Chapter 5

Impact of design and programming patterns in software piracy forensics

This chapter is an attempt to explore and study an interesting point of how to deal with the increasing use of design and programming patterns in general software piracy forensics and to propose some judiciary-friendly formats for presenting results to judicial officers. While this point presumably comes under some of the “doctrine” exceptions of AFC, to date it does not seem to have been explicitly examined (Baboo and Bhattathiripad, 2010). The ultimate objective of this chapter is to discuss this point, in the context of the limitations of AFC.

The proposed formats in this chapter are intended for presenting results of manual or non-automated comparison of ‘original’ and ‘pirated’ software. Finally, this chapter also points out the fact that (in the context of Management Information systems) automated comparison runs the risk of overlooking certain tangible elements of commonalities which may be dismissed as insignificant or incidental or predictable. Hence, this chapter emphasizes through out that it is good practice to supplement automated comparison with a manual one too.

The predicament of comparing two source codes arises usually when one party lodges a complaint of software piracy against the other. The ‘pirated’ source code is generally made available to the expert by the police or judiciary through a seizure procedure (Marcella and Menendez, 2008). A full-fledged forensic investigation of the given code has to be done to establish source code piracy. As it involves technical comparison, the judge would appoint an impartial technical expert for comparison. Given the source code from two different software systems, the technical expert concentrates on finding out the similarities and commonalities and digging out the pieces of potential evidences of

54 This chapter is an enhanced form of the peer-reviewed article by this researcher, published in the Journal of Digital Forensic Practice, USA, 2:4, pages 175 — 182 , (2009)
copyright infringement. This can be conveniently done along the parameters, shown and discussed below.

5.1 Elements of similarity and commonality in design and programming

Needless to say that the first task of the technical expert is to rigorously study the ‘original’ and the allegedly pirated source codes and systematically collate items that would form the basis for establishing elements of similarity and commonality. Similarity and commonality can be seen in a variety of design and programming entities / patterns / practices / traditions like the nomenclature tradition of the software company for naming various entities in the program, the logic of the program, the errors, and the comments.

5.1.1 Commonalities in Nomenclature

One of the important conventions followed by the companies in evolving their programs is for the programmers to use a pattern within the established nomenclature tradition of the company for naming the variables, database tables, fields, modules and procedures in the program. These commonalities are unavoidable because of the shared nature of the technology. Some of the conventions are as follows.

Most of the experienced programmers, for unique identification, include some amount of personal touch realized through a set of names of variables, database tables, fields, modules, procedures etc. that they often use. An analysis of these names may prove to be vital in confirming piracy as they are idiosyncratic. Digging out such vital elements from the ‘original’ software and establishing and confirming their duplication in the alleged software is the success of the technical expert. This is what most experts rely on to establish piracy (Spafford and Weeber, 1992). At the same time there are two major factors (see below as a & b) that many experts tend to discount as unreliable either because it is difficult to prove piracy conclusively by using them or because these elements are part of universally conventional commonalities involved. What is particularly stressed here is the fact that these two factors, when properly and intelligently
scrutinized by a determined expert, can still yield enough results at least to supplement confirmation of piracy, if not to unequivocally prove it.

a) It is universally accepted that the name of the variable or a database field should reflect the concept it bears. For example, let us take the case of banking software, a classic example of a Management Information System software. To store an Account Number in banking software, the programmers would opt for a variable with a name like ACC_NO, ACCOUNT_NO, ACCOUNT_NUMBER or ACCNT_NO, to name a few. Such preferences are universally accepted and widely used, and so, similarity in, say ACC_NO, in the ‘original’ and the ‘pirated’ software need not indicate piracy, let alone prove it. Nevertheless, despite all these commonalities, a discerning expert can still find a way for establishing possible piracy through a rigorous formulation of statistical occurrences of these variables. For instance, the report (Bhattathiripad, 2002) submitted to the court by the author, as the technical expert of a software piracy case, encountered such a situation and still was able to find material strongly favoring piracy. Table-5.1 below, which was extracted from that report (p.11), lists out several self-explanatory values, as percentages. These percentages motivated the author to suspect piracy (despite these fields being moderately universal names) and helped to add this finding to the list of supporting evidence. What is more important is the format of the table-5.1, which shows how rigorous statistical information can be presented in order to increase the reliability of comparison of two database tables. The format used in this table was devised by the author for this purpose as most judges would like to have the results presented in such a self-explanatory table structure. Such a format allows technical experts to present the results of investigation like how many fields in a ‘pirated’ database table bear exactly same names, data types, length, context, and sequence of appearance, as in the ‘original’. Interestingly, in certain cases, the names of the respective database tables may be different but the fields in them may show commonality, which also suggests piracy. In such cases also, the above format can be used.

b) Programmers usually name loop variables, temporary variables, index variables, database tables, fields, modules, procedures etc. based on standards set by the
employer software company. At the same time, within this frame work of company’s general directive, each project leader or the programmer can take the liberty of setting his/her own thumb impressions in the names. A company or the project leader can instruct that at least one part of the name of the variable should be spelt out completely, for instance, ACCOUNT_NO or ACC_NUMBER. The company or the project leader expects this to be their unique feature, identifiable as the company’s (or the project leader’s) thumb impression or finger print; but in reality, this need not be. As many companies and project leaders follow the same rule, the expert cannot consider such ‘thumb impressions’ to be uniquely and invariably attributable to any particular company or person. On the other hand, if, for example, a company, viz. SRP, instructs that any variable or procedure should bear name starting with SRP, then this identity (of the company) becomes the company’s thumb impression. The presence of such thumb impression in the original as well as the allegedly pirated source code suggests piracy.

<table>
<thead>
<tr>
<th>Similarity in the table name: Both tables bear the name ‘BANK’. So, 100% commonality.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length of the ‘original’ table: 98</td>
</tr>
<tr>
<td>Length of the ‘pirated’ table: 132</td>
</tr>
<tr>
<td>Field count of the ‘original’: 12</td>
</tr>
<tr>
<td>Field count of the ‘pirated’: 13</td>
</tr>
<tr>
<td>Commonality in the names of fields: 11 out of 12 names of fields in the ‘original’ are found in ‘pirated’ also. So, 91% commonality</td>
</tr>
<tr>
<td>Commonality in the data type of common fields: All the 11 common fields have the common data types too. So, 100% commonality</td>
</tr>
<tr>
<td>Commonality in the length of common fields: 9 out of 11 common fields have same names, data types and length. So, 81% commonality</td>
</tr>
<tr>
<td>Commonality in the default values set in the common fields: Among the 9 common fields only 2 were found bearing default values in the ‘original’ and the exactly same 2 fields in the ‘pirated’ also bear the same default values. So, 100% commonality.</td>
</tr>
<tr>
<td>Commonality in sequence: All the 11 common fields do occur in the same sequence.</td>
</tr>
<tr>
<td><strong>Inference:</strong> There is 81% similarity as 9 out of 11 fields bear exactly same names, data types, length, context, and sequence of appearance in this table.</td>
</tr>
</tbody>
</table>

Table -5.1: Result of comparison of two database tables (Post-piracy modifications are not considered) (Bhattathiripad, 2002)
The most important objective of the judge is to establish piracy. This he does by studying the opinion and the testimony of the experts, and making judgment on its credibility. This makes the expert’s testimony and thus his / her expertise crucial. Needless to say the more the expert’s recommendations are seen to be based on objective factors, the more its credibility and reliability will be. There have been instances where the expert’s testimony was successfully challenged in the court. The report (Bhattathiripad, 2002), especially the Annexure 7 of it, is a classic example of such a challenge, where incomplete comparison amounted to unreliability of evidence. During a subsequent cross examination of this suit in the court of law, the technical expert had to accept that there was still scope for further in-depth analysis. Obviously, events like this might cause inconveniences to the jury and delay the final judgment. (The final judgment of the above-referred suit is still pending in the court.) Hence, the first report of the technical expert itself should wherever possible be thorough, convincing and binding and it is the duty of the technical expert to avoid creating such inconveniences to the judiciary, with the possible consequence of an unintended delay or miscarriage of justice.

All the above suggest that all the database tables in the ‘original’ and the ‘pirated’ are to be compared and the inferences arrived at are to be presented in the format given in table-5.1, in order to ascertain nomenclature-level piracy. These percentages (as part of table-5.1) would help the judge to arrive more easily at a judgment. In other words, the judge would greatly appreciate tangible evidence like such figures and conclusions logically arrived at by the technical experts.

The same technique (the format devised by the author in table-5.1) used for database tables can be adopted for comparing other variables too in the ‘original’ and the ‘pirated’ source codes.

5.1.2 Commonalities in Logic Sequence

Any program poses the following logic related issues, which need to be addressed while analyzing and comparing the logic of the ‘original’ and the ‘pirated’, to ascertain piracy.
i. The programmer is forced to follow, strictly, the base logic demands of a problem specification.

ii. The programmer is forced to follow, strictly, the notations, conventions, coding styles, and the universal standards of the programming language, and, above all, the programming ethics.

iii. The programmer is forced to follow the logic specifications or standards set by the organization. For example, many firms specify that all database access codes should be consigned to the database script. These are thumb impressions of the employer, which need not always be unique as several firms follow the same rule. However, they can be helpful in suggesting piracy.

iv. The programmer should look out for the standards or specifications for the logic defined by the project leader, i.e. the thumb impressions of the project leader, which also need not be unique; but the presence of the same thumb impression in the original as well as the allegedly pirated source code is a clue of piracy.

While analyzing the two source codes to confirm piracy, any commonality pertaining to the above four areas may suggest possible piracy, but it may not be sufficient to establish it legally. Interestingly, the major part (approx. 90%) of the commonalities of the logic sequence would be in the above four areas (see 1 to 4, above). The expert needs to know precisely what features are expected to be invariant and what features are permitted to vary, but need not necessarily have to vary, and how generally the invariant and variant elements in the two fit together.

Despite its suspect nature, the above logic-related issues can still offer some clues that sometimes might help confirm piracy. Certain degree of uniqueness of logic sequence can become visible under tight scrutiny. For instance, the report (Bhattathiripad, 2002) submitted by the author, as the technical expert of a software piracy case, has several such instances, which illustrate how to dig out such material evidence and arrive at the statistics that strongly favor piracy. The table -5.2, below, which was extracted from that report (p.13), lists out several self-explanatory values as percentages, which forced the author to add this finding to the list of supporting evidence for suspected piracy. As mentioned above about table-5.1, what is more important is the format of the table-5.2,
which shows how rigorous statistical information can be presented in order to increase
the reliability of comparison of two procedures. The format used in this table was devised
by the author for this purpose with the intention of presenting the results of comparison.
Such a format allows technical experts to present the results of investigation like how
many parameters in a ‘pirated’ procedures bear exactly the same names, data types,
length, context, and sequence of appearance as in the ‘original’.

Discovering, identifying and confirming those suspect code segments from numerous
lines of source code is admittedly a difficult task, but achievable. If that is done, it would
result in significant means of providing evidence to confirm piracy. These elements have
to be invariably and invariantly associated and aligned in terms of their position,
sequencing and context in both the ‘original’ and the ‘pirated’ source codes so as to
provide evidence of piracy. This is where the real expertise of the technical expert really
counts. In addition, complexity of analyzing a program increases when the program is
written by one programmer and updated at various instances later, by others
programmers. In any case, as far as program logic is concerned, most programs are very
likely to have the thumb impressions of many programmers due to the frequent turnover
of employees in the software industry. These multiple thumb impressions, compatibly
identified in both the ‘original’ and the ‘pirated’ sources, can prove to be vital.
**Procedure comparison number 2:** The reason for comparing these two procedures is that both are doing the same job in the respective software packages (‘original’ as well as ‘pirated’). Moreover, no other procedure in the respective software package was seen doing this job. Thus, though there is only near 100% similarity (and not 100% similarity) in the names, it can be ascertained that they were made for executing the same process.

Name of the procedure in the ‘original’ software: DBPROCINSERTACCTTRANS.

Name of the procedure in the ‘pirated’ software: INSERTACCTTRANS_PROC

**Results of comparison of procedure number 2:**

Similarity in the name of the procedure has already been discussed, above.

All the 13 parameters in the ‘original’ are found exactly named and declared (except length, in case of NUMERIC definition) in the same manner in the ‘pirated’ also. The parameters, namely, HEAD CHAR(8), FINYEAR CHAR(4), VOUCHER CHAR(8), DEBIT NUMERIC(15,2), CREDIT NUMERIC(15,2), CHEQUE CHAR(10), XTYPE CHAR(2), DESCRIPTION CHAR(300), USERNAME CHAR(6), MACHINEID CHAR(10), ACCTRANSDATE DATE, ACCTRANSBILLNUMBER CHAR(11) and INTERNALENTRY CHAR(10) occur in both ‘original’ and ‘pirated’ in the same position and in the same sequence. Considering the variety in styles of nomenclature and thinking process of programmers, this is extra ordinary. The ‘pirated’ has 2 more additional parameters namely COSTCENTRE CHAR(2) and DIVISION CHAR(2), which may have added after the ‘piracy’ has occurred, as their positions come after the 13 parameters, given above. Further analysis of these 13 parameters have the produced the following statistics, in percentage:

- Similarity in names and data types: 100%.
- Similarity in names, data types & length: 84% (11 out of 13)
- Similarity in position of appearance: 100%
- Similarity in the sequence of appearance: 100%

There are exactly 3 sql statements present in these two procedures in the ‘original’ as well as the ‘pirated’. All of them look exactly similar in styles, positioning and sequence of appearance of codes (except the inclusion of the 2 additional parameters in the ‘pirated’). The ‘pirated’ neither has anything extra nor has anything lacking. So, the 3 sql statements in these two procedures are 100% similar.

**Inference:** Piracy is suspected.

Table -5.2: Result of comparison of two data base procedures (Post-piracy modifications are not considered) (Bhattathiripad, 2002)
5.1.3 Similarities in errors

An important evidence of piracy is the existence of same programming errors and blunders in the original as well as allegedly pirated source codes. There is a distinction to be made between errors and blunders.

A programming error found in well tested and implemented software can be a variable or a code segment or a field in a database table, which causes or produces wrong result during the execution of the program.

There is a specific type of error which is perhaps a greater indicator of piracy, which the author would like to call a blunder. A programming blunder found in well tested and implemented software can be a variable or a code segment or a field in a database table, which is hardly used or executed in the context of the application or the user’s functionality but unlike an error, it will be harmless. For example, if a variable with a universally uncommon name PXRN_CODE_AUCQ CHAR[6] is found in identical places of the identical procedures in the ‘original’ as well as the ‘pirated’ software, but not used anywhere else, that is an instance of blunder. Since it is only properly declared but not used anywhere in the program, it may not cause an erroneous result during execution. A technical expert can detect this. The occurrence of such blunders in both programs in identical contexts would be a serious indication of piracy. It is highly unlikely that two programmers will blunder or err exactly in the same way (See Chapter 6 for more details on programming blunders).

While performing the quality control of the source, the tester is normally expected to remove blunders and errors. However, harmless blunders and errors may remain unnoticed by the tester. Sometimes, the pirate may find and remove them. But, any slip-up on the part of the pirate that leaves the blunders and errors intact as in the ‘original’ is a strong evidence of software piracy. However, spotting such blunders and errors in the ‘original’ code is a challenging task.
5.1.4 Similarities in Comments

A comment or a remark is a non-programming, blocked, statement, done by the programmer in order to increase the readability of the program. As programmers may use the same set of indexical comments in their own style for further explaining program statements found in different parts of the source codes, they are considered as a programmer’s thumb impressions. Piracy can be strongly suspected from the presence of such identical remarks in the ‘original’ as well as the allegedly pirated source codes. The author was surprised to see such instances (p.16), which are listed in the table-5.3 below (Bhattathiripad, 2002). The four comments in the ‘original’ source code appear in the ‘pirated’ as well, with high degree of congruence in wordings, grammar, sentence structure, context and the sequence of appearance. Though the wordings and grammar may be quite universal, it is the similarity in the context and the sequence of appearing which qualify them for almost certain indicators of piracy.

The ‘original’ source code has the following four comments which are appearing in the ‘pirated’ as well, in the same context and sequence.

/* IF THE DATE BEING CLOSED IS TODAY, CHECK IF THE TIME BETWEEN LAST CLOSING AND NOW IS MORE THAN THE MINIMUM TIME */

/ * * ******** IT IS YEAR ENDING OF THDAY IS THE FINENDYEAR DATE IN FASETINGS ******** */

/* Get the new financial year */

/* TO GET END YEAR, GET THE DATE AFTER ONE YEAR. IF LEAP YEAR, ADD 365 ELSE ADD 364 */

Table-5.3: Result of comparison of comments / remarks (Bhattathiripad, 2002)
5.2 AFC and the elements of similarities and commonalities

As explained at the beginning of this chapter, none of the above interesting points (of design patterns and programming patterns in general software piracy forensics) has not been given due respect in AFC (and also in other software piracy forensics techniques). While this presumably comes under some of the “doctrine” exceptions in AFC, the elements that are sensitive to this are often filtered out in the filtration step of AFC, resulting in loosing important valuable evidence. This is a deficiency of AFC. Those elements whose design and programming aspects can in some way provide supporting evidence to establish piracy should not be filtered out. In order to make the filtration process sensitive to design and programming pattern considerations, the original AFC list of items to be filtered out needs to be modified. Chapter 7 discusses the necessary modifications and restructuring needed in the list of filtration. Moreover, chapter 7 also discusses the way the original and the pirated need to be compared for enlisting all actual similarities / commonalities from the design and programming considerations (for enlisting suspected piracy of thumb impressions, programming errors, programming blunders etc. in the pirated).