

Copyright is owned by the Author of the thesis. Permission is given for a copy to be downloaded by an individual for the purpose of research and private study only. The thesis may not be reproduced elsewhere without the permission of the Author.

M.Θ.Θ.T

META OBJECT
ORIENTATED TOOL

A NOVEL META-CASE
TOOL METHODOLOGY
REPRESENTATION
STRATEGY

A dissertation submitted in partial fulfilment of the
requirements for the degree of
Doctor of Philosophy in Computer Science

Massey University, New Zealand

David Charles Page

1998

Abstract

This thesis presents an investigation into current meta-CASE technology. The research focuses on CASE tool support for the concept of methodology, the representation of methodology syntax and semantics, and the support for re-use of methodology descriptions and software artefacts. A novel methodology representation strategy for meta-CASE tools is proposed and implemented with the development of a new meta-CASE tool (MOOT – Meta Object Orientated Tool).

The novel strategy propounded in this thesis uses an object-orientated meta-model and views methodology descriptions as potentially re-usable components. The coupling between methodology syntax and semantic descriptions is *minimised* so they can be re-used independently.

Two new modelling languages have been derived, to support the definition of syntax (NDL – Notation Definition Language) and semantics (SSL – Semantic Specification Language) of software engineering methodologies. Semantic descriptions are compiled to a platform independent representation (SSL-BC), which is executed on a purpose built virtual machine (SSL-VM). Late binding of syntax and semantic methodology descriptions is implemented with the development of Notation Semantic Mapping (NSM) tables. Two libraries of re-usable methodology description components, the Core Knowledge Base (CKB) and the Generic Object Orientated Knowledge Base (GOOKB), have been derived during this research.

Empirical results gained from applying the MOOT prototype demonstrated the flexibility, extensibility and potential of the novel methodology representation strategy. This approach permitted the implementation and modelling of UML and patterns, two recent advances of object technology that did not exist when the research commenced.

The novel strategy presented in this thesis is more than an untried theory. It has been implemented, applied and is being evaluated. Simply, it is real and it works.

DEDICATION

This thesis is lovingly dedicated to my parents

Michael Julius Page and Susan Evelyn Page

TABLE OF CONTENTS

INTRODUCTION	1
1.1 Introduction	1
1.2 Fundamental Terms	2
1.2.1 Software Engineering Development Methodology	3
1.2.2 Meta-Modelling	6
1.2.3 Computer Aided Software Engineering (CASE)	7
1.2.4 CASE Tool	7
1.2.5 Meta-CASE and Meta-CASE Tool	8
1.3 Object-Orientated Software Development Methodologies	8
1.4 CASE Technology	11
1.5 Methodology CASE Tools	14
1.5.1 Methodology Dependent CASE Tools	15
1.5.2 Multi-Methodology CASE Tools	15
1.5.3 Tools that Support More than One Methodology	16
1.5.4 Meta-CASE Tools	16
1.5.5 CASE Tool Generators	16
1.5.6 Modifiable CASE Environments	17
1.6 Limitations of Methodology CASE Tools	18
1.6.1 Limitations from the Organisational Perspective	20
1.6.2 Limitations from the CASE Tool Perspective	22
1.7 Objectives of the Research	23
1.8 Method	24
1.9 Outline of the Thesis	25
META-MODELLING AND META-CASE TOOLS	27
2.1 Introduction	27
2.2 Meta-Modelling	27
2.2.1 The OMG Meta Object Facility	29
2.2.2 Unified Modelling Language	31
2.2.3 COMMA	32
2.2.4 Open Modelling Language	33
2.2.5 OOram	33
2.2.6 CASE Data Interchange Format	34
2.2.7 ISO/CDIF Meta-Model	35

2.2.8	MetaData Interchange Facility	36
2.3	Meta-CASE Tools	36
2.3.1	Framework for Discussion of Meta-CASE Tools	39
2.3.2	MetaView	41
2.3.3	Meta-Edit and MetaEdit+	43
2.3.4	Alfabet	46
2.3.5	ToolBuilder	48
2.3.6	Graphical Designer Pro	50
2.4	Limitations of Current Meta-CASE Technology	51
2.5	Summary	54
 META OBJECT ORIENTATED TOOL		 56
3.1	Introduction	56
3.2	Method	56
3.3	Rationale and Goals of the MOOT Project	58
3.4	MOOT Methodology Descriptions	62
3.5	The CKB and GOOKB	65
3.6	Addressing the Limitations of Meta-CASE tools	68
3.7	Architecture of MOOT	71
3.7.1	CASE Tool Client	74
3.7.2	Methodology Development Tool	75
3.7.3	MOOT Core	76
3.8	The MOOT Prototype	77
3.9	Summary	79
 NOTATION DEFINITION LANGUAGE		 81
4.1	Introduction	81
4.2	Method	81
4.3	Models and Notations	82
4.4	Analysis of Notations	85
4.4.1	Symbols	86
4.4.2	Connections	88
4.4.3	Docking Areas	91
4.4.4	Groups	93
4.4.5	Presentation	94

4.4.6	Actions	95
4.5	Notation Definition Language	96
4.5.1	Requirements of NDL	96
4.5.2	Design of NDL	97
4.5.3	Describing Symbols in NDL	99
4.5.4	Support for Grouping	103
4.5.5	Docking Areas	105
4.5.6	Describing Connections in NDL	110
4.6	NDL Interpreter	113
4.7	Design of the NDL Interpreter	114
4.7.1	Representing Expressions	114
4.7.2	Segment Templates	116
4.7.3	Group Templates	117
4.7.4	Connection and Symbol Templates	117
4.8	Implementation of the NDL Interpreter	119
4.9	Summary	120
 SEMANTIC SPECIFICATION LANGUAGE		 121
5.1	Introduction	121
5.2	Method	121
5.3	Rationale and Goals of SSL	122
5.4	Requirements of SSL	124
5.5	Semantic Specification Language	126
5.5.1	Overview	126
5.5.2	MOOT Meta-Model	126
5.5.3	Module System	129
5.5.4	Memory Management	129
5.5.5	Messages	130
5.6	Semantic Specification Language Definition	130
5.6.1	Collections	131
5.6.2	Simple Expressions	131
5.6.3	Interface Module	134
5.6.4	Class Interface Definition	134
5.6.5	Implementation Module	135
5.6.6	Class Definition	135
5.6.7	Methods	136
5.6.8	Statements	139
5.7	SSL Compiler	140
5.8	Executing SSL	143

5.9	SSL Virtual Machine	145
5.9.1	Requirements of the SSL Virtual Machine	146
5.9.2	Architecture of the SSL Virtual Machine	146
5.9.3	SSL Virtual Machine Instruction Set	147
5.9.4	Internal Representation of Classes, Objects and Methods	148
5.9.5	Processing Messages on the Virtual Machine	151
5.10	Summary	153
 THE CORE KNOWLEDGE BASE AND GENERIC OBJECT ORIENTATED KNOWLEDGE BASE		 154
6.1	Introduction	154
6.2	Context of the Core Knowledge Base and the Generic Object Orientated Knowledge Base	154
6.3	Development of the Core Knowledge Base	156
6.3.1	Meta-Model of Methodology	156
6.3.2	Meta-Model of Modelling Language	158
6.3.3	Handling Exceptional Situations	165
6.4	Development of the Generic Object Orientated Knowledge Base	166
6.4.1	Object-Orientated Methodology Comparisons	167
6.4.2	Method used to Design the Generic Object Orientated Knowledge Base	170
6.4.3	Generic Object Orientated Knowledge Base	171
6.5	Implementing the Knowledge Bases	175
6.6	Summary	176
 REALISING METHODOLOGIES AND SOFTWARE ENGINEERING PROJECTS IN MOOT		 177
7.1	Introduction	177
7.2	Interaction Between CASE Tool Clients and the MOOT Core	177
7.2.1	CASE Tool Client Requests	178
7.2.2	MOOT Core Directives and Responses	180
7.3	Methodology Description Table	181
7.3.1	Composition of the Methodology Description Table	181
7.3.2	Applying the Methodology Description Table	184
7.4	Notation Semantic Mapping Tables	185
7.4.1	NDL vs. SSL	185
7.4.2	Composition of NSM Tables	187

7.4.3	Applying NSM Tables	191
7.5	Summary	199
VALIDATING THE MOOT APPROACH		201
8.1	Introduction	201
8.2	Defining the Coad and Yourdon Methodology	202
8.3	Supporting Patterns	210
8.4	Supporting UML	213
8.5	Preliminary Development of the Semantics Editor	216
8.5.1	Notation for the SSL Module Structure Modelling Language	219
8.5.2	Notation for the SSL Method Modelling Language	221
8.6	Toward Supporting Joosten Workflow Modelling	224
8.7	Summary	225
CONCLUSION AND FUTURE WORK		227
9.1	Introduction	227
9.2	Summary of the Thesis	228
9.3	Discussion	230
9.3.1	The Novel Meta-CASE Tool Methodology Representation Strategy	230
9.3.2	The MOOT Approach	231
9.3.3	The Notation Definition Language	236
9.3.4	The Semantic Specification Language	236
9.3.5	Core Knowledge Base and Generic Object Orientated Knowledge Base	237
9.4	Future Work	238
9.4.1	The Notation Definition Language	238
9.4.2	The Semantic Specification Language	240
9.4.3	Notation Semantic Mapping Tables	240
9.4.4	Support for Re-use	241
9.4.5	Cognitive Support	242
9.4.6	Meta-Modelling	242
9.4.7	Validation of a Complete Implementation of MOOT	243
9.5	Conclusion	244

APPENDICES

EVALUATION FRAMEWORK	246
I.1 Existing Evaluation Frameworks	246
I.2 A New Evaluation Framework	246
NDL GRAMMAR	249
II.1 Introduction	249
II.2 Reserved Words	249
II.3 Operators	249
II.4 Grammar	249
SSL GRAMMAR	253
III.1 Introduction	253
III.2 Reserved Words	253
III.3 Operators	253
III.4 Grammar	253
SSL EXAMPLES	258
IV.1 The Sieve of Eratosthenes Version 1	258
IV.1.1 Interface Module	259
IV.1.2 Implementation Module	259
IV.2 The Sieve of Eratosthenes Version 2	265
IV.2.1 Interface Module	265
IV.2.2 Implementation Module	266
SSL-VM INSTRUCTION SET	269
V.1 Introduction	269
V.2 Instruction Set	269

SSL COMPILER	277
VI.1 Introduction	277
VI.2 The SSL Compiler	277
VI.3 Representing Types in the SSL Compiler	279
VI.4 Representing Statements and Expressions in the SSL Compiler	280
VI.5 Representing Modules in the SSL Compiler	282
THE SSL VIRTUAL MACHINE	285
VII.1 Introduction	285
VII.2 The SSL Virtual Machine	285
VII.3 Representing SSL Types	287
VII.4 SSL Proxies	287
VII.5 Processing Messages	289
VII.6 Binding	291
VII.7 Garbage Collection	293
REFERENCES	297

LIST OF FIGURES

<i>Number</i>	<i>Page</i>
Figure 1-1 - Modelling	4
Figure 1-2 - Meta-modelling	6
Figure 1-3 - Classification hierarchy of CASE tool categories	14
Figure 1-4 - Thesis outline	26
Figure 2-1 - Four layer meta-modelling process	28
Figure 2-2 - CDIF Meta-metamodel (EIA CDIF, 1994b)	35
Figure 2-3 - CASE tool generators	37
Figure 2-4 - Architecture of a modifiable CASE environment	38
Figure 2-5 - Meta-CASE tools and the four layer meta-modelling architecture	39
Figure 3-1 - Mapping between goals and design decisions made regarding MOOT	60
Figure 3-2 - The relation between software projects, methodology descriptions and the description languages in MOOT	62
Figure 3-3 - Methodology descriptions and software engineering projects	64
Figure 3-4 - Knowledge bases in MOOT	65
Figure 3-5 - The relation between the CKB, the GOOKB, methodologies and software engineering projects in MOOT	66
Figure 3-6 - Meta-modelling architecture	67
Figure 3-7 - Addressing the limitations of meta-CASE tools	69
Figure 3-8 - The two roles of the MOOT system	71
Figure 3-9 - Moot system	72
Figure 3-10 - Proposed, top level, system architecture	73
Figure 3-11 - Architecture of the MOOT prototype	79
Figure 4-1 - A state transition diagram drawn in the notation of Booch and Feylock	84
Figure 4-2 - A simple diagram drawn with UML, Coad and Yourdon and Booch notations	86
Figure 4-3 - Three examples of a UML class symbol	87
Figure 4-4 - Topographical description of a UML class	87
Figure 4-5 - Coad and Yourdon and Booch symbols showing common sub-parts	88
Figure 4-6 - Two example connections	88
Figure 4-7 - Inheritance connection in UML	90

Figure 4-8 - An example UML sequence diagram	91
Figure 4-9 - A Jacobson Use Case diagram	91
Figure 4-10 - Docking areas on Coad and Yourdon Class&Object symbols	92
Figure 4-11 - Coad and Yourdon Subject Area: expanded (left) and collapsed (right)	93
Figure 4-12 - A UML class expressed with varying levels of detail	94
Figure 4-13 - Two example active areas	96
Figure 4-14 - (i) A symbol (ii) exploded Symbol (iii) templates	97
Figure 4-15 - Applying a template	98
Figure 4-16 - Topographical description of a UML class symbol	99
Figure 4-17 - A UML class symbol with active areas	101
Figure 4-18 - Template describing a UML class symbol	102
Figure 4-19 - Coad and Yourdon class and Class&Object symbols	103
Figure 4-20 - Identified common sub-parts in Coad and Yourdon's notation	103
Figure 4-21 - Group templates	104
Figure 4-22 - Coad and Yourdon class symbol	104
Figure 4-23 - Docking at a point	106
Figure 4-24 - Anatomy of a point docking area	106
Figure 4-25 - Docking on a line	107
Figure 4-26 - Anatomy of a line docking area	107
Figure 4-27 - Representing valid directions for a line docking area	108
Figure 4-28 - Docking on an arc	108
Figure 4-29 - Anatomy of an arc docking area	109
Figure 4-30 - Representing valid directions for an arc docking area	109
Figure 4-31 - Two example connections	110
Figure 4-32 - Connection symbol template	110
Figure 4-33 - Coad and Yourdon connection symbol line docking area (i) with a single connection (ii) with multiple connections	111
Figure 4-34 - Connection terminator templates for Coad and Yourdon Gen-Spec and message connections	112
Figure 4-35 - NDL connection templates for Coad and Yourdon Gen-Spec and message connections.	112
Figure 4-36 - Components of the NDL interpreter	113
Figure 4-37 - The Expression class hierarchy	115
Figure 4-38 - Template segment hierarchy	116

Figure 4-39 - The different types of template	118
Figure 4-40 - NDL interpreter using an NDL description of the Rumbaugh instance and object diagram	119
Figure 4-41 - NDL interpreter using an NDL description of the Coad and Yourdon class diagram	120
Figure 5-1 - Mapping between goals and design decisions made regarding features of SSL	124
Figure 5-2 - MOOT meta-metamodel	127
Figure 5-3 - The built-in SSL variables	128
Figure 5-4 - SSL collection and iterator types	131
Figure 5-5 - Partial SSL implementation of a list class	135
Figure 5-6 - SSL implementation of the list class	137
Figure 5-7 - Implementing SSL create operations	138
Figure 5-8 - Example loop and if statements	140
Figure 5-9 - SSL compiler	141
Figure 5-10 - Processing actions	143
Figure 5-11 - Architecture of the SSL virtual machine	146
Figure 5-12 - SSL class	149
Figure 5-13 - SSL method	150
Figure 5-14 - SSL object	151
Figure 5-15 - Processing messages on the SSL-VM	152
Figure 6-1 - The three tier structure of the information processed by MOOT	155
Figure 6-2 - Methodology meta-model	157
Figure 6-3 - Transitions	158
Figure 6-4 - Meta-model of modelling language	159
Figure 6-5 - Representing a whole-part relation	160
Figure 6-6 - Representing a class diagram with instances of classes from the CKB	161
Figure 6-7 - Detailed meta-model of modelling language	162
Figure 6-8 - Extended meta-model of modelling language	163
Figure 6-9 - Core Knowledge Base	164
Figure 6-10 - Situations	165
Figure 6-11 - Critics	166
Figure 6-12 - Taxonomy of object-orientated methodology comparisons	168
Figure 6-13 - Number of comparisons for $N \times N$ methodologies	169
Figure 6-14 - Representing classes and objects	171

Figure 6-15 - Representing object-orientated relations	172
Figure 6-16 - The Generic Object Orientated Knowledge Base	173
Figure 6-17 - Representing an object model with classes from the GOOKB	174
Figure 6-18 - Module structure of the CKB and GOOKB	175
Figure 7-1 - The communication between CASE tool clients and the MOOT core	179
Figure 7-2 - Methodology Description Table	182
Figure 7-3 - Creating a new software engineering project	184
Figure 7-4 - The create concept map	187
Figure 7-5 - The create relation map	188
Figure 7-6 - The add map	189
Figure 7-7 - The action map	190
Figure 7-8 - The SSL object creation map	190
Figure 7-9 - The SSL object update map	191
Figure 7-10 - The Notation Semantic Mapping Table	192
Figure 7-11 - Creating a new model	193
Figure 7-12 - Creating a new concept	195
Figure 7-13 - Successful update of a field	196
Figure 7-14 - Failed attempt to update a field	197
Figure 7-15 - Propagating server side update	198
Figure 8-1 - Supporting the Coad and Yourdon	202
Figure 8-2 - Methodology description table for Coad and Yourdon	203
Figure 8-3 - The select methodology dialogue box	203
Figure 8-4 - Symbol template for the Coad and Yourdon Class&Object symbol	204
Figure 8-5 - Representing the Coad and Yourdon message connection	206
Figure 8-6 - NSM table for Coad and Yourdon	207
Figure 8-7 - Implementation of the addAttribute operation	208
Figure 8-8 - Adding an attribute	209
Figure 8-9 - An Object-Orientated Analysis model of Object-Orientated Analysis (Coad and Yourdon, 1991a)	210
Figure 8-10 - Extending the GOOKB to support Patterns	212
Figure 8-11 - UML v1.1 Foundation: CORE: Backbone + Foundation: CORE: Extension Mechanisms + Foundation: CORE: Auxiliary Elements	214
Figure 8-12 - UML v1.1 Behavioural Elements: Collaborations	215
Figure 8-13 - UML v1.1 Common Behaviour: Common Behaviour	216
Figure 8-14 - SSL modelling languages	217

Figure 8-15 - Representing SSL as an extension of the GOOKB	218
Figure 8-16 - NSM table for the SSL module modelling language	219
Figure 8-17 - Supporting SSL with MOOT	220
Figure 8-18 - Explain method of the <i>ComplexCritic</i> class in the CKB	223
Figure 8-19 - An example SSL method model	224
Figure 8-20 - Joosten trigger model (Joosten, 1995)	225
Figure I-1 - Dimensions of the evaluation framework	248
Figure IV-1 - Sieve of Eratosthenes version 1	258
Figure IV-2 - Sieve of Eratosthenes version 2	265
Figure VI-1 - The main components of the SSL compiler	278
Figure VI-2 - Representing types in the SSL compiler	280
Figure VI-3 - Statements and expressions in the SSL compiler	281
Figure VI-4 - Representing modules, classes, operations and methods in the SSL compiler	282
Figure VII-1 - Components of the SSL-VM	286
Figure VII-2 - Representing SSL objects and SSL classes	288
Figure VII-3 - The classes involved in processing a message on the SSL-VM	290
Figure VII-4 - Executing a method on the SSL-VM	291
Figure VII-5 - Binding a message to a method on the SSL-VM	292
Figure VII-6 - Implementation of the reference counting garbage collection scheme	294
Figure VII-7 - Implementation of the SSL Instance Proxy class	295

LIST OF TABLES

Table 1-1 - First generation object-orientated methodologies	8
Table 1-2 - Second generation object-orientated methodologies	9
Table 1-3 - History of CASE tools (Ferguson, 1998)	13
Table 2-1 - Four layer meta-modelling architecture	28
Table 2-2 - Meta-CASE tools	40
Table 5-1 - SSL-VM types	147
Table 5-2 - SSL-BC instruction set	148
Table 7-1 - Correspondance between syntax and semantic elements	186
Table 9-1 - Practical work completed during the research	230
Table VII-1 - Implementation of SSL types in the SSL-VM	287

ACKNOWLEDGMENTS

Colours

You reap what you sow. Put your face to the ground.
Here come the marching men. Your colours wrapped around.

The Sisters of Mercy

The research detailed in this thesis was supported by two PGSF funded research projects (MAU-503, MAU-807). Financial assistance was also received with a New Zealand postgraduate scholarship.

The following people deserve special mention.

My supervisors, who inspired me and taught me a great deal

Assoc. Prof. Daniela Mehandjiska-Stavreva, Prof. Mark Apperley

The Masters and Honours students who were also involved in this research

Paul Clark, Steven Adams, Hong Yu, Duane Griffin, Sarisha Dasari, Mi Duk Choi,
Jonathan Ham

Two people that selflessly proof-read the thesis

Rachel Page, Wendy Browne

My family, without whom this would not have been possible

Michael Page, Susan Page, Audrey Isaac, Rachel Page, Jonathon Page, Ruth Page

My friends (you know who you are), who all helped without knowing it

Extra thanks to: Nick Earle, Paul Clark, Duane Griffin, Lisabeth Weston, Shamus Smith, Luke Usherwood, Marion Moore, Andrew Turvey, James Fulton, Steven Adams

GLOSSARY

The content of the glossary has been derived from a range of dictionaries (Collins, 1995; Nuttals, 1902; Readers Digest, 1988; Oxford, 1993; Mirriam-Webster, 1998), the Dictionary of Object Technology (Firesmith and Eykholt, 1995) and (D'Souza and Wills, 1998; Jacobson *et al.*, 1995; Pressman 1997; Schach, 1993, 1997; Somerville, 1996).

Abstraction. Any model that includes the most important, essential, or distinguishing aspects of something while suppressing or ignoring less important, immaterial, or diversionary details. The result of removing distinctions so as to emphasise commonalities.

Arity. The cardinality of something. For example the arity of a relation specifies the number of concepts that are involved in the relation.

Attribute. Any named property used as a data abstraction to describe its enclosing object, class or extent.

Behaviour. Anything that an organism does involving action and response to stimulation. The way in which someone behaves; also: an instance of such behaviour.

Bind. To place under certain constraints. To cohere or cause to cohere. To place under obligation; oblige.

Binding. Any selection of the appropriate method for an operation on receipt of a corresponding message.

Browser. Any view that allows you to access hierarchically organised and indexable information.

CASE Tool. A) Any computer based tool for software planning, development and evolution. This includes all examples of computer-based support for the managerial, administrative, or technical aspects of any part of a software development project. B) Products that assist the software engineer in developing and maintaining software.

CASE. An acronym that stands for Computer Assisted Software Engineering.

CKB. Core Knowledge Base. A library of methodology semantic components that implements a meta-model of methodology.

Class. Any uniquely identified abstraction (i.e. a model) of a *set* of logically related instances that share the same or similar characteristics. The combination of a type interface and associated type implementation.

Classification. The act of forming into a class or classes; a distribution into groups such as classes, orders, families, etc., according to some common relations or affinities.

Cohesion. The degree, to which something models a single abstraction, localising only features and responsibilities related to that abstraction.

Component. A) Any standard, reusable, previously implemented unit that is used to enhance the programming language constructs and to develop applications. B) An independently deliverable unit of software that encapsulates its design and implementation and offers interfaces to the out-side, by which it may be composed with other components to form a larger whole.

Coupling. The degree to which one thing depends on another. Low coupling is desirable because it produces better encapsulation, maintainability, and extendibility with fewer objects needlessly affected during iteration.

Encapsulation. To enclose in or as if in a capsule; the act of enclosing in a capsule. The physical localisation of features.

Engineering. The application of scientific principles to such practical ends as the design, construction, and operation of efficient, economical structures, equipment and systems. The application of science to the design, building, and use of machines, constructions etc.

GOOKB. Generic Object Orientated Knowledge Base. A library of methodology semantic components that implements a meta-model of concepts germane to all object-orientated methodologies.

Identity. Individuality.

Information Hiding. The deliberate and enforced hiding of information (e.g. design decisions, implementation details) from clients. The limiting of scope so that some information is invisible outside of the boundary of the scope.

Inheritance. The incremental construction of a new definition in terms of existing definitions without disturbing the original definitions and their clients.

Instance. Anything created from or corresponding to a definition.

Interface. The visible outside, user view of something.

Language. Any method of communicating ideas, as by a stream of signs, symbols, gestures or the like. The special vocabulary and usage of a scientific professional or other group. The speech or expression of ideas.

MDT. An acronym that stands for Methodology Description Table. The Methodology Description Table provides an index of the methodologies supported by MOOT.

Message Send. The sending of a message to an object.

Message. Any communication sent or received by an object.

Meta. A Greek prefix signifying beyond, after, with, among and frequently expressing change. Going beyond or transcending. Used with the name of a discipline to designate a

new but related discipline designed to deal critically with the original one. Of a higher or second-order kind.

Meta-CASE Tool. A) A meta-CASE tool is any tool that provides automated or semi-automated support for developing CASE tools. B) ... are CASE tools which are used to generate other CASE tools. C) A CASE tool that operates on CASE tools.

Meta-language. The natural language, formal language, or logical system used to discuss or analyse another system. A form of language used to discuss a language.

Method. A) Mode of procedure, logical arrangement, orderly arrangement, system of classification. A means or manner of procedure, especially a regular and systematic way of accomplishing anything. The procedures and techniques characteristic of a particular discipline or field of knowledge. A special form of procedure esp. in any branch of mental activity. B) A way of carrying out a complete phase such as such as design or integration. C) The hidden implementation of an associated operation.

Methodology CASE Tool. A CASE tool that supports one or more software development methodologies and attempts to span most of the software development life-cycle.

Methodology. The science of scientific method of classification. From the Greek method and logis (science). The system of principles, practices, and procedures applied to any specific branch of knowledge. The science of method; a body of methods used in a particular branch or activity.

Model. A) Archetype; a description or analogy used to help visualise something that cannot be directly observed; a system of postulates, data, and inferences presented as a mathematical description of an entity or state of affairs. A preliminary pattern or representation of an item not yet constructed. A tentative framework of ideas describing something intangible and used as a testing device. B) A model clarifies – for a person or group of people – some aspect or perspective on a thing or event.

MOOT. Meta Object Orientated Tool. A new meta-CASE tool developed as a result of this research.

NDL. Notation Definition Language. A new language used to define the syntax of a methodology in MOOT.

Notation. A system of characters, symbols, or abbreviated expressions used in an art or science or in mathematics or logic to express technical facts or quantities.

NSM. An acronym that stands for Notation-Semantic Mapping. NSM tables are used to implement late binding of NDL and SSL methodology descriptions.

Object. Any abstraction that models a single thing.

Operation. Any service that may be requested.

Polymorphism. The ability of a single name to refer to different things having different forms.

Process. A) A system of operations in the production of something. A series of actions, changes or functions that bring about an end or result. A course of action or proceeding, esp. a series of stages in manufacture or some other operation. B) ... the way we produce software. It starts with concept exploration and ends when the product is finally retired. C) ... the set of activities and associated results which produce a software product.

Relation. Connection by consanguinity or affinity; kinship; relationship; as, the relation of parents and children; an abstraction belonging to, or characteristic of, two entities or parts together.

Semantic. Of, or relating to, meaning in language.

Software Development Life-cycle (SDLC). A process by which software engineers build computer applications.

Software Engineering. A) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is the application of engineering to software. B) ... is concerned with the theories, methods and tools that are needed to develop software for computers. C) A discipline whose aim is the production of quality software that satisfies the user's needs, and is delivered on time and within budget.

Software Project. A software project consists of a set of models (built using a particular methodology) which collectively define the software being constructed.

SSL. Semantic Specification Language. A new object-orientated language used to define the semantics of a methodology in MOOT.

SSL-BC. SSL Byte Code. A platform independent, binary, representation of SSL, which is generated by the SSL compiler.

SSLC. SSL compiler.

SSL-VM. SSL Virtual Machine. A new virtual machine which supports efficient processing of SSL.

State. Any status, situation, condition, mode, or life-cycle phase of an object or class during which certain rules of overall behaviour (e.g. response to messages) apply.

Syntax. The way in which linguistic elements (as words) are put together to form constituents (as phrases or clauses).

Tool. A thing used in an occupation or pursuit. Any instrument of use or service.

Type. A lower taxonomic category selected as a standard of reference for a higher category. The declaration of the interface of any set of instances (e.g. objects) that conform to this common protocol.

PUBLICATIONS

The results of this research have been presented in the following refereed publications.

- Phillips, C.H.E., Adams, S., **Page, D.** and Mehandjiska, D. (1998) The Design of the Client User Interface for a Meta Object-Oriented CASE tool, Proceedings of TOOLS Pacific'98, Monash Printing Services, Victoria, pp145-157
- Page, D.**, Mehandjiska, D. and Phillips, C.H.E. (1998) Methodology Independent OO CASE: Supporting Methodology Engineering, Proceedings of Software Engineering: Education and Practise (SE:E&P'98), IEEE Computer Society Press, Dunedin, New Zealand, pp373-380
- Phillips, C.H.E., Adams, S., **Page, D.** and Mehandjiska, D. (1998) Design of the User Interface for a Methodology Independent OO CASE Tool, Proceedings of OZCHI'98, IEEE Computer Society Press, Los Alamitos, California, pp106-114
- Phillips, C.H.E., Mehandjiska, D. and **Page, D.** (1998) The Usability Component of a New Framework for the Evaluation of Object-Oriented CASE tools, Proceedings of Software Engineering: Education and Practise (SE:E&P'98), IEEE Computer Society Press, Dunedin, New Zealand, pp131-141
- Page, D.**, Griffin, D., Usherwood, L. and Mehandjiska, D. (1997) Implementation of a Semantic Specification Language Interpreter for a Methodology Independent OO CASE Tool, Proceedings of IASTED International Conference on Software Engineering (SE'97), ACTA Press, San Francisco, USA, pp239-242
- Mehandjiska, D., **Page, D.**, Griffin, D. and Usherwood, L. (1997) Methodology Knowledge Representation and Interpretation for a Methodology Independent OO CASE Tool, Proceedings of IASTED International Conference on Software Engineering (SE'97), ACTA Press, San Francisco, USA, pp243-247
- Mehandjiska, D., **Page, D.** and Choi, M. D. (1996) Meta-Modelling and Methodology Support in Object-Oriented CASE Tools, Proceedings of 3rd International Conference on Object-Oriented Information Systems (OOIS'96), Eds. Patel, D., Sun, Y. and Patel, S., Springer-Verlag, London, pp370-386
- Mehandjiska, D., **Page, D.** and Dasari, S. (1996) Generic Knowledge Base for a Methodology Independent Object-Oriented CASE Tool, Proceedings of the IASTED International Conference on Artificial Intelligence, Expert Systems and Neural Networks, Ed. Hamza, M., IASTED/Acta Press, Honolulu, Hawaii, pp23-26

- Mehandjiska, D., Apperley, M. D., Phillips, C.H.E., Dasari, S. and **Page, D.** (1996) Advancing information technologies through CASE, Proceedings of the 19th Australasian Computer Science Conference (ACSC'96), Ed. Ramamohanarao, K., Melbourne, Australia, pp213-222.
- Dasari, S., Mehandjiska, D. and **Page, D.** (1995) Construction of a Generic Knowledge Base for a Methodology Independent CASE Tool, Addendum to the Proceedings of The Second NZ International Two-Stream Conference on Artificial Neural Networks and Expert Systems (ANNES'95), Dunedin, pp466-473
- Mehandjiska, D., Apperley, M.D., Phillips, C., **Page, D.** and Clark, P. (1995) A Methodology independent object oriented CASE tool, New Zealand Journal of Computing, Vol. 6, pp95-105
- Mehandjiska, D., **Page, D.** and Ham, J. (1995) Template generator for methodology independent object oriented CASE tool, Proceedings of 2nd International Conference on Object-Oriented Information Systems (OOIS'95), Eds. Murphy, J. and Stone, B., Springer-Verlag, Dublin, Ireland, pp431-440
- Page, D.**, Clark, P. and Mehandjiska, D. (1994) An Abstract Definition of Graphical Notations for Object Orientated Information Systems, Proceedings of 1st International Conference on Object-Oriented Information Systems (OOIS'94), Eds. Patel, D., Sun, Y. and Patel, S., Springer-Verlag, London, pp266-276
- Mehandjiska, D., **Page, D.** and Clark, P. (1994) An Intelligent Object Oriented CASE Tool, Proceedings of 1st International Conference on Object-Oriented Information Systems (OOIS'94), Eds. Patel, D., Sun, Y. and Patel, S., Springer-Verlag, London, pp168-172