

AUTHOR: Ashley Thomas Ward DEGREE: Ph.D.

**TITLE: Interaction with Meaningful State:
Implementing Dependency on Digital Computers**

DATE OF DEPOSIT:

I agree that this thesis shall be available in accordance with the regulations governing the University of Warwick theses.

I agree that the summary of this thesis may be submitted for publication.

I **agree** that the thesis may be photocopied (single copies for study purposes only).

Theses with no restriction on photocopying will also be made available to the British Library for microfilming. The British Library may supply copies to individuals or libraries, subject to a statement from them that the copy is supplied for non-publishing purposes. All copies supplied by the British Library will carry the following statement:

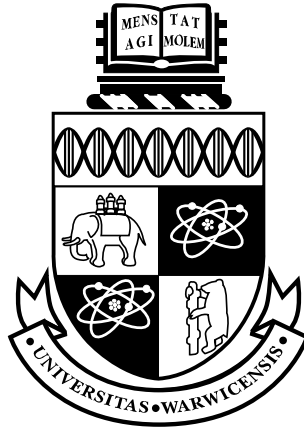
“Attention is drawn to the fact that the copyright of this thesis rests with its author. This copy of the thesis has been supplied on the condition that anyone who consults it is understood to recognise that its copyright rests with its author and that no quotation from the thesis and no information derived from it may be published without the author’s written consent.”

AUTHOR’S SIGNATURE:

USER’S DECLARATION

1. I undertake not to quote or make use of any information from this thesis without making acknowledgement to the author.
2. I further undertake to allow no-one else to use this thesis while it is in my care.

DATE	SIGNATURE	ADDRESS
.....
.....
.....
.....
.....



**Interaction with Meaningful State:
Implementing Dependency on Digital Computers**

by

Ashley Thomas Ward

Thesis

Submitted to the University of Warwick

for the degree of

Doctor of Philosophy

Department of Computer Science

May 2004

THE UNIVERSITY OF
WARWICK

*To Uncle Adam, Uncle Colin,
Mr Cheal, Mikki Larcombe
and Meurig with thanks*

Contents



List of Figures	iv
List of Tables	viii
List of Listings	ix
Acknowledgements	xi
Declaration	xii
Abstract	xiii
Abbreviations	xiv
Chapter 1 Introduction	1
1.1 Dependency in concept: EM, Radical Empiricism and the making of meaning	3
1.2 Dependency in application: Interaction with Meaningful State	10
1.3 Dependency in development: a novel abstraction	19
1.3.1 Modelling requirements	21
1.3.2 Program comprehension and validation through abstraction	22
1.4 Dependency in engineering: simple and consistent relationships	26
1.5 Related work	32
1.6 Thesis aims	41
1.7 Thesis outline	43
Chapter 2 The Abstract Definitive Machine, ADM	46
2.1 Concepts of the LSD notation and the ADM	47
2.1.1 LSD accounts	47
2.1.2 1-agent modelling with the ADM	52
2.1.3 ‘Programming’ the ADM	54
2.1.4 From LSD accounts to ADM scripts	58
2.2 Implementations of the ADM	60
2.2.1 Evaluation/storage implementation strategies for definitive systems	60
2.2.2 Existing implementations of the ADM	61
2.2.3 A hardware implementation?	63
2.3 Operational semantics	64
2.3.1 Operational semantics of LSD	64

2.3.2	Invalid transitions in the ADM	67
2.3.3	What's in a transition?	69
2.3.4	Divisible command lists and the 'Authentic' ADM (AADM)	74
2.4	The ADM and UNITY	81
2.5	Background/Sources	88
2.A	Using the <code>am</code> implementation	91
Chapter 3 The Definitive Assembly Maintainer (DAM) machine		97
3.1	The DMM and the BRA	99
3.1.1	State and the DMM	99
3.1.2	Transitions and the block redefinition algorithm (BRA)	103
3.1.3	Efficiency and generality of the DMM	107
3.2	The DAM machine, from the bottom up	109
3.2.1	The platform basis for the DAM machine	110
3.2.2	Values	112
3.2.3	Operators	112
3.2.4	Data structure	113
3.2.5	DAM execution	117
3.2.6	External agency	120
3.3	!Donald	123
3.3.1	!Donald overview	124
3.3.2	DoNaLD to DAMscript	127
3.3.3	DAMscript to DAM data structure	130
3.3.4	Graphical action execution in !Donald	131
3.4	Definitive programming in !Donald2	140
3.4.1	Extending !Donald	140
3.4.2	Using raw DAMscript: the parabola example	141
3.4.3	DAM machine operators in BASIC and the dereference problem	149
3.4.4	!Donald performance	155
3.5	Geometry of definitions in the DAM machine store	162
3.5.1	Dependency between multi-word data types	162
3.5.2	Dependency on variable length data	163
3.5.3	Visualising the store	166
3.5.4	Exploiting the visualisation	170
3.A	The Script Digraph	178
Chapter 4 The Engine for Definitive Notations, EDEN		194
4.1	EDEN, chronologically to 1999	196
4.1.1	The early history	196
4.1.2	Formula variables and actions	199
4.1.3	<i>texteditorYung1987</i>	201
4.1.4	DoNaLD, SCOUT and ARCA pipeline translators	209
4.1.5	The early <code>tkeden</code>	214
4.1.6	Distributed <code>tkeden</code> : <code>dtkeden</code>	216

4.1.7	Fifty versions: an overview	218
4.2	Developments to EDEN since 1999	223
4.2.1	The beginnings: Sasami, motivating “in itself”	223
4.2.2	A database in EDEN: EDDI	229
4.2.3	A front-end parser in EDEN: the Agent Oriented Parser	231
4.2.4	Generalising the parser solution: the notations framework	239
4.2.5	Making the parser dependency driven?	240
4.2.6	Demonstrating EDEN with presentations	242
4.2.7	Interface improvements	245
4.2.8	Novel interfacing	250
4.3	EDEN execution and scheduling	259
4.3.1	Terminology mostly explicitly dismissed	260
4.3.2	Requirements of an EDEN-like definition maintainer	262
4.3.3	Black-box analysis of EDEN	264
4.3.4	Differences between definitions and actions in Eden	269
4.3.5	Inside the machine: definitions and actions	269
4.3.6	The EDEN scheduler algorithm	270
4.3.7	Higher Order Definitions (HODs) in EDEN	275
4.A	Pseudo-code of the EDEN scheduler algorithm	277
Chapter 5 Problematic issues in dependency maintenance		280
5.1	Concurrent definition maintenance	280
5.1.1	Mapping evaluation agency to definition-agents	281
5.1.2	Synchronisation of concurrent definition-agents	287
5.2	Moding	294
5.2.1	Problems with definitive lists	294
5.2.2	Meziani’s DENOTA and the “mode of definition” of a variable	301
5.3	The tri-box framework for higher-order definitive state	303
5.3.1	Motivation	304
5.3.2	Concept and examples	309
5.3.3	Implementation issues	321
5.A	Concurrent definition maintenance in SR	328
Chapter 6 Conclusions, contributions and future work		343
6.1	Conceptual machine models for EM	344
6.2	Empirical study of the EM tools and background scholarship	345
6.3	Identification of subtleties associated with dependency	347
6.4	Strengthening connections with other work	348
6.5	Future work: dependable computing, psychology and foundations	349
Bibliography		353
Index		372
Colophon		386

List of Figures

1.1	An Ishikawa fishbone diagram	5
1.2	The Empiricist Framework for Learning	7
1.3	The Phillips machine in the London Science Museum	9
1.4	A physical planimeter shown in the London Science Museum	13
1.5	Charles Care's Planimeter ISM	13
1.6	The Temposcope ISM	15
1.7	The digital watch ISM	15
1.8	A spreadsheet program and EDEN, showing observables, dependency and agency	17
1.9	Synchronisation techniques and language classes	24
1.10	A digital combinatorial circuit, definitive script and 'i/o equivalent' computer program	27
1.11	A single gate, compared with 'i/o equivalent' computer program	29
1.12	A 'pulse generator' combinatorial circuit where propagation time is significant	31
1.13	Count of articles including the word 'spreadsheet' in the title	35
2.1	BNF describing syntax for an LSD account	49
2.2	LSD account of a train carriage door and passenger	51
2.3	BNF describing syntax for the ADM	56
2.4	D , A and P in the ADM	57
2.5	Sequential <code>am</code> implementation and parallel conception of the ADM	74
2.6	Excerpts from Y.P. Yung's railway passenger LSD account and ADM script	75
2.7	Syntax for redefinition and evaluation from three implementations of the ADM	76
2.8	Partial BNF for UNITY	82
3.1	Architecture of an Acorn Archimedes 400 series computer, predecessor of the Risc PC	111
3.2	DAM machine data representation and interface routines	115
3.3	State transitions in the DAM machine	118
3.4	A DAM machine operator cleaning partial state	120
3.5	Overview of !Donald application components	125
3.6	The !Donald application	126

3.7	Compilation, Loading and “Execution” in !Donald	128
3.8	Selecting a !Donald viewport	134
3.9	A screen shot of !Donald with Listing 3.3 loaded, showing the DAM store	134
3.10	DAM machine data structure shown in Figure 3.9	136
3.11	The !Donald2 user interface (compare Figure 3.6, p.126)	141
3.12	A screenshot of the parabola	145
3.13	The “off screen” label in the parabola DAMscript	147
3.14	Graphical display from “Dereference” DAMscript, before and after a change to <code>yspace</code>	152
3.15	“Dereference” after the change to <code>name1</code>	153
3.16	“Dereference” after the change to <code>a</code>	154
3.17	Structure of the “numeric” script	160
3.18	DAM machine operator configuration for summation with double length data types	163
3.19	Visualisation of the “parabola” DAMscript DAM store	167
3.20	Visualisation of the “engine” DoNaLD script DAM store	168
3.21	DAM store mapping to pixels	169
3.22	Lookup of an ASCII character code glyph	173
3.23	Spreadsheet form of “text” DAMscript	175
3.24	Visualisation of the “text” DoNaLD script DAM store	176
3.25	Simple graph cycles	179
3.26	<code>xvcg</code> visualisation of <code>roomviewerYung1991</code>	181
3.27	Wong’s Dependency Modelling Tool, showing the ATM script	182
3.28	Acyclic digraphs graphically enumerated, $3 \leq N \leq 5$	187
4.1	<code>texteditorYung1987</code> running in <code>ttyeden-1.52</code>	201
4.2	<code>texteditorYung1987</code> with some example redefinitions	209
4.3	The original DoNaLD system	211
4.4	ARCA, SCOUT and DoNaLD combined on a pipeline	212
4.5	<code>tkyeden-dec151997</code> running <code>cruisecontrolBridge1991</code>	215
4.6	<code>tkyeden</code> architecture	215
4.7	“God’s eye view” in <code>claytontunnelSun1999</code> on the <code>dtyeden</code> server	217
4.8	<code>dtyeden</code> and <code>claytontunnelSun1999</code> in use with school children during the ACE week in January 1999	218
4.9	Code size of EDEN from 1988 to 2004 (1984 <code>hoc</code> shown for comparison)	219
4.10	<code>tkyeden</code> running on Solaris, Linux, Windows and Mac OS X	221
4.11	Data from logging of local <code>tkyeden</code> usage and downloads	222
4.12	The Sasami Rubik’s cube, <code>rubiksCarter1999</code>	224
4.13	Data flow in Sasami	226
4.14	The effect of changing the <code>small_cube_size</code> variable in <code>rubiksCarter1999</code>	226
4.15	EDDI syntax	230
4.16	AOP EDDI to EDEN translation	234

4.17	Observational structure of the AOP	236
4.18	Harfield’s parser builder in <i>agentparserHarfield2003</i>	237
4.19	Harfield’s parser visualisation in <i>agentparserHarfield2003</i>	238
4.20	<code>tkeden</code> input window with a selection of the multiple notations now available installed	239
4.21	Notation input data flow in EDEN	241
4.22	The first presentation environment slide	242
4.23	The ‘jugs’ model in a presentation slide	244
4.24	<code>tkeden</code> interface changes: <code>tkeden-dec151997</code> contrasted with <code>tkeden-1.46</code>	246
4.25	<code>tkeden-1.46</code> menus	246
4.26	Part of a simplified <code>tkeden</code> interface constructed in SCOUT	248
4.27	Sasami window linked to a USB steering wheel using a definition	252
4.28	Jon McHale and <i>carparkingsimMcHale2003</i>	252
4.29	Chris Rose and the model railway hardware	254
4.30	A partial screenshot of <i>modelrailwayRose2004</i>	255
4.31	Communication between EDEN and an Interactive Process notation translator	258
4.32	Requirements of an EDEN-like definition maintainer	263
4.33	Depth- and breadth- first topological sorts	264
4.34	Definitive script graphs containing only definitions systematically constructed to examine DM evaluation strategies	266
4.35	An equivalent definition and action	267
4.36	Definitive script graphs containing only actions systematically transformed from Figure 4.34, which contains only definitions	268
4.37	EDEN machine data, operations, data flow and control flow	272
4.38	The EDEN machine states considered more statically	273
5.1	The script graph for <code>a is b+c</code> , together with an associated extended script graph devised by adding a definition-agent node	282
5.2	Two possible definition-agent arrangements for a simple two-line definitive script	283
5.3	A script graph, the corresponding “monolithic” and one-to-one configurations of definition-agent nodes	284
5.4	Decomposition of the polynomial definition $f = ax^3 + bx^2 + cx + d$ into two possible definitive scripts	286
5.5	Synchronisation for indivisible observation of state in the single definition case	290
5.6	EDEN implements lists “horizontally” in the symbol table	298
5.7	Transformations implemented by the <code>%edens1</code> translator	299
5.8	The <code>%edens1</code> translator transforms references to lists, causing EDEN to locate them “vertically”	300
5.9	O-, C- and U-agents interacting with the state S	304
5.10	O-, C- and U-agents observe & act on subsets of S and are synchronised	308
5.11	Script graph and tri-box diagram for <code>a is b+c</code>	309

5.12	Tri-box diagrams of the “power” script	312
5.13	Tri-box diagram showing an on-screen character glyph representation linked by dependency to a box containing a character code	314
5.14	Two U-agents observing sub-sets of a list	316
5.15	An <code>if</code> conditional HOD in the tri-box framework	318
5.16	Two O-agents perceiving different dependencies	320
6.1	Dependency and agency in the current EM tool architecture	350
6.2	Spectrum of empirical work	351

List of Tables

2.1	LSD and ADM terminology compared	59
3.1	An example script in LLDN	100
3.2	Numerical !Donald2 operators	142
3.3	Geometrical !Donald2 operators	142
3.4	Miscellaneous !Donald2 operators	142
3.5	<code>tkeden</code> and !Donald performance compared when running on different machines	157
3.6	<code>tkeden</code> and !Donald performance compared when running on the <i>same</i> machine	157
3.7	<code>tkeden</code> and !Donald2 performance when running the “numeric” non-graphical script (Listing 3.9)	160
3.8	Number of acyclic digraphs with N unlabelled nodes	185

List of Listings

1.1	A description file for the <code>make</code> tool	38
2.1	Two ADM entities that can compute GCD	55
2.2	Slade's ADM algorithm	69
2.3	An interaction with <code>am</code> demonstrating evaluation in <i>S</i> state	73
2.4	A proposed algorithm for 'Authentic' ADM (AADM) execution (Ward, after Beynon and Slade)	79
2.5	Y.P. Yung's <code>am</code> cat flap script	92
2.6	An interaction with <code>am</code> and Y.P. Yung's cat flap	96
3.1	ARM code for a DAM machine <code>add</code> operator	113
3.2	A simple DoNaLD script with multiple viewports	133
3.3	DAMscript translation of the DoNaLD script shown in Listing 3.2	133
3.4	ARM code for the <code>!Donald line</code> graphical action operator	138
3.5	The "parabola" DAMscript	144
3.6	Definitions relating to the "off screen" label, written in a fictional language	148
3.8	The BASIC functions used by the Listing 3.7 DAMscript	151
3.7	The "dereference" DAMscript	151
3.9	"Numeric" Eden script and DAMscript	159
3.10	A DAM machine 'lookup' operator	164
3.12	<code>chargraphic</code> BASIC function used by "text" DAMscript (Listing 3.11)	173
3.11	"text" DAMscript	173
4.1	An interaction with <code>hoc</code>	198
4.2	A selection of Read-Write Variables from <i>texteditorYung1987</i>	202
4.3	Some Formula Variables in <i>texteditorYung1987</i>	203
4.4	Some actions in <i>texteditorYung1987</i>	204
4.5	Some procedures in <i>texteditorYung1987</i>	205
4.6	Highlights from the <i>texteditorYung1987</i> <code>edit</code> procedure	206
4.7	Some possible redefinitions to make to <i>texteditorYung1987</i>	207
4.8	Eden output corresponding to the Sasami definition <code>vertex v (a+10)/20 -c 1.0</code>	225
4.9	Interaction with the <code>eddip</code> translator	232

4.10	EDEN script corresponding to part of Figure 4.22	243
4.11	EDEN script that instantiates the slide shown in Figure 4.22	244
4.12	Using the USB HID facilities in <code>tkeden</code>	251
4.13	Eden code to communicate with the train PIC	254
5.1	SR code implementing the synchronisation shown in Figure 5.5	291

Acknowledgements

Most of all, I would like to thank my supervisor Meurig Beynon for his trust, patience and inspiration. Mine has been a wonderful apprenticeship. Steve Russ has also played a major part in my supervision and I am deeply grateful for his thoughtful advice.

Many other people have also assisted with this thesis, directly and indirectly.

I would like to thank the selection of previous EM researchers whose work I have had the pleasure of studying closely for this thesis. If I criticise their work, I do so only in an attempt to stand on their shoulders. In chronological order of their contribution: Y.W. ‘Edward’ Yung, Samia Meziani, Mike Slade, Y.P. ‘Simon’ Yung, Dominic Gehring, James Allderidge, Pi-Hwa ‘Patrick’ Sun and Richard Cartwright. I am also very grateful to Richard for introducing me to this area.

I am pleased to acknowledge Russell Boyatt, Chris Brown, Ben Carter, “the other” Ashley Chaloner, Josie Clement, Richard Cunningham, Carlos Fischer, Nathan Griffiths, Antony Harfield, Tim Heron, Mark Lloyd, Ria Lloyd, Izellah Macintosh, Jon McHale, Rod Moore, Roger Packwood, Chris Roe, Chris Rose, Dorian Rutter, Stuart Valentine and Paul Williamson for technical assistance related to this thesis. I also thank my other fellow EM-ers for many thought-provoking seminars and models, as well as Abhir Bhalerao, Peter Forbrig, Katie Lucas, Steve Matthews, Simon Rawles, Hans Roeck, Matt Street and Rob West for useful input.

I would especially like to thank Chris Roe for his companionship as we explored this landscape together. Many thanks also to my trusted friend Benjohn Barnes for many wide-ranging discussions over the course of this work.

Since May 2002 I have been working part-time on this thesis — my Teaching Fellow ‘day job’ has involved the teaching of Computer Organisation and Architecture (COA). I am indebted to my COA colleagues Chang-Tsun Li, Franc Buxton, Gerry Biggs and TRM, RAP, SGV, RC, PW already mentioned above. I also thank my other colleagues and friends in the department. I particularly appreciate the generosity and trust of Graham Nudd.

I am extremely grateful to my family and friends for moral support. Thanks to Mum and Dad for (amongst many other things), instilling organisational skills and encouraging me to ask ‘why’.

Finally, thank you, Linda, for the unconditional love and support, providing the stability without which this thesis would never have been completed.

Thanks to my external examiners Mike Holcombe and Jean Bacon for the timely viva and helpful comments.

Declaration

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. All quoted text remains the copyright of the original attributed author. I have taken the liberty of correcting typographical errors, spelling and grammar where necessary. All trademarks are acknowledged.

None of the material in this thesis has been previously published, but some of the motivating ideas have appeared in joint work published in [BCSW99, BWM⁺00, BCH⁺01, BRWW01, BBRW03].

Interaction with Meaningful State:
Implementing Dependency on Digital Computers

Abstract

A perception of an agent as follows — that the act of changing the value of one observable indivisibly entails a predictable accompanying change in that of another — can be termed a *dependency*. Indivisibility in change allows us, as agents, to make our experience of interaction with the ‘world’ meaningful.

Empirical Modelling (EM) is the name we have given at Warwick to the activity of building artefacts to embody patterns of observables, dependencies and agent actions that are encountered in experience. EM involves the progressive development of understanding through interaction, whereby meaning is continually refined in the light of additional experience. An EM artefact can be physical (*cf.* the Phillips machine [Phi00]) or computer-based (*cf.* spreadsheets).

This thesis investigates how dependency can be effectively implemented on a digital computer through the critical evaluation of three contrasting tools to support EM developed at Warwick over the last 16 years. The thesis contribution has three aspects: conceptual insight; critical, historical and empirical review; and technical and practical development.

Slade’s original Abstract Definitive Machine (ADM) concept [Sla90] and its implementations are reviewed and linked to the original motivation for its development — animating LSD accounts. The main emphasis of the ADM is on agency. Subtleties and inconsistencies in the way in which the ADM concept has developed over time are exposed and an algorithm for an ‘Authentic’ ADM (AADM) is proposed.

Cartwright’s design and implementation of the Definitive Assembly Maintainer (DAM) machine [Car99] is analysed and critiqued. The DAM machine emphasises only dependency. The DAM machine is extended to allow lower-level interaction using a special-purpose code and to exploit the video hardware to create dependency at a very low level.

The primary tool of the EM research group, EDEN [Yun93, Yun90], successfully combines both dependency and agency. Its development is reviewed in its historical context and various extensions, including novel interfacing, are described. The first exposition of EDEN’s internal operation is given.

The analysis of existing EM tools motivates the examination of some problematic issues regarding concurrency, moding and higher-order dependency. Some proposals for design and implementation to address these issues are described.

Throughout the thesis, consideration of technical issues is motivated by the possibility that human engagement with computers can have qualities similar to engagement with ‘real-world’ artefacts, manifest in interaction with meaningful state that is at all times intelligible to the human interpreter, unlike the typically meaningless intermediate states that are generated in the execution of a conventional program. The insights that exploring this possibility brings may be significant in producing programs that are more robust under change.

Abbreviations

AADM	Authentic ADM
ADM	Abstract Definitive Machine [Sla90]
AOP	Agent-Oriented Parser [Har02, Bro01]
ARCA	A definitive notation named after Arthur Cayley (1821–1895), which “might be charitably regarded as ‘An Aid for the Realisation of Combinatorial Artefacts’” [Bey83]
BRA	Block Redefinition Algorithm [Car99] (my abbreviation)
CADNO ¹	Computer-Aided Design Notation ([Bey89b], first named in [Bey89a])
DAM machine	Definitive Assembly Maintainer machine [Car99]
DAMscript	A type of Low Level Definitive Notation used to program the DAM machine
DM	Dependency Maintainer
DMM	Dependency Maintainer Model [Car99]
!Donald	An implementation of DoNaLD running on the DAM machine
DoNaLD	Definitive Notation for Line Drawing [BABH86]
dtkeden	An implementation of EDEN that allows distributed communication [Sun99]
EDDI	Eden Definition Database Interpreter [Tru96]
EDEN	Engine for DEfinitive Notations [Yun90] — in this thesis, upper case ‘EDEN’ denotes an implementation
Eden	The language implemented by EDEN [Yun89]
EM	Empirical Modelling

¹Also Welsh for “fox”.

empublic	A local directory containing our tools installation and collected archive of models, partially accessible via the web [WRB]
FV	Formula Variable [Yun90] (in Eden)
HOD	Higher Order Dependency/Definition
ISM	Interactive Situation Model [Sun99]
LLDN	Low Level Definitive Notation [Car99] (my abbreviation)
LSD	Language for Specification and Description [Bey87a]
RWV	Read-Write Variable [Yun90] (in Eden)
S, S^*, S' states	States before, during and after a transition
Sasami	A definitive notation for EDEN allowing use of 3D graphics, implemented using OpenGL [Car00]
SCOUT	Screen layOUT — a definitive notation [Yun93]
tkeden	An implementation [Yun93] of EDEN with a graphical user interface implemented using Tcl/Tk [Ous94]
ttyeden	An implementation of EDEN with a terminal-based ‘curses’ interface
UNITY	Unbounded Nondeterministic Iterative Transformations — a computational model and proof system [CM88]
VM	Virtual Machine