

# A Serendipitous Software Framework for Facilitating Collaboration in Computational Intelligence

by

Edwin S. Peer

Submitted in partial fulfilment of the requirements for the degree Magister Scientiae

in the Faculty of Engineering, Built Environment and Information Technology

University of Pretoria

Pretoria, South Africa

October 2004

# A Serendipitous Software Framework for Facilitating Collaboration in Computational Intelligence

by

Edwin S. Peer

## Abstract

A major flaw in the academic system, particularly pertaining to computer science, is that it rewards specialisation. The highly competitive quest for new scientific developments, or rather the quest for a better reputation and more funding, forces researchers to specialise in their own fields, leaving them little time to properly explore what others are doing, sometimes even within their own field of interest. Even the peer review process, which should provide the necessary balance, fails to achieve much diversity, since reviews are typically performed by persons who are again specialists in the particular field of the work. Further, software implementations are rarely reviewed, having as a consequence the publishing of untenable results. Unfortunately, these factors contribute to an environment which is not conducive to collaboration, a cornerstone of academia — building on the work of others.

This work takes a step back and examines the general landscape of computational intelligence from a broad perspective, drawing on multiple disciplines to formulate a collaborative software platform, which is flexible enough to support the needs of this diverse research community. Interestingly, this project did not set out with these goals in mind, rather it evolved, over time, from something more specialised into the general framework described in this dissertation. Design patterns are studied as a means to manage the complexity of the computational intelligence paradigm in a flexible software implementation. Further, this dissertation demonstrates that releasing research software under an open source license eliminates some of the deficiencies of the academic process, while preserving, and even improving, the ability to build a reputation and pursue funding.

Two software packages have been produced as products of this research: i) CILib, an open source library of computational intelligence algorithms; and ii) CiClops, which

is a virtual laboratory for performing experiments that scale over multiple workstations. Together, these software packages are intended to improve the quality of research output and facilitate collaboration by sharing a repository of simulation data, statistical analysis tools and a single software implementation.

**Keywords:** Computational Intelligence, Design Patterns, Open Source, CILib, CiClops

**Supervisor:** Prof. A. P. Engelbrecht

**Co-supervisor:** Dr. F. van den Bergh

Submitted in partial fulfilment of the requirements for the degree Magister Scientiae.

## Acknowledgements

I would like to take this opportunity to thank the following:

- First and foremost, The Lord God Almighty for giving me the ability to do this work.
- My parents, Molly and Richard, for providing me with an ideal home environment.
- Hilary, my sister who fortuitously is an English studies graduate, for proof reading and checking up on my language usage. Not having a background in Computer Science must have made the work rather boring reading material, making her efforts all that much more appreciated.
- Andries, my supervisor, for providing me with guidance and putting up with all the changes in the direction that this work took to get completed.
- My good friend and co-supervisor Frans, who provided me with continuous motivation to get this work finished.
- Nathan, another good friend as well as my music teacher, for his patience during the music lessons in which I lacked the proper practice due to the time spent on research and writing for this dissertation.
- All those who have made contributions to CILib<sup>1</sup>, particularly the early adopters of the software.
- SourceForge<sup>2</sup> for hosting the CILib project.
- The University of Pretoria, for providing me with an excellent undergraduate foundation to build upon and an environment with the necessary resources to conduct this research.
- The National Research Foundation, for providing financial support. The opinions expressed in this work and the conclusions arrived at are my own and should not necessarily be attributed to the National Research Foundation.

---

<sup>1</sup><http://cilib.sourceforge.net>

<sup>2</sup><http://www.sourceforge.net>

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Project History . . . . .	2
1.2	Motivation . . . . .	3
1.3	Scope . . . . .	4
1.4	Contribution . . . . .	4
1.5	Dissertation Layout . . . . .	5
<b>2</b>	<b>Computational Intelligence</b>	<b>6</b>
2.1	Problem Classes . . . . .	7
2.1.1	Optimisation . . . . .	8
2.1.2	NP-Complete Problems . . . . .	10
2.1.3	Supervised Learning . . . . .	13
2.1.4	Unsupervised Learning . . . . .	14
2.2	Neural Networks . . . . .	17
2.2.1	Feed Forward Neural Networks . . . . .	17
2.2.2	Different Network Architectures . . . . .	20
2.2.3	Learning Vector Quantiser . . . . .	21
2.2.4	Self Organising Feature Maps . . . . .	22
2.3	Evolutionary Computing . . . . .	25
2.3.1	Genetic Algorithms . . . . .	27
2.3.2	Genetic Programming . . . . .	29
2.3.3	Evolutionary Programming . . . . .	30
2.3.4	Evolutionary Strategies . . . . .	31
2.3.5	Cultural Evolution . . . . .	32
2.3.6	Coevolution . . . . .	33

2.4	Swarm Intelligence . . . . .	34
2.4.1	Particle Swarm Optimisation . . . . .	35
2.4.2	Ant Systems . . . . .	37
2.5	Fuzzy Systems . . . . .	42
2.5.1	Fuzzy Sets . . . . .	42
2.5.2	Fuzzy Controllers . . . . .	45
2.6	Other Paradigms . . . . .	47
2.7	Hybrid Approaches . . . . .	48
2.8	Software Requirements . . . . .	50
<b>3</b>	<b>Design Patterns</b>	<b>52</b>
3.1	Creational Patterns . . . . .	54
3.1.1	Abstract Factory . . . . .	54
3.1.2	Builder . . . . .	55
3.1.3	Prototype . . . . .	56
3.1.4	Singleton . . . . .	57
3.2	Structural Patterns . . . . .	58
3.2.1	Adapter . . . . .	58
3.2.2	Composite . . . . .	59
3.2.3	Decorator . . . . .	60
3.2.4	Facade . . . . .	61
3.2.5	Proxy . . . . .	63
3.3	Behavioural Patterns . . . . .	64
3.3.1	Interpreter . . . . .	64
3.3.2	Iterator . . . . .	65
3.3.3	Observer . . . . .	67
3.3.4	Strategy . . . . .	68
3.3.5	Template Method . . . . .	69
3.3.6	Visitor . . . . .	70
3.4	Discussion . . . . .	71
<b>4</b>	<b>Open Source Software (OSS)</b>	<b>73</b>
4.1	Licenses . . . . .	74
4.1.1	Academic Free License (AFL) . . . . .	75

4.1.2	Apache Software License (ASL)	76
4.1.3	Artistic License (AL)	77
4.1.4	BSD Licenses	77
4.1.5	Common Public License (CPL)	77
4.1.6	GNU General Public Licenses (GPL and LGPL)	78
4.1.7	MIT License	79
4.1.8	Mozilla Public License (MPL)	79
4.1.9	Open Software License (OSL)	80
4.2	The Open Source Ecosystem	80
4.3	Business Models	81
4.4	Open Source in a South African Context	83
4.5	University of Pretoria Intellectual Property	85
4.6	Credits	87
<b>5</b>	<b>Languages and Tools</b>	<b>89</b>
5.1	XML (eXtensible Mark-up Language)	89
5.1.1	Well Formed Documents	91
5.1.2	Document Types and Schemas	91
5.1.3	Document Object Model (DOM)	94
5.2	Java	94
5.3	Java 2 Enterprise Edition (J2EE)	99
5.3.1	Persistence Layer	100
5.3.2	Application Layer	101
5.3.3	Presentation Layer	102
5.3.4	Deployment	102
5.4	XDoclet	103
5.5	JUnit	104
5.6	Summary	106
<b>6</b>	<b>CILib - Computational Intelligence Library</b>	<b>107</b>
6.1	Coding Conventions	108
6.2	Implementation Details	110
6.2.1	Domains and Types	110
6.2.2	Problem Classes	116

6.2.3	Algorithms . . . . .	119
6.2.4	Particle Swarm Optimisers . . . . .	123
6.2.5	Stopping Conditions . . . . .	129
6.2.6	Measurements . . . . .	131
6.2.7	Simulator . . . . .	133
6.3	Collaborations . . . . .	136
6.4	Limitations . . . . .	138
<b>7</b>	<b>CiClops - Collaborative Laboratory</b>	<b>142</b>
7.1	Architectural Overview . . . . .	143
7.2	Data Model . . . . .	145
7.3	Workers . . . . .	147
7.4	Client . . . . .	148
7.5	Status . . . . .	150
<b>8</b>	<b>Conclusion</b>	<b>151</b>
8.1	Summary . . . . .	151
8.2	Future work . . . . .	153
<b>A</b>	<b>List of Acronyms and Abbreviations</b>	<b>166</b>
<b>B</b>	<b>Unified Modelling Language</b>	<b>170</b>
<b>C</b>	<b>The Open Source Definition</b>	<b>172</b>
C.1	Free Redistribution . . . . .	172
C.2	Source Code . . . . .	172
C.3	Derived Works . . . . .	173
C.4	Integrity of The Author’s Source Code . . . . .	173
C.5	No Discrimination Against Persons or Groups . . . . .	173
C.6	No Discrimination Against Fields of Endeavor . . . . .	173
C.7	Distribution of License . . . . .	173
C.8	License Must Not Be Specific to a Product . . . . .	173
C.9	License Must Not Restrict Other Software . . . . .	174
C.10	License Must Be Technology-Neutral . . . . .	174



<b>D GPL Approval Letter</b>	<b>175</b>
<b>E Popular Open Source Licenses</b>	<b>176</b>
E.1 Academic Free License (AFL) . . . . .	176
E.2 Apache Software License (ASL) . . . . .	180
E.3 Artistic License (AL) . . . . .	184
E.4 BSD License . . . . .	188
E.5 Common Public License (CPL) . . . . .	189
E.6 GNU General Public License (GPL) . . . . .	194
E.7 GNU Lesser General Public License (LGPL) . . . . .	202
E.8 MIT License . . . . .	212
E.9 Mozilla Public License (MPL) . . . . .	213
E.10 Open Software License (OSL) . . . . .	224

# List of Figures

2.1	Computational Intelligence Paradigms . . . . .	7
2.2	Example TSP Network (not to scale) . . . . .	11
2.3	TSP Optimal Tour (length = 28) . . . . .	12
2.4	Hierarchical Clustering . . . . .	16
2.5	Three Layer Feed Forward Neural Network . . . . .	18
2.6	Two Layer Learning Vector Quantiser . . . . .	22
2.7	5x5 Self Organising Feature Map . . . . .	24
2.8	Example U-matrix plot . . . . .	25
2.9	Crossover Operators . . . . .	28
2.10	Genetic Program Tree Representation . . . . .	29
2.11	Cultural Algorithm . . . . .	32
2.12	Typical Neighbourhood Topologies . . . . .	36
2.13	Membership Functions for <i>Age</i> Linguistic Variable . . . . .	44
2.14	Fuzzy Controller Architecture . . . . .	45
3.1	Abstract Factory . . . . .	54
3.2	Builder . . . . .	55
3.3	Prototype . . . . .	56
3.4	Singleton . . . . .	57
3.5	Adapter . . . . .	59
3.6	Composite . . . . .	60
3.7	Decorator . . . . .	61
3.8	Facade . . . . .	62
3.9	Proxy . . . . .	63
3.10	Interpreter . . . . .	65

3.11	Iterator . . . . .	66
3.12	Observer . . . . .	67
3.13	Strategy . . . . .	68
3.14	Template Method . . . . .	69
3.15	Visitor . . . . .	70
5.1	A Simple XML Phone Book Document . . . . .	90
5.2	Phone Book Document Type Definition (phonebook.dtd) . . . . .	92
5.3	Phone Book Schema (phonebook.xsd) . . . . .	93
5.4	EJB Entity Relationship . . . . .	101
5.5	JUnit Composite Test Framework . . . . .	105
6.1	Partial Domain Grammar . . . . .	111
6.2	Domain Composite/Interpreter . . . . .	111
6.3	Domain Visitor Interface . . . . .	113
6.4	Partial Type System . . . . .	114
6.5	Domain Builder . . . . .	115
6.6	Problem Interfaces . . . . .	116
6.7	Solution Classes . . . . .	117
6.8	Optimisation Problems . . . . .	118
6.9	Fitness Classes . . . . .	119
6.10	Algorithm, Stopping Conditions and Events . . . . .	119
6.11	Optimisation Algorithms . . . . .	121
6.12	Overview of PSO Architecture . . . . .	123
6.13	Particle Decorators . . . . .	125
6.14	Velocity Updates . . . . .	126
6.15	Particle Visitors . . . . .	128
6.16	Stopping Conditions . . . . .	130
6.17	Measurements . . . . .	132
6.18	XML Object Factory . . . . .	134
6.19	Simple Simulator Configuration . . . . .	135
6.20	More Complex Simulator Configuration . . . . .	137
7.1	CiClops Overview . . . . .	144

7.2	CiClops Data Model . . . . .	146
7.3	Configuring a CILib simulation using CiClops . . . . .	148
7.4	CiClops monitoring CILib simulations . . . . .	149
B.1	Example UML Class . . . . .	170
B.2	UML Relationships . . . . .	171

# List of Tables

2.1	Fuzzy Set Theoretic Operators . . . . .	43
4.1	Open Source License Characteristics . . . . .	76
4.2	Instrumental Open Source Software . . . . .	87
5.1	NastyPSO Performance . . . . .	97
6.1	Legal Algorithms for Stopping Conditions . . . . .	131
6.2	CILib Contributors . . . . .	138

# List of Algorithms

1	Neural Network Back-propagation Training . . . . .	19
2	Learning Vector Quantiser Training . . . . .	23
3	General Evolutionary Computing Framework . . . . .	26
4	Particle Swarm Optimiser . . . . .	37
5	Ant Colony Optimiser for TSP . . . . .	39
6	Ant Colony Clustering . . . . .	41