Chapter 1

Introduction

"PLAN, v.t. To bother about the best method of accomplishing an accidental result." — Ambrose Bierce, Devil's Dictionary

Some of the most significant discoveries are those stumbled upon unintentionally. History is scattered with examples of such discoveries that have apparently come about by accident [97]. Archimedes determined a method of calculating the volume of irregular shaped objects, using displacement, when he noticed the water level rising while getting into a bath. Another example is Newton's inspiration for the theory of gravity resulting from the falling of an apple. The inspiration for and the discovery of many inventions, ranging from velcro to penicillin, was due to the sagaciousness of inventors to recognise the value of something unexpected during another, usually unrelated, activity. The phenomenon of making discoveries in this manner has become known as the "Serendipity Effect" [48].

WordNet defines serendipity as "accidental sagacity; the faculty of making fortunate discoveries of things you were not looking for". Although this work may not be as significant to mankind as the discovery of penicillin, it definitely turned out to be more important to the Computational Intelligence Research Group at the University of Pretoria (CIRG@UP)[1] than its original focus.

The following section takes the reader through the history of this research detailing how the project serendipitously grew into something more ambitious than initially intended. Next, the importance of this research is covered in Section 1.2. Since this work

[1]http://cirg.cs.up.ac.za

is only the first step in a collaborative effort, a careful scoping of what is and is not covered by this dissertation are discussed in Sections 1.3. This introduction concludes with the contribution of this research in Section 1.4 and a breakdown of the dissertation layout in Section 1.5.

1.1 Project History

This research set out with the very specific goal of creating a taxonomy of existing Particle Swarm Optimisers (PSOs, refer to Section 2.4.1) and performing an empirical analysis of their performance on various optimisation problems. To accomplish this, several PSOs and benchmark problems were implemented in C++, dubbed PSOLib, with the intention of having a flexible object oriented design to facilitate experimentation by making every aspect of the platform as configurable as possible. The lack of reflection features in C++, however, made it very difficult to configure properties of objects and their compositions at run time, leading to the investigation of Java as an implementation platform.

Java turned out to be a viable platform for multiple reasons. Most importantly, its reflection API (Application Programming Interface) enabled classes to be dynamically instantiated and composed according to a run time configuration file. Further, the built-in XML (eXtensible Mark-up Language, refer to Section 5.1) processing API was convenient, since XML was chosen as the representation for configurations. Finally, Java's performance was found to compare favourably to C++ for implementing PSOs (refer to Section 5.2).

Work began on porting the existing PSOLib code to Java, while at the same time generalising the platform to support the needs of a wider audience, at which point it became known as CILib (Computational Intelligence Library, refer to Chapter 6) and the focus of this work shifted away from PSOs. Initially, CILib was made available to other members of the CIRG@UP and it was later decided to release the software under an open source license (refer to Chapter 4) in an attempt to promote collaboration with third parties outside of the research group. Later, it became evident that there are strong merits for such a collaborative research platform, which ultimately became the subject of this research.

1.2 Motivation

The following problems, which were identified during a survey of several PSO papers [89], serve as motivation for effective collaborative research tools:

- **Duplication of effort:** In the restricted context of a research group, duplication of effort equates to lost productivity. In general, the science is better served if researchers can expend their efforts on developing new algorithms instead of writing implementations for software that already exists elsewhere. A collaborative code base can save researchers from reinventing the wheel. Further, an awareness of what is happening in industry can reduce the likelihood of duplicating work in academia which is already in active use by industry players.

- **Failure to take latest developments into account:** A collaborative code base increases awareness of what others are doing, in effect providing all participants with a more generalised view even though they specialise on their own specific work.

- **Insufficient testing on problems:** The No Free Lunch (NFL) theorem [120, 121] implies that algorithms should be tested on many problems to determine which problems they are best suited for, since all algorithms are on average equivalent when all possible problems are considered. Thus, large amounts of empirical data will need to be generated, which may have value if shared, to draw conclusions about the relative merit of different algorithms.

- **Poor parameter choices:** Good parameter choices for algorithms can be communicated as default values in a shared implementation platform. Further, a shared repository of simulation results can make researchers aware of the best results obtained for a given algorithm by other researchers.

- **Conflicting results:** Ignoring the fact that results cannot be in conflict if everyone shares the same implementation, a collaborative platform will undergo more stringent peer review and is likely to be far more reliable than throw away research code.

- **Invalid statistical inference:** Shared statistical analysis tools, which provide decision support for the best analysis method to use in a given context, can reduce the risk of researchers making incorrect assumptions about the applicability of statistical tests.

1.3 Scope

Building a collaborative framework to support the needs of a large research community requires a broad view of the subject matter in order to make it general enough to suit all parties.

For this reason, the computational intelligence field is examined in detail. Design patterns are examined as a means to manage the complexity of this broad field, ensuring a flexible software design capable of supporting the subject matter. Open source licensing is studied for the benefits it brings to a collaborative software development process. Further, this work draws on other software, tools and best practices from industry, which are unlikely to be found in scientific circles, but provide significant benefits for the software implementation.

The software implementation, however, is only discussed within the context of particle swarms, which was the original focus of this work, as a specific example demonstrating the more general framework. Further, an in depth knowledge of Object Oriented (OO) [21, 49] programming is assumed.

1.4 Contribution

The primary contribution of this work is two software components:

- **CILib** (Computational Intelligence Library), which is a shared collaborative framework for implementing computational intelligence software. Publishing it under an open source license maximises its visibility and its availability to potential collaborators.

- **CiClops** (Computational Intelligence Collaborative Laboratory Of Pantological Software), which is intended to further the collaborative goal by providing a scalable simulation environment, a shared repository of empirical data and statistical support tools.

Finally, this dissertation shows how the above mentioned frameworks facilitate collaboration in computational intelligence, while serving the dual purpose of providing documentation, introducing the framework to potential collaborators.

1.5 Dissertation Layout

The remainder of this dissertation is organised as follows:

- **Chapter 2:** The computational intelligence field is examined, illustrating its complexity and highlighting requirements for a flexible software framework capable of handling this complexity.
- **Chapter 3:** Patterns are explored as a mechanism for implementing good software design by drawing on the experience of experts.
- **Chapter 4:** Open source licensing is investigated as a means to facilitate collaboration while exposing software developers to reputation rewards and profit opportunities.
- **Chapter 5:** The languages and tools which are prerequisites for working with the software developed for this research are discussed.
- **Chapter 6:** The implementation of CILib is discussed with particular reference to the platform's use of patterns.
- **Chapter 7:** CiClops is introduced as a mechanism to address some implementation specific limitations of CILib while improving its viability as a collaborative platform.
- **Chapter 8:** This dissertation is concluded and ideas for future work are discussed.