Full citation: MacDonell, S.G., Buckingham, D., Gray, A.R., & Sallis, P.J. (2002) Software forensics: extending authorship analysis techniques to computer programs, Journal of Law and Information Science 13(1), pp.34-69.

Software Forensics: Extending Authorship Analysis Techniques to Computer Programs

Stephen G. MacDonell[1] Donna Buckingham[2] Andrew R. Gray[3] Philip J. Sallis[1]

[1] *Auckland University of Technology*
Private Bag 92006, Auckland, New Zealand
stephen.macdonell@aut.ac.nz
philip.sallis@aut.ac.nz

[2]*University of Otago*
Faculty of Law
PO Box 56, Dunedin, New Zealand
donna.buckingham@stonebow.otago.ac.nz

[3] *University of Otago*
Department of Information Science
PO Box 56, Dunedin, New Zealand
agray@infoscience.otago.ac.nz

Abstract

Software forensics is the analysis of the syntactic, structural and semantic form of software in order to identify, characterise and discriminate between the authors of software products for some legal purpose. Determining software authorship may be important in several contexts: civil litigation involving allegations of software theft or plagiarism or apportioning liability for software failure; criminal litigation in relation to computer fraud or software attacks on computer systems using viruses and other similar means. Our focus is on forensic analysis of software source code, the structured English-like implementation of the algorithm selected to undertake the task at hand. We use a fictionalised version of a recent case to illustrate the potential of software forensics to provide evidence and also review in detail the judicial reception of such material.

Keywords: Software forensics, authorship analysis, software metrics, source code, evidence

1. INTRODUCTION

Software affects the convenience and safety of our lives in applications both trivial and critical, from supermarket checkouts to airliner control systems. Software can also be used to circumvent or cause disruption in processes and systems. Sophisticated fraud can be perpetrated in banking and insurance systems through stealth software or a virus may be written and disseminated throughout an organisation by a disgruntled former employee.

Issues of authorship and ownership can therefore arise in a number of legal contexts, for instance, civil litigation involving allegations of software theft or plagiarism or the apportioning of liability for software failure; or criminal litigation in relation to computer fraud or software attacks on computer systems using viruses and other similar means. Analysis of the form and content of software source code can address these issues. Although the vocabulary and range of programming structures available are to a certain extent constrained by the rules of the underlying language, there remains significant and sufficient flexibility to allow programmers to express themselves distinctively. This level of individualism in program code makes software forensics analysis possible. Our definition of software forensics is broad – it incorporates analysis of code in terms of the syntax used, the form and layout of the code, and the semantics of the chosen vocabulary and its use.

The remainder of this paper is structured as follows:

- the form and nature of software are described as a precursor to a discussion of the methods used in software forensics;
- the possible applications of software forensics are presented, including the types of inference that may be made on the basis of the data and information obtained;
- an example of a litigation context is described and discussed ;
- the issues this kind of evidence raises in the trial forum are then considered in detail.

2. THE FORM AND NATURE OF SOFTWARE

Software is the computer-based realisation of an idea or task. That idea or task is an abstract notion, which must be transformed through one or more stages: perhaps a high level design, a more detailed algorithm and then into program source code. Source code is generally written using a programming language, which has vocabulary, syntax and grammar, like any other language. Programs can therefore be analysed from several perspectives,

incorporating aspects of form, structure and semantics. Programming languages are also characterised by their generation which generally reflects the time that they were devised (for instance, assembly language is first-generation, COBOL third-generation), and their type (procedural, object-oriented, declarative or functional).

While programming languages are by and large more formal and restrictive than spoken or written languages, programmers still retain substantial flexibility when writing source code. Individual choices can affect the manner in which the task is achieved (the steps and ordering of the algorithm used), the layout of the source code (including spacing, indentation, bordering characters used to set off sections of code), and the stylistic manner of algorithm implementation (including the choice of program statements used or variable names).

This flexibility is illustrated in the two segments of C++ source code in Figure 1.[1] Both deliver the same user functionality: calculating the mathematical function *factorial(n)* (or *n!*) of an integer value provided by the user. Each author has solved the problem differently. The first program is a relatively simple reverse loop from 1 to the user-provided value, while the second uses a more complex recursive definition.[2] The stylistic differences include the use or absence of comments, the form of variable names, use of white space and indentation, and overall levels of readability in each function. These fragments illustrate the fact that programmers can and do write very different programs to perform the same task. They are also likely to reflect the differences that would be commonly evident between the programs of their respective authors.

Although source code is the principal entity for such analysis, other code products may provide additional insights into authorship. Source code is generally not directly executable. It is often compiled into object code that is then combined (or linked) into the delivered executable, as shown in Figure 2. This executable is what end-users 'see' and run. Information about authorship can be extracted from the object/executable code by decompiling it into source code (albeit with considerable information loss).[3] This may indicate the development environment, source language and compiler used, the platform on which the software was intended to run, any libraries incorporated into the program in the compilation and linking process, and the general efficiency of the code in terms of memory and processor requirements. All of these factors reflect decisions taken by programmers, system architects or managers and thus provide a further range of indicators that may enable us to differentiate between authors.

```
// Factorial takes an integer as an input and returns
// the factorial of the input.
// This routine does not deal with negative values!

int Factorial (int Input)
{
       int Counter;
       int Fact;
       Fact=1;  // Initalises Fact to 1 since factorial 0 is 1
       for (Counter=Input; Counter>1; Counter=Counter-1)
       {
                     Fact=Fact*Counter;
       }
       return Fact;
}
```

```
int f(int x){
int a, y=1;
if (!x) return 1; else return   x*f(x-1);}
```

Figure 1. Two different C++ program segments providing identical functionality

Some programming languages and most scripting languages work on an interpreter system[4], instead of using the compiling process described above. In these cases the executable is the source code itself, since the program is executed within an environment that translates the code into machine-understandable instructions as the program operates. [5]

Irrespective of the form the program takes – source code, compiled code or linked executable – there is sufficient potential for variability in the numerous decisions made when writing code to enable us to characterise, and in some cases identify, the program author.[6] The

[1] Gray, A.R., Sallis, P.J., and MacDonell, S.G. (1998). IDENTIFIED (Integrated Dictionary-based Extraction of Non-language-dependent Token Information for Forensic Identification, Examination, and Discrimination): A dictionary-based system for extracting source code metrics for software forensics. In *Proceedings of SE:E&P'98 - Software Engineering: Education & Practice*. Dunedin, NZ, IEEE CS Press: 252-259.

[2] A sequence of instructions that can loop back to the beginning of itself and continue running until a condition has been satisfied.

[3] Generally, compilers optimise code, removing programmer-imposed structure and replacing variable names with symbols. Thus some indicators of authorship are lost in the process.

[4] A script is a sequence of instructions that is interpreted and run via another program rather than directly by the computer processor (as a compiled program is). In general, scripting languages are easier and faster to code in than the more structured and compiled programming languages and are ideal for programs of very limited capability or that can reuse and tie together existing compiled programs. However, a script takes longer to run than a compiled program since each instruction is handled by another program first (requiring additional instructions) rather than directly by the basic instruction processor.

[5] This is less common for programming languages as such but is very popular for scripting languages. In general, however, the term *executable* refers to a compiled program.

[6] In fact, as Sallis *et al*. note (supra n 1) a reasonable proportion of the work already carried out in computational linguistics for text corpus authorship

information available will obviously depend on the program's form (source code or object code), and for different purposes one form may be more useful than another. Source code generally provides the greatest amount of information.

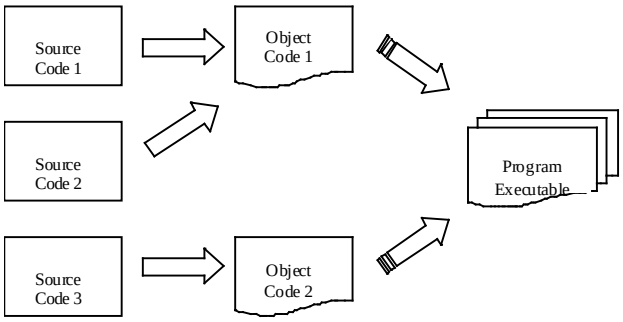

Figure 2. Program source code being compiled and linked into an executable

3. SOFTWARE FORENSICS IN AUTHORSHIP ANALYSIS

There are four principal aspects of code authorship analysis in terms of software forensics:[7]

1. ***Author identification.***

The goal here is to determine the likelihood of a particular author having written a piece of code, usually based on the characteristics of other code samples from that programmer. In order to provide a greater level of confidence in any assertion of identity it may be necessary to analyse code samples from several programmers and to determine the likelihood or even the statistical probability of the piece of code in question having been written by each. This is very similar to, for example, the attempts to verify the authorship of the Shakespearean plays or certain historical passages using linguistic and structural analysis. A digital example would be the ascribing of authorship of a new piece of source code (such as a computer virus) to a specific programmer, given that its characteristics match those in other pieces of code written by that author.

2. ***Author discrimination.***

This is the task of deciding whether one or more pieces of code were written by a single author or by a number of authors, and an estimate of the number involved. It does not extend, however, to actually identifying the authors, so large samples of code are not required. Discrimination involves determining the degree of similarity between the code segments or programs and the estimation of the level of between- and within-subject variability. The converse application can be used to test for plagiarism.[8]

3. ***Author characterisation.***

Rather than identifying a specific author, the goal of characterisation is to establish the type of person likely to have written the code in question. The aim is to determine certain characteristics of the author of a code fragment, such as gender, personality and educational background, based on their programming style. For instance, the second example shown in Figure 1 is more likely to have been written by a programmer with greater exposure to computer programming than the author of the first example, as it uses a more complex algorithm to deliver the required functionality.

4. ***Author intent determination.***

It may also be possible to determine whether code that has had an undesired effect was written maliciously or was the result of an error. Since the software development process is never error free and some errors can have catastrophic consequences, such questions can arise frequently. This analysis could also be extended to check for negligence, where erroneous code is found to be much less rigorous than that normally produced by a programmer.

4. THE PRACTICE OF SOFTWARE FORENSICS

Software forensics borrows extensively from the areas of software measurement (also known as software metrics[9]) and computational linguistics.[10]

4.1 Software measurement/software metrics

In terms of source code, software measurement/metrics focuses in the first instance on extracting a range of largely quantitative measures to construct a profile that

analysis has parallels for source code. Similarly, some of the techniques used in software measurement research and practice are transferable to software forensics. See section 4 of this paper.

[7] Gray, A.R., Sallis, P.J., and MacDonell, S.G. (1998). IDENTIFIED (Integrated Dictionary-based Extraction of Non-language-dependent Token Information for Forensic Identification, Examination, and Discrimination): A dictionary-based system for extracting source code metrics for software forensics. In *Proceedings of SE:E&P'98 - Software Engineering: Education & Practice*. Dunedin, New Zealand, IEEE CS Press: 252-259.

[8] For extensive discussion of plagiarism detection in program code, see Whale, G. (1990) Software metrics and plagiarism detection. *Journal of Systems and Software*. 13: 131-138 and Krsul, I., and Spafford, E.H. (1997). Authorship analysis: Identifying the author of a program. *Computers & Security*. 16(3): 233-256.

[9] The measurement of attributes of products, processes and resources involved in the development and use of computer software and systems.

[10] The use of automated methods to extract and analyse characteristics of language expressed either verbally or in written form.

Measure/Indicator	Description
Layout	
WHITE	Proportion of lines that are blank
SPACE-1	Proportion of operators with white space on both sides
SPACE-2	Proportion of operators with white space on left side
SPACE-3	Proportion of operators with white space on right side
SPACE-4	Proportion of operators with white space on neither side
LOCCHARS	Mean number of characters per line
Style	
CAPS	Proportion of letters that are upper case
LOC	Non-white space lines of code
DBUGSYM	Debug variables per line of code (LOC)
DBUGPRN	Commented out debug print statements per LOC
COM	Proportion of LOC that are purely comment
INLCOM	Proportion of LOC that have inline comments
ENDCOM	Proportion of end-of-block braces labelled with comments
Structure	
GOTO	Gotos per non-comment LOC (NCLOC)
COND-1	Number of #if per NCLOC
COND-2	Number of #elif per NCLOC
COND-3	Number of #ifdef per NCLOC
COND-4	Number of #ifndef per NCLOC
COND-5	Number of #else per NCLOC
COND-6	Number of #endif per NCLOC
COND	Conditional compilation keywords per NCLOC
CCN	McCabe's cyclomatic complexity number
DEC-IF	if statements per NCLOC
DEC-SWITCH	switch statements per NCLOC
DEC-WHILE	while statements per NCLOC
DEC	Decision statements per NCLOC

Table 1. Measures and indicators extracted to enable software forensic analysis

reflects a programmer's approach to programming. A vast number of different stylistic and structural measures can be extracted. Some examples are shown in Table 1 (for the C++ programming language). These measures have been chosen not only for their potential in enabling us to identify, characterise and discriminate between authors, but also because they can all be extracted automatically using a code parsing tool.[11] This is an important pragmatic requirement since, for all but the smallest programs, the volume of code to be analysed could be very high, rendering human analysis impractical.

4.2 Computational linguistics

The measures and indicators listed in Table 1 do not reflect code semantics. This form of analysis falls more appropriately within the bounds of computational linguistics[12] and could be especially valuable where extensive comments are available within or supplementary to the code. A number of characteristics may be useful to consider:

- writing style, the level of language used, unusual forms of expression
- preference for short or long statements/sentences
- frequency and severity of errors in spelling and grammar
- use of profanities, corrupted forms
- the degree to which comments match the code
- meaningfulness of identifiers and variable names
- the degree of code optimisation
- the complexity of the flow of control through the code.

It should be evident that it would generally not be possible to automate collection of such 'measures' and an expert would be needed to determine relevant values for each.[13] Several other measures and indicators could also be useful, depending on the code available and on the purpose of the analysis being undertaken.[14] This might

[11] A code parser is a program that takes each statement that a developer has written and divides it into parts (for example, the main command, options, target objects, and so forth) that can then be used for developing further actions or for creating the instructions that form an executable program.

[12] For example, see Ledger, G. (1995) An exploration of differences in the Pauline epistles using multivariate statistical analysis. *Literary and Linguistic Computing* 10:85-98; Waugh, S., Adams, A. and Tweedie, F. (2000) Computational stylistics using artificial neural networks. *Literary and Linguistic Computing*. 15:187-198.

[13] In any analysis there is likely to be a trade-off between the ease of data extraction through automation and the richness of the resulting data set. Use of only those measures shown in Table 1 would mean that the entire data set could be extracted automatically, but the measures of code semantics could in some cases provide a much stronger body of potential evidence on the issue of authorship.

[14] These and other similar measures have been suggested by: Spafford, E.H. (1989). The Internet worm program: an analysis. *Computer Communications Review*. 19(1):

include measures of the depth of nesting[15] in the code, the number of each type of data structure[16] used and the use of system and library calls.[17].

4.3 Analysis of code - early studies

'Software forensics' was originally coined by the computer security community to describe the measurement-based analysis of code specifically written with malicious intent. The two most discussed incidents where such code has been examined in order to identify or characterise the program authors relate to 'worms': stand-alone programs that propagate by making copies of themselves. These studies illustrate both the viability of software forensics and the type of information that can be obtained.

Internet Worm[18]

Spafford's analysis of the Internet Worm (released onto the Internet in November 1988) using a number of forensics measures led to the following conclusions:

- The code was not well written and contained many errors and inefficiencies.
- The code contained little error-handling behaviour, suggesting that the author was sloppy and performed little testing. Alternatively the worm's release may have been premature.
- The data structures used were all linked lists that were inefficient and indicated a lack of advanced programming ability and/or tuition.
- The code contained redundancy of processing.
- A section of the program that performed cryptographic functions was exceptionally efficient and provided functionality not used by the worm. According to Spafford (1989) this did not appear to have been written by the author of the rest of the worm.

This impressive list of observations indicates the amount of knowledge that can be extracted from such source code. Especially important in terms of author characterisation are the observations of the lack of ability of the author, the poor quality of the code and the evidence of dual or multiple authorship.

The WANK and OILZ worm[19]

These worms were released in 1989, principally attacking two US government systems. Both were written in the same language (DEC's Digital Command Language, or DCL), with the WANK worm preceding OILZ by about two weeks. The fact that the worms were written in DCL, a scripting language, and were therefore not compiled, provided much more information than would have been available from a compiled version. After drawing the overall conclusion that three distinct authors had worked on the two worms, Longstaff and Schultz were also able to suggest the following:

- Author one:

  employed an academic style of programming

  used descriptive and lower case variable names

  produced complex program flow based on variables, GOTOs, and subroutines

  had a high level of understanding

  was intent on experimentation rather than malice

- Author two:

  wrote malicious code with hostile intent

  made use of profanities in her/his code

  employed capitalisation

  adopted a simple programming style

- Author three:

  combined the others' code

  employed mixed case

17-49; Whale, G. (1990). Software metrics and plagiarism detection. *Journal of Systems and Software*. 13: 131-138; Spafford, E.H., and Weber, S.A. (1993). Software forensics: can we track code to its authors*? Computers & Security*. 12: 585-595; Krsul, I. (1993). *Authorship Analysis: Identifying the Author of a Program*. Technical Report CSD-TR-94-030, Department of Computer Sciences, Purdue University; Longstaff, T.A., and Schultz, E.E. (1993). Beyond preliminary analysis of the WANK and OILZ worms: a case study of malicious code. *Computers & Security*. 12: 61-77; Thomson, R. and Murachver, T. (2001) Predicting gender from electronic discourse. *British Journal of Social Psychology* 40:193-208; de Vel, O., Anderson, A., Corney, M. and Mohay, G. (2001) Mining e-mail content for author identification forensics. *ACM SIGMOD Record* 30(4): 55-64.

[15] In programming, 'nested code' describes instructions that perform a particular function and that are contained within code that performs a broader function. The level of code nesting describes the depth to which functions are hierarchically incorporated within one another.

[16] The way in which data values are stored, influencing the time it takes to write and retrieve data – examples are B-trees and linked lists.

[17] Programs can include instructions, or *calls*, that execute functions which are available externally as part of the operating system or from libraries of pre-built components.

[18] Spafford, E.H. (1989). The Internet worm program: an analysis. *Computer Communications Review*. 19(1): 17-49.

[19] Longstaff, T.A., and Schultz, E.E. (1993). Beyond preliminary analysis of the WANK and OILZ worms: a case study of malicious code. *Computers & Security*. 12: 61-77.

used non-descriptive variable names

wrote simple code resembling BASIC

attempted to correct bugs in the code - the OILZ worm corrected some bugs evident in WANK.

These pieces of evidence could have been of considerable value in an investigation of the attack. Particularly important in this regard is the existence of multiple authors and the differences in both style and intent.

4.4 Feasibility of analysis - recent developments

If software forensics is to be effective or even feasible, much of the data extraction needs to be automated. To this end we have constructed a software environment, called IDENTIFIED, that uses measure/indicator dictionaries to collect frequency data from programs.[20] The IDENTIFIED software parses the code, counting the number of occurrences of the measures/indicators listed in a language dictionary file[21]. Once these are extracted, a number of different modelling techniques, including cluster analysis, logistic regression, and discriminant analysis, can be used to derive authorship classification or prediction models.

MacDonell and Gray[22] reported on a study of the authorship of 351 programs written by seven different authors using the set of indicators listed in Table 1. Measurement data was extracted from half of the 351 programs using IDENTIFIED and predictive models were built using three modelling methods – a neural network,[23] multiple discriminant analysis,[24] and case-based reasoning.[25] All three models correctly predicted the author of between 81% and 88% of the 175 programs remaining in the test sample.

Kilgour *et al.*[26] describe a pilot study analysis employing a combined set of automatically derived objective measures of form and structure and expert-assigned subjective indicators of code semantics (including the degree of match between comments and code), as evident in a sample of textbook programs written by two authors. To enable greater differentiation, the assessment of each subjective factor was expressed using one of five fuzzy values[27], ranging from 'Never' through to 'Always'. As this was a pilot study, no predictive models of authorship were built. The hybrid approach to analysing authorship did, however, enable a richer and more diverse set of data to be collected so that greater distinction could be made between the programs.

5. A LITIGATION EXAMPLE

We have recently applied the techniques of software forensics to litigation in which there had been an accusation of code theft.[28] A former employee (A) of a custom systems development company developed a software product that competed directly with that of his former employer (B). B asserted that A had stolen source code from the original product base while employed and had used it in developing his competing product. The two

---

[20] For a fuller description of the tool set, see Gray, A.R., Sallis, P.J., and MacDonell, S.G. (1998). IDENTIFIED (Integrated Dictionary-based Extraction of Non-language-dependent Token Information for Forensic Identification, Examination, and Discrimination): A dictionary-based system for extracting source code metrics for software forensics. In *Proceedings of SE:E&P'98 - Software Engineering: Education & Practice*. Dunedin, New Zealand, IEEE CS Press: 252-259.

[21] A list of commands, reserved words and characteristics that are relevant for a specific programming language. Use of separate dictionary files means that the parser itself is language-independent.

[22] MacDonell, S.G. and Gray, A.R. (2001). Software forensics applied to the task of discriminating between program authors. *Journal of Systems Research and Information Systems* 10: 113-127.

[23] A neural network is a system that approximates the operation of the human brain. A neural network usually involves a large number of processors operating in parallel, each with its own small sphere of knowledge and access to data in its local memory. Typically, a neural network is initially trained or fed large amounts of data and rules about data relationships. Depending on its structure a program can then tell the network how to behave in response to an external stimulus, or it may learn patterns and behaviour from the training data.

[24] Multiple discriminant analysis is useful for building a predictive model of group membership based on observed characteristics of each instance. It is a statistical procedure that generates a set of discriminant functions based on linear combinations of the predictor variables that provide the best discrimination between the groups.

[25] Case based reasoning is a method for modelling the relationship between a series of independent variables and a dependent variable by storing and retrieving cases (observations) in a database. When presented with a new observation, the cases that are similar in terms of the independent variables are retrieved and the dependent variable calculated from them using either the nearest neighbour or some form of 'averaging' process.

[26] Kilgour, R.I., Gray, A.R., Sallis, P.J., and MacDonell, S.G. (1997). A fuzzy logic approach to computer software source code authorship analysis. In *Proceedings of the Fourth International Conference on Neural Information Processing - The Annual Conference of the Asian Pacific Neural Network Assembly (ICONIP'97).* Dunedin, New Zealand, Springer-Verlag: 865-868.

[27] Numbers or labels that are imprecise, such as "about 40", "always", or "large". Use of such numbers or labels enables better incorporation of uncertainty and subjectivity in our measurement.

[28] As there remains a need for confidentiality we here use fictional labels to represent the parties involved in the dispute. Apart from this the facts of the dispute and our analysis are faithfully reported.

systems were examined to determine the level of evidence to support or refute this assertion.

Our analysis involved the following four steps:

1. Examination of the fundamental form and structure of the two systems
2. Comparison of the source code to determine whether a significant amount of identical code existed in both products, perhaps implying that one product had been copied from the other
3. Consideration of the technical implementation of each product and the distinctions that might have arisen from any differences in their implementation
4. Stylistic and structural comparisons of the source code.

Step 1:

Our straightforward review of the form and structure of the two systems enabled us to draw two overall conclusions. First, the interfaces of the two systems were developed using different languages (Pascal and C++), using different component libraries (VCL and MFC respectively). The effort to convert between these would have been prohibitive, implying that each had been written independently. It was also clear that the structure of the respective system engines[29] was *very* different. A's product principally used C++ and an object-oriented[30] style of coding with a small amount of procedural C code, whereas B's system was written entirely in C using a procedural style.

Step 2:

We then undertook a detailed comparison of the lines of code in the two systems. The degree of correspondence of code was minimal, at approximately 3.5% of the total lines of source code. This was no more than could have been expected to occur by (i) coincidental matching (which occurs when possibilities for expression are constrained by limited syntax as they are in programming languages); and (ii) the fact that both systems had at their foundation fundamental public domain programming work done by yet another set of authors in the early to mid 1990s (public A and B respectively). The degree of correspondence in lines of code is represented in Figure 3.

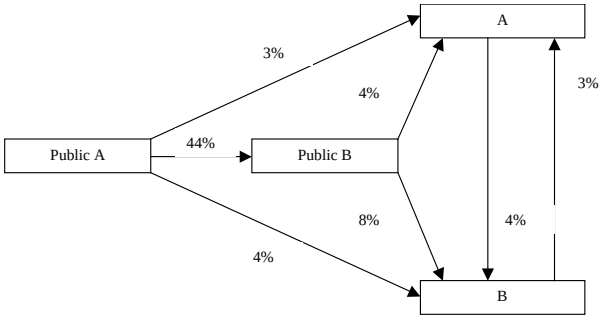

Figure 3. Extent of source code correspondence between products of parties A and B

Step 3:

The fundamental differences in form and structure and the lack of correspondence of source code indicated entirely different implementations of, admittedly, a similar set of functional modules. However, even given a similar overall aim for the two systems, a number of significant differences in the technical configuration of the two systems were evident:

- the clear and deliberate separation of interface and implementation in A's product was in direct contrast to the move towards integration evident in B's product
- the distinct use of structured tables in A's product
- B's product relied on objects being checked at the time they were run, an approach not evident in A's product.
- the basic data type sets[31] in the two products were different.

Step 4:

It is in this step that the forensics methods were applied. Stylistic and structural examination used 24 of the 26 indicators listed in Table 1, extracted automatically using the IDENTIFIED product. (As the products were written in different languages use of the LOCCHARS and LOC measures was not appropriate.) A summary of the results compiled from the 175 files in A's product and the 28 in B's product are shown in Table 2. All values are percentages except for CCN, the mean number of unique execution paths though a program or system.

In terms of this analysis the following differences in style were observed:

- B's code contained proportionally more white space lines than A's code

[29] The part of a software system that processes data. This may be contrasted with a system interface, which handles input and output, or a database, which can be used to store the data.

[30] Object-oriented programming (OOP) is organized around 'objects' rather than 'actions', data rather than logic. Historically, a program was viewed as a logical procedure that took input data, processed it, and produced output data. The programming challenge was seen as how to write the logic, not how to define the data. Object-oriented programming takes the view that what we really care about are the objects we want to manipulate rather than the logic required to manipulate them. Examples of objects range from human beings (described by name, address, and so forth) to buildings and floors (whose properties can be described and managed) down to the little widgets on your computer desktop (such as buttons and scroll bars).

[31] A basic data type in a programming language is a set of data with values having predefined characteristics. Examples of data types are: integer, character, string, and pointer. Usually, a limited number of such data types come built into a language. The language usually specifies the range of values for a given data type, how the computer processes the values, and how they are stored.

- different patterns of white space around operators were evident, with B's code especially more likely to have spaces on both sides or the right side (although the median values are greater for all space patterns apart from no space)
- B's code contained proportionally more upper case characters than A's code
- A's code contained proportionally more comments
- the comments in B's code were more likely to be inline comments than block comments.[32]

The following differences in structure and flow were observed:

- B's code made proportionally greater use of compiler directive conditionals than A's source code
- the cyclomatic complexity[33] of B's code was much higher than A's code
- B's code made proportionally greater use of conditional decisions, specifically the IF and WHILE statements.

The structural and stylistic analysis enabled us to conclude that the two sets of code followed different conventions for comments and white space. Further, B's code contained proportionally more branching and looping structures, thus producing higher values for cyclomatic complexity at the program module level.

We found virtually no evidence to support the assertion that A had stolen source code from the original product base while employed by B and had then used it in the development of his competing product. When this information was presented, B elected to withdraw from litigation.

5.1 Confounding issues from the software analyst's perspective

While there is significant and increasing potential in the application of software forensics to legal issues, there are several factors that could confound such analyses.

- There is an undetermined lower limit in terms of the volume of code written by various authors that is required in order to enable an author to be identified.
- It is almost certain that novice programmers who have undertaken some form of tuition (either self-taught or via instruction) adopt the approaches and styles learned during their training. Their programs are therefore likely to reflect the characteristics of code seen in textbooks or help files or as taught to them. Similarly, an organisation may have a strict set of coding standards to which all programmers adhere, thus reducing the distinctions that might arise if programmers were allowed to use their 'natural' approach.
- While the degree to which source code reuse is actually practised remains unclear, the influence of reused code does need to be borne in mind.
- Authors may collaborate on the production of a piece of source code, perhaps blurring the particular characteristics of an individual programmer.

Measure/Indicator	**Mean**			**Median**		
	Party A	Party B	*Difference*	Party A	Party B	*Difference*
Layout						
WHITE	11.7	14.3	*-2.6*	10.6	13.6	*-2.9*
SPACE-1	9.5	16.1	*-6.6*	6.7	14.9	*-8.2*
SPACE-2	21.6	21.4	*0.2*	14.4	19.6	*-5.1*
SPACE-3	17.5	34.3	*-16.8*	17.1	34.0	*-16.9*
SPACE-4	58.3	33.8	*24.4*	57.8	34.7	*23.2*
Style						
CAPS	21.8	24.1	*-2.3*	19.6	20.8	*-1.3*
DBUGSYM	1.3	0.1	*1.2*	0.0	0.0	*0.0*
DBUGPRN	0.0	0.0	*0.0*	0.0	0.0	*0.0*
COM	34.0	27.8	*6.2*	31.7	23.4	*8.3*
INLCOM	3.9	10.4	*-6.5*	1.1	1.8	*-0.8*
ENDCOM	0.0	0.2	*-0.2*	0.0	0.0	*0.0*
Structure						
GOTO	0.0	0.0	*0.0*	0.0	0.0	*0.0*
COND-1	0.1	0.7	*-0.7*	0.0	0.0	*0.0*
COND-2	0.1	0.2	*-0.2*	0.0	0.0	*0.0*
COND-3	0.4	2.2	*-1.8*	0.0	0.6	*-0.6*
COND-4	1.9	0.9	*1.0*	0.8	0.0	*0.8*
COND-5	0.2	0.5	*-0.3*	0.0	0.0	*0.0*
COND-6	2.4	3.8	*-1.5*	1.5	2.0	*-0.5*
COND	5.0	8.5	*-3.5*	3.0	4.5	*-1.5*
CCN	28.4	47.0	*-18.7*	5	6.5	*-1.5*
DEC-IF	1.8	5.4	*-3.6*	0.0	4.1	*-4.1*
DEC-SWITCH	0.1	0.1	*0.0*	0.0	0.0	*0.0*
DEC-WHILE	0.2	0.5	*-0.3*	0.0	0.0	*0.0*
DEC	2.1	6.0	*-4.0*	0.0	4.2	*-4.2*

Table 2. Differences in style and structure between products of parties A and B

- Some measures of coding style (such as WHITE, COM, and the SPACE- measures) are fairly trivial

[32] Inline comments: normally brief notes written by the programmer that are included on the same line as a program instruction. Block comments: documentation included within the code but on lines separate from the program's executable instructions.

[33] A measure which reflects the number of paths through a set of code instructions. It was formulated from graph theory by McCabe in 1976, and is said to reflect the structural complexity of code resulting from jumps, branches and loops.

(although relatively time consuming) to manipulate. In this way a programmer may be able to disguise their code so that it has a different profile to that which they normally produce. It is therefore important that other measures related to code semantics, programming structure and style, which are more difficult to manipulate without affecting the functionality of the software (such as the DEC- measures and CCN), are also included alongside the layout-oriented indicators to provide as comprehensive a profile as possible.

The fundamental assumption of software forensics is that programmers tend to have coding styles that are distinct, at least to some degree. As such, these styles and features are often recognisable to their colleagues, or to experts in source code analysis who are provided with samples of their code.[34] This leads to the evidentiary question: is there sufficient information available using these techniques to provide adequate authorship evidence for use at trial?

6. THE RECEPTION OF SOFTWARE FORENSICS EVIDENCE IN LITIGATION

Code authorship, as an aspect of software forensics, has yet to figure large in New Zealand case law. In both the civil[35] and criminal[36] contexts, litigation has tended to focus on ownership or establishing the presence of unauthorised material on a defendant's computer or proving that it was used to access particular information.

Five evidentiary rules[37] will be triggered by the eventual tender of evidence which analyses the similarities/differences between code incontrovertibly created by a particular author and the code under forensic scrutiny:

- *Common knowledge rule*: does the tribunal of fact need expert help on this issue or can it rely on its general knowledge and common sense?
- *Expertise rule*: does the witness have knowledge and experience sufficient to entitle them to express an opinion on this issue?
- *Area of expertise rule*: is the proposed content of the opinion sufficiently accepted or recognised by others capable of evaluating its theoretical basis?
- *Ultimate issue rule*: is the effect of the opinion equivalent to subverting the function of the tribunal of fact in determining the issue before the court?
- *Basis rule*: to what extent can the opinion found itself on matters not directly within the expert's own observations?

6.1 The common knowledge rule

In New Zealand, the rule is a matter of degree rather than a rigid exclusionary regime:[38]

---

[34] Spafford and Weber (1993) comment that evidence of identity may remain even after the attempt to disguise; some aspects of a programmer's style cannot be changed if they are to program in an effective manner.

[35] For example, *Pacific Technology Ltd v Perry Group Ltd* [2004] 1 NZLR 164 (CA) considers ownership but only on the formally agreed basis of whether an order for delivery up of the source code was appropriate and includes discussion of the technical issues inherent in establishing authorship. The central issue was whether, in the context of the commissioning of the source code, prior copyright in some incorporated elements still subsisted.

[36] As in other jurisdictions, classifying electronic acts within existing offences has challenged existing criminal law and reform has been necessary. The New Zealand Law Commission provided impetus in *Computer Misuse* (1999) NZLC R 54. The Crimes Amendment Act 2003 creates the offences of accessing a computer system for a dishonest purpose, damaging or interfering with a computer system, making, selling or distributing software for unauthorised access to a computer system in order to commit crime, and accessing a computer system without authorisation. Until its coming into force on 1 October 2003, the existing provisions of the Crimes Act 1961 were flexed to sanction digital offences. Much of the judicial discussion focused on the interpretation of 'document' in a digital context. In *R v Misic* [2001] 3 NZLR 1 the NZ Court of Appeal ruled that using a computer program to make unpaid international toll calls constituted fraudulently using a 'document' to obtain a pecuniary advantage. A computer password and log-in were also ruled a 'document' in *R v Garrett (No 2)* [2001] DCR 912.

[37] This comment in *R v Flaws* (1998) 16 CRNZ 216, 219 neatly captures the rules: "Generally it will be appropriate to instruct the jury that expert evidence is an exception to the rule that witnesses must speak only as to observed facts and are not permitted to express their opinions or beliefs. We would normally expect a jury to be directed about the fact that opinion evidence is received on a subject which requires special study or experience (being beyond the ordinary experience of jurors) and that the expert witness has particular qualifications which enable that person to express an opinion. Also that the expert's opinion to be of probative value must be based on a properly established evidential foundation. The jury will be reminded that it is for them to be satisfied about the essential ingredients and that they are not bound to accept the opinion even of the most highly qualified expert."

[38] *R v Decha-Iamsakun* [1993] 1 NZLR 141, 145 where the NZ Court of Appeal ruled admissible expert evidence of a language difficulty. Other examples include *R v Tipene* (2001) 19 CRNZ 93 where the same Court ruled that the jury was legitimately assisted on identification by expert evidence in relation to still photographs taken from a bank video; *Attorney General v Equiticorp Industries Ltd* [1995] 2 NZLR 135 where expert evidence on professional legal standards was regarded as helpful in focusing the issue. In *Police v Sinclair* [1991] 3 NZLR

> Matters which to a considerable extent are within the experience of a Judge trying the facts or a jury can arise, yet expert evidence may help materially in coming to a conclusion. The ordinary experience test not need be interpreted so as to exclude some evidence. The information provided may well be outside ordinary experience and cause the Judge or jury to review impressions or instinctive judgments based on ordinary experience, and to do so in the direction of either confirmation or doubt of what ordinary experience suggests.

The trigger for questioning the tribunal of fact's own resources is often the issue of 'usual' human behaviour or 'normal' personality characteristics or the psychiatric evaluation of credibility of testimony.[39] Forging the evidentiary lines between what is admissible and what should be excluded has drawn on substantial reserves of judicial (and sometimes legislative[40]) energy in an attempt to regulate admissibility on some principled basis.[41] However, profiling the construction of computer source code is an area far removed from the common knowledge of a tribunal of fact. Given that the underlying basis of this stage of the admissibility enquiry is whether or not the fact-finder can form a conclusion without help, the need for particular expertise would seem self-evident. If fingerprint or voice identification is beyond common knowledge[42] then the kind of digital 'fingerprinting' inherent in the comparison of code is even further removed.

6.2 The expertise rule

Computing expertise has been received in a number of trial contexts.[43] None involve code authorship as a fact in issue, although the principle of qualification remains the same. The New Zealand Court of Appeal has recently reaffirmed that a proposed witness may acquire expertise by a formal or informal route: "A witness need not undertake a course of scientific study to qualify as an expert and that his or her knowledge may be acquired from experience, as distinct from a professional course of studies. … [P]ersons may qualify themselves in respect of the subject matter of a particular case."[44] Where code authorship is at issue, such an approach might precipitate interesting forays into the world of 'geeks', although there is no case law in which this has yet occurred. It is certainly not a closed possibility.

6.3 The area of expertise rule

The New Zealand Court of Appeal in *R v B*[45] has explored the requirement that the content of the opinion lies within an accepted area of expertise:

> As a precondition of admissibility the subject-matter to which the expert opinion relates must be a

569 Tipping J suggested expert evidence of the driving standard required in an off-road rally would have been desirable rather than 'judicial intuition' being used to determine the driver's degree of care.

[39] E.g. in *R v Mesui* CA 471/99, 2/12/1999, the Court of Appeal supported the trial judge's decision in a murder trial to receive expert evidence of the effect on a Tongan person, in a cultural context, of the victim's offensive statements about the accused and his family at a church function in the presence of other Tongan men.

[40] In relation to child complainants in sexual cases, s 23G Evidence Act 1908 provides for expert evidence relating to: the intellectual attainment, mental capability and emotional maturity of the complainant based upon pre-trial examination or upon observation of the complainant giving evidence; the general developmental level of children of the same age group; and whether evidence given by any other witness about the child's behaviour is consistent or inconsistent with that of sexually abused children of the same age group.

[41] *R v Makoare* [2001] 1 NZLR 318, 323 is the most recent redrawing of common knowledge parameters in the context of human behaviour by the NZ Court of Appeal, endorsing expert evidence in relation to 'counter-intuitive' behaviour (that which 'does not conform with what a layperson might expect'). It was again endorsed in *R v Hurihanganui* CA 81/03, 24 October 2003 in relation to Asperger's syndrome and schizo-affective disorder and their relevance to the jury's assessment of the reliability and credibility of a confession.

[42] *R v Buisson* [1990] 2 NZLR 542 (fingerprint analysis). In *R v Carroll (No 28)* HC Auckland T002481 2 May 2003, Williams J provides a review of the current NZ approach to voice identification, particularly in relation to the production of an evidential transcript from tapes obtained via an interception warrant.

[43] Examples include the description of the operation of a program: *R v Garrett* [2001] DCR 955; the analysis of computer use in relation to the behaviour of news groups: *R v Millwood* [2000] DCR 633.

[44] *R v Tipene* (2001) 19 CRNZ 93, 97 citing *R v Menzies* [1982] 1 NZLR 40, 49 and *R v Howe* [1982] 1 NZLR 618, 627 together with a number of Canadian and English decisions. An unusual example of informal expert witness qualification is offered by *R v MacDonald* CA 55/95, 6 July 1995 where 2 prosecution witnesses testified that they had seen and used cannabis before and that the substance at issue in the trial looked like cannabis. However the Court of Appeal did concede it was somewhat artificial to label the two as 'experts'. More recently in *Hume v Police* AP 24/99 High Court Invercargill 2 September 1999, the High Court accepted expertise in the development of bruising could arise from an amateur undertaking (here world class expertise in marital arts) rather than arising from a profession or a course of scientific study. Recently, the NZ Court of Appeal has held that there is no rule that evidence of a well-qualified expert becomes inadmissible simply because a better-qualified person exists who could have given evidence: *R v Lapalapa* (2003) 20 CRNZ 115.

[45] [1987] 1 NZLR 362, 367

> sufficiently recognised branch of science at the time the evidence is given. For this reason the fields on which expert evidence will be allowed may be expected to be enlarged as research establishes the accuracy of knowledge in that field. Whether the area on which the witness seeks to express an opinion is properly the subject of expert opinion and whether an individual witness is an expert in a field will be for the Court to decide in the light of the knowledge prevailing at the time the opinion is proffered.

In *R v Makoare*[46] the same Court stated: "It is not enough for a witness, however eminently qualified in his or her field, simply to advance a theory or offer an explanation in the absence of supporting literature or other verification of the pedigree of their opinion." These observations presage the issue of 'novel' scientific evidence, which will arise if the Court were to treat the practice of software forensics as new scientific theory or methodology.[47]

R v Calder[48] reflects the current position on 'novel' scientific endeavour. Charges of attempted murder and of causing poison to be taken with intent to cause grievous bodily harm were based on the alleged administration of acrylamide (a poison). The Crown sought to rely on both clinical symptoms and hair and blood analysis. The defence objected to admission of the hair analysis, arguing it breached the area of expertise rule as novel scientific evidence. The novelty arose from the technique being applied to a body part other than blood. Tipping J adopted the concept of the judge as 'gatekeeper' of an evidentiary threshold which is crossed in discrete steps:

a) Does the evidence logically tend to show that a fact in issue is more or less likely? This governs all admissibility inquiries and is trial context dependent.

b) Does the proposed evidence show a sufficient claim to reliability? This involves ascribing a quality of helpfulness. The decision cites as useful factors listed in the Canadian cases of *R v Johnston*[49] and *R v Melaragni*[50] although these were not reproduced or individually applied.

c) Is the evidence more probative than prejudicial?

This overarching exclusionary jurisdiction applies to all admissibility questions, although Tipping J indicated that it would be rare[51] for evidence which crosses the relevance/helpfulness threshold nonetheless to be excluded.[52]

R v Calder echoes the general judicial criticism of a stricter test of 'general acceptance of the technique or theory within its scientific community', a standard set by *Frye v US.*[53] This requirement dogged United States case law until 1993 when its influence was moderated, at least in terms of the Federal Rules of Evidence, by the United States Supreme Court decision in *Daubert v Merrill Dow Pharmaceuticals.*[54] The decision also incorporates the position of the New Zealand Law Reform Commission:[55] "[T]he theory need not be accepted by all or most scientists working in the area. That is too high a standard. Theories which are newly developed or which represent the views of a minority may still be reliable and helpful." Therefore the innovative nature of the methodology for attributing code authorship will not of itself preclude forensic consideration.

---

[46] [2001] 1 NZLR 318, 324

[47] The NZ Court of Appeal observed in *R v Zhang* CA 216/98, 12 August 1998: "We do not consider evidence of the observance of actions by card players gives rise to the same issues as evidence drawing upon new fields of science, technology, psychology and the like". The reference to 'technology' suggests that novel methods of computer forensics may well fall on the other side of the line.

[48] High Court, Christchurch, T 154/94, 12 April 1995, Tipping J.

[49] *Regina v Johnston* (1992) 69 CCC (3d) 395 (DNA profiling going to the issue of identification).

[50] *Regina v Melaragni* (1992) 73 CCC (3d) 348 (the approximate bullet entry point through the rear vehicle window).

[51] Supra n 847, page 13. *R v Iese* CA 96/02, 29 August 2002 is a recent example of exclusion of evidence while acknowledging its helpfulness and probative force. The expert was a police officer and the area of expertise was the operation of and gang participation in 'tinnie' (cannabis) houses. The NZ Court of Appeal adverted to the danger that, in the absence of other evidence, a jury might treat the evidence of a tendency of a particular group to commit such crimes as probative of the charge against one accused (an admitted gang member).

[52] Applying this analysis, Tipping J ruled the evidence admissible. It was 'relevant' in that it showed the putative victim had a much higher hair concentration of CEC than the control group and therefore logically tended to establish the ingestion of acrylamide. The following factors were relevant to 'helpfulness' (the sufficient claim to reliability): hair analysis had been performed many times worldwide for other chemical compounds; neither the accused's blood analysis nor that of the control group was challenged; the extension of the isolating technique from blood to hair rested on 'perfectly intelligible scientific reasoning' (supra, note 47, page 9). The submission that the evidence was more prejudicial than probative (because it could not demonstrate any link between CEC in the hair and the alleged oral ingestion of acrylamide) failed since the medical evidence in particular excluded any external cause for the presence of CEC.

[53] (1923) 293 F 1013

[54] (1993) 509 US 579. *Daubert* established that general acceptance was not a necessary pre-condition to admissibility, simply one factor.

[55] *Evidence Law: Expert Evidence and Opinion Evidence* (1991) NZLC PP18

Assuming the need for a *Calder* style analysis, an opinion on code authorship will be highly relevant. It would go directly to identity, the likely fact in issue in any proceedings in which it is offered (whether civil or criminal). In terms of helpfulness/sufficient claim to reliability, the issue becomes less straightforward. The factors enumerated in the Canadian cases of *R v Johnston*[56] and *R v Melaragni*[57] or drawn from the general guidelines in *Daubert*[58] relate to evidence based on scientific theory or technique (e.g. falsification by empirical testing or rate of error). This is qualitatively different from evidence based on specialised knowledge or skills.[59] For example, evidence of DNA profiling *(Johnston)* or the analysis of a bullet's trajectory *(Melaragni)* is designed to help the fact finder deal appropriately with a piece of real evidence. The techniques are capable of replication. Can this be said of expert opinion as to code authorship, based on a comparative analysis between code whose authorship is known and the code which is itself a material fact?

Certainly some of the factors in those cases could be adapted relatively painlessly to help determine threshold reliability. For example, relevant *Johnston* factors might include:

- the existence and maintenance of standards;
- the expert's qualifications and stature;
- the existence of specialised literature;
- the nature and breadth of the inference adduced;
- the clarity with which the technique may be explained;
- the extent to which the basic data may be verified by the court and jury;
- the availability of other experts to evaluate the technique;
- the probative significance of the evidence.

The relevant *Melaragni* factors bear a more direct fact-finder focus, but raise similar issues:

- is the evidence likely to assist fact-finding or to confuse and confound?
- is the jury likely to be overwhelmed by the mystic infallibility of the evidence, or will it be able to objectively assess worth?
- will the evidence, if accepted, conclusively prove an essential element of the crime which the defence is contesting or is it part of a larger puzzle?
- what degree of reliability has the proposed scientific technique or body of knowledge achieved?
- are there sufficient experts available so that the defence can retain its own?
- can the defence independently test the scientific technique or body of knowledge?
- are there clear policy or legal grounds which would render the evidence inadmissible despite its probative value?
- will the evidence cause undue delay or result in the needless presentation of cumulative evidence?

Most of these factors would present as relevant whether the fact finder is a jury or a judge sitting alone and are capable of adaptation to the civil context (where the judge almost always sits alone and will therefore assume the fact finding function).

In terms of the final probative value/prejudice inquiry, *R v Calder* notes that the capacity of the evidence to provide a logical step against the accused is not sufficient for the exercise of the exclusionary discretion; the prejudice must be illegitimate. For example, "where the impugned evidence has little probative value but may lead the jury to an erroneous process of reasoning or may lead the jury to conclude that the accused is guilty on an insecure or improper basis."[60]

This is redolent of the 'mystic infallibility' reference in *Melaragni. R v Calder* finally observes that the precise weight to be given to the evidence, once admitted, depends upon its testing by cross-examination and counter evidence. "By admitting the testimony the Court is not warranting that it is necessarily accurate or reliable. The Court, acting as gatekeeper, must simply decide whether the material is worthy of consideration."[61] This observation introduces the ultimate issue rule, designed to preserve the exclusive domain of the court as arbiter of fact.

6.4 The ultimate issue rule

Evidence law demarcates the function of providing evidence from the function of ascribing evidentiary value.[62] In principle therefore the expert must not express

[56] Supra, n 948

[57] Supra, n 50

[58] Supra, n 54

[59] The distinction between evidence based on scientific theory or technique and evidence based on specialised knowledge and skills and the applicability of the guidelines on the former to the latter is recognized by the Law Commission in its commentary on the proposed Evidence Code: *Evidence: Evidence Code and Commentary* NZLC R 55 Volume 2 C100 – 101.

[60] Supra n 847 at page 13

[61] Supra n 847 at page 6

[62] The rationales for this exclusionary regime include avoiding unstated assumptions about matters in dispute, avoiding the expert effectively becoming an advocate, preventing the proliferation of opinions, overwhelming or confusing the tribunal of fact or going beyond the witness' area of expertise. See Freckleton & Selby, *Expert Evidence*, The Law Book Company 1993- para 10.10

an opinion on an issue which is for the tribunal of fact to decide. Where code authorship is in issue as a material fact, the expert opinion therefore runs the risk of being denied admission.[63] However, in New Zealand the potentially rigid exclusionary effect of the ultimate issue rule has been largely sidelined.[64] Two decades ago, the Court of Appeal in *R v Howe*[65] began to describe it as 'very much eroded'. In *Attorney-General v Equiticorp Industries Group Ltd*[66] the same Court again downplayed the absolute exclusionary nature of the rule in favour of a focus on the 'helpful' quality of the proposed evidence. The rule now focuses on whether the proposed expert evidence brings the ultimate issue into sharper relief or avoids the court inappropriately becoming its own expert. Only where the evidence of the expert is untested (for example by the absence of conflicting expert opinion) will the Court feel uneasy and perhaps retreat to the over-arching exclusionary discretion. There are hints of this in the *Equiticorp* decision where the Court was careful in the civil context to preserve the discretion to exclude where there is a risk of true usurpation.[67]

6.5 The basis rule

The essence of the role of the expert is to give an opinion based on certain facts. Consequently the expert must either prove the facts on which the opinion is based or state the factual assumptions inherent in it (for which an evidential foundation is laid by other evidence). The rule is directed to the need for the tribunal of fact to critically evaluate the opinion and give it due weight. Where the expert relies on the knowledge or experience of someone who is unavailable to the court to establish a fact on which the opinion is based, the hearsay rule is then triggered and the expert evidence becomes vulnerable to exclusion. However again this rule is not absolute in its effect in New Zealand.

Courts accept the use of factual information which is not personal to the expert but which is part of the received knowledge in the area of expertise:[68] "It is true that expert evidence, founded as it is on study and experience, necessarily involves the acceptance by the expert of the opinions of others. In that sense his expertise and so his opinions are based to a significant extent on hearsay information." Therefore it is acceptable to rely on material which has been peer reviewed and then published.

In terms of unacceptable reliance (where another's knowledge of facts particular to the litigation are relied upon by the expert), discrete exceptions have been carved. This has been particularly so with diagnostic history (statements of fact made by a patient to enable a medical witness to form an opinion). *R v Rongonui* rehearses the current position:[69]

> The inclusion of hearsay evidence in expert testimony is not necessarily fatal to its admission. …. If it is largely non-contentious and the surrounding circumstances make it probable that it is true, it may be unduly technical to exclude it. That will often be the case with medical and family histories given to a psychiatrist or psychologist by the accused. … The proper course is not to exclude the opinion evidence and statement of the facts upon which it is based, but to admit it subject to a warning to the jury that the absence of direct evidence to prove the diagnostic facts may affect the weight to be given to the opinion evidence based upon them.

Whether this 'non technical' approach can be exported in an untrammelled fashion to areas outside the diagnostic history context is moot. There is no case law which touches even tangentially on the kind of research process inherent in determining code authorship and the extent to

---

[63] The only case where computer expertise has directly arisen is *Advanced Management Systems Ltd v Attorney General* HC Auckland, CP 371-SW/00, 27 April 2001, Potter J. Here application was made under Rule 324 of the High Court Rules which permits the Court to appoint an independent expert. The area of opinion would have involved the isolation of the original source code in a piece of software which had since been modified – a material issue in a case on ownership of software. However the application was declined on the basis that the request presupposed a particular legal view of the contractual arrangements under which the modifications had been done and these had yet to be established in the litigation.

[64] The New Zealand Law Commission recommends abolition of the common knowledge and ultimate issue rules (adopting a similar approach to that of the Evidence Act (Cth) 1995 in Australia). The Commission's draft Evidence Code (yet to be enacted) would replace the filtering function with a test as to whether the proposed evidence is likely to "substantially help" the fact finder to understand other evidence or ascertain a material fact. It is considered that this will more consistently fulfill the function of these two rules (to prevent usurping the fact-finding function and time wasting): *Evidence: Reform of the Law* (1999) NZLC R55 – Volume 1.

[65] *R v Howe* [1982] 1 NZLR 618, 627

[66] [1995] 2 NZLR 135

[67] Supra n 6, p 140. The decision acknowledges the line between expert evidence on matters outside the Court's knowledge and argument by an expert. Recent examples in the criminal context include: *R v J* CA 51/03 4 August 2003 where the psychiatrist-expert in a sexual offences trial effectively commented on the complainants' credibility; *R v A* CA 136/03 24 July 2003 where a doctor's impermissible opinion on the issue of consent in a rape trial was partly the basis of a successful appeal.

[68] *Holt v Auckland City Council* [1980] 2 NZLR 124, 127. The proposed Evidence Code would preserve this differentiation between the general body of knowledge/skill comprising the witnesses' expertise (not necessary to prove) and particular facts upon which the opinion is based (requiring proof): *Evidence: Evidence Code and Commentary* NZLC R 55 Volume 2, C104 – 105.

[69] [2000] 2 NZLR 385, 403

which the so-called 'basis rule' might activate a hearsay objection. Reliance on the published work of others to establish the genesis of the methodology used would certainly be acceptable. Less acceptable might be the reliance on the work of a colleague in relation to particular litigation who is potentially available to the court but who is not called. There is persuasive authority which suggests that where scientific tests are run by assistants (who are available to the court), they must be called to give evidence as to their results as a necessary precondition of an expert giving evidence thereon.[70] However even in this situation, the matter is not free from doubt. In passing, the Court of Appeal has observed:[71]

> Scientific conclusions are frequently the result of team activity. It is conceivable that there is, or should be, a common law exception to the hearsay rule in circumstances where a conclusion is expressed by an informed and responsible member of a scientific team and where the opposing party has not objected to evidence in that form after adequate prior notice.

Since this observation was made without the benefit of argument, the wisest course might be to make available each team member where it is clear that the work of one relies on that of another.[72] Of course, the inclusion of hearsay evidence in expert testimony is also permissible if the Court can find an existing exception to the hearsay rule to justify admission. Such exceptions are numerous, both at common law and resulting from statutory intervention.[73]

7. RECENT DEVELOPMENTS

In civil proceedings an exchange of witness briefs will almost always occur prior to trial.[74] In mid-2002 a Code of Conduct was introduced, setting out the expert's duty to the Court in High Court proceedings (to impartially assist and to refrain from advocacy) and the matters their evidence must address.[75] Some effectively incorporate aspects of the five common law rules which regulate admissibility: e.g. to state the witness' qualifications as expert; to state that the evidence the witness addresses is within this area of expertise; to state the facts and assumptions upon which the opinion is based; to specify literature or other material used or relied on in support of that opinion and to describe examinations, tests or other investigations relied upon and identify and give the qualifications of the person who carried them out.

To ensure transparency, the Code also requires that the expert state any qualification upon their evidence which may affect its completeness or accuracy; likewise if the opinion is not conclusive because of insufficient research or data or for any other reason.[76] These are matters which previously might only be elicited in cross-examination. In addition the Court may direct that expert witnesses confer to attempt agreement on matters in issue and to prepare a joint witness statement which sets out areas of agreement and disagreement and reasons for the latter.[77] A panel approach to the presentation of expert evidence is provided, once both parties have elicited the facts.[78]

---

[70] *R v Jackson* [1996] 2 Cr App Rep 420, quoting the Report on the Royal Commission on Criminal Justice (UK) 1993 (voicing concern that the rule be changed) and rehearsing alternative pre-trial strategies available to avoid such a difficulty.

[71] *R v Mokaraka* [2002] 1 NZLR 793, 802. The litigation concerned fingerprint identification and the expert referred in evidence-in-chief to 'peer review', effectively a check of her work by three other members of an identification team whom the Crown elected not to call. The Court of Appealaccepted that the evidence was hearsay (the purpose of tender was to imply that these peers had supported the conclusions of the witness) but felt that the judge had dealt adequately with the issue in summing up.

[72] See post, for a discussion of the High Court Amendment Rules 2002 in relation to expert evidence in civil proceedings. Clause 3(g) of the Code of Conduct for Expert Witnesses requires the expert to identify and give details of the qualifications of persons who carried out examinations, tests or other investigation upon which the expert relies. This seems an implicit endorsement of the Court of Appeal's obiter comment.

[73] For example, the Evidence Amendment Act (No 2) 1980 establishes a general statutory regime for admission of documentary or oral hearsay in both civil and criminal proceedings, while preserving the common law exceptions including statements against interest, statements in the course of a duty, pedigree statements, statements of public or general right and dying declarations.

[74] Rules 441A to 441I High Court Rules provide a specialized procedure. Rule 434(3) District Court Rules 1992 is the correlative general power in the District Court.

[75] High Court Amendment Rules 2002 (SR 2002/132). The changes occurred after wide consultation by the Rules Committee established under s 51B Judicature Act 1908. The Code of Conduct (Schedule 4) overlaps with the guidelines issued by the Federal Court of Australia as part of a practice direction. The ultimate source for these is the judgment of Creswell J in *Ikranian Reefer* [1993] 2 Lloyds Rep 68, 81. Clauses 1 and 2 state the duty to the Court. Clause 3 contains the 7 matters which the expert must address in their evidence (including an acknowledgement of the Code and agreement to abide by its provisions). Clauses 4 – 5 require the expert to state any qualifications upon their evidence or the conclusiveness of their opinion. Clauses 6 – 7 incorporate the duty to confer with another expert witness where so directed by the Court. Corresponding provisions are not yet in force in the District Court although the Rules Committee proposed similar change in that jurisdiction.

[76] Clauses 4 – 5, Code of Conduct for Expert Witnesses

[77] Rule 330B High Court Rules

[78] Rule 330D High Court Rules. This has been dubbed the 'hot tub' and was developed by the Australian Competition Tribunal, and adopted by the Federal Court of Australia in Order 34A, Rule 3(2). The procedure is

There are few decisions yet on the new Rules[79] and there are opposing views of their potential as an instrument of change in truly educating the court and avoiding the 'gun for hire' syndrome.[80]

While in criminal proceedings expert evidence can be supplied in advance so that the other party can elect whether to challenge admissibility,[81] there is no equivalent Code which regulates both the content and manner of presentation. However, given the current procedural climate of limited pre-trial disclosure obligations on the defence, it is understandable that this has not occurred.

8. CONCLUSION

Expert opinion as to code authorship will variously enliven the five exclusionary rules. The common knowledge rule requires little flexing as the arcane world of computer code resides well beyond the resources of the trier of fact. Whether the proposed witness has the requisite knowledge and skill is an issue of fact, but unconventional routes to such expertise do not disqualify. Computer code analysis is a recognised field and so the area of expertise is legitimately 'expert'. But the forensic methodology proposed here may in itself be 'novel' and therefore attract the tripartite *Calder* analysis (relevant/helpful/more probative than prejudicial). Certainly addressing some of the factors in that case and those it relied upon may smooth the path to admissibility since these may provide rational tools for evaluating the evidence. Expert opinion on code authorship might appear to run foul of the ultimate issue rule in some trial contexts. However an opinion coextensive with the essential question for decision is not a fatal characteristic. Finally, the intellectual basis of the opinion must be clearly addressed, either in the evidence of the expert or that of other witnesses or by admissible hearsay.

directed to ensuring the experts deal with the case on the basis of the evidence adduced. *Robert Hicks Pty Limited v Melway Publishing Pty Limited* (1999) 21 ATPR 41-668 is an example of the process. At first instance, each part was directed to close its case subject to the calling of expert witnesses. The experts were informed of the evidence given and sworn. Each was then examined and cross-examined.

[79] *Air Chathams Ltd v Civil Aviation Authority of New Zealand* (2003) 16 PRNZ 676 is one of the first decisions to consider the obligations under the common law (see n. 75) and their correlatives in the Code of Conduct, finding that almost all were breached by the expert's brief and ruling it inadmissible.

[80] *The Role and Use of Expert Witnesses*, Legal Research Foundation Seminar Series, Auckland, 7 November 2002. See in particular Baragwanath J, *The New Rules: Judicial thought on the changes*; Tom Weston QC, *The new High Court Rules.*

[81] Section 344A Crimes Act 1961 provides for an interlocutory order relating to the admissibility of evidence on the application of either party. In *R v Yu* [1998] DCR 1077 the Crown applied pre-trial to determine the admissibility of proposed expert evidence on the basis of the proposed defence objection.

SCIENTIFIC REFERENCES

de Vel, O., Anderson, A., Corney, M. and Mohay, G. (2001) Mining e-mail content for author identification forensics. *ACM SIGMOD Record* 30(4): 55-64.

Gray, A.R., Sallis, P.J., and MacDonell, S.G. (1998). IDENTIFIED (Integrated Dictionary-based Extraction of Non-language-dependent Token Information for Forensic Identification, Examination, and Discrimination): A dictionary-based system for extracting source code metrics for software forensics. In *Proceedings of SE:E&P'98 - Software Engineering: Education & Practice*. Dunedin, New Zealand, IEEE CS Press: 252-259.

Kilgour, R.I., Gray, A.R., Sallis, P.J., and MacDonell, S.G. (1997). A fuzzy logic approach to computer software source code authorship analysis. In *Proceedings of the Fourth International Conference on Neural Information Processing - The Annual Conference of the Asian Pacific Neural Network Assembly (ICONIP'97)*. Dunedin, New Zealand, Springer-Verlag: 865-868.

Krsul, I. (1993). *Authorship Analysis: Identifying the Author of a Program*. Technical Report CSD-TR-94-030, Department of Computer Sciences, Purdue University.

Krsul, I., and Spafford, E.H. (1997). Authorship analysis: Identifying the author of a program. *Computers & Security*. 16(3): 233-256.

Ledger, G. (1995). An exploration of differences in the Pauline epistles using multivariate statistical analysis. *Literary and Linguistic Computing*. 10: 85-98.

Longstaff, T.A., and Schultz, E.E. (1993). Beyond preliminary analysis of the WANK and OILZ worms: a case study of malicious code. *Computers & Security*. 12: 61-77.

MacDonell, S.G. and Gray, A.R. (2001). Software forensics applied to the task of discriminating between program authors. *Journal of Systems Research and Information Systems* 10: 113-127.

Sallis, P.J. (1994). Contemporary computing methods for the authorship characterisation problem in computational linguistics. *New Zealand Journal of Computing.* 5(1): 85-95.

Sallis P., Aakjaer, A., and MacDonell, S. (1996). Software forensics: old methods for a new science. In *Proceedings of SE:E&P'96 - Software Engineering: Education & Practice*. Dunedin, New Zealand, IEEE CS Press: 367-371.

Spafford, E.H. (1989). The Internet worm program: an analysis. *Computer Communications Review*. 19(1): 17-49.

Spafford, E.H., and Weber, S.A. (1993). Software forensics: can we track code to its authors? *Computers & Security*. 12: 585-595.

Thomson, R. and Murachver, T. (2001) Predicting gender from electronic discourse. *British Journal of Social Psychology* 40:193-208

Waugh, S., Adams, A. and Tweedie, F. (2000) Computational stylistics using artificial neural networks. *Literary and Linguistic Computing*. 15:187-198.

Whale, G. (1990). Software metrics and plagiarism detection. *Journal of Systems and Software*. 13: 131-138.

Wolfe, H.B. (1994). Viruses: what can we really do? In *Proceedings of the 10th Annual IFIP Security Conference*. Curacao, Netherland Antilles, IFIP: 692-706.

Wolfe, H.B. (1996) The Internet: sources of threat and protection. In *Proceedings of the Surveillance Exp "96.* McLean VA, USA, Ross Engineering: 98-118.