

Contents

List of Boxes	iii
List of Figures	iv
List of Listings	vii
List of Tables	viii
Acknowledgement	ix
Declarations	x
Abstract	xi
Abbreviations	xii
Introduction.....	1
A RADICAL SHIFT IN PERSPECTIVE ON COMPUTER USE?.....	1
INTRODUCING MODELLING WITH DEFINITIVE SCRIPTS	4
CONTEXT FOR THE THESIS.....	8
CONTENT OF THE THESIS.....	10
1 Selected dependency-based applications.....	15
1.1 DEPENDENCY IN CLOSED WORLD AND OPEN DEVELOPMENT	15
1.2 SPREADSHEETS	19
1.3 GRAPHICAL MODELLING	22
1.4 RELATIONAL QUERY LANGUAGES.....	24
1.5 OTHER DEFINITION-BASED APPLICATIONS	26
1.6 MODELLING WITH DEFINITIVE SCRIPTS (MWDS).....	29
2 Principles of MWDS	38
2.1 WHAT IS A DEFINITIVE SCRIPT?.....	38
2.1.1 The characteristics of definition and dependency	40
2.1.2 Review of computer support tools	43
2.1.3 Illustrative examples of using definitive scripts.....	52
2.2 SINGLE-AGENT MODELLING	57
2.2.1 General characteristics of single-agent MWDS	58
2.2.2 The role of MWDS in construal.....	68
3 Abstract Definitive Modelling.....	77
3.1 THE ABSTRACT DEFINITIVE MODELLING FRAMEWORK.....	77
3.2 ILLUSTRATING THE ADM FRAMEWORK	87
3.2.1 Observation and interpretation in single-agent modelling	87

3.2.2	Comprehension of state in multi-agent modelling	94
3.2.3	The internal semantic relation	100
3.2.4	Open and closed interaction	108
4	MWDS as an activity	114
4.1	A CASE STUDY IN MWDS	114
4.1.1	Constructing the data model	115
4.1.2	Developing the interface	118
4.1.3	Using the instrument	122
4.2	CHARACTERISTIC FEATURES OF MWDS	125
4.2.1	Interaction	125
4.2.2	Comprehension	128
4.2.3	Discovery	129
4.2.4	Extension	130
5	Representing state and behaviour in MWDS	132
5.1	OPEN-DEVELOPMENT AND CLOSED-WORLD VIEWS ON DEVICES	133
5.2	THE SEMI ASPECTS OF STATE IN MWDS	139
5.3	ADM DEVICES AND PROGRAM DEVICES	145
5.3.1	Explicit & internal aspects of state in ADM and program devices	146
5.3.2	Situational & mental aspects of state in ADM and program devices	154
6	MWDS and the theory of computation	162
6.0	OVERVIEW	163
6.1	INTRODUCTION TO THE HEAPSORT ALGORITHM	163
6.1.1	Conventional implementation of heapsort	164
6.1.2	The structure of the definitive Heap model	166
6.1.3	Introducing agency to the model	171
6.2	RELATING EXPERIENTIAL AND FORMAL VIEWS OF HEAPSORT	174
6.2.1	Integrating the Heap model with a formal specification	174
6.2.2	The Heapsort device as a non-standard heapsort model	177
6.2.3	Construals of heapsort	179
7	Problematic issues and possible solutions	181
7.1	PROBLEMATIC ISSUES IN MWDS	181
7.1.1	Problems with constructing a model with definitive scripts	181
7.1.2	Problems in the maintenance of large scripts	188
7.1.3	Evaluation efficiency	192
7.2	POSSIBLE SOLUTIONS	193
7.2.1	Techniques to help in comprehending definitive models	194
7.2.2	Techniques to help in automatically generating definitive scripts	202
	Conclusion	205
	Bibliography	208
	Appendix A	223
	Appendix B	225

List of boxes

Box 6-1: The pseudo code for the heapsort process.....	198
Box 6-2: Eden actions to represent the automatic agents in heapsort	207
Box 6-3: A formal specification for the heapsort process cited from [BRS00]	209
Box B-1: The scenario of the Clayton Tunnel railway accident (from [Bey99]).....	227

List of Figures

Figure 1: A definitive script for a simple electrical circuit	6
Figure 2: Visualisation for a switch	6
Figure 1-1: A diagram to illustrate the concept of a declarative constraint	17
Figure 1-2: A diagram to illustrate explicit acyclic dependency.....	19
Figure 1-3: Acyclic dependency to specify a constraint satisfaction method	20
Figure 1-4: A simple agentsheet.....	30
Figure 1-5: The interpretation of state in open-development modelling.....	37
Figure 1-6: The interpretation of state in closed-world modelling.....	38
Figure 1-7: Scripts to illustrate modelling activity in open-development sense	39
Figure 2-1: A transition of state in definitive scripts.....	48
Figure 2-2: The dependency network diagrams	49
Figure 2-3: MWDS in tkeden.....	53
Figure 2-4: The integration between Eden, DoNaLD and Scout in (d)tkeden	58
Figure 2-5: The Jugs model with an array of columns to display the current values of key variables	61
Figure 2-6: Scripts and screenshots of a geometric model representing a ‘door’	63
Figure 2-7: The dependency network diagram for the script in Figure 2-6	64
Figure 2-8: Illustrating the use of Eddi	65
Figure 2-9: (a) the tabular view of the ALLFRUITS relation (b) a graphical presentation of the fruit of current interest (i.e. ‘granny’) (c) the various extracts from types of definitive script in the model.....	66
Figure 2-10: The explorer’s understanding of a situation	69
Figure 2-11: Modelling the modeller’s understanding.....	70
Figure 2-12: Two views of the semantic relation (a) within the computer state, (b) between the model and the external situation	72
Figure 2-13: Various representations of a triangle taken from [FP89]	75
Figure 2-14: A screenshot of two different perspectives on representing a triangle.....	76
Figure 2-15: The seed for the Number model	76
Figure 2-16: Several example variants of the Number model.....	77
Figure 2-17: MWDS for computer-based construal.....	81
Figure 2-18: Definitive script as observer’s model of state (‘one-agent’ modelling)	84
Figure 2-19: Definitive script as observer’s model of state (‘multi-agent’ modelling).....	86
Figure 2-20: An LSD account of the electrical circuit (cf. Figure 1-4).....	87
Figure 3-1: The Jugs-and-Door artefact	93

Figure 3-2: An LSD account/specification of the Jugs model/program.....	94
Figure 3-3: Screenshot and script extract for the text-interface Jugs model	95
Figure 3-4: The ADM from the machine perspective	97
Figure 3-5: The ADM from the human perspective.....	98
Figure 3-6: An ADM artefact for Jugs	100
Figure 3-7: A screenshot of the Room Viewer model and its script	104
Figure 3-8: A screenshot of the Lines model and its script.....	108
Figure 3-9: A screenshot of Classroom and pupil's behaviour decision diagram.....	112
Figure 3-10: Screenshots of the Railway Accident model running on different machines	115
Figure 3-11: The data representation diagram for a specific car (i.e. Ford_Escort).....	120
Figure 3-12: Transforming a car from a saloon to a convertible.....	121
Figure 3-13: The use of Eddi in the Car History model.....	121
Figure 3-14: A screenshot of the Timeline Record model	125
Figure 3-15: (a) Screenshots of the form design application, (b) using i option	126
Figure 3-16: A screenshot of the MBF model.....	130
Figure 3-17: A screenshot of the Digital Watch model.....	132
Figure 4-1: A screenshot of running the Temposcope	143
Figure 4-2: Illustrating automatic script generating with virtual agents	144
Figure 4-3: The Temposcope – an interactive timetabling model.....	147
Figure 4-4: (a), (b) Changing the availability of the staff member	152
Figure 4-5: A client application for a staff member to input his or her availability.....	156
Figure 5-1: The four aspects of state in using a device (adapted from [BRWW01]).....	163
Figure 5-2: Situational, Explicit and Mental aspects of state in digital watch use (adapted from [BRWW01]).....	164
Figure 5-3: A snapshot of the Jugs device with its script (for internal and explicit).....	169
Figure 5-4: Explicit state for the Jugs device	169
Figure 5-5: Changing the Jugs model to illustrate the mental state	171
Figure 5-6: The Jugs device with the time constraint	172
Figure 5-7: Distributed use of the Jugs device (taken from [Sun99])	173
Figure 5-8: The Jugs model with the evaporating liquid.....	174
Figure 5-9: Data representation in (a) a traditional program, and (b) MWDS.....	180
Figure 5-10: The Jugs model: (a) conventional, (b) definitive approach	181
Figure 5-11: The Jugs device: (a) pixel per unit display, (b) pixel per 2 units display	182
Figure 5-12: The richer visualisation for the Jugs device in Figure 5-11	183
Figure 5-13: The CSP specification for the VMC from [Hoare85, p30].....	189
Figure 5-14: The definitive VMC model	192
Figure 6-1: An array as a heap	196

Figure 6-2: A screenshot of the definitive Heap model	200
Figure 6-3: The geometrical representations of the array and the associated tree.....	202
Figure 6-4: Agents to automate heapsort	206
Figure 6-5: An extended version of heapsort with its formal specification	210
Figure 7-1: The state changing in definitive scripts	220
Figure 7-2: The file structures and their timestamps.....	223
Figure 7-3: Delaying dependency maintenance	224
Figure 7-4: An example of a triggered action	226
Figure 7-5: Illustrating of using <i>split-tkeden</i>	235
Figure 7-6: Screenshots of running the reporting tools on the Car History script.....	237
Figure 7-7: The Car History model with additional script generated from the tools	239
Figure 7-8: Special-purpose script extension of the Lines model	241
Figure 7-9: Generating scripts using a template.....	244
Figure B-1: A screenshot of the Room Viewer model developed by MacDonald.....	228
Figure B-2: A screenshot of the extended Room Viewer Model by Carter	228

List of Listings

Listing 2-1: Ttyeden input and output for the file script.e	52
Listing 3-1: LSD account for Classroom Interaction	114
Listing 3-2: The adm definition for Classroom Interaction.....	114
Listing 3-3: Script to record details of cars	119
Listing 3-4: The script for computing whether a given MBF is planar computable	129
Listing 4-1: Examples of definitions from the data model in the Temposcope.....	138
Listing 4-2: Examples of definitions from the data capture in the Temposcope.....	139
Listing 4-3: Examples of definitions for developing the interface in the Temposcope.....	142
Listing 4-4: Script extract generated by the technique described in Figure 4-2.....	144
Listing 5-1: Converting a definitive script to a procedural program.....	184
Listing 5-2: Extract of the script for the VMC model.....	191
Listing 7-1: Extracts from the scripts in the car-history (.e, .d, .s) files	235

List of Tables

Table 1-1: The advantages and disadvantages of spreadsheets.....	24
Table 5-1: Contrasting the characteristics of typical devices and ADM artefacts	160
Table 7-1: A list of reporting tools.....	236

Acknowledgements

I would like to express my great gratitude to my supervisor, Meurig Beynon, who always encouraged and guided me during these years of research. Thanks to Meurig for picking me up out of the depths of despair and for his invaluable help throughout the research and writing of the thesis.

I also would like to thank to my friends and colleagues in the Empirical Modelling group for providing a warm, friendly and stimulating research environment during my study here.

I am so very thankful to Beth who helped in proof reading my entire thesis. Without her generous help, this thesis could not be completed.

Finally, my deepest gratitude goes to my parents, brothers and sisters for their love and consistent support. Without their encouragement and love, I could not possibly complete such a challenge as this study.

Further thanks goes to Martin Loomes and Jane Sinclair for their constructive comments during the viva and to Meurig Beynon for his useful help on this final version.

Declarations

This thesis is presented in accordance with the regulations for the degree of Doctor of Philosophy. It has been composed by myself and has not been submitted in any previous application for any degree. The work in this thesis has been undertaken by myself except where otherwise stated.

Abstract

Modelling with definitive scripts (MWDS) is an alternative approach to computer-based modelling. It enables the modeller to develop a computer-based artefact in which his/her experience of a situation or subject of interest is embodied, in much the same way that the experimental scientist devises a physical model to represent a phenomenon, or an investor creates a spreadsheet to model his financial situation. In MWDS, the dependencies amongst observables in the situation are faithfully reflected in the relationships amongst variables that are linked by spreadsheet-like definitions. The values of such variables may themselves be directly associated with observable features of the computer. As a treatise on modelling with definitive scripts, the thesis will frame the concept of MWDS, discuss its background and history, introduce the tools that have been developed to support it, illustrate and document its use with reference to many diverse examples, compare and contrast MWDS with alternative approaches to modelling and programming, and identify and – to some extent – address problematic issues surrounding its application.

The abstract definitive modelling framework (ADM), based on the concepts of observable, dependency and agency, is introduced. With reference to an ADM artefact, the versatility, flexibility and extensibility of MWDS is illustrated and elaborated to show the potential for using MWDS to support universal agent-oriented modelling. The model developed in this framework is open-ended, flexible to change and easy to extend to reflect changes both in the modeller's perspective and in the external situation.

Rather than focusing on representing behaviour, MWDS emphasises the representation of state as experienced and perceived. This makes MWDS distinctively different from classical procedural computation in character. Every state in a definitive model is interpretable, whereas not every state in a classical procedural program can be. However, MWDS may help in comprehending the program if the program is treated as a physical artefact whose behaviour is to be construed by the modeller.

The thesis describes how MWDS provides an open-ended framework for the modeller to interactively cultivate and refine a computer-based artefact in which several roles of agents and modes of observation can be embodied. It also discusses how such artefacts can be specialised and treated as devices with specific modes and conventions of use. In this context, the development of the model can be seen as the collaborative integration of the roles of the designer and the user.

Abbreviations

adm – Abstract Definitive Machine

ADM – Abstract Definitive Modelling framework

ARCA – Definitive notation for displaying geometric diagrams

CSP – Communicating Sequential Processes

DoNaLD – Definitive Notation for Line Drawing

Eddi – Eden Definition Database Interpreter

Eden – Evaluator for Definitive Interpreter

EM – Empirical Modelling

GUI – Graphic User Interface

HCI – Human-Computer Interaction

ISBL – Information Systems Base Language

ISM – Interactive Situation Model

L.E.G.O. – An interactive computer graphics system for teaching geometric

LSD – Language for Specification and Description

MVC – Model View Controller

MWDS – Modelling with definitive scripts

OLE – Object Linking and Embedding

OO – Object-Oriented; Object-Oriented

PDL – Pictorial Description Language

PRTV – The Peterlee Relational Test Vehicle

Scout – Definitive Notation for Screen Layout

UML – Unified Modelling Language