows screenshots of a student diagram before and after marking. Students can also read textual comments given by the examiner on each component in their diagram.

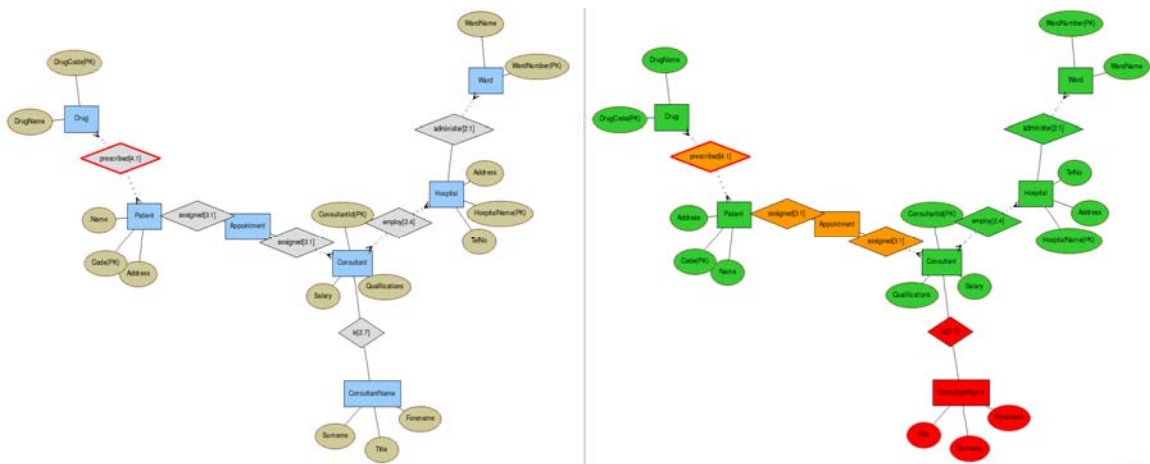


Figure 7.6 A student diagram with feedback (Stone et al. 2009)

The marking environment of the assessment tool supports only the partial marking style. The full marking style is not supported since the components of the student diagrams can only have direct references.

### 7.5.1 Using the tool and results

The diagram editor has been used with two scenarios by a first year class of 200 students in 2009. This represents 4 separate sessions since the first year class had to be split into four groups of 50 students. At the beginning of each 50min practical session a short demonstration of the system was given showing how to make one entity, one attribute and one relationship. The students were then asked to create ER diagrams for the first scenario (hitherto unseen) and, if possible, go on to do the same for the second scenario. By the end of the session most students had finished the first scenario and some had finished both.

**Table 7.11 Summary of diagram marking for two scenarios**

Scenario No	Total Component #	Component Group #	Diversity Rate %	Auto marked Group #	Manual marked Group #	Efficiency Rate %
1	5356	708	13	468	240	96
2	3707	607	16	317	290	92

The examiner marked the student diagrams using the marking tool in less than two hours. Table 7.11 shows a summary of the marking task. The tutor marked 240 components out of 5356. That makes the efficiency of the tool for the first scenario approximately 96 percent. The efficiency for the second scenario is 92 percent. The difference between 96% and 92% could be due to noise or this could be interpreted such that the system's efficiency increases when more students draw ER diagrams for the same scenario. However, to draw the conclusion about the relationship between the system efficiency and the number of students, more experiments are needed.

The table also shows the diversity rate. This rate is calculated by using both component group and total component numbers. If student diagrams are similar to each other then the diversity rate decreases, otherwise it increases. The diversity rate could be used as feedback for the scenarios. Student solutions for the similar scenarios should have the same rate. If they are very different, the tutor may analyse and revise the scenario text to find out the reason for the diversity.

The marking tool produces a detailed report for any chosen scenario. The report has a list of every distinct element, how it was marked and the number of students

whose diagrams included that element. This reveals, for example, how many students made the 'same mistake'. So, for example, it was clear that something in the way the first scenario was worded caused 50 students to wrongly identify "Consultant Name" as an entity and go on to make related mistakes with attributes. The report contains aggregate marks e.g. entities (green 82%, amber 7%, red 11%) attributes (green 82%, amber 10%, red 8%) and relationships (green 34%, amber 59%, red 7%) showing that it is the precise identification of relationships (amber 59%) that caused the most problems.

In feedback sessions, the students were able to see their marked diagrams. The colour coding of tutor comments was extremely well received by the students and led to lively, positive discussion of the principles involved with interpreting the scenarios which was very beneficial. A simple questionnaire about the editor and the associated marking feedback was given to the students at end of the term. The results from 67 returns show that the students were favourably disposed to the editor and they liked the coloured feedback. The questionnaire given and the result can be found in Appendix D.

Students have accepted the concept of using the editor as a way of submitting their work. They receive more detailed personalised feedback than the feedback from traditional methods. The marking tool has decreased the number of diagram components the examiner needs to mark.

The current tool is a basic system where the components of the diagram are directly referenced to the scenario. The next stage is to improve the interface so that the user can create indirect referenced components by splitting and merging existing elements.

## 7.6 Summary

This chapter discussed the evaluation of the proposed semi-automatic system in the thesis. Three components of the system were investigated separately. The first component of the system is for scenario writing. It was used by lecturers and they produced natural scenarios based on a given template. It has verified that the use of the scenario writing environment is possible. The second component of the system is for diagram drawing. It was used by students. The students produced their

solutions based on question scenarios. The scenarios are specially written by using templates in order to increase the automatic marking. The student solutions have shown that the diagram editor can be used properly in order to produce appropriate solutions for the scenario. The last component is for marking diagrams. Experiment results have shown that the participants managed to mark complex student solutions partially. It demonstrated that the new partial marking style does not cause any major difficulties.

Each component needs to be improved in order to be used in a real assessment situation in the future. Some parts of scenario writing environment should be automated. It could suggest some sentences based on a chosen template and components in the template can be labelled automatically. This will increase the acceptability of the tool. The environment will also enable the author to modify current templates or to create their own templates after getting feedback from the marking editor.

The diagram editor has been shown to be usable by students. However, experiments revealed that the automatic diagramming section of the editor needs to be improved. The algorithm used in the editor automatically draws student diagrams in order to give the best layout for the diagrams. This sometimes makes significant changes to the layout of diagram and makes students confused. A suitable algorithm needs to be developed which automatically draws diagrams with the minimum changes in their layout each time students add new components.

The naming ambiguity problem is highlighted in section 3. A naming convention is suggested as a solution in order to improve the quality of the student diagrams. The same problem is revisited in the section 4 since it may cause some inconsistency in marking. Two different solutions are suggested in that section.

A basic implementation of the semi-automatic system uses an additional component for giving appropriate feedback to students and the teacher. The feedback component should be improved so that it can use the examiner marks from the marking editor and grade them based on marking schemes. It should also explain how their marks are given.

The experiments with the tool have not focused on the “ease of use” aspect of either the drawing or the marking tool, so the interfaces need to be made more user-friendly before the system starts being used in a taught module.

The next chapter gives the conclusion of the thesis. It summaries the research contributions and provides the future directions of the research.

# CHAPTER 8

## Conclusions and Future Work

### 8.1 Introduction

Diagrams are increasingly used in many design methods, and are being taught in a variety of contexts in higher education such as database conceptual design or software design in computer science. They are the key part of many assessments. Personalised and detailed feedback to students is very important for the formative assessment. The increasing number of students in HE increases the assessment load of lecturers. This thesis proposed an advanced solution to the process of assessing student diagrams. The solution uses human-computer collaboration by providing a semi-automatic assessment environment.

This chapter gives a brief review of the thesis, summaries the achievements and points out some directions for future work. It is organised as: 8.2 gives a review of the thesis, 8.3 summarises the major contributions and 8.4 outlines the limitations of the thesis with some future directions.

### 8.2 Thesis Review

The research aims to develop a semi-automatic assessment framework which enables the use of a computer to support the assessment process of diagrammatic solutions, with the focus of ensuring consistency of feedback on the solutions.

To achieve the research aim, the following three objectives were set in Chapter 1:

Objective 1: To identify the repetitive tasks in the assessment process.

Objective 2: To develop techniques to reduce the repetitive tasks or remove them completely where possible.

Objective 3: To develop a novel framework that provides a platform where different intelligent techniques work together to support the assessment process of diagrammatic solutions.



The underlying framework of existing computer aided assessment research was identified in Chapter 2. Available technologies in computer aided assessment area were studied, which included marking students' essay-type work and software code as well as diagrammatic solutions. Some fully automatic diagram marking research (Hall and Gordon, 1998) (Suraweera, 2001), which uses a constraint based reasoning technique (Freuder and Mackworth 1994), was discussed in detail. A semi-automatic approach for assessment of diagrammatic solutions was decided for this research to deal with complex question types and to improve the consistence of the marking and feedback.

A conventional, manual diagram-assessment process was studied and the problems of the process were identified in Section 3.2 in Chapter 3. The repetitive tasks were identified in Section 3.3.1 and Section 3.3.4 to achieve Objective 1 of this research. A substantial amount of lecturers' time in higher education is occupied with preparing questions for diagram-type work and marking student solutions when the number of students increases. In the assessment context, the process of marking a diagram means that components of the teacher and student solutions are matched against each other. If some computer support is provided to the diagram matching process, then it has the potential to shorten the assessment time and improve the consistency of the human markers' grading and feedback. Thus, the thesis focused on computer assisted diagram matching. The thesis also proposed to modify the authoring of the questions in order to improve the marking stage of the process.

The limitations of automatic marking research for the assessment were identified in Section 2.4 in Chapter 2. Case-based reasoning (Kolodner, 1993) was proposed as a replacement of constraint based reasoning in order to overcome the shortcomings in Section 3.3.3 in Chapter 3. Case-based reasoning was employed to achieve Objective 2, which reduces the repetitive marking task of the assessment process. The requirements of the proposed approach were identified and discussed in Section 3.3.2, including (1) contextual attributes of diagram components, (2) controlling the reasoning diversity, (3) manual partial marking and (4) automatic partial marking. The contextual attribute was the most important requirement for the semi-automation while developing the ability for partial marking became the major challenge of the thesis. A novel framework was proposed for the approach in Section 3.3.3 to achieve Objective 3. The framework provided a platform to enable

the integration of a number of the technologies for the assessment of student solutions for various types of scenario texts. Detail of the framework was developed in Chapter 4 and Chapter 5.

To support this framework, the research area of requirements traceability (Ramesh and Jarke, 2001) was studied and a novel trace model was proposed which defines the design traces and production of student design traces for the contextual attribute of diagram components in Chapter 4. The concept of an action function was developed for automatic online trace capturing. The use of action functions was designed to enable users to modify their diagrams at a higher level during design and enable the examiner to capture the students' reasoning and self-explanations.

A novel marking process model was proposed in Chapter 5 for the developed design trace model. The model prescribes five important sages: (1) the segmentation of student diagrams, (2) segment grouping, (3) automatic marking of the possible segments, (4) manual marking of the remaining segments, and (5) assembling the marked segments. The problems of manual partial marking were identified and tackled successfully by introducing manual full marking alongside the partial marking. A case definition was created for automatic marking. The concept of generic case was developed for the case adaptation phase of the CBR cycle. The requirements of generic case production were discussed. A set of guidelines was suggested for writing scenario texts in order to increase the use of generic cases in Section 5.3.

Components of scenario templates for writing similar problem scenarios were identified and discussed, which include (1) scenario sections, (2) statement types, (3) existing and alteration conditions, (4) condition satisfiers, and (5) sentence structure. Some scenario templates for scenario text writing were developed and demonstrated, which required action functions to be used to reach the correct design solution. Using scenario templates was proposed to also control the difficulty level of scenario based questions.

A rudimentary scenario writing editor was developed for proof of concept testing in Section 6.4 in Chapter 6. Three volunteer examiners used the editor to write scenario text which was compatible with the given scenario template. The results showed that the examiners could follow the guideline which enables the writing similar scenario texts.

A prototype diagram editor, which is based on automatic graph drawing (Tamassia, et al, 1988), was developed Section 6.2 in Chapter 6 to evaluate the proposed design trace model. Three case studies were performed in Section 7.3 in Chapter 7. Students were given a different scenario text for each case. The first and second scenario texts were written in such a way that the student had to use one of action functions to express their design and the third scenario makes the uses of action functions optional. The first two cases were designed to see whether the cognitive load of the editor was acceptable. The third case was designed to find out the diversity in students' reasoning. The first two case studies showed the cognitive load of the editor is not high since they all managed to get the correct solutions. The study of student solutions in the third case showed that the variety of student reasoning is limited and there is common reasoning among student solutions. Student solutions were successfully segmented and grouped, based on their reasoning, by using the proposed case definition.

A prototype marking editor was developed Section 6.3 in Chapter 6 to evaluate the proposed partial marking technique. Three case studies were performed in Section 7.4 in Chapter 7. The human markers were given three reference diagrams as a contextual attribute of the components. Each reference diagram contained one action function. The cases were designed to see whether the representation of design traces is understandable and correctly used by human markers and that the cognitive load of it is acceptable. The results showed that partial marking is possible and the cognitive load is not too high. The partial marking enables consistency of grades and feedback, a fair application of the marking scheme, and integration of intelligent support in the assessment process. These three features are critical in the manual assessment process in the absence of a full automatic assessment system.

### 8.3 Summary of Contributions

This section reiterates the contributions of the thesis mentioned in Chapter 1. Additionally, references to the related chapters are given for each contribution.

- Through the application of assessment in the diagrammatic solution domain, it has contributed to an enhanced understanding of "semi-automatic assessment". The concept of reducing repetitive tasks in the question preparation and marking stage of the assessment process is

introduced to the semi-automatic assessment research of diagrammatic solutions, which is necessary to increase the quality and consistence of feedback given to students. The discussion about the semi-automatic approach can be found in Section 2.4 and Section 3.3.

- The new assessment framework is proposed, which gives a platform where a variety of technologies can be used to increase automation of the assessment of diagrammatic solutions. Increasing the automation is desirable in order to make the framework more acceptable in the assessment community. The automation part of the framework achieves Objective 3 identified in Chapter 1. The framework adds a new automation approach to the list of current computer aided assessment research discussed in the Section 2.2.2. The detail of the framework can be found in Chapter 6. The complete system view of the framework is given in Section 6.5.
- A novel trace model is developed, which is a part of the proposed framework and necessary for the semi-automation.. The model captures design traces of student solutions and enables construction of contextual information of components. The model contributes the achievement of Objective 3. The trace model is developed in Chapter 4.
- A new generic case concept is defined, which enables scalable adaptation rules. It contributes to the case-based reasoning method by defining a new way of indexing natural language text, which is a question text describing the system requirements. The generic case concept is used as a part of the proposed framework in order to achieve Objective 3. The generic case concept is introduced in Section 5.2.
- The novel partial marking style of student diagrams creates a new research direction in the computer aided assessment community and adds a new type into the list of computer aided assessment types mentioned in Section 2.2.1. The partial marking style is essential for the marking process model used in the proposed framework. The partial marking style is explained in Section 3.3.3.

- A novel marking process model is developed, which integrates full and partial marking styles. The process model describes all steps needed to follow during the marking in the proposed framework. The process model ensures the consistency of the feedback given to students which is the part of the thesis's aim. The model can be seen in Section 5.4.
- The thesis contributes to the online assessment area by presenting the requirements of a new online diagram marking tool. The tool allows component-based marking as well as diagram-based marking, which is necessary for the marking process model. The requirements can be found in Section 6.3
- A set of guidelines for writing question text is introduced for diagrammatic solutions. The use of the guidelines is not mandatory for the proposed framework. However they increase the automation. Besides the use of question text written based on the guidelines enables formative assessment. Section 3.2.1 highlighted the importance of the feedback in formative assessment. Section 3.3.2 underlined the benefit of the proposed framework for formative assessment. Section 5.3 gives the details of the guidelines.

## 8.4 Limitations and Future Directions

The following issues are currently being addressed, or should be addressed in future work:

### 8.4.1 Marking based on design traces

The manual partial marking uses students' design traces. The design traces allow naming discrepancy as discussed in Chapter 7. The proposed framework should include a name checking mechanism to be able to use in summative assessment. When students name any component wrongly, the editor sends a message alert. One possible solution is to use an ontology for each question. To avoid increasing question setup cost, the ontology could be automatically created. If the questions previously used in the formative assessment are asked in the summative assessment, the ontology for the questions could be generated automatically from the names used in the student solutions.

### **8.4.2 Handling multiple graph-based diagram types**

This thesis focused on the entity-relationship diagram type. It used ER diagrams to evaluate the proposed framework. The framework can be applicable to other graph based diagram types (e.g. DFD, Class Diagrams). Action functions for the proposed design trace model should be checked to see whether they are sufficient for the new diagram types. New action functions may be defined when it is necessary or suggested.

A meta-process model can be developed for the adaptation process of the design trace model to the new diagram type.

### **8.4.3 Feedback generation**

Partial marking made the use of a detailed marking scheme possible. However, the marking scheme and standard feedback have been left for the examiners to prepare. A simple feedback presentation has been used in the prototype editor. An alternative feedback presentation could be developed to use the full potential of design traces. The feedback could be interactive so that students may enquire about their design actions. This feedback can even be used as a teaching tool. Students may learn from their own or other student's mistakes.

### **8.4.4 User interface design**

A prototype diagram editor was developed for the online production of design traces. The editor is based on automatic graph drawing to reduce the cognitive load of action functions. A prototype marking environment uses mouse and screen instead of pen and paper. The project has not focused on the "ease of use" aspect of either the drawing or the marking tool. Separate research (Stone et al., 2009) has already started to develop a more user friendly interface to the editor. The use of single touch and multi touch screens has been under investigation to track students' design activity more naturally.

### **8.4.5 Deep-knowledge assessment tool**

Action functions are used to record the design activities. They are sufficient to give the contextual information about the components. The action function can also be used to obtain detailed self-explanations explicitly from the students. The self-

explanation can be used for the assessment of students' deep-knowledge. The action function can be extended to identify the causes of actions and the decision rationale. When the action button is used, an appropriate explanation list could be displayed and students could select one of them.

#### **8.4.6 Extensions to the guideline of writing scenario text**

The scenario writing guideline is proposed to ensure having a number of similar scenario texts in order to produce generic cases, which increase the automation. The guideline could be extended to create more scenario templates. Student solutions for scenario texts used previously could be analysed to find out more factors which affect student reasoning. The analysis could also reveal the difficulty levels of question text and the templates. The students could be given scenario texts ranging from easy to hard as a formative assessment.

#### **8.4.7 New application areas of the partial marking style**

The partial marking style could be investigated in its application to other assessment domains (e.g. essay, program code). For example, students could be asked to first produce a mind map diagram and then write their short essays which link to components of the mind map. For assessment of program code, they could be asked to first draw a flowchart or activity diagram and then write a small program which links to components of the diagram. Later on, student solutions (essays or program code) could be grouped, based on the corresponding diagram components. The examiner would mark solution segments from each group. For each solution type, a new question authoring, solution editor and marking tool needs to be designed. If the diagram type for the assessment domain doesn't exist then a new diagram notation could be developed. For example, a problem solving diagram for algebra questions could be developed to assess the student solution steps instead of just assessing the final answers.

### **8.5 Overall Conclusion**

This thesis has proposed an advanced solution to the assessment process of diagrammatic solutions. The semi-automatic assessment system required for the solution was developed and evaluated. The evolution results of the proposed system

in Chapter 7 showed that the research achieved all its objectives and successfully met its aim. The system successfully tackled the main problem areas of semi-automatic assessment. The developed manual partial marking style creates a new research direction in the assessment community, and its application in diagrammatic solutions indicates the importance of contextual information about the diagram components.



## REFERENCES

Aamodt, A & Plaza, E 1994, 'Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches'. *AI Communications*, IOS Press, vol. 7, no.1, pp 39-59.

Achour, CB 1998, 'Writing and Correcting Textual Scenarios for System Design', *Proceedings of the Natural Language and Information Systems (NLIS'98) Workshop*, Vienna, Austria, pp.166-170.

Arnold D, Barshay O 1999, 'On-line programming examinations using Web to teach', *Proc. of the 4th annual SIGCSE/SIGCUE on Innovation and Technology in Computer Science Education*, ACM press, krakow, poland, pp. 21-24.

Bancroft, P, Hynd, J, Reye, J & Dal Santo, F 2003, 'Web-based assignment submission and electronic marking'. *Proc. of the HERDSA Annual Conference*, Christchurch, New Zealand, pp.41-52.

Batini, C & Lenzerini, M 1989, 'A Comparative Analysis of Methodologies for Database Schema Integration', *ACM Computing Surveys*, Vol. 18, No. 4, pp.323-364.

Batmaz, F & Hinde, CJ 2008, 'A Method For Controlling The Scenario Writing For The Assessment Of Conceptual Database Model', *Proc. of Computers and Advanced Technology in Education*, ACTA press, Calgary, Canada, pp.614-804.

Batmaz, F, Stone, R, & Hinde, C 2009, 'Personalised Feedback With Semi-Automatic Assessment Tool For Conceptual Database Model', *Proc. of the 10th Annual Conference of Information and Computer Sciences*. Higher Education Academy Subject Centre, University of Ulster, UK, pp.115-120.

Batra, D, & Davis, J 1992, 'Conceptual data modelling in database design: similarities and differences between expert and novice designers', *Int. Jour. Man-machine Studies*, vol.37, no.3, pp.395-417.

Bligh B 2002, *Automatic Assessment of Diagrams: Feasibility Report*, University of Nottingham, Nottingham, UK.

## References

- Bloom, B 1956, *Taxonomy of Educational Objectives*, David McKay Company, New York, USA.
- Bohner, SA 1991, 'Software Change Impact Analysis for Design Evolution', *8th International Conference on Software Maintenance and Re-engineering*, IEEE CS Press, Los Alamitos, CA, pp. 292-301.
- Boyle, A & O'Hare, D 2003, 'Assuring quality computer-based assessment development in UK higher education', *7th International CAA Conference*, Loughborough University, Loughborough, UK.
- Boyle, A, Hutchison, D, O'Hare, D & Patterson, A 2002, 'Item selection and application in higher education', *Proc. of 6th International CAA Conference*, Loughborough University, Loughborough, UK, pp.269-281.
- Brown, S & Race, P 1996, *500 Tips on assessment*, Cogan Page, London, UK.
- Brown, S, Race, P & Bull, J 1999, *Computer Assisted Assessment in Higher Education*, Kogan Page, London, UK.
- Brusilovsky, P & Higgins, C 2005, 'Preface to the Special Issue on Automated Assessment of Programming Assignments', *Journal of Educational Resources in Computing*, vol.5, no.3. pp.1-3.
- Brusilovsky, P & Sosnovsky, S 2006, 'Individualized exercises for self-assessment of programming knowledge: An evaluation of QuizPACK', *Journal of Educational Resources Computing*, vol.5, no.3, Article no.6.
- Bull J 1999, 'Update on the National TLTP3 Project: The implementation and evaluation of computer-assisted assessment', Keynote, *Proc. of 3th International CAA Conference*, Loughborough University, Loughborough, UK, viewed on 15<sup>th</sup> september 2006, <<http://www.lboro.ac.uk/service/ltd/flicaa/conf99/pdf/keynote.pdf>>.
- Bull, J & Collins, C 2002, 'The use of computer-assisted assessment in engineering: some results from the CAA national survey conducted in 1999', *International Journal Of Electrical Engineering Education*, vol.39; no.2, pp.91-99.
- Bull, J & McKenna, C 2004, *Blueprint for computer-assisted assessment*, Routledge Falmer, London, UK.

## References

- Bunt, A, Conati, C & Muldner, K 2004, 'Scaffolding self-explanation to improve learning In exploratory learning environments', *Proc. 7th ITS*, LNCS 3220, Springer-Verlag, Heidelberg, Berlin, Germany, pp.656-667.
- Burstein, J, Kukich, K, Wolff, S, Lu, C, & Chodorow, M 1998. 'Enriching automated essay scoring using discourse marking'. *Proc. of the Workshop on Discourse Relations and Discourse Markers*, Annual Meeting of the Association of Computational Linguistics, Montreal, Canada, pp.90-97.
- Canup, MJ & Shackelford, RL 1998, 'Using software to solve problems in large computing courses', *SIGCSE Bull*, vol.30, no.1, pp.135-139.
- Cerbah, F & Euzenat, J 2001, 'Using terminology extraction to improve traceability from formal models to textual requirements', *Lecture notes in Computer Science no.1959*, Springer-Verlag, Germany.
- Charman, D & Elmes, A 1998, *Computer Based Assessment: A guide to good practice-Volume I*, SEED Publications, Plymouth, UK.
- Chen, PP 1976, 'The Entity-Relationship Model - Toward a Unified View of Data', *ACM Transactions on Database Systems*, vol.1, no.1, pp. 9-36.
- Cheng, P, & Lowe, R, & Scaife, M, 2001, 'Cognitive Science Approaches To Understanding Diagrammatic Representations', *Artificial Intelligence Review*, Vol.15, no.1, pp. 79-94.
- Chi, MTH, Bassok, M, Lewis, M, Reinmann, P & Glaser, R 1989, 'Self-Explanations: How students study and use examples in learning to solve problems', *Cognitive Science*, vol.13, no.2, pp.145-182.
- Christiansen, H & Have, CT 2007, 'From use cases to UML class diagrams using logic grammars and constraints'. *RANLP '07: Proc. Intl. Conf. Recent Adv. Nat. Lang.*, Borovets, Bulgaria, pp.128-132.
- Conati, C & VanLehn, K 2000, 'Toward Computer-based Support of Meta-cognitive Skills: A Computational Framework to Coach Self-Explanation', *International Journal of Artificial Intelligence in Education*, vol.11, no.4, pp. 389-415.
- Conklin, J 1989, 'Design rationale and maintainability', *Proc. 22nd International Conference on System Science*, IEEE Computer Society, Kailua-Kona, Hawaii, pp. 533-539.

## References

- Conole, G & Warburton, B 2005, 'A review of computer-assisted assessment', *ALT-J, Research in Learning Technology*, vol.13, no.1, pp. 17-31.
- Cooper, RL & Macrae, J 2003, 'Software Systems to Support the Teaching of the Use of Relational Database Systems', *Proceedings of Teaching, Learning and Assessment of Databases (TLAD)*, Coventry Techno Centre, Coventry, UK, pp. 4-12.
- Cox, K & Phalp, K 2000, 'Replicating the CREWS use case authoring guidelines experiment', *Empirical Software Engineering*, vol 5, no.3 pp.245-267.
- CREWS 1999, 'Co-operative Requirements Engineering With Scenarios', *EU funded ESPRIT project*, no.21903.
- Davies, P 2001, 'CAA must be more than multiple-choice tests for it to be academically credible?', *Proc. of 5th International CAA Conference*, Loughborough University, Loughborough, UK, pp.145-154.
- Davies, P 2002, 'There's no confidence in multiple-choice testing', *Proc. of 6th International CAA Conference*, Loughborough University, Loughborough, UK, pp. 119-130.
- DEAP Project 2007 , Diagram interpretation research, viewed 30<sup>th</sup> October 2007, <<http://mcs.open.ac.uk/Diagrams/>>.
- Dellen, B 1999, 'Change Impact Analysis Support for Software Development Processes'. PhD thesis, University of Kaiserslautern, Germany.
- Dessus, P, Lemaire, B & Vernier, A 2000, 'Free Text Assessment in a Virtual Campus,, *Proc. of the 3rd International Conference on Human System Learning*, Paris, France, pp.61-75.
- Duke-Williams, E & King, T 2001, 'Using computer-aided assessment to test higher level learning outcomes', *Proc. of 5th International CAA Conference*, Loughborough University, Loughborough, UK. pp.181-191.
- Elawar, MC, & Corno, L 1985, 'A factorial experiment in teachers' written feedback on student homework: Changing teacher behaviour a little rather than a lot', *Journal of Educational Psychology*, vol.77, no.2, pp.162-173.
- Ellson, J, Gansner, E, Koutsofios, L , North, SC & Woodhull, G 2002, 'Graphviz – open source graph drawing tools', Mutzel, P, Jünger, M & Leipert, S. (eds.), *Graph Drawing LNCS vol.2265*, Springer, Heidelberg , Germany, pp.483-484.

## References

Elmasri, R, Weeldreyer, J & Hevner, A, 1985, 'An extension to the entity-relationship model', *Data Knowledge Eng.*, vol.1, no. 1, pp. 75-116.

*Encyclopedia Britannica* 1911, 11th edition, University Press, Cambridge, UK.

Fan, S & Tanimoto S 2007, 'A Framework for Automated Diagram Assessment in Online Learning', *Proc. of 7th IEEE International Conference on Advanced Learning Technologies (ICALT 2007)*, IEEE Computer Society, Niigata, Japan, pp.51-53.

Fischer, G, Lemke, A, McCall, R & Morch, A 1995 'Making Argumentation Serve Design', in T Moran & J Carroll (ed.), *Design Rationale Concepts, Techniques, and Use*, Lawrence Erlbaum Associates, New Jersey, pp. 267-294.

Freuder, E & Mackworth, A 1994, *Constraint-based reasoning*. MIT Press, Cambridge, MA.

Fulford, H. 2001. 'Developing Document Analysis and Data Extraction Tools for Entity Modelling'. *Proc. of the 5th international Conference on Applications of Natural Language To information Systems-Revised Papers, LNCS, vol. 1959*, Springer-Verlag, London, pp.265-275.

Gibbs, G & Simpson, C 2004, 'Conditions under which assessment supports students' learning', *Learning and Teaching in Higher Education*, vol.1, pp.3-31.

Gotel OCZ & Finkelstein, A 1994, 'An Analysis of the Requirements Traceability Problem', *Proc. International Conference on Requirements Engineering (ICRE)*, IEEE CS Press, Colorado Springs, Colorado, USA, pp. 94-101.

Haladyna, T 1997, *Writing Test Items to Evaluate Higher Order Thinking*, Allyn and Bacon, Boston.

Hall, L & Gordon, A 1998, 'A virtual learning environment for entity relationship modelling', *Proc. of the 29th SIGCSE Technical Symposium on Computer Science Education, SIGCSE '98*, ACM Press, New York, NY, pp.345-349.

Heinrich, E & Lawn, A 2004 'Onscreen marking support for formative assessment', *Proc. Ed-Media (2004)*, Association for the Advancement of Computing in Education, Chesapeake, USA, pp.1985-1992.

Heinrich, E, Wang, Y 2003, 'Online Marking of Essay-type Assignments', *Proc. of Ed-Media 2003*, Association for the Advancement of Computing in Education, Norfolk, USA, pp. 768-772.

## References

- Herman I, Melancon, G & Marshall, M 2000, 'Graph visualization and navigation in information visualization: A survey', *IEEE Transactions on Visualization and Computer Graphics*, vol.6, no.1, pp. 24-43.
- Higgins, C, Hegazy, T, Symeonidis, P, & Tsintsifas, A 2003, 'The CourseMaster CBA system: Improvements over Ceilidh', *J. Edu. Inf.Technol*, vol.8, no.3, pp.287-304.
- Higgins, CA & Bligh, B 2006, 'Formative computer based assessment in diagram based domains', *Proc. of the 11th Annual SIGCSE Conference on innovation and Technology in Computer Science Education, ITICSE '06*. ACM Press, New York, NY, pp.98-102.
- Hinde, CJ, Batmaz, F & Stone, R 2008, 'A Web-Based Semi-Automatic Assessment Tool For Conceptual Database Model', *HEA development fund 2008/09*, viewed on 10<sup>th</sup> March 2009 [http://www.ics.heacademy.ac.uk/projects/development-fund/fund\\_details.php?id=125](http://www.ics.heacademy.ac.uk/projects/development-fund/fund_details.php?id=125).
- Hoggarth, G & Lockyer, M 1998 'An automated student diagram assessment system', *SIGCSE Bull*, vol.30, no.3, pp.122-124.
- Hollingsworth, J 1960, 'Automatic graders for programming classes', *Commun. ACM*, vol.3, no.10, pp.528-529.
- Jamnik, M 1998, 'Automatic diagrammatic proofs of arithmetic arguments', PhD thesis, University of Edinburgh, UK.
- Jayal, A & Shepperd, M 2008, 'The Problem of Labels in e-Assessment of Diagrams', *ACM J. of Educational Resources in Computing*, vol.8, no.4, article.12.
- Joy, M & Luck, M 1995, 'On-line submission and testing of programming assignments', *Innovations in Computing Teaching*, J. Hart (Ed.), SEDA, London, UK.
- Joy, M, Griffiths, N, Boyatt, R 2005, 'The boss online submission and assessment system', *Journal on Educational Resources in Computing (JERIC)*, vol.5 no.3, pp.2-5,
- King, G 1994, 'Developing adaptive tests for school children', in Drasgow, F & Olson-Buchanam, j (eds), *Innovations in Computerised Assessment*. Lawrence Erlbaum Associates inc, New jersey, pp.93-116.
- Knethen, A & Paech, B 2002, *A Survey on Tracing Approaches in Practice and Research*, IESE-Report No. 095.01/E, Fraunhofer, Germany.

## References

- Knethen, A. 2002, '*Change-Oriented Requirements Traceability. Support for Evolution of Embedded Systems*', PhD Thesis in Experimental Software Engineering, Fraunhofer IRB, Germany.
- Kolodner, J 1993, *Case-based reasoning*, Morgan Kaufmann Publishers Inc., San Francisco, CA.
- Kulpa, Z 1994, 'Diagrammatic Representation and Reasoning', *Machine Graphics & Vision*, vol.3, no.1, pp.77-103.
- Landauer, TK, Foltz, PW & Laham, D 1998. 'An introduction to latent semantic analysis', *Discourse Processes*, vol25, no.2, pp.259-284.
- Larkey, LS 1998, 'Automated essay grading using text categorization techniques', *Proc. of the 21<sup>st</sup> Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, CIIR publications, Melbourne, Australia, pp.90-95.
- Laurillard, D 2002, *Rethinking University Teaching A Conversational Framework For The Effective Use Of Learning Technologies*, 2nd edn, Routledge Falmer, London, UK.
- Lilley & Barker 2003, 'An Evaluation of a Computer Adaptive test in a UK University Context', *Proc. of the 7th CAA Conference*, Loughborough University Loughborough, UK, pp.171-182.
- Lindvall, M & Sandahl, K 1996, 'Practical Implications of Traceability', *Software Practice and Experience*, vol. 26, No. 10, pp. 1161-1180.
- Lindvall, M 1994, '*A Study of Traceability in Object-Oriented Systems Development*', PhD thesis, Linköping University, Sweden.
- Low, A, Hatch, A & Burd, L 2009, 'Technologically Enhanced Demonstrator Support or Tools for Support Demonstrators in First Year Programming Lab Classes', *Proc. of the 10th Annual Conference of Information and Computer Sciences*. Higher Education Academy Subject Centre, University of Ulster, UK, pp.30-35.
- Manning, CD & Schütze, H 2002, *Foundations of Statistical Natural Language Processing*, The MIT Press, Cambridge, Massachusetts.
- Mason D & Woit D 1998, 'Integrating technology into computer science examinations', *Proc. of the 29th SIGCSE Technical Symposium on Computer Science Education*, ACM press, atlanta, GA USA, pp.140-144.

## References

Mason, DV & Woit, DM 1999, 'Providing mark-up and feedback to students with online marking'. *The Proc. of the 30<sup>th</sup> SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '99, ACM Press, New York, NY, pp.3-6.

McAlpine, M 2002, *Principles of Assessment*, CAA Centre, Luton, UK.

McKenna, C 2000, 'Using computers to assess humanities: some results from the national survey into the use of computer-assisted assessment', *Computers and Texts*, vol18, no.19, pp. 6-7.

Miller, GA 1956, 'The magic number seven plus or minus two: some limits on our capacity to process information'. *Psychological Review*, vol.63, no.2, pp.81-97.

Mills, C, Potenza, M, Fremer, J & Ward, C 2002, *Computer-based testing – building the foundation for future assessment*, Lawrence Erlbaum Associates, New York, NY.

Miriyala, K & Harandi, MT 1991, 'Automatic derivation of formal software specifications from informal descriptions', *IEEE Trans. Soft. Eng.*, vol.17, no.10, pp.1126-1142.

Mitrovic, A 2003, 'An intellignet SQL tutor on the Web', *Int. J. AI Edu.* vol.13, no.2, pp.173-197.

Nicol, D & Macfarlane-Dick, D 2004, 'Rethinking Formative Assessment in HE: a theoretical model and seven principles of good feedback practice', in C Bryan & K Clegg (ed.), *Innovative assessment in higher education*, Routledge, Abingdon, New York, pp.64-77.

Page, EB 1966, 'The imminence of grading essays by computer', *Phi Delta Kappan*, vol.48, no.1, pp.238-243.

Pennock, J & Tabrizi, MHN 2008, 'A Survey of Input Sensing and Processing Techniques for Multi-Touch Systems', *Proc. of The 2008 International Conference on Computer Design*, IEEE CS Press, Las Vegas, Nevada, USA, pp.10-16.

Phalp, KT, Vincent, J, & Cox, K 2007, 'Improving the quality of use case descriptions: empirical assessment of writing guidelines', *Software Quality Control*, vol.15, no.4, pp.383-399.

Pinheiro, F 1996, 'Design of a Hyper-Environment for Tracing Object-Oriented Requirements', PhD thesis, University of Oxford, UK.



## References

- Plimmer, B & Mason, P 2006, 'A pen-based paperless environment for annotating and marking student assignments', *Proc. of the 7th Australasian User interface Conference*, vol. 169, ACM International Conference Proceeding Series, Darlinghurst, Australia, pp.37-44.
- Pohl, K 1996, *Process-Centered Requirements Engineering*, 2<sup>nd</sup> Edition, John Wiley & Sons, New York, NY, USA.
- Pritchett, N 1999, 'Effective question design', In S. Brown, P. Race & J. Bull (Eds), *Computer assisted assessment in higher education*, Kogan Page, London, UK, pp.29-37.
- Question Mark Computing Ltd 2004, 'Perception: Windows based authoring', viewed 29 August 2004 <[http://www.questionmark.com/uk/perception/authoring\\_windows.htm](http://www.questionmark.com/uk/perception/authoring_windows.htm)>.
- Raikes, N, Grotorex, J & Shaw, S 2004, 'From Paper to Screen: some issues on the way', *International Association of Educational Assessment Conference*, viewed 2 August 2007 <[http://www.cambridgeassessment.org.uk/ca/Our\\_Services/Research/Conference\\_Papers](http://www.cambridgeassessment.org.uk/ca/Our_Services/Research/Conference_Papers)>
- Ramamoorthy, CV, Usuda, Y, Prakash, A & Tsai, WT 1990, 'The evolution support environment system', *IEEE Transactions on Software Engineering*, Vol. 16., No. 11, pp. 1225-1234.
- Ramesh, B & Jarke, M 2001, 'Towards Reference Models for Requirements Traceability', *IEEE Transactions on Software Engineering*, vol. 27, No. 1, pp. 58-92.
- Ramesh, B 1998, 'Factors influencing requirements traceability', *Communications of the ACM*, vol. 41, no. 12, pp. 37-44.
- Sadler, DR 1989, 'Formative assessment and the design of instructional systems', *Instructional Science*, vol.18, no.2, pp.119-144.
- Sclater, N & Howie, K 2003, 'User requirements of the ultimate online assessment engine', *Computers and Education*, vol.40, no.3, pp.285-306.
- Sclater, N 2004, *Final report for the Item Banks Infrastructure Study*, IBIS, JISC, Bristol.
- Simas R & McBeath R 1992, *Constructing Multiple Choice Items*, In *Instructing and Evaluating in Higher Education*, Educational Technology Publications, Englewood Cliffs, New Jersey.

## References

- Smith, N, Thomas, PG & Waugh, K 2004, 'Interpreting Imprecise Diagrams', *Diagrams 2004 - Third International Conference on the Theory and Application of Diagrams*, Springer, University of Cambridge, UK, pp.239-241.
- Stephens, D & Mascia, J 1997, *Results of a Survey into the Use of Computer-Assisted Assessment in Institutions of Higher Education in the UK*, Loughborough University. Loughborough, UK.
- Stone, R, Batmaz, F & Hinde, C 2009, 'Drawing and Marking Graph Diagrams', *Italics*, vol.8, no.2, pp.45-53.
- Suraweera, P & Mitrovic, A 2002, 'KERMIT: a Constraint-based tutor for database modelling', *Proc ITS'2002, LCNS 2363*, Springer, Biarritz, France, pp.377-387.
- Suraweera, P 2001, 'An intelligent teaching system for database modelling', MSc Thesis, University of Canterbury, New Zealand.
- Tamassia, R, Di Battista, G & Batini, C 1988, 'Automatic graph drawing and readability of diagrams', *IEEE Transactions on Systems, Man and Cybernetics*, vol.18, no.1, pp.61-79.
- Thomas, P 2003, 'Evaluation of Electronic Marking of Examinations', *Proc. of the 8th Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE 2003)*, ACM press, New York, NY, pp.50-54.
- Thomas, P 2004, *Drawing Diagrams in Online Examinations*, Technical Report, TR2004/14, Computing Department, Open University, UK.
- Thomas, P 2004, *Grading Diagrams Automatically*, Technical Report, TR2004/01, Computing Department, Open University, UK.
- Thomas, PG, Smith, N, & Waugh, K. 2008, 'Automatically assessing graph-based diagrams.' 'Learning, Media and Technology, Volume 33 Issue 3' pp 249-267
- Thomas, P, Waugh, K, & Smith, N 2005, 'Experiments in the Automatic marking of E-R Diagrams', *Proc. of the 10th Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE 2005)*, ACM press, New York, NY, pp.158-162.
- Thomas, PG, Waugh, K, & Smith, N 2006, 'Using Patterns in the Automatic Marking of ER-Diagrams', *Proc. of the 11th Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE 2006)*, ACM press, New York, NY, pp.403-413.

## References

TOIA 2004, '*Technologies for Online Interoperable Assessment (TOIA)*', JISC funded Project ,viewed 31August 2004 <<http://www.toia.ac.uk>>.

Tselonis C, Sargeant J, McGee Wood M 2005, 'Diagram Matching for Human-Computer Collaborative Assessment', *Proc. of the 9th CAA Conference*, Loughborough University, Loughborough, UK, pp. 441-456.

Tsintsifas A 2002, '*A framework for the computer-based assessment of diagram-based coursework*', PhD thesis, University of Nottingham, UK.

Watson, I 1997, *Applying Case-Based Reasoning: Techniques for Enterprise Systems*, Morgan Kaufmann, San Francisco, USA.

Wieringa, RJ 1995, *An introduction to requirements traceability*, Technical Report IR-389, Faculty of Mathematics and Computer Science, Vrije Universiteit, Amsterdam.

Williams R 2001, 'Automated essay grading: An evaluation of four conceptual models', In A Herrmann & M Kulski (Eds), *Expanding Horizons in Teaching and Learning*, Curtin University of Technology, Perth, Australia, pp.139-140.

Woodford, K & Bancroft, P 2005, 'Multiple choice questions not considered harmful'. *Proc. of the 7th Australasian Conference on Computing Education ACE 2005*, Australian Computer Society, New South Wales, Australia, pp.109-116.

Yates, RB & Neto, BR 1999, *Modern Information Retrieval*, 1<sup>st</sup> edn, ACM Press Series/Addison Wesley, New York.

# Appendix A

## 1. The scenario writing tool

This section shows the environment of the scenario writing tool which is used in the experiment in Section 7.2. It is used to produce scenario text for an event management system. It consists of ten pages. They are one plan, eight section and one production pages. In the experiment, the tool uses a split template. The template consists of eight diagram templates. They are used to write each section for a scenario.

### Plan page:

A domain, a scenario template and a diagram type are chosen in this page. The page displays the related diagram template, an example scenario and its diagram. Figure A.1 shows the screenshot of the page. No text entry is required in this page. The diagram template has three entities: "Event", "Member" and "Entity X". Entity names are given manually for the experiment.

### The Scenario Planner

Chose A Domain

Chose A Scenario Template

Chose A Diagram

### The diagram template

---

### Scenario text for the example diagram

The Blue Computer Training School (BCTS) provides a wide range of computer training short courses. BCTS's manager gives you the following description of the business:

The administrator records the details of any new course: course code, course name, description, level, tuition fee, and starting date. The details of new students are kept into the student file. The school needs to know their name, address and qualification. Each student is assigned a unique student id. A student may enrol on several courses. At the end of a course, the student is assessed and the grade achieved is recorded.

Same course is offered several times a year. Students select a suitable starting date of the course during the enrolment. If necessary, the tuition fee of the same course is adjusted whenever it is offered.

Page 1/10

### The example diagram

Figure A.1 Plan page 1

The related scenario text and its diagram are placed in the environment manually for this proof of concept tool.

**Section Pages:**

Figure A.2 shows the diagram template for the “Event” entity. The user needs to enter text into the system which describes the entity. The example diagram and sample text are given to help the user for text writing on Page 2.

**Figure A.2 Section page 2 for “Event” entity type**

Figure A.3 shows the diagram template for the “Member” entity. The user needs to enter text into the system which describes the entity. The example diagram and sample text are given to help the user for text writing on Page 3.

**Figure A.3 Section page 3 for “Member” entity type**

Appendix A

Figure A.4 shows the diagram template for the relationship between the “Member” and “Event” entities. The user needs to enter text into the system which describes the relationship. The example diagram and sample text are given to help the user for text writing on Page 4.

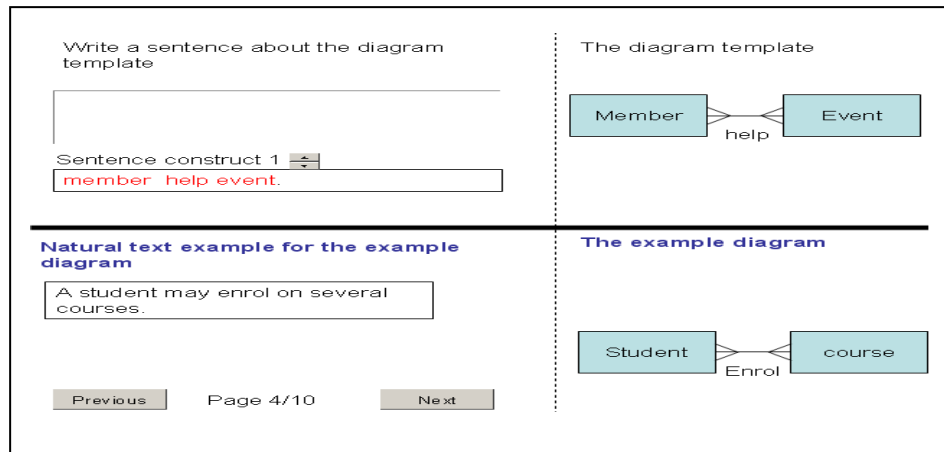


Figure A.4 Section page 4 for “Help” relationship type

Figure A.5 shows the diagram template for the “Date” attribute of the “Event” Entity, which is a multi value attribute. The user needs to enter text into the system which describes the attribute. The example diagram and sample text are given to help the user for text writing on Page 5.

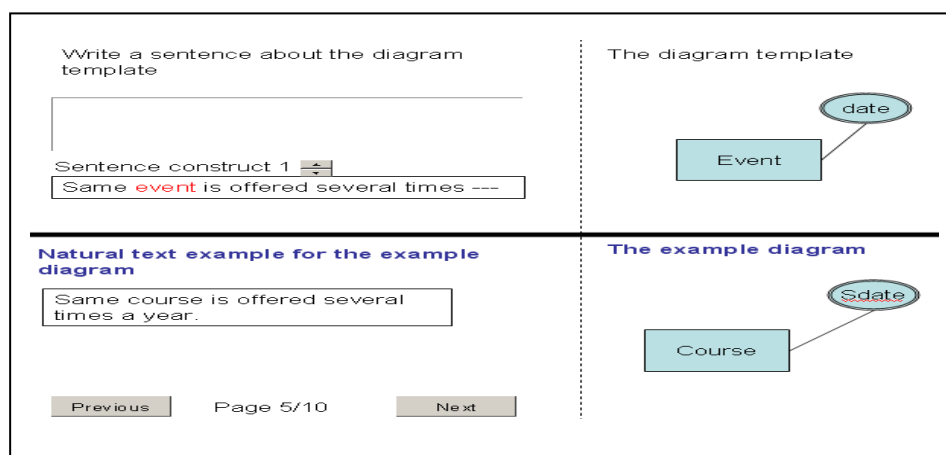


Figure A.5 Section page 5 for “Date” attribute type

Figure A.6 shows the diagram template for the “Date” attribute of no named entity X, which is a single value attribute. The template also shows the relationship between the entity X and “Event” entity. The diagram template on this page is an alternative representation of the diagram on Figure A.5.

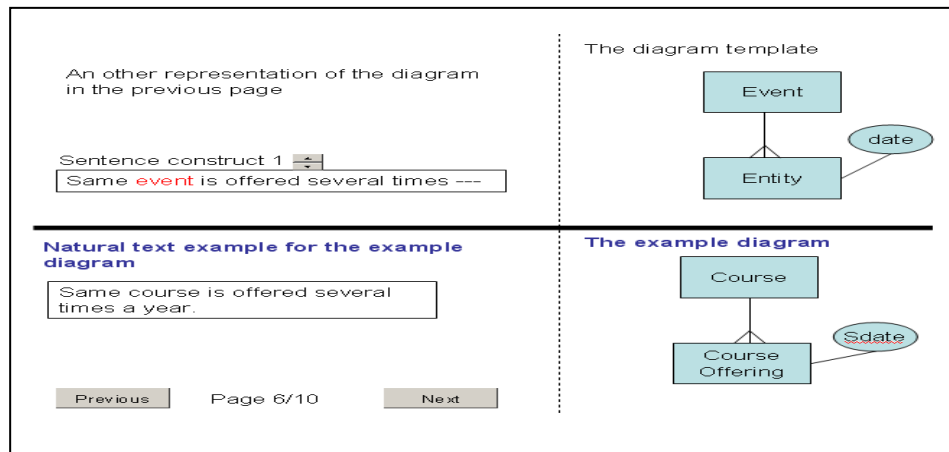


Figure A.6 Section page 6 for relationship and attribute types

Figure A.7 shows the diagram template for the relationship between the entity X and “Member” entity. User needs to enter text into the system which describes the relationship. The example diagram and sample text are given to help the user for text writing on Figure A.7.

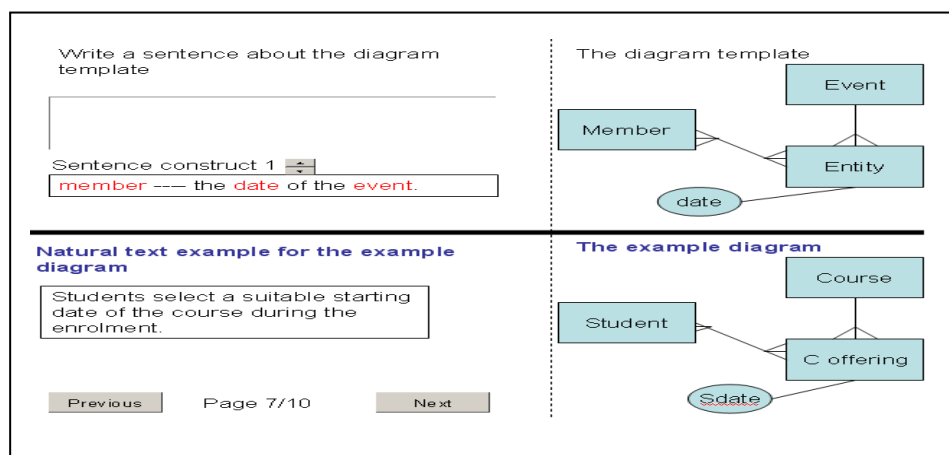


Figure A.7 Section page 7 for relationship type

Appendix A

The following picture shows the diagram template for the “Fee” attribute of the entity X. User needs to enter text into the system which describes the attribute. The example diagram and sample text are given to help the user for text writing on Figure A.8.

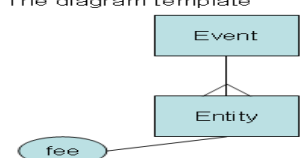
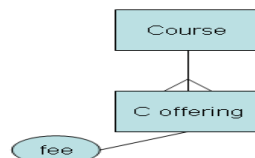
<p>Write a sentence about the diagram template</p> <div style="border: 1px solid black; height: 20px; width: 100%;"></div> <p>Sentence construct 1 <input type="text" value=""/></p> <div style="border: 1px solid black; padding: 2px;"> <p>The <b>fee</b> of <b>event</b> is adjusted whenever it is offered.</p> </div>	<p>The diagram template</p> 
<p><b>Natural text example for the example diagram</b></p> <div style="border: 1px solid black; padding: 2px;"> <p>If necessary, the tuition fee of the same course is adjusted whenever it is offered.</p> </div>	<p><b>The example diagram</b></p> 
<p><input type="button" value="Previous"/> Page 8/10 <input type="button" value="Next"/></p>	

Figure A.8 Section page 8 for “Fee” attribute type

Figure A.9 shows the diagram template which includes all the components. The user needs to write an introduction about the system. The text entered into this page will be the first paragraph of the scenario text.

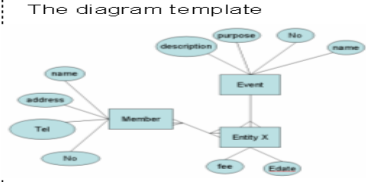
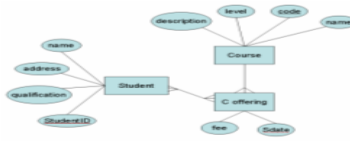
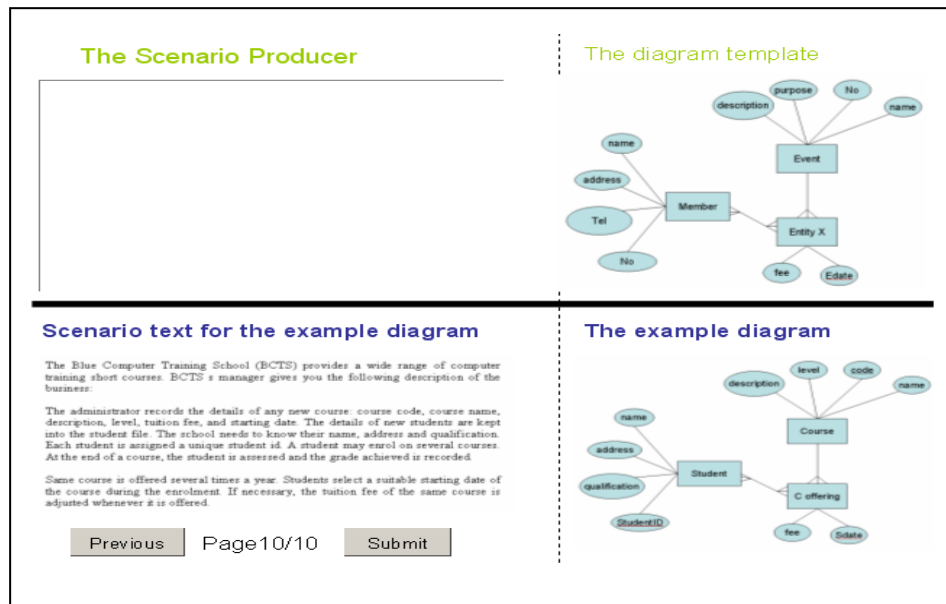
<p>Write a sentence to introduce the system</p> <div style="border: 1px solid black; height: 20px; width: 100%;"></div> <p>Sentence construct 1 <input type="text" value=""/></p> <div style="border: 1px solid black; padding: 2px;"> <p>X provide <b>event</b>. X gives you the following description of the business:</p> </div>	<p>The diagram template</p> 
<p><b>Natural text example for the example diagram</b></p> <div style="border: 1px solid black; padding: 2px;"> <p>The Blue Computer Training School (BCTS) provides a wide range of computer training short courses. BCTS s manager gives you the following description of the business:</p> </div>	<p><b>The example diagram</b></p> 
<p><input type="button" value="Previous"/> Page 9/10 <input type="button" value="Next"/></p>	

Figure A.9 Section page 9 for the whole of the diagram



**Production Page:**

Figure A.10 shows a production page. All the text entered in each section pages are put together as a scenario text on the production page. The scenario text is represented to the user. When the separately written text parts come together, they may be disjoint and not natural. Therefore the production page allows the user to modify the text.



**Figure A.10 Production page 10**

The scenario text on the production page is the composition of the section pages in the following order:

The user text on Page 9 on Figure A.9 makes up the first paragraph of the scenario. It is the introduction of the scenario.

The user text on Page 2 on Figure A.2, on Page 3 on Figure A.3 and on Page 4 on Figure A.4 are the paragraph 2, 3 and 4 of the scenario.

The user text on Page 5 on Figure A.5, on Page 6 on Figure A.6 , on Page 7 on Figure A.7 and on Page 8 on Figure A.8 make up the last paragraph of the scenario.

## **2. Scenario Writing**

This section shows three scenarios produced in the experiment in Section 7.2. The scenario writing environment in the previous section is used.

### **Scenario X**

This scenario text was produced by Lecturer X. It has five sections. The scenario requires the use of “Split” function. Text in Section 5 of the scenario shows that the name (or the ideal name) for the “split” entity is not mentioned in the scenario text as required for the experiment.

#### **Section 1:**

Members are involved in a range of events, each of which can be held a number of times. These are recorded by the event manager.

#### **Section 2:**

The details are recorded for an event. These show the date of the event, the purpose, the name of the event and the fee charged. An ID number is given and a description of the event is held.

#### **Section 3:**

Each member's ID number, name, telephone number and address are recorded.

#### **Section 4:**

A member can be involved in one or more events and each event can have one or more members involved in it.

#### **Section 5:**

The event can take place on a number of dates and the event can cost different amounts on different occasions. When a member is involved in an event, the date he or she is involved and the fee paid is recorded (by the events manager)

## **Scenario Y**

This scenario text was produced by Lecturer Y. It has five sections. The scenario requires the use of the “Split” function. The text in Section 5 of the scenario shows that the name (or the ideal name) for the “split” entity is not mentioned in the scenario text as required for the experiment.

### **Section 1:**

Blue Sky Events (BSE) runs events for their members. A manager of BSE has given the following description of her company's operation:

### **Section 2:**

An event has the following attributes: a unique ID number (No), a description, purpose, name, start date and fee.

### **Section 3:**

The details of each member are kept. Users need to know the member's address, name and telephone number. Each member is assigned a unique ID number.

### **Section 4:**

A member may take part in several events and an event can be attended by several members.

### **Section 5:**

An event can be offered on several different dates. Members choose a particular date they wish to attend the event on. The fee of the event is adjusted whenever it is offered.

## Scenario Z

This scenario text was produced by Lecturer Z. It has five sections. The scenario requires the use of the “Split” function. The Text in Section 5 of the scenario shows that the name (or the ideal name) for the “split” entity is not mentioned in the scenario text as required for the experiment.

### Section 1:

True Colour (TC) is a charitable trust. It organises events to increase the awareness of animal abuse for children. TC’s manager gives you the following description of the business:

### Section 2:

Events are planned by the trustees. They decide the details like: event name, description, purpose, date. The executive management board (EMB) calculates the cost and set the fee of the event. The charity secretary records event information with a unique event number to the event file.

### Section 3:

TC keeps the record of its members. A member’s name, address and phone number are taken on the membership form. At the end of the membership process, each member is given a member number.

### Section 4:

The charity gets the members’ help for each event. The members who contribute to an event are recorded. A special “thank you” card is sent to their addresses.

### Section 5:

Popular events are repeated several times a year. If necessary, the fee of an event is adjusted whenever it repeats.

# Appendix B

## Scenario 1

This scenario text is used for Experiments 1 and 3 in Chapter 7. It consists of 6 sections and 15 noun phrases. The scenario requires the use of “Split” function. The name (or the ideal name) for the “split” entity is “course offering” in the teacher solution.

### **Title: Computer Training**

#### **Section 1:**

The Blue Computer Training School (BCTS) provides a wide range of computer training short courses. BCTS s manager gives you the following description of the business:

#### **Section 2:**

The administrator records the details of any new course: course code, course name, description, level, tuition fee, and starting date.

#### **Section 3:**

The details of new students are kept into the student file. The school needs to know their name, address and qualification. Each student is assigned a unique student id.

#### **Section 4:**

A student may enroll on several courses. At the end of a course, the student is assessed and the grade achieved is recorded.

#### **Section 5:**

Same course is offered several times a year. Students select a suitable starting date of the course during the enrolment.

#### **Section 6:**

If necessary, the tuition fee of the same course is adjusted whenever it is offered.

**Noun phrases:**

Table B.1 shows the list of the noun phrases in each section for Scenario 1. The total number of noun phrases is given at the end of each section. Noun phrases in the lists are used to create direct components.

**Table B.1 The list of noun phrases for Scenario 1**

Section No	Noun Phrase	Section No	Noun Phrase
2	Administrator	4	Student
	Course		Course
	Course Code		Grade
	Course Name	5	Course
	Description		Student
	level		Starting Date
	Tuition Fee	6	Tuition Fee
	Starting Date		Course
Total number of Phrases =8			
3	Student		
	Student File		
	School		
	Name		
	Address		
	Qualification	Student ID	
Total number of Phrases =7			

**Solution:**

Figure B.1 is an entity relationship diagram showing the solution of Scenario 1. It has three entities, two relationships and eleven attributes. The Names of the attributes and two entities are picked from the list given by the scenario. The “Course offering” was typed by the examiner.

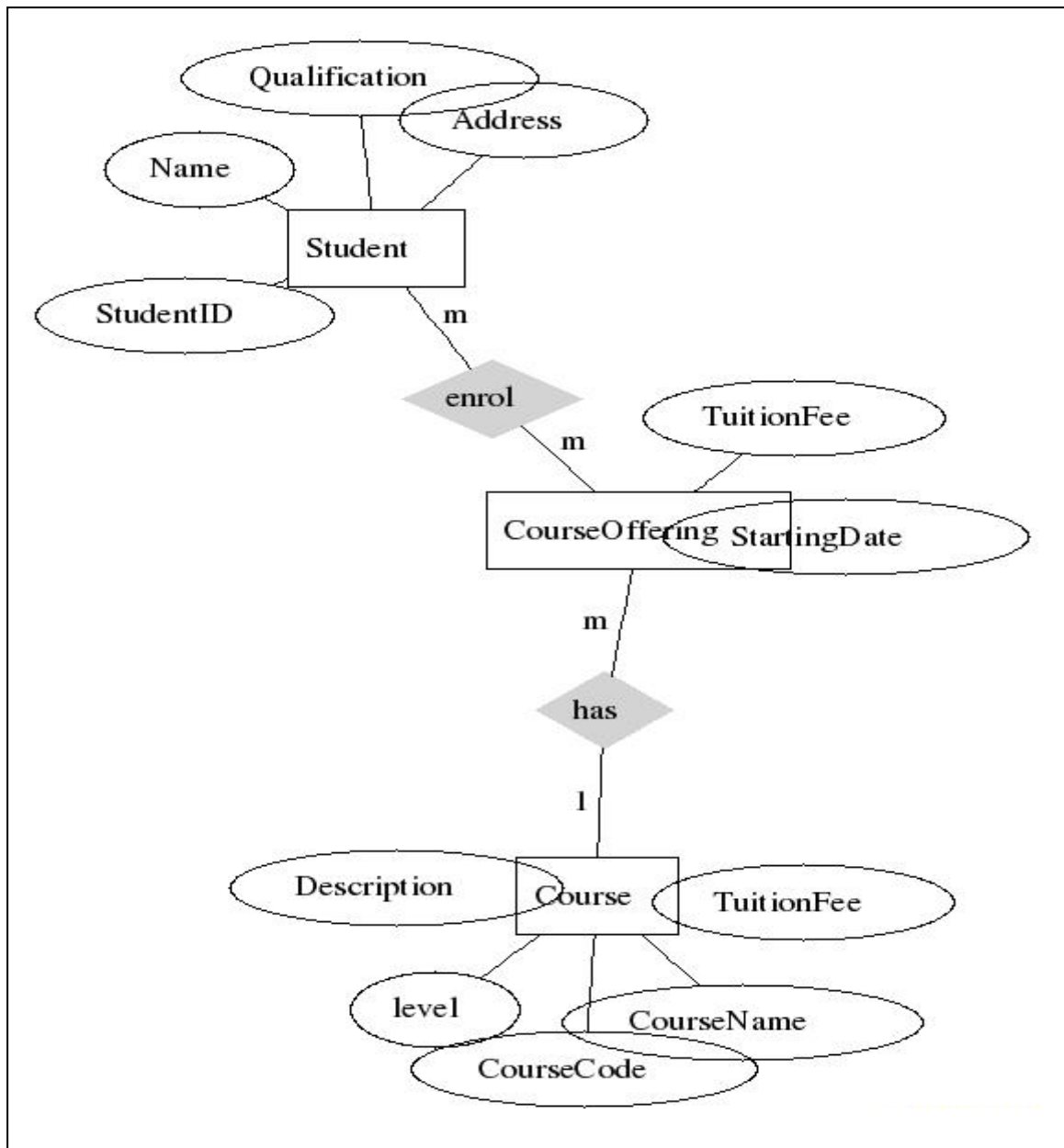


Figure B.1 The solution of Scenario 1

### Reference Diagrams:

Figure B.2 shows two ideal reference diagrams for the entity components in the teacher solution of Scenario 1. The first diagram shows that “Student” entity has a direct reference to the scenario text in section 3. The second one shows that “Course Offering” entity has an indirect reference with a “split” action to the text in section 2.

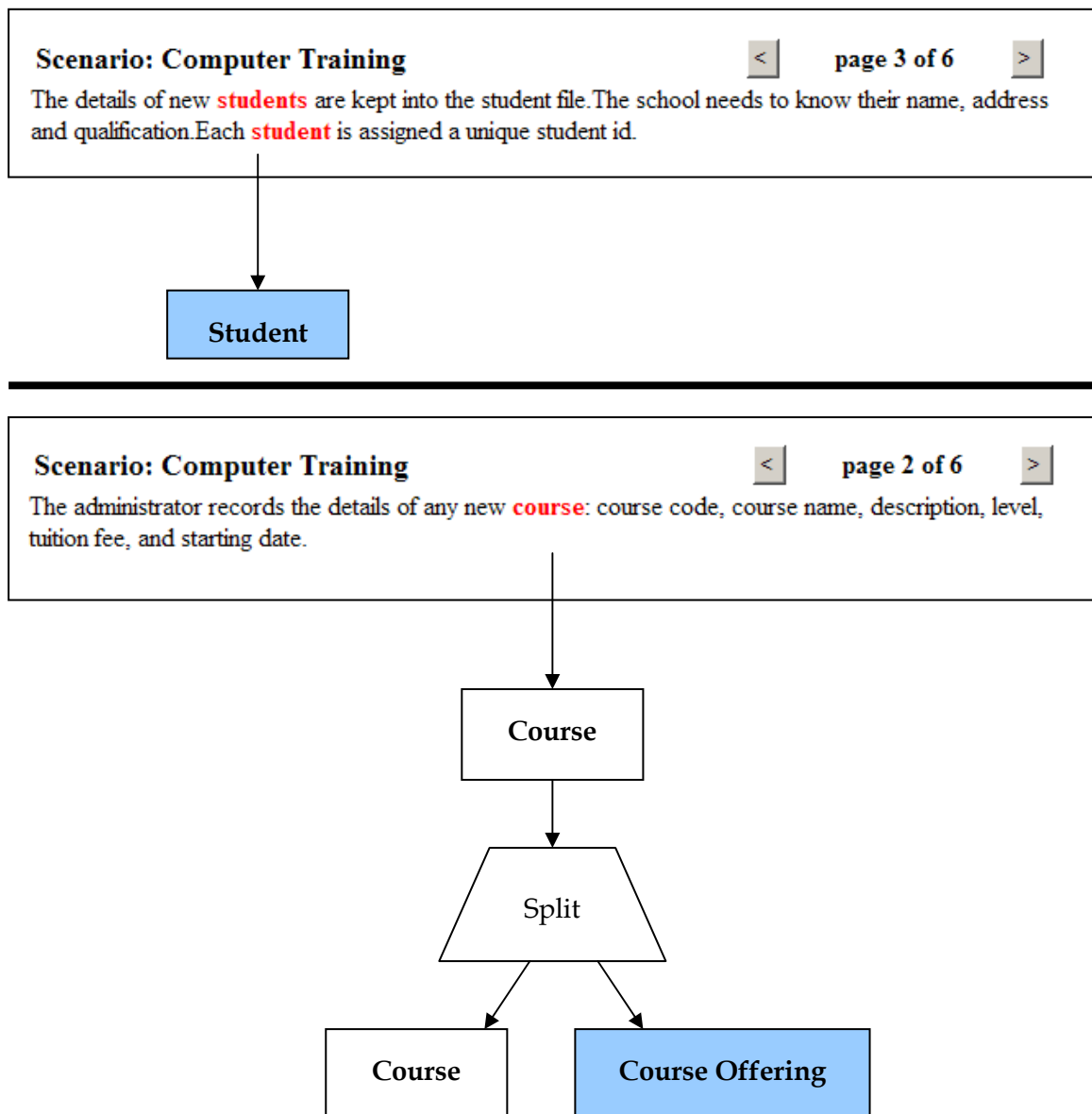


Figure B.2 Ideal reference diagrams for the entity components



## **Scenario 2**

This scenario text is used for Experiments 1 and 3 in Chapter 7. It consists of 4 sections and 14 noun phrases. The scenario requires the use of the “Merge” function. The name (or the ideal name) for the “Merge” entity is “staff” in the teacher solution.

### **Title: Loughborough University**

#### **Section 1:**

Loughborough University wants to keep the information about support staff and academics for each department.

#### **Section 2:**

Academics are given a unique number by the university. Their name, address, gender and date of birth are stored.

#### **Section 3:**

Support staff and academics can work in only one department. Support staff’s id number, name, address should be kept in the database. Their job title and age are also important to store.

#### **Section 4:**

Each department has got a unique name, address and main office phone number.

**Noun phrases:**

Table B.2 shows the list of the noun phrases in each section for Scenario 2. The total number of noun phrases is given at the end of each section. Noun phrases in the lists are used to create direct components.

**Table B.2 The list of the noun phrases for Scenario 2**

Section No	Noun Phrases	Section No	Noun Phrases	
1	Lboro University	3	Support Staff	
	Support Staff		Academic	
	Academic		Department	
	Department		ID number	
2	Academic		Total number of Phrases =8	Name
	University			Job Title
	Number			Age
	Name			Address
	Address	4	Department	
	Gender		Name	
	Date Of Birth		Address	
Total number of Phrases =4		Total number of Phrases =4	Phone Number	

**Solution:**

Figure B.3 is an entity relationship diagram showing the solution of Scenario 2. It has two entities, one relationship and nine attributes. The names of the attributes and “department” entity are picked from the list given by the scenario. The “Staff” name was typed by the examiner.

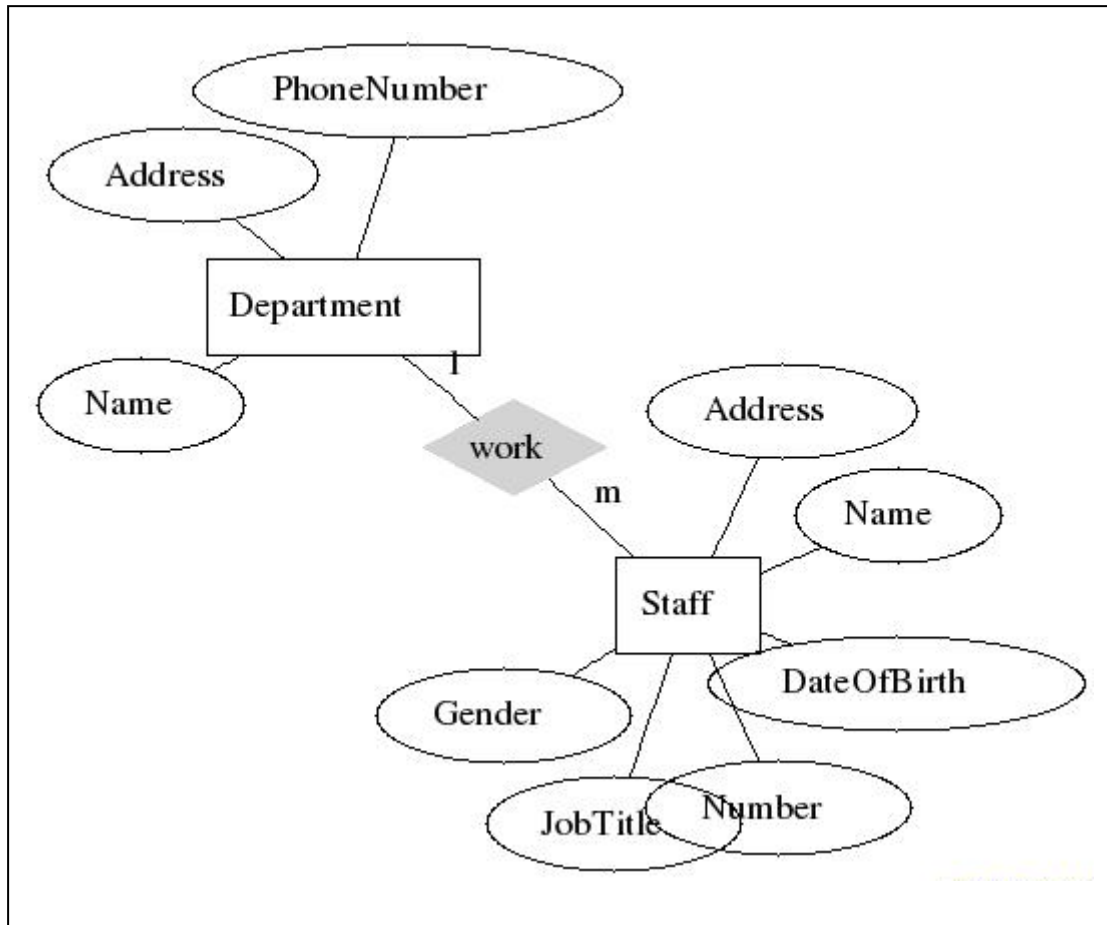


Figure B.3 The solution of Scenario 2

### Reference Diagrams:

Figure B.4 shows two ideal reference diagrams for the entity components in the teacher solution of Scenario 2. The first diagram shows that “Department” entity has a direct reference to the scenario text in section 4. The second one shows that “Module” entity has an indirect reference with a “merge” action to the text in section 2 and 3.

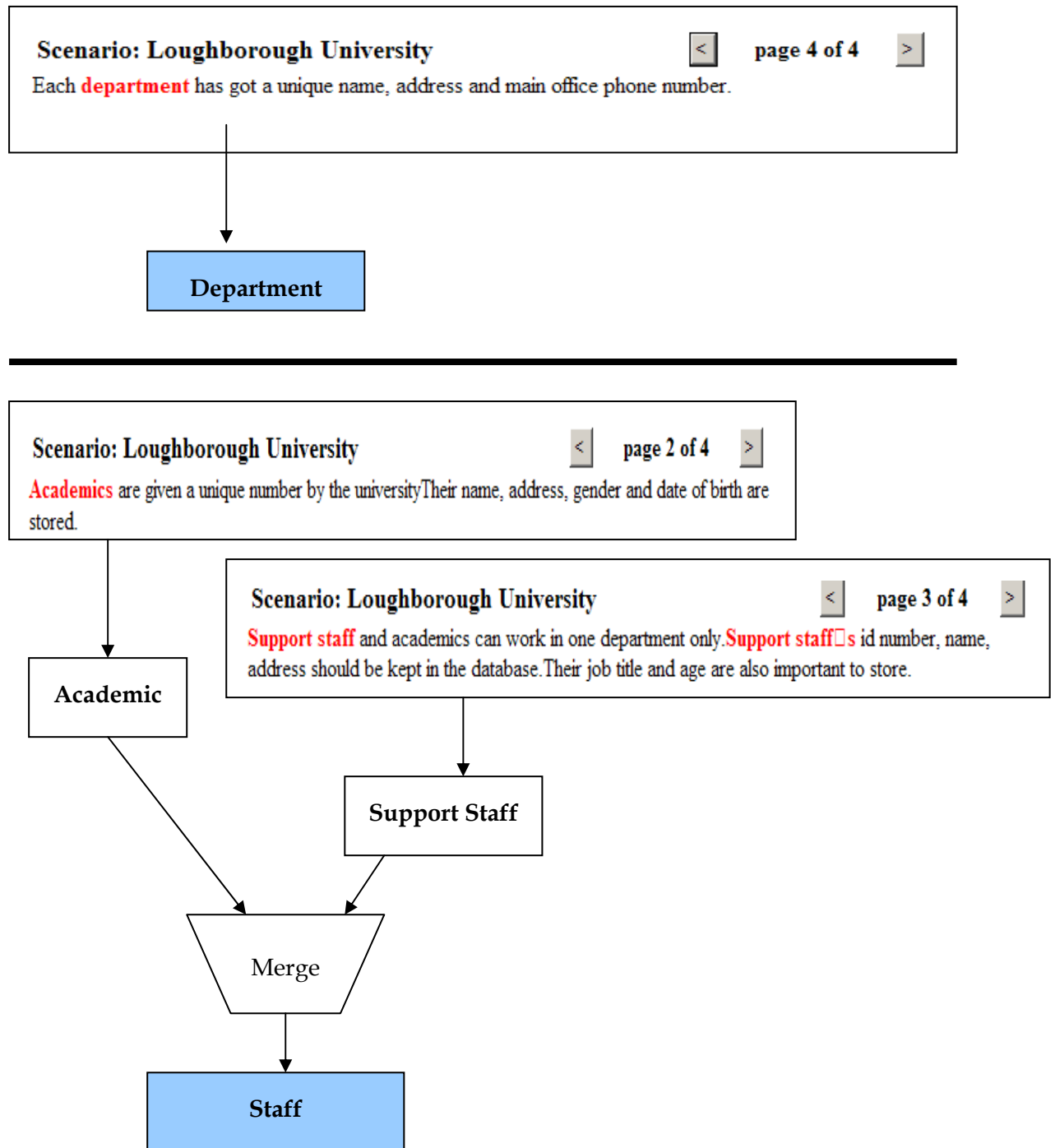


Figure B.4 Ideal reference diagrams for the entity components

## Scenario 3

This scenario text is used for Experiment 2 in Chapter 7. It consists of 5 sections and 23 noun phrases. The scenario requires the use of the “Split” function. The name (or the ideal name) for the “Split” entity is “popular event” in the teacher solution.

### **Title: Event Organiser**

#### **Section 1:**

True Colour (TC) is a charitable trust. It organises events to increase the awareness of animal abuse for children. TC’s manager gives you the following description of the business:

#### **Section 2:**

Events are planned by the trustees. They decide the details like: event name, description, purpose, date. Executive management board (EMB) calculates the cost and set the fee of the event. The charity secretary records event information with a unique event number to the event file.

#### **Section 3:**

TC keeps the record of its members. A member’s name, address and phone number are taken on the membership form. At the end of the membership process, each member is given a member number.

#### **Section 4:**

The charity gets the members’ help for each event. The members who contribute an event are recorded. A special “thank you” card is sent to their addresses.

#### **Section 5:**

Popular events are repeated several times a year. If necessary, the fee of an event is adjusted whenever it repeats.

**Noun phrases:**

Table B.3 shows the list of the noun phrases in each section for Scenario 3. The total number of noun phrases is given at the end of each section. Noun phrases in the lists are used to create direct components.

**Table B.3 The list of the noun phrases for Scenario 3**

Section No	Noun Phrases	Section No	Noun Phrases
2	Event	3	TC
	Trusty		Member
	Event Name		Name
	Description		Address
	Purpose		Phone No
	Date		Form
	EMB		Member No
	Cost	4	Charity
	Fee		Member
	Secretary		Card
	Information		Address
	Event Number		Event
	Event File	5	Event
	Fee		
Total number of Phrases =13		Total number of Phrases =7	
		Total number of Phrases =5	
		Total number of Phrases =2	

**Solution:**

Figure B.5 is an entity relationship diagram showing the solution of Scenario 3. It has three entities, two relationships and ten attributes. Names of the attributes and two entities are picked from the list given by the scenario. The “Repeated event” name was typed by the examiner.

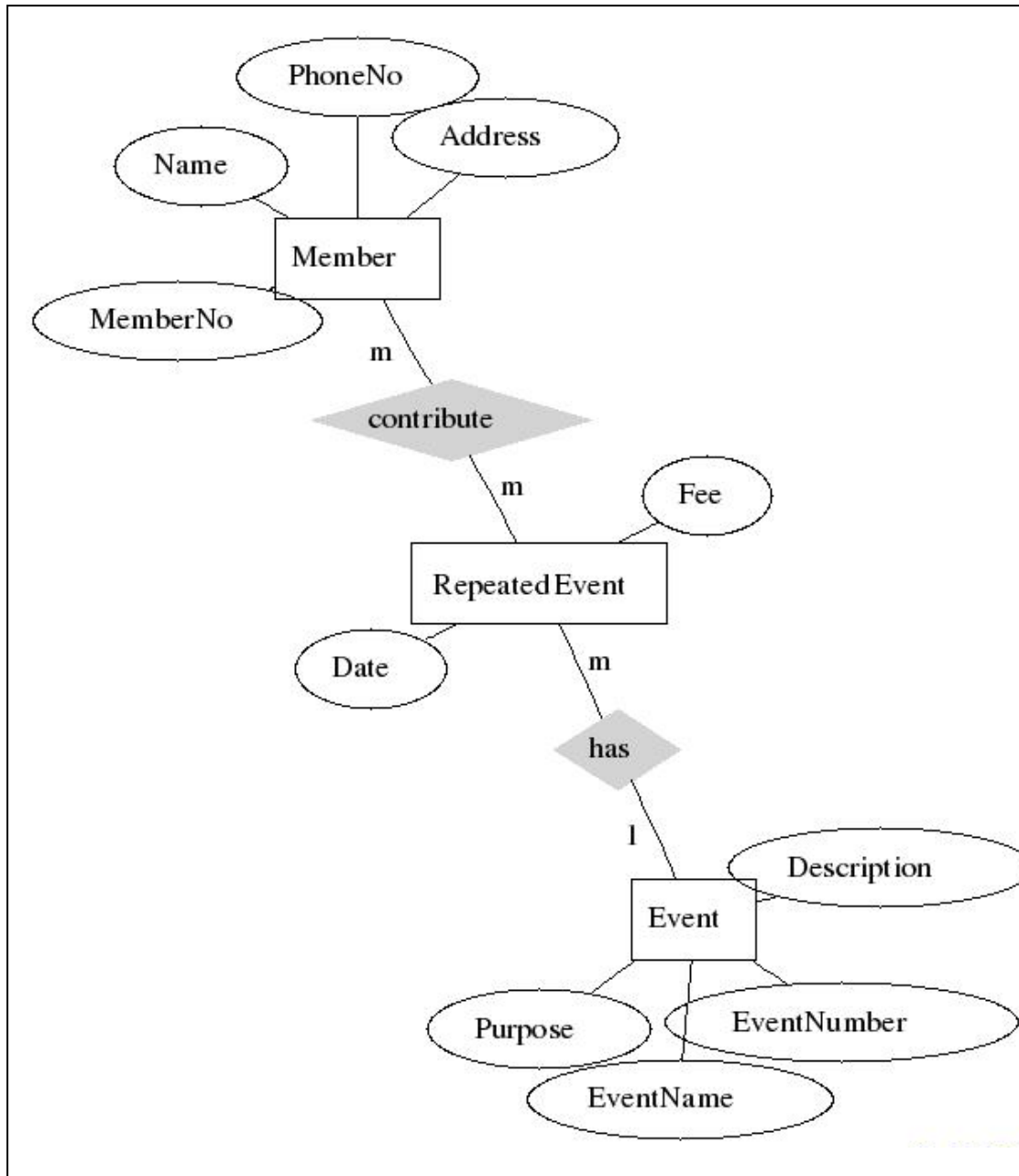


Figure B.5 The solution of Scenario 3

### Reference Diagrams:

Figure B.6 shows two ideal reference diagrams for the entity components in the teacher solution of Scenario 3. The first diagram shows that the “Member” entity has a direct reference to the scenario text in section 3. The second one shows that the “Repeated Event” entity has an indirect reference with a “split” action to the text in section 2.

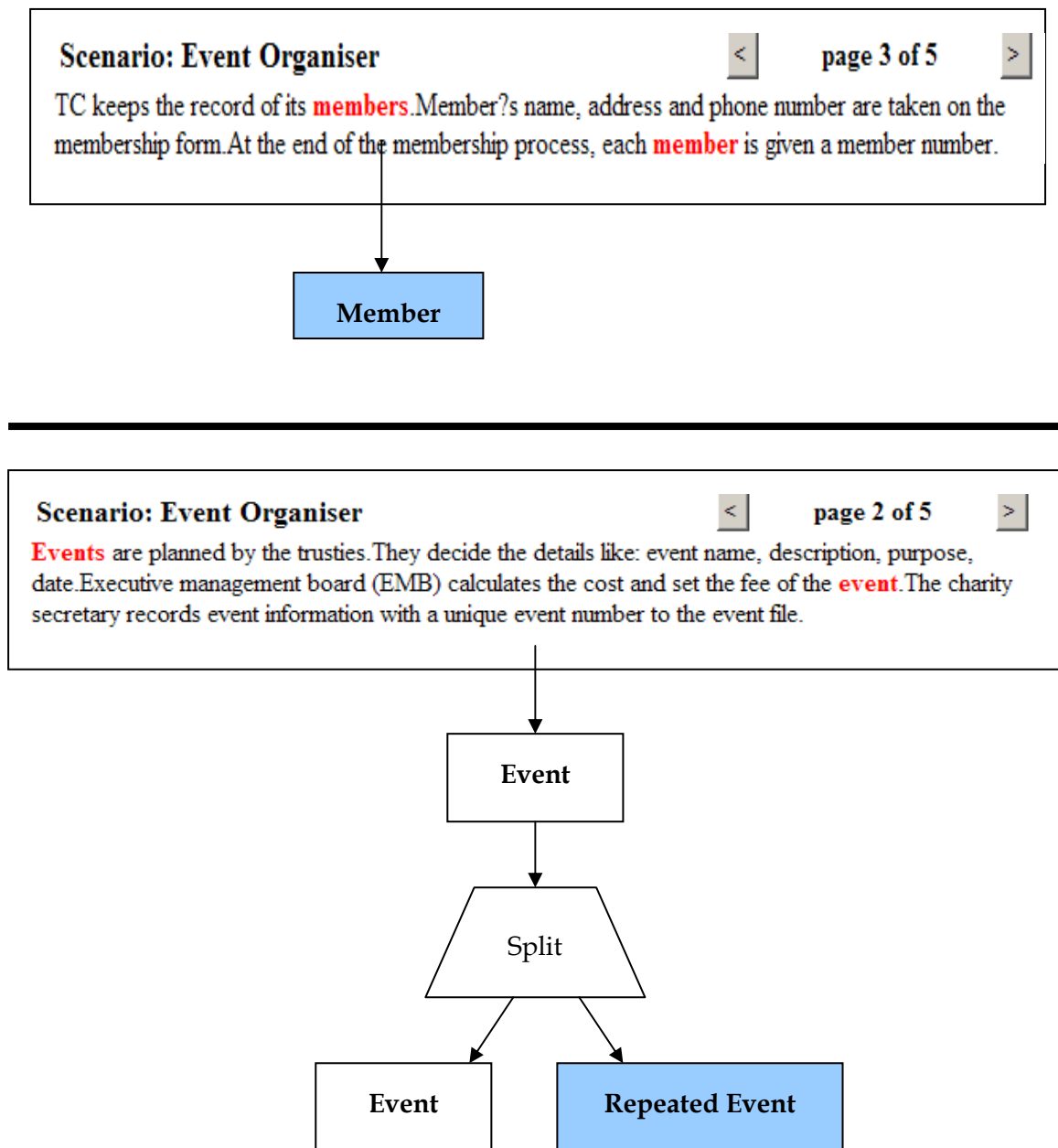


Figure B.6 Ideal reference diagrams for the entity components



## **Scenario 4**

This scenario text is used for Experiment 2 in Chapter 7. It consists of 4 sections and 15 noun phrases. The scenario requires the use of “Merge” function. The name (or the ideal name) for the “Merge” entity is “module” in the teacher solution.

### **Title: Computer Science Department**

#### **Section 1:**

The computer science department wants to keep the information about core modules and options of each postgraduate programme.

#### **Section 2:**

Core modules are given a unique code by the programme manager. Their title, credit, assessment style are stored.

#### **Section 3:**

Core modules and options can be given in more than one programme. The code, name and type of each option need to be stored. The option’s credit and evaluation method are kept together with its other information.

#### **Section 4:**

The name, content and brief information of each programme are prepared by the programme organiser

**Noun phrases:**

Table B.4 shows the list of the noun phrases in each section for Scenario 4. The total number of noun phrases is given at the end of each section. Noun phrases in the lists are used to create direct components

**Table B.4 The list of the noun phrases for Scenario 4**

Section No	Noun Phrases	Section No	Noun Phrases	
1	Department	3	Code	
	Core Module		Name	
	Option		Type	
	Programme		Credit	
2	Core Module		Total number of Phrases =8	Evaluation
	Code		4	Name
	Manager			Content
	Title			Information
	Credit	Programme		
	Assessment	Organiser		
3	CoreModule	Total number of Phrases =5		
	Option			
	Programme			

**Solution:**

Figure B.7 is an entity relationship diagram showing the solution of Scenario 4. It has two entities, one relationship and nine attributes. The attribute names and name of one entity are picked from the list given by the scenario. The “Module” name was typed by the examiner.

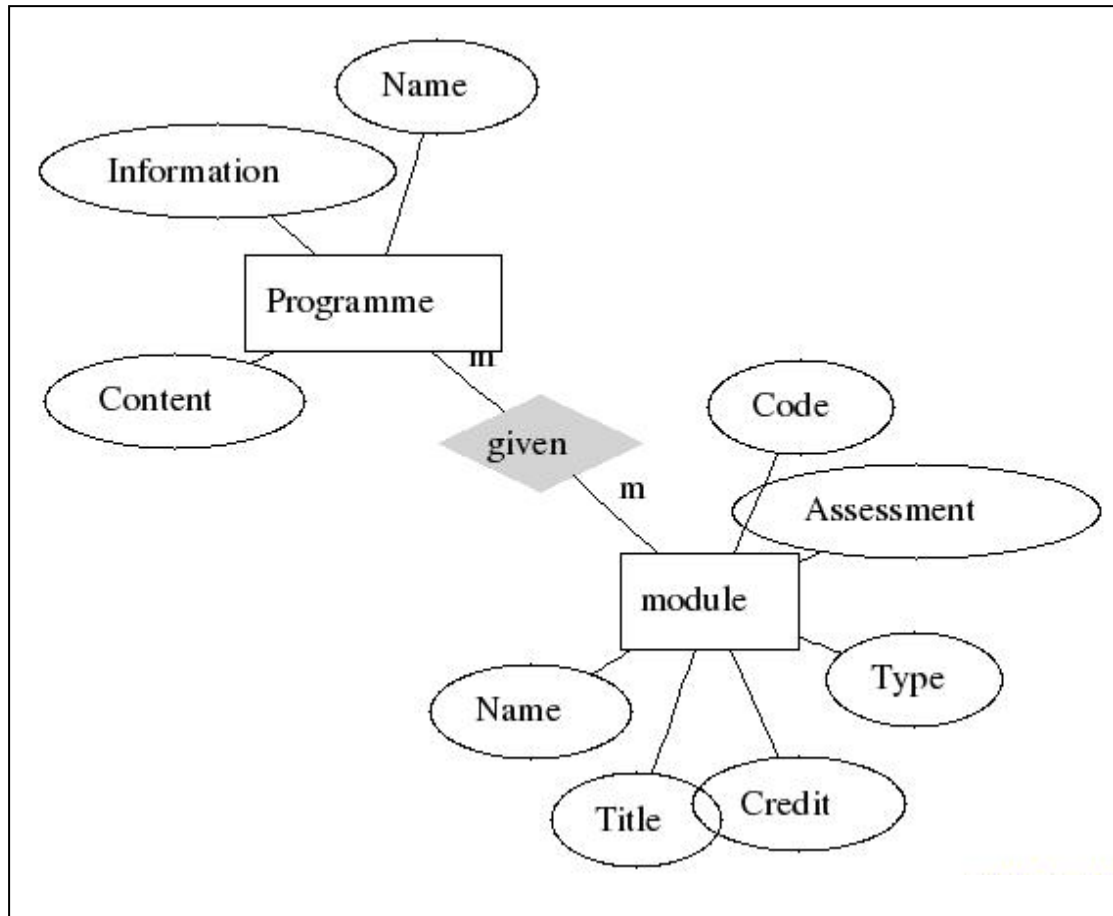


Figure B.7 The solution of Scenario 4

## Reference Diagrams:

Figure B.8 shows two ideal reference diagrams for the entity components in the teacher solution of Scenario 4. The first diagram shows that the “Member” entity has a direct reference to the scenario text in section 3. The second one shows that the “Repeated Event” entity has an indirect reference with a “split” action to the text in section 2.

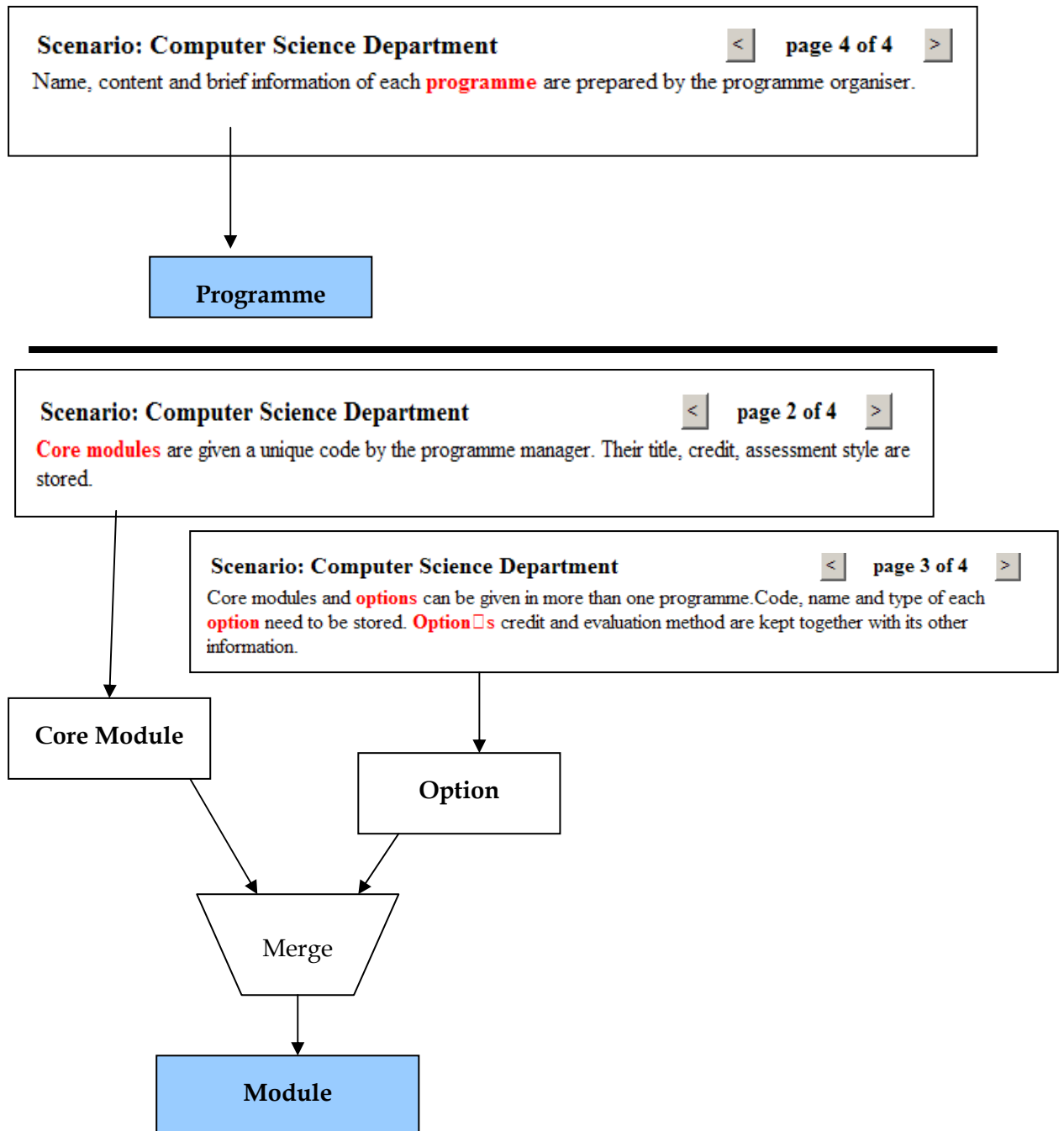


Figure B.8 Ideal reference diagrams for the entity components

## Appendix C

This appendix is for entity relationship diagrams produced by the participants in Experiments 1 and 2 in Chapter 2. The diagram components are shown in tabular format. Attributes of the diagram components are omitted in the table for simplicity.

### Diagram Solutions for Scenario 1

Table C.1 shows the diagram components of the participants' solutions for Scenario 1 in the experiment 1 in Section 7.3. Entity 1 and Entity 2 are the direct referenced components and Entity 3 is the indirect referenced components. Relationship 1 is the relationship between Entity 1 and Entity 3. Relationship 2 is the relationship between Entity 2 and Entity 3.

**Table C.1 Diagram solutions for Scenario 1**

Participant No	Entity1	Entity 2	Entity 3	Relationship 1 (E1-E3)	Relationship 2 (E2-E3)
1	student	course	cOff	enroll	has
2	student	course	courseofferings	can	has
3	student	course	courseofferings	has	has
4	student	course	courseofferings	enrols	has
5	student	course	courseofferings	enrols	may have
6	student	course	courseofferings	enrols	has
7	student	course	courseofferings	enrols on	has
8	student	course	courseofferings	has	is_in

Appendix C

Participant No	Entity1	Entity 2	Entity 3	Relationship 1 (E1-E3)	Relationship 2 (E2-E3)
9	student	course	courseofferings	enrols	has
10	student	course	courseofferings	enrolls on	has
11	student	course	courseofferings	enrol	offered
12	student	course	courseofferings	enrols	has
13	student	course	courseofferings	enrol	have
14	student	course	offerings	enrols	related to
15	student	course	courseofferings	enroll	has
16	student	course	courseofferings	enrol	has
17	student	course	courseofferings	enrols	has
18	student	course	courseofferings	can be on	can be offered
19	student	course	courseofferings	enrols_in	has
20	student	course	courseofferings	enrol	has

## Diagram Solutions for Scenario 2

Table C.2 shows the diagram components of the participants' solutions for Scenario 2 in Experiment 1 in Section 7.3. Entity 1 is the direct referenced components and Entity2 is the indirect referenced components. Relationship is the relationship between Entity 1 and Entity 2.

**Table C.2 Diagram solutions for Scenario 2**

Participant No	Entity 1	Entity 2	Relationship (E1-E2)
1	department	staff	works
2	department	acsupport	workin
3	department	staff	works
4	department	staff	works
5	department	mergeattributes	works
6	department	staff	works
7	department	staffmember	works
8	department	staff	work_in
9	department	staff	works
10	department	staff	works
11	department	staff	worksin
12	department	staff	worksin
13	department	staff	works_within
14	department	people	work

*Appendix C*

<b>Participant No</b>	<b>Entity 1</b>	<b>Entity 2</b>	<b>Relationship(E1-E2)</b>
15	department	people	worksfor
16	department	member of staff	works_in
17	department	staff	works_in
18	department	staff	works_in
19	department	staff	works_in
20	department	staff	work_in



## Diagram Solutions for Scenario 3

Table C.3 shows the main diagram components of the participants' solutions for Scenario 3 in Experiment 2 in Section 7.3. Entity 1 and Entity 2 are the direct referenced components and Entity 3 is the indirect referenced components. Relationship 1 is the relationship between Entity 1 and Entity 3. Relationship 2 is the relationship between Entity 2 and Entity 3. Some students have come up with new components, which are not in the teacher solutions. The table doesn't show these components for simplicity.

Table C.3 Diagram solutions for Scenario 3

Participant No	Entity 1	Entity2	Entity 3	Relationship1 (E1-E3)	Relationship2 (E2:E3)
1	member	event	eventde	helps	has
2	member	event	event offerings		
3	member	event	event offerings		
4	member	event	eventfee		has
5	member	event	event offerings		has
6	member	event	fee		has
7	member	event	eventfile		recorded_in
8	member	event	setevent		has
9	member	event	eventhas	contributes	has details
10	member	event	eventoffer	contribute	have
11	member	event	scheduled_event	helps with	has
12	member	event	events taken	have	has been
13	member	event	repeats		can

Appendix C

<b>Participant No</b>	<b>Entity 1</b>	<b>Entity2</b>	<b>Entity 3</b>	<b>Relationship1 (E1-E3)</b>	<b>Relationship2 (E2:E3)</b>
14	member	event	repeat	helps	can
15	member	event	event fee		set for
16	member	event	eventcost		expenditure
17	member	event	event offerings		for
18	member	event	cost		can have
19	member	event	event repeats		has
20	member	event	popularevent		repeated

## Diagram Solutions for Scenario 4

Table C.4 shows the main diagram components of the participants' solutions for Scenario 4 in Experiment 2 in Section 7.3. Entity 1 is the direct referenced components and Entity2 is the indirect referenced components. Relationship is the relationship between Entity 1 and Entity 2. Some students have come up with new components, which are not in the teacher solutions. The table doesn't show these components for simplicity.

**Table C.4 Diagram solutions for Scenario 4**

Participant No	Entity1	Entity 2	Relationship (E1-E2)
1	programme	module	Given
2	programme	coreoption	GivenIn
3	course	Coremodule	given
4	programme	Modules	part of
5	programme	merged entity	given in
6	programme	option	has
7	programme	programmeitem	includes
8	programme	modules	given_in
9	programme	options	has
10	programme	module	has
11	programme	modules	belongs_to
12	programme	module	has
13	programme	subject	module taught

Appendix C

<b>Participant No</b>	<b>Entity1</b>	<b>Entity 2</b>	<b>Relationship (E1-E2)</b>
14	programme	choice	have
15	programme	program_part	part of
16		deptoptions	
17	programme	module	given_to
18	programme	course	can be part of
19	programme	op_core	given_in
20	programme	module	given

## Appendix D

This appendix is for the questionnaire given to the students in order to get feedback about the semi-automatic assessment tool used in Chapter 7.

### Questionnaire

There are six questions on the questionnaire form. The form was based on a five item Likert scale, giving the user options on how much they would agree with the statements given. The first three questions are about the usability about the editor and the last three questions are about the personalised feedback they received about their works.

Following are the questions on the form.

#### A) The editor:

1. The diagram editor is easy to use.

*[Strongly agree] [Agree] [Neutral] [Disagree] [Strongly disagree]*

2. I like the drag and drop feature for diagramming.

*[Strongly agree] [Agree] [Neutral] [Disagree] [Strongly disagree]*

3. I like the auto diagramming.

*[Strongly agree] [Agree] [Neutral] [Disagree] [Strongly disagree]*

#### B) The feedback:

4. The feedback given for my diagram is clear to understand.

*[Strongly agree] [Agree] [Neutral] [Disagree] [Strongly disagree]*

5. The feedback is sufficient.

*[Strongly agree] [Agree] [Neutral] [Disagree] [Strongly disagree]*

6. I like the colour coded feedback.

*[Strongly agree] [Agree] [Neutral] [Disagree] [Strongly disagree]*

## The results

The questionnaire was given to students at end of the term. 67 of them returned the form. The following diagrams give the results for each question on the form.

**Question 1:** Figure D.1 shows the result of Question 1. The average of the result is 2.33 and the standard deviation is 0.87. The research doesn't deal with the usability issue. Many alternative editors could be designed to get student diagrams. Nevertheless, students are generally happy with the editor. They didn't find it to be a difficult to use the editor.

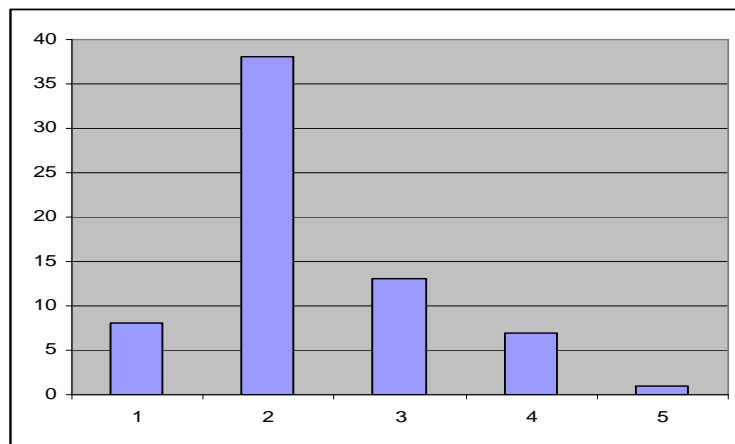


Figure D.1 The result of Question 1

**Question 2:** Figure D.2 shows the result of Question 2. The average of the result is 2.09 and the standard deviation is 0.86. The editor enables students drag noun phrases from the scenario text and drop them on the tool box in order to create diagram components. The result shows that students are happy with the drag and drop feature.

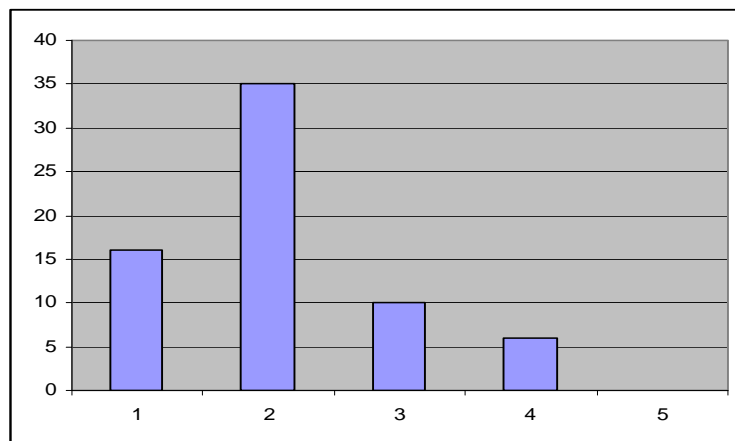


Figure D.2 The result of Question 2

**Question 3:** Figure D.3 shows the result of Question 3. The average of the result is 2.37 and the standard deviation is 0.84. The editor has an auto diagramming feature. The editor changes the diagram layout too much each time new component is added. Some students didn't like this part of the editor.

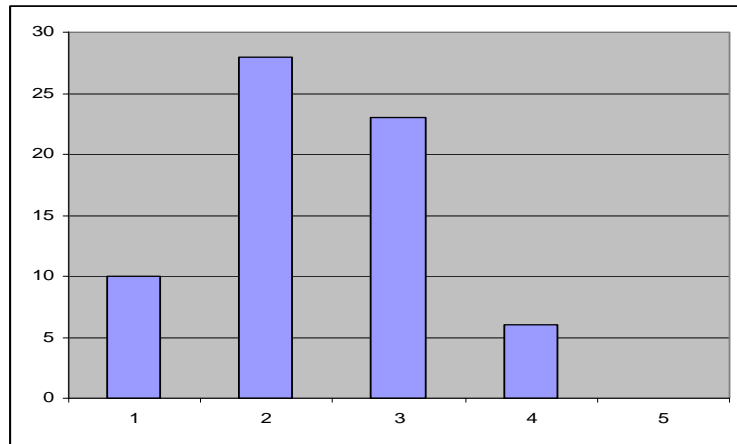


Figure D.3 The result of Question 3

**Question 4:** Figure D.4 shows the result of Question 4. The average of the result is 2.35 and the standard deviation is 0.81. The editor shows the colour coded feedback about student diagrams when they are marked. Most of the students find the feedback clear to understand. The result also shows that the presentation of the feedback needs to be improved.

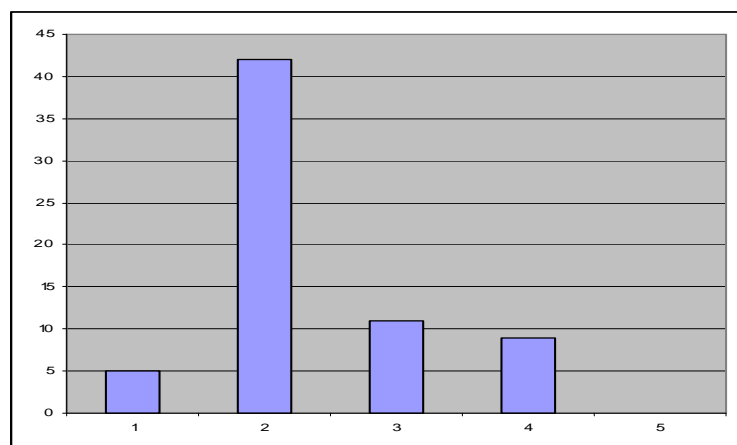


Figure D.4 The result of Question 4

Appendix D

**Question 5:** Figure D.5 shows the result of Question 5. The average of the result is 2.43 and the standard deviation is 0.78. Students can read textual comments given by the examiner on each component in their diagram as well as colour code. The examiner didn't give detailed comments on the student work for these tutorial questions although the editor supports for this. The result shows that the editor should have an automatic commenting feature in order to increase the student satisfaction and the editor should not rely on only the examiner comments.

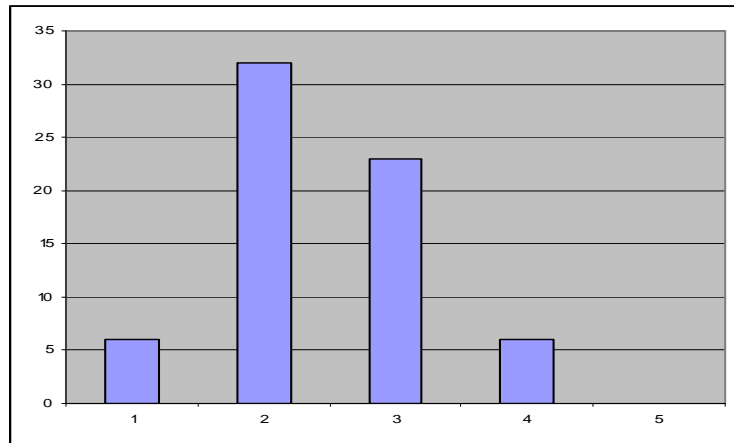


Figure D.5 The result of Question 5

**Question 6:** Figure D.6 shows the result of Question 6. The average of the result is 2.15 and the standard deviation is 0.74. The editor uses a colour code for presentation of feedback about student diagrams. Most of the students like the coloured feedback.

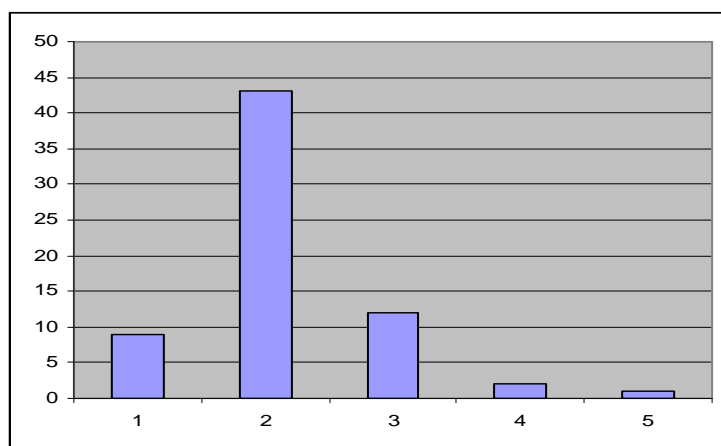


Figure D.6 The result of Question 6



## **Appendix E**

This appendix is for the significant code from the system developed for the proposed framework in the thesis. The system's details are discussed in Chapter 6 and used to evaluate the framework. The evaluation's results are given in Chapter 7.

The system mainly consists of a diagram editor and a marking environment. The important parts of the diagram editor are the function buttons and the automatic diagram drawing components. The important parts of the marking environment are grouping diagram components, matching components and the displaying reference diagrams. A prototype tool for the authoring part of the proposed framework was developed using the Microsoft PowerPoint tool without using any program code and algorithm. The detail of the tool's interface is discussed in Chapter 6. There is not any additional information provided about the authoring tool in this appendix.

The following section is for the conceptual database diagram of the developed system. The diagram helps the understanding of the SQL statements provided in the subsequent sections of this appendix.

### **Conceptual database diagram of the developed system**

The diagram editor of the developed assessment system for this thesis keeps the student diagrams and the scenarios in a relational database. The editor records all the design activities into this database during the diagramming. Figure E.1 shows the entity relationship diagram of the developed system's database. The marking environment of the system uses the same database during marking.

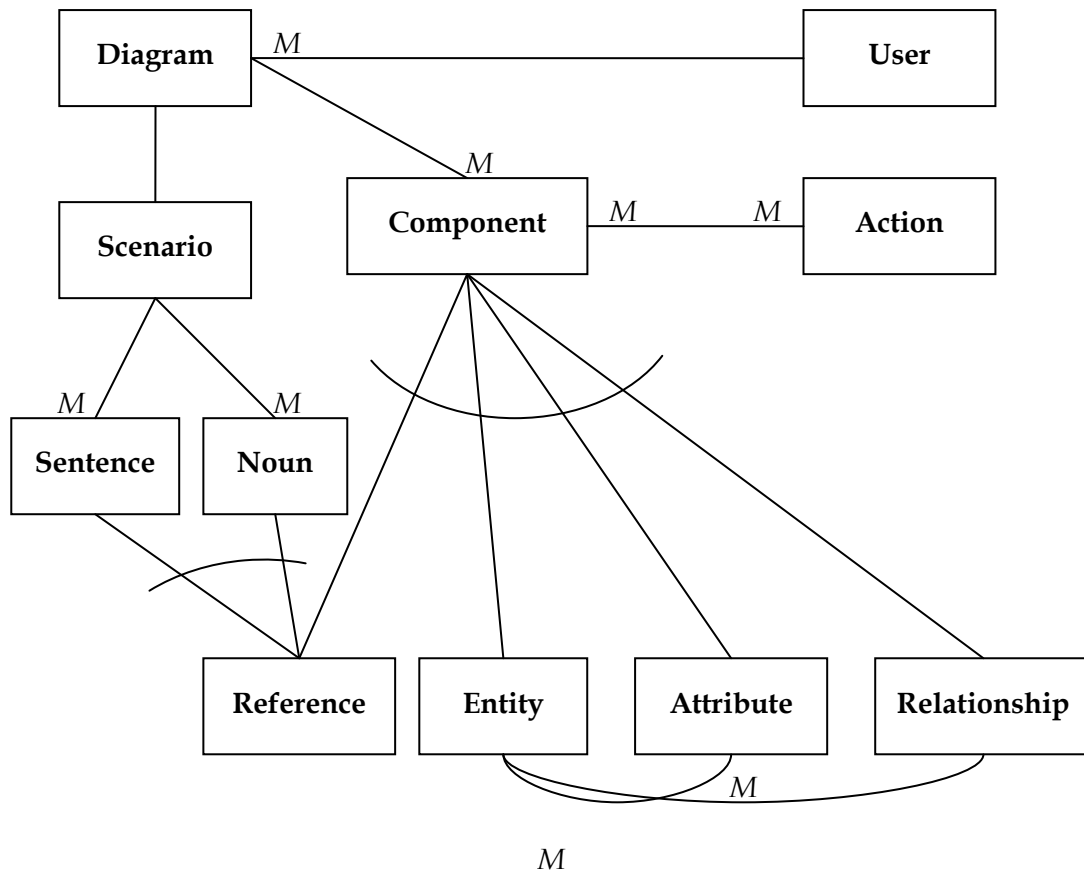


Figure E.1 Entity relationship diagram of the developed system’s database

Following gives a brief explanation of each entity in Figure E.1:

“**User**” entity keeps the user information of the system (e.g. user name and password). A user can be either the examiner or student type.

“**Diagram**” entity keeps the general information about the about diagrammatic slotuons (e.g. user , scenario and submit date).

“**Component**” entity keeps all the components the users produce. Components can be intermediate or final types. The final component is the one the user sees on the diagram canvas of the editor. The intermediate component is the one which is removed from the canvas after the “merge” or “split” operations. The editor keeps the intermediate components and uses during the grouping in order to find the contextual information of the component. The concept of the contextual information concept is explained in Chapter 3.

“**Action**” entity keeps link data to combine the intermediate components. Its pro-action and pre-action attributes keep the identities of the intermediate components.

## *Appendix E*

The action type attribute of the entity is used during generating of the reference diagram of a component.

**“Reference”**, **“Entity”**, **“Attribute”** and **“Relationship”** entity types keeps some additional information about the entries in the “component” entity depending on the type of the diagram component. For example, if the component is an attribute, it keeps the identity of its entity.

**“Sentence”** entity keeps the sentence references of all scenarios used in the assessments and **“noun”** entity keeps the noun phrase references of all the scenarios. The entries in “Reference” entity have a link to either the “sentence” or “noun” entities.

**“Scenario”** entity keeps all the scenario text used in the system. The text of the chosen scenario is displayed on the diagram editor during the diagramming.

## The diagram editor

The diagram editor uses the Graphviz engine, open source graph drawing tool, (Ellson et al. 2002). When a new component is entered into the system, the editor generates a code (namely dot-file) and sends to the Graphviz engine. The engine reads the dot-file and creates a diagram picture file and sends the picture to the editor. The editor displays the picture on the diagram canvas.

### Generation of a dot-file:

The editor reads the components of the user diagram from the system database in order to produce a dot-file for the Graphviz engine. The following PHP program is for the generation of the dot-file for the entities and their attributes of a diagram.

<to create a file to write the code into>

```
$myfile = 'pics/' . $userid . 'basic.dot';
$fh = fopen($myfile, 'w') or die ("cannot open the file");
$Graph = "Graph ";
$ER = "ER";
$openbracket = " {\n";

$startbit = "node [shape=box];";
$attbit = "node [shape=ellipse];";
$relbit = "node [shape=diamond, style=filled, color=lightgrey];";
fwrite($fh, $Graph);
fwrite($fh, $ER);
fwrite($fh, $openbracket);
fwrite($fh, $startbit);
fwrite($fh, "\n");
```

<to write all entities of the diagram to the file>

```
$sql = "SELECT * FROM Component
WHERE did = '$userdiagram' AND type='entity' AND postaction IS NULL";
$res = mysql_query($sql) or die (mysql_error());
while ($myrow = mysql_fetch_array($res)) {
    $id = $myrow['id'];
    $name = $myrow['name'];
    $entstr = "node [label=".chr(34)."$name".chr(34)."] ".$name.$id;

    fwrite($fh, $entstr);
    fwrite($fh, ',');
}
}
```

<to write all attributes of the diagram to the file>

## Appendix E

```
$sql = "SELECT * FROM Component
      WHERE did = '$userdiagram' AND type='attr' AND postaction IS NULL";
$res = mysql_query($sql) or die (mysql_error());
fwrite($fh, "\n");
fwrite($fh, "\n");
fwrite($fh, "\n");
fwrite($fh, $attrbit);
fwrite($fh, "\n");
while ($myrow = mysql_fetch_array($res)){
    $attrname = $myrow['name'];
    $attrid = $myrow['id'];
    $attrstr="node [label=".chr(34)."$attrname".chr(34). " ] $attrname$attrid";
    fwrite($fh, $attrstr);
    fwrite($fh, ';');
    fwrite($fh, "\n");
}
```

<to create links between attributes and entities>

```
$sql = "SELECT * FROM Component
      WHERE did = '$userdiagram' AND type='entity' AND postaction IS NULL";
$entres = mysql_query($sql) or die (mysql_error());
while ($myrow = mysql_fetch_array($entres)){
    $entityid = $myrow['id'];
    $entity= $myrow['name'];
    $entstr=$entity.$entityid;
    $sql2 = "SELECT * FROM AttrName WHERE eid = '$entityid'";
    $attrresult = mysql_query($sql2) or die(mysql_error());
    while ($attrrow = mysql_fetch_array($attrresult)){
        $attrid= $attrrow['aid'];
        $attrname= $attrrow['name'];
        $attrstr= "$attrname$attrid";
        fwrite($fh, "\n");
        fwrite($fh, $entstr);
        fwrite($fh, '---');
        fwrite($fh, $attrstr);
        fwrite($fh, ';');
    }
}
```

<to write the code into the file and send the the Graphviz engine >

```
fwrite($fh, "\n");
fclose($fh);

echo "<img src='/cgi-bin/webdot/users/cofb/ER/" . $myfile . ".neato.png?foo=".md5(time())."'> ";
```

**“Merge” and “Split” actions**

A new component is entered by using one of the command buttons into the system. Later, these components can be merged or split by using function buttons. Following PHP code is the main implementation for these function buttons

<to merge two entities>

```

$userdiagram=$_SESSION['did'];
$entity1 = $_POST['entity1'];
$entity2 = $_POST['entity2'];
$newentityname=$_POST['newentity'];
// testing if entity name is missing
    if ($newentityname == ''){
        echo "<font color =red>*please enter a name for this relationship*";
        echo "<br>";
    }
    else {
// add a new action
        $reftypeno="merge";
        $actionid=addnewaction($reftypeno);
// create merged entity comp
        $entityid =addnewcomp($newentityname,$actionid,"entity",$actionid);
        merge_entity($actionid,$entityid,$entity1);
        merge_entity($actionid,$entityid,$entity2);

function merge_entity($actionid,$mergedentityid,$oldentityid){
    $userdiagram=$_SESSION['did'];

    updateentity($actionid,$oldentityid);

    $query = "SELECT * FROM Attribute WHERE eid = '$oldentityid'";
    $result = mysql_query($query) or die(mysql_error());
    while ($attrow = mysql_fetch_array($result)){
        $attrid=$attrow['aid'];
        duplicateattr($mergedentityid,$attrid);
    }

    $query = "SELECT DISTINCT rid FROM RelParticipant WHERE eid = '$oldentityid'";
    $result = mysql_query($query) or die(mysql_error());
    while ($partrow = mysql_fetch_array($result)){
        $relid=$partrow['rid'];
        duplicaterel($mergedentityid,$relid,$oldentityid);
    }
}

```

<to split an entity>

## Appendix E

```
$newentityname=$_POST['newentname'];
$splitentity= $_POST['entity'];
$s_ent_name=getentityname($splitentity);
$s_ent_actionid=getentityref($splitentity);
// add new action
    $reftype="split";
    $actionid=addnewaction($reftype,$resonno,$rsno);
// create split entity comps
    $newentity =addnewcomp($newentityname,$actionid,"entity",$actionid);
    $c_entity =addnewcomp($s_ent_name,$s_ent_actionid,"entity",$actionid);

    updateentity($actionid,$splitentity);
    split_entity($actionid,$splitentity,$newentity,'newattr');
    split_entity($actionid,$splitentity,$c_entity,'attr');
    addreltoent($splitentity,$c_entity,$actionid);
}

function split_entity($actionid,$splittedentity,$newentityid,$newattr){
    $userdiagram=$_SESSION['did'];
    if (isset($_POST[$newattr])){
        foreach ($_POST[$newattr] as $attr) {
            duplicateattr($newentityid,$attr);
        }
    }
}
```

The next section focuses on the marking environment of the system. The Dot-file generation part of the diagram editor is reused in the marking editor to draw the reference diagram of a component. Each reference diagram is presented to the examiner during the manual partial marking.

## Marking Environment

The marking environment component of the system consists of two parts. The first part processes the student diagrams. It groups the diagram components and match them each other. The second part presents the reference diagrams of the unmatched components to the examiner.

Grouping and matching of the diagram components are implemented by SQL statements. Generation of the reference diagrams are implemented by the PHP code

### Grouping diagram components

The flowing SQL statements find the reference groups of the direct components. The database diagram in Figure E.1 may clarify the SQL statements.

#### <Grouping direct entities>

```
Select count(b) as ent, refid as entity_ref from Reference,  
    (Select R.id as a ,E.id as b from Component as E, Component as R  
        Where E.preaction=R.postaction  
        and R.type="ref"  
        and E.type="entity"  
        and E.postaction is null  
        and Diagram.sid=scenarioid )  
    ) as M  
where a=userrefid  
group by entity_ref;
```

#### <Grouping attributes of direct entities>

```
Select refid as attr_ref,count(a) as numattrgrp, from Reference,  
    (Select R.id as r ,A.id as a from Component as A, Component as R  
        Where A.preaction=R.postaction  
        and R.type="ref"  
        and A.type="attr"  
        and A.postaction is null  
        and Diagram.sid=scenarioid)  
    ) as M
```



## Appendix E

```
where r=userrefid
group by attr_ref
order by numattrgrp;
```

The following SQL statements find the reference groups for indirect components.

### <Grouping indirect entities (Merge)>

```
Select C1.refid, C2.refid from
  (Select postaction, refid from Reference
    ,(Select R.id as a ,E.postaction as postaction
      from Component as E, Component as R
      Where E.preaction=R.postaction
      and R.type="ref"
      and E.type="entity"
      and E.postaction is not null) as M
    where a=userrefid) C2,
  (Select postaction, refid from Reference
    ,(Select R.id as a ,E.postaction as postaction
      from Component as E, Component as R
      Where E.preaction=R.postaction
      and R.type="ref"
      and E.type="entity"
      and E.postaction is not null) as M
    where a=userrefid) C1
where C1.postaction=C2.postaction
and C1.refid <> C2.refid
and C1.postaction in
  (Select actionid From Component, Action
  Where Component.preaction=Action.actionid
  And Component.postaction is null
  and Action.type="merge"
  and Component.type="entity"
  and Component.did in (Select Diagram.did from User, Diagram
  Where User.uid=Diagram.uid and Diagram.sid=scenarioid))
group by C1.refid,C2.refid;
```

## Appendix E

### <Grouping indirect entities (Split)>

Select C1.refid from

```
(Select postaction, refid from Reference
,(Select R.id as a ,E.postaction as postaction
from Component as E, Component as R
Where E.preaction=R.postaction
and R.type="ref"
and E.type="entity"
and E.postaction is not null) as M
where a=userrefid) C1
where C1.postaction in
(Select actionid From Component, Action
Where Component.preaction=Action.actionid
And Component.postaction is null
and Action.type="split"
and Component.type="entity"
and Component.did in
(Select Diagram.did from User, Diagram Where
User.uid=Diagram.uid and Diagram.sid=scenarioid ))
group by C1.refid;
```

### <Grouping relationships>

Select refid as entity\_ref, count(b), from Reference,

```
(Select R.id as a ,E.id as b from Component as E, Component as R
Where E.preaction=R.postaction
and R.type="ref"
and E.type="relation"
and E.postaction is null
and E.did in (Select Diagram.did from User, Diagram Where
User.uid=Diagram.uid and Diagram.sid=scenarioid))
```

### Matching diagram components

The following SQL statements are used to match entity components of the diagrams. Each component, which is matched with any component in teacher solution is accepted as a correct component. They use the scenario references of the components. The database diagram in Figure E.1 may clarify the SQL statements

#### <Matching Direct entity>

Select \* from

(select a.refid,d.sid from directentref a , Diagram d  
where d.did=a.did

group by d.sid, a.refid) S,

(select a.refid, d.sid from directentref a , Diagram d  
where d.did=a.did

and d.uid = **idealsolution#**

group by d.sid, a.refid) T

where S.refid=T.refid

#### <Matching merged entity>

Select \* from

(Select ref1, ref2, sid from Diagram, mergeentbrefs

Where Diagram.did= mergeentbrefs.did

group by sid, ref1,ref2) S,

(Select ref1, ref2, sid from Diagram, mergeentbrefs

Where Diagram.did= mergeentbrefs.did

and Diagram.uid = **idealsolution#**

group by sid, ref1,ref2) T

Where S.ref1=T.ref1 and S.ref2=T.ref2

#### <Matching split entity>

Select \* from

(Select refid, sid from Diagram, splitentrefs

Where Diagram.did= splitentrefs.did

group by sid,refid) S,

(Select refid, sid from Diagram, splitentrefs

## Appendix E

Where Diagram.did= splitentrefs.did

and Diagram.uid = idealsolution#

group by sid,refid) T

Where S.refid=T.refid

The similar SQL statement are written to use for matching the attribute and relationship components in the systems

### Generation of the reference diagrams

This part of the system creates the reference diagrams of the components, which is not match with any components of the teacher solutions. The reference diagrams are presented to the examiner for manual marking. The examiner may accept or reject these components.

The following PHP code is used to generate the reference diagrams of the components.

```
function showref($entity){
    $Graph = "digraph ";
    $ER= "g";
    $openbracket = " {\n";
    $closebracket = "\n }\n";
    $startbit = "node [shape=box];";
    $operation = "node [shape=ellipse];";

    $userdiagram=$_SESSION['did'];
    // start making the dot file
    $myfile = 'ref.dot';
    $fh = fopen($myfile,'w') or die("cannot open the file");

    fwrite($fh,$Graph);
    fwrite($fh,$ER);
    fwrite($fh,$openbracket);

    makepic($entity,$fh);

    fwrite($fh, $closebracket);

    echo "<img src='/cgi-bin/webdot/users/cofb/ER/ref.dot.dot.png?foo=".md5(time())."'> ";
}
```

The main function of the code is “makepic” . The following is an extract from the function.

<Initialisation>

```

$writerel=false;
$firstent=false;
$sgneeded=false;
$sql = "SELECT * FROM Component WHERE id='$compid'";
$entres = mysql_query($sql);
$entrow = mysql_fetch_array($entres);
$actionid=$entrow['preaction'];
$comptype=$entrow['type'];
$name=$entrow['name'];
$id=$entrow['id'];
if ($comptype=="entity") {
    $compshape="box";
} else if ($comptype=="attr") {
    $compshape="ellipse";
} else if ($comptype=="relation") {
    $compshape="diamond";
}

```

<creating the component picture>

```

$compstr="$name$id [shape=$compshape, label=\"$name\"]; ";
fwrite($fh,$compstr);

```

<creating the actiont picture [1 for merge and 2 for split action] >

```

$sql = "SELECT * FROM Action WHERE actionid='$actionid'";
$actres = mysql_query($sql);
$actionrow = mysql_fetch_array($actres);
$actiontype=$actionrow['type'];
if ($actiontype==1) {
    $atype="Scenario";
    $ashape="hexagon";
} else if ($actiontype==2) {
    $atype="Merge";
    $ashape="invtrapezium";
} else if ($actiontype==3) {
    $atype="Split" ;
    $ashape="trapezium";
    $sgneeded=true;
}
$actvar="action$atype$actionid";
$actionstr="node [shape=$ashape, label=\"$atype\"] $actvar;";
fwrite($fh,$actionstr);

```

<creating links between the action shape and the component shapes>

```

    $sql = "SELECT * FROM Component WHERE postaction='$actionid' ORDER BY type DESC";
    $compres = mysql_query($sql);
    while ($comprow = mysql_fetch_array($compres)){
        $type = $comprow['type'];
        if ($type == 'entity') {
            $sid = $comprow['id'];
            makepic($sid,$fh);
            $name = $comprow['name'];
            $comp=$name.$sid;

            fwrite($fh,"\n");
            fwrite($fh,$comp);
            fwrite($fh,'->');
            fwrite($fh,$actvar);
            fwrite($fh,');');
            $firstent=True;
        }else if ($type == 'relation') {
            $sid = $comprow['id'];
            if (checkrelation($sid)){
                $relid=$sid;
                $name = $comprow['name'];
                $relstr=$name.$sid;

                $writere1=True;
            }
        }
    }
    // this is to show a relationship between two entities
    if ($writere1){
        if ($firstent){
            makepic($relid,$fh);
            fwrite($fh,"\n");
            fwrite($fh,$relstr);
            fwrite($fh,'->');
            fwrite($fh,$actvar);
            fwrite($fh,');');
            $writere1=False;
        }
    }
}

```

This part of the “makepic” function is recursive. Since the reference diagrams can be malformed.

This developed system was used to evaluate the framework. The light version of the system was also developed and used in a taught module. However its sample code is not provided here since the approach is the similar to the system presented here.

## PUBLICATIONS

This section gives the details of the publications produced as a result of this research. The abstract of each publication are also included and the links to the thesis is made.

### **Peer Reviewed Conference**

#### **Paper 1: A Diagram Drawing Tool for the Semi-automatic Assessment of Conceptual Database Diagrams**

This is the first paper published. The paper introduces a tool which is a standalone diagram editor. It is a proof of concept tool. It is used in the experiment 3 in Section 7.3. Google Scholar shows that the paper has been cited 12 times since 2006.

#### Abstract

The increased number of diagram based questions in higher education has recently attracted researchers to look into marking diagrams automatically. Student diagrammatic solutions are naturally very dissimilar to each others. However, it has been observed that there are a number of identical diagram components. This observation forms the basis of our semi-automatic assessment. Identifying identical diagram components in student diagrams needs contextual information about each component. This paper proposes a diagram tool which obtains the contextual information of each component in a conceptual database diagram.

#### Reference

Batmaz, F & Hinde, CJ 2006, 'A diagram drawing tool for semi-automatic assessment of conceptual database diagrams', *Proc. of the 10th CAA Conference*, Loughborough University. Loughborough, UK pp.71-84.

#### **Paper 2: A Web-Based Semi-Automatic Assessment Tool for Conceptual Database Diagram**

An online version of the previous diagram editor in the first paper is developed. The tool is used in the experiment 1 and 2 in section 7.3. The paper covers the initial findings of the experiments and also introduces the marking part of the semi-

## Publications

automatic approach mentioned in Chapter 6. Google Scholar shows that the paper has been cited 4 times since 2007.

### Abstract

The increased number of diagram-type student work in higher education has recently attracted researchers to look into the automation of diagram marking. This paper proposes a new (semi-automatic) marking approach to reduce number of the diagram component marked by the human marker. We believe this approach improves the marking consistency and has potential to provide individualised and detailed feedback to students with mark-ups. We have developed a prototype web-based diagram drawing and marking tools for the approach. The initial experiment and findings for the tools are described in the paper.

### Reference

Batmaz, F & Hinde, CJ 2007, 'A\_Web-Based Semi-Automatic Assessment Tool for Conceptual Database Diagram', *Proc. of the 6th Web-Based Education conference*, ACTA Press, Anaheim, CA, USA, pp.427-432.

## **Paper 3: A Method For Controlling The Scenario Writing For The Assessment Of Conceptual Database Model**

The paper covers some parts of the scenario writing in section 5.3. It introduces the scenario writing environment in Chapter 6. It gives the initial findings of the experiment in Section 7.2. Google Scholar shows that the paper has been cited 1 time since 2008.

### Abstract

This paper proposes a method for semantically controlling scenario text writing in natural language. The scenario text is used for semi-automatic assessment of student translation of those scenarios into database diagrams. These scenarios increase the automation of the marking process and enable the scenario texts to be categorised in difficulty levels. An experimental tool has been implemented for this method. The initial experiments and findings for the interface are described in the paper.



## Publications

### Reference

Batmaz, F & Hinde, CJ 2007, 'A Method For Controlling The Scenario Writing For The Assessment Of Conceptual Database Model', *Proc. of Computers and Advanced Technology in Education*. ACTA press, Calgary, Canada, pp.614-804.

### **Paper 4: Personalised Feedback With Semi-Automatic Assessment Tool For Conceptual Database Model**

A new version of the web-based semi-automatic assessment tool in the paper 2 has been developed. The tool is for simple scenario typed questions. It gives colored feedback to students. The paper highlights the feedback features of the tool which is briefly mentioned in section 7.5. It also gives the initial findings of the experiment in section 7.5.1.

### Abstract

The increased presence of diagram-type student work in higher education has recently attracted researchers to look into the automation of diagram marking. This paper introduces web-based diagram drawing and marking tools for a new (semi-automatic) assessment approach. The approach reduces the number of diagram components marked by the human marker and provides individualised and detailed feedback to students. The tools which have been used in tutorials of a first year database module in the Computer Science department at Loughborough University are described together with findings from the usage of the tools.

### Reference

Batmaz, F, Stone, R & Hinde, CJ 2009, 'Personalised Feedback With Semi-Automatic Assessment Tool For Conceptual Database Model', *Proc. of the 10th Annual Conference of the Higher Education Academy Subject Centre for Information and Computer Sciences*, University of Ulster, UK, pp.115-120.

### **Paper 5: A Multi-Touch ER Diagram Editor to Capture Students' Design Rationale**

A new user interface is developed for the diagram editor in Paper 4. The interface is outside of the main research in the thesis. It has been mentioned in 8.4 Future Directions section. The interface uses multi touch technology. The paper introduces this new interface and gives the results of the initial experiments.

## Publications

### Abstract

The increased presence of diagram-type student work in higher education has recently attracted researchers to look into the automation of diagram marking. Research into the semi-automatic diagram assessment at Loughborough University has identified the requirements of a diagram editor in order to capture the students' design rationale. To fulfil these requirements, several experimental diagram editors have been developed. This paper introduces an ER diagram editor which uses multi touch technology. The initial experiments and findings for the editor are described in the paper.

### Reference

Stone, R, Batmaz, F & Rickards, T 2010, 'A Multi-Touch ER Diagram Editor to Capture Students' Design Rationale', *Proc. of International Conference on Education and Information Technology*.

## **Peer-Reviewed Journal**

### **Paper 6: Drawing and Marking Graph Diagrams**

A new version of the web-based semi-automatic assessment tool in Paper 2 has been developed. The tool is for simple scenario typed questions. It gives coloured feedback to students. The paper highlights the user interface part of the tool which is briefly mentioned in section 7.5. It also gives the initial findings of the experiment in section 7.5.1.

### Abstract

The marking of graph diagrams (that is to say diagrams that are composed of nodes, possibly joined by edges) is tedious if the diagrams are presented on paper. If the key content of the diagrams is available in electronic form then the marking can be much more efficient. This is achieved because the tutor only has to mark each different diagram element once and this mark is transmitted to all diagrams that contain the element. This benefit to the tutor is obtained by requiring the students to use a diagram drawing program of some kind. However using such an editor can simplify the process for the students by allowing them to concentrate more on the problem and less on its graphical representation. The students can also be rewarded for going to this extra effort by receiving a much more detailed, personalised

## Publications

commentary on their work than would have been possible before, given the same amount of tutor time.

Keywords: Marking, diagrams, ER diagrams, graphs, UML, drag-and-drop

### Reference

Stone, R, Batmaz, F & Hinde, C 2009, 'Drawing and Marking Graph Diagrams', *Italics*, vol.8, no.2, pp.45-53.

## **Paper 7: Personalised Feedback With Semi-Automatic Assessment Tool For Conceptual Database Model**

Paper 4 was awarded best paper at the 10th annual conference of the ICS HE academy and published in the *Italics* journal.

### Reference

Batmaz, F, Stone, R & Hinde, CJ 2009, 'Personalised Feedback With Semi-Automatic Assessment Tool For Conceptual Database Model', *Italics*, vol.9, no.1, pp.105-110.

## **Grant awarded**

The semi-automatic assessment of student diagrams developed in the research helps the consistency of feedback on the solutions. A HEA development fund grant has been given to this research to develop a web assessment tool. The tool is available to all university students in the UK. The findings of using the tool in the class are published in Paper 4 and Paper 6.

### Project Brief

The intention of this work is to provide computer assistance not only to the marking phase but also to other phases of the current manual diagram assessment process. The aim is to reduce or remove as many of the repetitive tasks in any phase of the process as possible. As the same tasks are performed less (possibly only once) by the examiners, consistency of grades and feedback on the solutions are achieved.

### Reference

Hinde, C.J., Batmaz, F, Stone, R, 2008, A Web-Based Semi-Automatic Assessment Tool For Conceptual Database Model, HEA development fund, 2008/09, [http://www.ics.heacademy.ac.uk/projects/development-fund/fund\\_details.php?id=125](http://www.ics.heacademy.ac.uk/projects/development-fund/fund_details.php?id=125).