Testing for Copyright Protection and Infringement in Non-Literal Elements of Computer Programs

Arjun Krishnan†
National Law School of India University, Nagarbhari, Bangalore 560 072

Received 19 July 2004

Computer software is one field in which India enjoys a competitive edge over other countries due to a fast growing industry and skilled manpower. One of the forms of protection of intellectual property in software is copyright protection. It has long been recognized that non-literal infringement of a work is also possible. Despite the importance of the software industry, no test has been enunciated in relation to non-literal infringement of computer programs. This article examines US and UK case law relating to non-literal infringement of computer software and the different tests laid down by the courts in these countries. Divergence of judicial opinion as to what is the most appropriate test for non-literal infringement has raised several doubts relating to some of the fundamental tenets of copyright jurisprudence. This article examines applicability of these tests to the Indian context in order to locate an appropriate test for India, in the light of Indian copyright jurisprudence.

Keywords: Copyright, computer software programs, non-literal infringement, idea – expression dichotomy, abstraction.

Copyright law has often been praised for its ability to adapt and respond to technological change. This ability to meet the challenges of modern technology is perhaps nowhere better exhibited than in the field of copyright over computer programs. Development in this regard has occurred both through legislative initiatives and judicial intervention.

This paper deals with a specific aspect of copyright in computer software programs – that of non-literal infringement and the tests that have been propounded in different jurisdictions for adjudicating instances of the same.

Historical background of computer program copyright in the three jurisdictions, the United States, the United Kingdom, and India, is examined first.

In the United States, the subject matter of copyright is codified in § 17 USC 102(a). An illustrative listing of the subject matter of copyright was given under the 1976 copyright legislation, but computer programs were not recognized as such. However, there was broad agreement that computer programs were eligible for copyright protection as ‘literary works’, and this is also borne out by the legislative history of the law.† Prior to the 1976 enactment, the United States Congress set up the National Commission on New Technological Uses (CONTU) of Copyrighted Works‡, to assess the adequacy of copyright law in relation to modern technological advances, in particular, photocopying and computer technology. The CONTU report stated that computer programs deserved protection, though it did not recommend that any significant changes would be required to accord protection, i.e., the pre-existing literary works framework was sufficient to protect computer programs. However, the CONTU report did recommend that the protection accorded to computer programs ought to be made explicit by the addition of the definition of ‘computer program’§.

In the United Kingdom, the 1956 Copyright legislation did not specifically mention computer programs, though protection could certainly be extended to them as literary works. However, computer industry concerns led to express recognition for computer programs as literary works in the Computer Software (Amendment) Act, 1985 and eventually in the Copyright, Designs and Patents Act, 1988 as well.¶

In India, the Copyright Act has specifically included computer programs within the ambit of a literary work.¶¶

Computer programs are susceptible to copyright infringement. Modern technology facilitates not only piracy of computer programs, but also easy copying of entire program code once it is revealed.¶¶ The
modes of infringement can be characterized as ‘literal infringement’. However, non-literal elements of computer software may also be infringed. Infringement of this type does not involve the copying of a program’s code or the language in which the program is written, and hence will serve for the user. This stage is often called the requirements definition stage. The second stage is the determination of the functions that are required to be performed by the program in order to satisfy the requirements definition.\textsuperscript{12}

Thirdly, the programmer would design the overall structure of the program by breaking down each of the functional requirements into subtasks\textsuperscript{13}. Each subroutine is further broken down into further subroutines until the programmer is left with several subroutines that individually perform relatively simple tasks and which can be written relatively easily. Once this is done, various sub-routines are arranged in an organized manner – usually using a flow chart.\textsuperscript{14}

Thereafter, the programmer must implement the design of the program by actually writing out the individual lines of code for the various sub routines.\textsuperscript{15} In doing this, certain sets of rules for the accomplishment of a given logical process, also known as ‘algorithms’ are used.\textsuperscript{16} Finally, possible errors in the program are detected and rectified in a continuous and ongoing process known as ‘debugging’.\textsuperscript{17}

Four Tests for Non-Literal Element Infringement in Computer Programs

In cases of non-literal infringement of copyright protections, the issues that arise have been characterized in the form of two kinds of questions\textsuperscript{18}.

(i) The \textit{broader question} of whether copyright protection should at all be accorded to the non-literal elements of computer programs.

(ii) The \textit{narrower question} of what constitutes, or what is the proper test for infringement of the non-literal elements of a software program.

On the issue of whether copyright protection should be extended to non-literal elements, courts in the United States have consistently held that the structure, sequence and organization, being the non-literal elements of a computer program are protected and capable of being infringed.\textsuperscript{19} The fundamental basis for extending copyright protection to non-literal elements in computer programs has the same rationale as for other works such as plays, books and movies. This rationale is founded in the belief that the rights of the copyright holder cannot be allowed to be infringed by minor variations in his work that would result from the restriction of copyright to strictly the ‘text’ of the work.\textsuperscript{20} In \textit{Computer Associates v Altair}\textsuperscript{21}, perhaps the leading computer software non-literal infringement decision, the court placed heavy reliance

The Hierarchical Structure of Computer Programs

The importance of an understanding of the structure of computer programs has been stressed repeatedly by the courts.\textsuperscript{8} Modern computer programs are highly complex systems. In order to make the task of the creation of such a complex system, the programmer breaks down the program into various parts. This is done because it is virtually impossible to understand a program in terms of individual lines of programing language or ‘code’.\textsuperscript{9} The process of breaking down a program into various sub-tasks\textsuperscript{10} involves a descent by levels of hierarchy. This descent is a multi step process that begins with the programmer deciding on what purpose the software will serve for the user. This stage is often called the requirements definition stage.\textsuperscript{11}
on cases where there were findings of the fundamental essence or structure of one work being duplicated in another.\textsuperscript{22}

However, the arguments against extending copyright protection to non-literal elements of computer programs are also weighty, albeit now academic. Broadly speaking, supporters of this viewpoint suggest that the structure of a program closely resembles an idea, and therefore, should be outside the pale of copyright.\textsuperscript{23} Other commentators also argue that such protection is overprotective and economically inefficient (and therefore undesirable)\textsuperscript{24} and not suitable in the background of industry standardization\textsuperscript{25}.

In any event, it seems unlikely that the Courts in the United States at least will seriously debate whether non-literal elements of software programs can be the subject matter of copyright. While there has been a consensus as far as the broader question of copyrightability of non-literal elements is concerned, there has been less unanimity as to what constitutes substantial taking and infringement in this context, and what part of the non-literal elements can be protected by copyright.

\textbf{Whelan Associates v Jaslow Dental Laboratory}

The Third Circuit Court of Appeals became the first court to seriously consider the question of non-literal infringement in the United States in \textit{Whelan Associates v Jaslow Dental Laboratory Inc}\textsuperscript{26}. The dispute concerned two competing programs for the management of dental laboratories called Dentalab (created by Whelan and marketed by Jaslow Inc) and Dentcom (created and marketed by Whelan). It was shown that the latter program was written in a different language and for a different computer system. The main question to be answered in appeal was whether copyright could subsist in non-literal elements\textsuperscript{27} and whether there was infringement in the case. The court proceeded to develop a test for what parts of the non-literal elements could be protected, based largely on the idea-expression dichotomy. Briefly, this principle states that ideas are not the subject of copyright, only their expression is. The court relied on \textit{Baker v Selden}\textsuperscript{28}, a leading case in the United States in this regard. The US Supreme Court had held that what part of the work as idea and what expression depended on the nature of the work. In a utilitarian work, the function or purpose of the work would be the idea, and everything not necessary to the purpose or function would be within the realm of protectable expression. The \textit{Whelan} court followed the dicta in \textit{Baker v Selden}, and formulated a test on this basis. The \textit{Whelan} court held that in a computer program, the function or purpose of the program would be the idea, and everything else was part of expression, and therefore copyrightable. According to the court, conclusion was that the concept of having a program for managing the Dental Lab would be the idea, and therefore beyond the scope of copyright. The structure of the program, however, would be protected expression.\textsuperscript{29}

However, the \textit{Whelan} decision has been severely criticised and rarely followed. The primary criticism levelled against the \textit{Whelan} formulation that it is overbroad in its protection of program structure, and is based on the false premise that only one idea underlies a given program. Though there is little doubt that the purpose or function of a program is an idea, this does not mean that other elements in the structure and design of the program are necessarily in the realm of expression.\textsuperscript{30} For instance, each module may perform a separate function, and each function may be conceptualized as a separate idea.\textsuperscript{31}

\textbf{Brown Bag v Symantec}

The Ninth Circuit Court of Appeals in \textit{Brown Bag Software v Symantec Corp}\textsuperscript{32} essentially applied the general test of copyrightability to the computer program context.\textsuperscript{33} The dispute in this case concerned an ‘outlining’ program that was created by an independent programmer for Brown Bag, who later produced and sold a similar program to Symantec. The court evolved a two-part test for determining substantial similarity.

\textit{First}, the court dissected the program in question to determine which parts were unprotected. This phase is known as the ‘extrinsic test’ and is objective, allowing for the adducing of expert testimony to aid the court in the dissection process. This test defines the scope of the plaintiff’s copyright.

The \textit{second} part of the test is known as the ‘intrinsic test’. It comes into play only where the court finds objective similarities in the two programs being compared after the extrinsic test.\textsuperscript{34} This part of the test involves the comparison of the objectively similar parts of the program to find “substantial similarity in expression… depending on the response of an ordinary reasonable person”.\textsuperscript{35} Whether there has been substantial similarity, is to be decided on the ‘total concept and feel’ of the two works.\textsuperscript{36} The second part of the test is therefore subjective in nature.
The extrinsic-intrinsic test has been criticised – particularly the second intrinsic limb, on the grounds that it is too vague and amorphous. Little guidance is given by the court as to how the potentially difficult ‘analytical dissection’ of the program in the extrinsic test is to be done. The oversimplified nature of the test, which is based more on the judge’s instinct rather than analysis has also been criticised as unsuitable for complex works such as computer programs.

**Lotus v Paperback Software**

The main issue before the District Court of Massachusetts in *Lotus Devp Corp v Paperback Software* when it formulated its three part test to determine copyrightability of non-literal elements was whether the user interface of the popular spreadsheet program Lotus 1-2-3 was protected by copyright. The district court held that there could be copyright in non-literal elements of a program and propounded a test consisting of three steps, based on the idea-expression dichotomy in copyright law.

The First step involves the making of the distinction between idea and expression within the program by the court, along a scale moving from the most general to the most particular conception. This step is essentially an adaptation of Hand’s Abstraction test formulated in *Nichols v Universal Pictures*. This essentially involves separating the various elements of a program, moving from the general to the particular, for the purpose of distinguishing the idea (the general) from the expression (the particular).

The second step involves focussing on whether the alleged expression of the idea is limited to the elements essential to the expression of the idea or also includes identifiable elements that are not essential to every expression of the idea. This step incorporates the doctrines of merger and *scenes a faire* which are traditional exceptions to copyright.

In the third step, having identified the elements of expression not essential to every expression of the idea, the decision maker determines whether these elements are a substantial part of the allegedly copyrightable work. The test laid down in paperback was further refined and developed in the Second Circuit’s decision in *Computer Associates v Altai*, which is examined in detail in the following section.

**Computer Associates v Altai**

In *Computer Associates v Altai* the Second Circuit was called upon to determine whether there was non-literal infringement of Computer Associates’ operating system compatibility component program ‘Adapter’ by Altai Inc’s Oscar 3.5 program which performed the same function. Developing on the same lines as the *Paperback* test, the Court evolved a three stage test, commonly known as the ‘Abstraction – Filtration – Comparison’ (AFC) Test.

In the first Abstraction stage, Learned Hand’s Abstraction Test is applied to the allegedly infringed computer program to separate the various elements in order of increasing generality – moving from object code to the source code, the parameter lists, the services required, to the general outline of the program – in ‘a manner resembling reverse engineering’.

The second Filtration stage involves examining the structural components of the program at each level of abstraction, so as to filter out elements that would not be protected by copyright. Elements that are unprotectable would be of two categories – ideas (not protected, as per conventional copyright jurisprudence) and certain kinds of expressions which do not merit copyright protection, due to the operation of some copyright law doctrines, in particular, *scenes a faire* and merger. Ideas would include those elements that are at the more general levels of abstraction. That part of the expression that is dictated by concerns of efficiency is to be excluded under the doctrine of merger. This essentially means that if a particular way of doing something in a program meets the user’s needs in the most efficient manner, then it is not the subject matter of protection.

The doctrine of *scenes a faire* also stands in the way of allowing these non-literal elements from being protected by copyright in that there are certain requirements external to the work (program) which should be excluded from the scope of protection. These external factors that circumscribe the programmer’s freedom of design choice:

- Mechanical specifications of the computer on which the program is to be used (hardware standards);
- Compatibility requirements with other applications and operating systems (software standards);
- Computer manufacturer’s design standards;
- Target industry practices, i.e., business practices and technical requirements of the end-user;
- Computer industry programming standards;

Also excluded from protection are ‘expressions’ that are already part of the public domain, since these
are not original and therefore not the subject matter of copyright.  
In the last Comparison stage, once the court has sifted out the elements of the allegedly infringed program which are non-protectable through the filtration process, it is left with a kernel of protected expression. At this stage, the court makes an inquiry into whether the defendant has copied any aspect of the protected expression, and also the relative importance of the copied portion’s relative importance with respect to the overall program.

The Altai case holds the predominant position in its field and has been widely praised followed, and approved.

There are two broad criticisms of Altai’s AFC test. The first of these is that the ‘efficiency demands of programmers’ – i.e. programmers would not infringe if they copy certain efficient elements in another’s program – and the demands of external factors are not valid defences to copyright and have no basis in statute. This argument can be countered at two levels. Firstly, these aspects of the filtration stage have their basis in the doctrines of merger and scenes a faire, which is a well established exception to protection of expression in copyright law, at least in the United States. Secondly, copying of certain non-literal portions (those dictated by efficiency and external factors) ought not to be condemned because the expediency in the software industry demands this.

The other major criticism of the Altai test is that it is under-protective of program structure. Some commentators have observed that the protection afforded to non-literal elements in software does not resemble copyright protection at all, rather some new sui generis form of copyright protection for software. It has been said that the dissection of the computer program and elimination of unprotected programs before comparison may miss or eliminate protectable elements, particularly, the manner in which the various elements are organized and combined as a whole. However, the Altai analysis has the advantage of being more organized, detailed and comprehensive than the other tests for determining the idea-expression divide in non-literal elements of software. The test is also not unduly complicated in that it involves breaking down the enormous task of determining which non-literal elements should be protected, into smaller tasks for the application of known copyright principles.

In Softel v Dragon Medical & Scientific Communications Inc, the Second Circuit also accepted the possibility of adopting a ‘compilations’ approach in holding that the combination of various elements individually unprotectable (after applying the ‘Filtration’ test of Altai) could accorded protection. The addition of such a ‘compilations’ limb to the Altai test has also proposed as an effective answer to the possibility of the test being under-protective. In any event, commentators also feel that erring in favour of under-protection may be preferable in light of the fact that computer programs involve only a limited range of potential expressions, and computer program structure should be given less protection than structure in other literary works.

Non-Literal Elements Infringement in the United Kingdom
The discussion now proceeds across the Atlantic to UK, where courts have begun to grapple with the issues of non-literal elements protection. Two cases will be discussed briefly that embody different approaches to the problem, and indeed copyright law itself. In John Richardson Computers v Flanders, the court followed an approach resembling the Altai ‘AFC’ test, with some modifications. The case accepted that the approach adopted in the United States (in the Altai case) could have application in the UK as well. However, in Ibcos Computers v Barclays Mercantile Finance, the court held that the American case law could have no application in UK, because the basis of separating which non-literal elements were protected – the idea-expression dichotomy – had little relevance in English copyright law.

In the UK as in the United States, the broader question of copyright in non-literal elements of software has been answered in the affirmative by courts. The discussion that follows restricts itself to those aspects of the cases dealing with the narrower question the test to be applied for non-literal infringement of computer programs.

John Richardson v Flanders
The case revolved around competing programs for managing labelling and stock control in pharmacies. The basic contention of the plaintiffs was that the defendants had copied the ‘look and feel’ of their program in the non-literal sense.

In determining the proper method of determining the scope of copyright protection to be granted, Ferris J turned to the American case law already discussed
above, in particular *Computer Associates v Altai*. The court broadly agreed with the need to separate the idea from the expression in the non-literal parts of the computer program. However, the court diverged from the *Altai* test in the application of the ‘Abstraction’ stage, in its analysis. Instead of dissecting the program into layers of generality and detail in order to test for copyright and infringement, the court inquired first into whether the plaintiff’s program as a whole is entitled to copyright. It then decided whether there was any similarity attributable to copying which was to be found in the defendant’s program amounted to the copying of a substantial part of the plaintiff’s program. In rejecting the abstraction approach, the court placed significant reliance on *Ladbroke (Football) Ltd v William Hill (Football) Ltd*. In the facts of this case, Ladbrokes admitted having copied some parts of Hill’s fixed-odds football betting coupons, but denied that Hill were entitled to copyright in those parts. They admitted that certain other parts of Hill’s coupons were protected by copyright, but they had not copied these parts. It was held by the House of Lords that the coupon had to be looked at as a single literary work and that, having regard to the skill and effort involved in working out certain parts of Hill’s coupon the coupon as a whole was an original compilation which was protected by copyright.

Thereafter, in determining whether there was substantial similarity, the court took into account the distinction between idea and expression, and applied parts of the filtration stage of the test proposed in *Altai*. The similarities between the programs were assessed and divided into four categories:

(i) Similarities which were by themselves the result of copying a substantial part of the plaintiff’s program;

(ii) Similarities which were the result of copying, but which did not by themselves involve the copying of substantial parts of the plaintiff’s program;

(iii) Similarities which may be the result of copying, although the judge was not satisfied as to this, but which, even if copied, did not by themselves involve the copying of substantial parts of the plaintiff’s program;

(iv) Similarities not the result of copying.

Thereafter, in the final stage of comparing to see whether there had been substantial copying, the test formulated by Ferris involved seeing whether the similarities in category (ii) would indicate copying of a substantial part of the plaintiff’s program to a greater extent than only the similarities in category (i).

An important reflection on this approach has been that had the *Altai* test been applied in full, the similarities in category (ii) would have been discounted automatically before evaluating substantial similarity, by the application of the filtration stage. Despite this ‘drawback’ the judgment has been commended for introducing in the UK ‘a limiting analysis’ in non-literal infringement cases.

**Ibco Computers v Barclays Mercantile Finance**

Although the facts of *Ibco Computers v Barclays Mercantile Finance* revolve mostly around a situation of literal infringement of copyright in programs, the philosophical approach and dicta in this case have a bearing on the issues at hand. At issue in this case was whether the plaintiff’s program suite was infringed by a similar compendium belonging to the defendant. The court held that not only was there copyright in the individual programs, but also in the compilation as a whole. In relation to the test for substantial similarity, the court commented on the position of the idea-expression dichotomy in English copyright law. Jacob J stated that the idea-expression dichotomy was not expressly recognized in UK as it had been in the United States. Indeed, it was not the expression that was protected but the ‘detailed idea’. However, Ferris J is not the first to question the idea-expression dichotomy itself.

In their seminal work, the *Modern Law of Copyright*, Laddie, Prescott and Vitoria too have seriously questioned the existence of the doctrine in English law. According to these commentators, copyright exists not in the expression, but in the more detailed collection of ideas. The authors support their idea through the provisions of the CDPA which state only that copyright subsists only in the original work, and infringement occurs when there is substantial taking of the same. ‘Ideas-patterns’ and concrete compilations original enough to constitute original works are protected.

Ferris used much of this logic in refusing to follow the American authorities cited before him, particularly *Altai*. The test for substantial infringement used by him was one of ‘over-borrowing’ of the skill labour and judgment that went into the work, and the idea of the ‘core of protective expression’, would according to him unduly complicate matters.
Idea/Expression – Dichotomy or Fallacy

The *Ibcos* case and its philosophical stance on the idea-expression dichotomy raise several concerns regarding the appropriate test for non-literal elements. The arguments of those who take this view are two-fold. The first is that it is difficult to accept the logic of the dichotomy in light of judicial pronouncements, and the second is that the doctrine has no basis in English law. Both these arguments are unfounded.

It is argued that the law itself does not follow the logic of the dichotomy to its full extent. For instance, it is argued that if the dichotomy was to be taken to its logical conclusion, the mere expression of the same book in slightly different language or for that matter the expression of a novel as a play using different dialogues, ought not to be infringement, when in fact it has been held to be so. However, such a view of the dichotomy takes too narrow a view of what is meant by ‘expression’. Expression is not restricted to form or medium. In the dichotomy, expression refers to the putting of the abstract idea in a more concrete form, which includes fixation of the idea in a medium.

Furthermore, it is argued that the separation of ideas and expressions provides no aid in differentiating between the “series of conceivable gradations ranging from the very abstract to the utterly concrete”, all of which may be found in a work. However, the commentators do not demonstrate how a dichotomy between general and detailed ideas can help draw this line any better than the idea-expression dichotomy, particularly in cases relating to computer program design and structure (non-literal elements). Indeed, it seems that there is, in practical terms no difference between differentiating between ideas and expressions and general ideas and detailed ideas.

With respect to the argument that English law does not recognize the dichotomy, it is submitted that in many cases English courts have expressly recognized the same. Even assuming, as some would have it, that this is mere ‘lip service’, the question seems quite redundant in light of many overarching treaties, particularly the WIPO Copyright Treaty, 1996, and the EU Software Directive, both of which expressly affirm the dichotomy.

The *Ibcos* judgement has also been criticised, primarily on the grounds that the test of ‘over-borrowing’ is too vague and over-protective, and does not provide any real guide in determining what the requirement of substantiality is. In *Cantor Fitzgerald v Tradition*, though the case and approach itself were approved, the measure of substantial similarity used was thought to be inappropriate as there was a risk of making an error when adapting well-known principles developed in the context of normal literary works and applying them uncritically to computer programs whose purpose is to make a machine operate in a specific manner. In essence, *Cantor Fitzgerald* seems to clarify that the test of substantiality ought to be qualitative and not qualitative. Pumfrey, J held that there could be no copyright in an idea, but the originality of the idea would have a bearing on what constituted infringement. In other words, the expression of a highly original idea will be more easily infringed than that of a less original idea.

Therefore, the balance of authorities in computer software cases in the UK seems to suggest that some element of copyright exists in ideas also. However, both *Cantor Fitzgerald* and *Ibcos* decisions need not to take into account the Article 1(2) of the EU Software directive. Article 1(2) expressly states that ideas and principles in computer programs can not be the subject of copyright protection. This certainly goes a long way in putting to rest the idea-expression controversy, and undermines the authority of cases like *Ibcos* and *Cantor Fitzgerald* that read some element of protection into ideas in computer programs.

Locating Appropriate Non-Literal Elements Test for India

So far, copyright in computer software has been largely untested by the courts in India. Though copyright in computer programs has been expressly recognized in the Indian Copyright Act, 1957, under the category of literary works, it remains to be seen how the issue of copyright in non-literal elements of computer software will be dealt with. In all probability, the Indian courts will follow the lead of US and UK courts in affirming the existence of copyright in these aspects. However, it would be more difficult to decide as to which of the approaches would be acceptable in the light of copyright precedents in India.

It now remains to be seen what kind of approaches have been taken in the Indian courts.

Almost all discussion relating to the law of copyright infringement must begin with the seminal case in this regard. In *R G Anand v Delux Films*, the
issue to be decided was whether the copyright in the plaintiff’s play had been violated by the defendant’s film. After a lengthy examination of the authorities cited before him, including several Indian authorities, the court stated that it was a well established principle of copyright law that there could be no copyright in ideas and only expressions could be protected. According to the court, the best test for copyright infringement, in such cases, was ‘to see if the reader, spectator or the viewer after having read or seen both the works is clearly of the opinion and gets an unmistakable impression that the subsequent work appears to be a copy of the original’. By stating this, the court seems to approve of the ‘look and feel’ approach. However, when deciding on the facts and circumstances of the case, more than one of the approaches outlined above seem to have been used. For instance, when dealing with the allegedly infringed play, Fazal Ali J, seems to employ Learned Hand’s Abstraction test from Nichols v Universal Pictures. The judge identified various levels of generality in the script of the play, the most general theme being that of provincialism, the next (less general) level being that of love between two persons coming from different backgrounds and so on.

However, at later stages in the judgment, where the court tried to determine whether substantial copying has occurred, the court compared the various similarities and dissimilarities in the two works by listing them out individually, and came to the conclusion that there was nothing to show that the similarities in the defendant’s work were the result of copying. The court held that the similarities arose out of the common themes underlying both works. This resembled the approach of Ferris J in John Richardson Computers.

The most common instances of non-literal infringement are to be found in situations where one kind of copyright – for instance a novel – resembles another kind, such as a dramatic performance on a similar theme. One recent example of this kind of a situation is Barbara Taylor Bradford v Sahara Media Limited. In this case, the plaintiff alleged the infringement in the copyright of her novel by the defendant’s television soap opera. The court affirmed the idea-expression dichotomy and proceeded to apply it to the facts of the case. As in R G Anand, the court applied a combination of the Abstraction and ‘look and feel’ tests. The court held that although both works shared the same theme. The court held that although the two works had a common theme – that of a woman dealing with adversity, there was sufficient dissimilarities in the details of the plot and characters so as to show that there was no infringement.

However in most other cases, courts have relied squarely on the dicta in the R G Anand case and examined whether an ordinary spectator would think that the two works are similar. The quality and substantiality of the infringements are used as the touchstones for examining whether the works are substantially similar. In Raja Pocket Books v Radha Pocket Books, the Delhi High Court held that an ordinary reader would feel that the defendant’s comic book was substantially similar to that of the plaintiff.

In Anil Gupta v Kunal Dasgupta, the facts revolved around the alleged copying of a script for a reality-marriage show in which prospective brides and grooms were matched on television. The case is important because it hints at the possibility of their being copyright in ideas in much the same vein as Ibcos Computers. However, the court also admits that it is a well-established principle of copyright law that there is no copyright in ideas but there is copyright in their original expression.

On an overview of the case law it seems clear that much of the controversy surrounding the idea-expression dichotomy that had arisen in UK law before Article 1(2) of the EU Software directive will probably be avoided in India. The Doctrine of the idea-expression dichotomy has gone largely unchallenged in India. In all probability, the phrase ‘concept fledged out in adequate detail’ mentioned in Anil Gupta v Kunal Dasgupta probably refers to the expression of an idea. In any case, as has already been pointed out, such a distinction between protected ‘detailed ideas’ and protected ‘expression’ should in practical terms make little difference. However, there does not seem to have been any Indian case where the Abstraction test has been expressly followed or applied. This certainly casts doubt on whether the AFC test of Altai has a firm footing as far as Indian copyright precedents is concerned. This does not, however, restrict the application of something like the test used in John Richardson or Brown Bag v Symantec being applied to the Indian scenario. The ‘look and feel’ approach also seems to be the predominant test for copyright infringement in India. The basic aim of such an approach is to look
at the work as a whole, and see if it has been infringed or infringes.

The protection accorded to compilations in copyright law, it is submitted, can be used equally well to protect the computer program as a whole. Such a ‘compilations prong’ would also go a long way in ensuring that the Altai test is not under-protective.

It is also important to look at the issue normatively, i.e. which is the best test per se that should be applied in India, regardless of what Indian courts have held previously in cases that have not dealt with software copyright issues. That software copyright deserves at least slightly differential treatment because of its essentially functional qualities is evident. The Abstraction test seems to be particularly suited to the field of computer software. Furthermore, because the nature of the work in question is highly technical, it is not possible to use the test of an ordinary spectator or viewer. Perhaps there is a need to use something like the requirement in patent law of a ‘person with ordinary skill in the art’ in the field of software copyright.

Conclusion

The highly functional nature of computer software poses a variety of problems to copyright law. This is particularly evident in the case of according protection for copyright protection for non-literal elements i.e. computer program design and structure. There seems to be little doubt that non-literal elements should be given protection, but problems arise as to what portions should be accorded protection (separating ‘idea’ and ‘expression’ in the computer program), and how much protection is appropriate (i.e., what constitutes infringement). The test propounded by the US 2nd Circuit Court of Appeals in Computer Associates v Altai Inc seems to have gained widespread approval. However, the first, and perhaps the most important stage, that of Abstraction, has not been applied widely in India.

In the courts of the United Kingdom some fundamental doubts had been raised about the existence of the idea-expression dichotomy, in which the Abstraction test has its roots. This problem will not be faced in India, as there is widespread use and approval of the idea-expression dichotomy by the courts. Given this, there can be little objection to the importation of the Altai test in India. It has the advantage of being more organized, detailed and comprehensive than the other tests for delineating the protectable elements in a work of computer programing. It remains to be seen whether Indian courts will continue to apply the widespread ‘look and feel’ doctrine in the sphere of non-literal infringement of computer software, or will switch over to the more complex, but better suited three-stage Altai test.

Acknowledgement

Special thanks to Mr Dev S Gangjee, Oxford University, and Professor T Ramakrishna, National Law School of India University, Bangalore, for their assistance and guidance.

References

1 17 USC § 101 defines literary works to include works “expressed in… numbers, or other… numerical symbols or indicia, regardless of the nature of material objects, such as… disks, … in which they are embodied.” The Report of the House of Representatives stated that the category of ‘literary works’ included “computer programs to the extent that they incorporate… the programmer’s expression of original ideas.” See H R Report No. 1476 (1976) at 54, as cited in Julian Velasco, The copyrightability of nonliteral elements of computer programs, in 94 Colommbia Law Review, 242, 1994, 250 (hereafter “VELASCO”)

2 Popularly known as ‘CONTU’

3 17 USC § 101 defines computer program as a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result


5 Section 2(o), See also Section, 2(fb), (f(f), which define computer and computer program respectively

6 Karjala Dennis S, Recent United States and international developments in software protection - I, European Intellectual Property Review, 13, 1994, 14

7 “It is of course essential to any protection of literary property, whether at common-law or under the statute, that the right cannot be limited literally to the text, else a plagiarist would escape by immaterial variations” per Learned Hand J in Nichols v Universal Pictures, 45 F. 2d 119, at 121. See also Rees v Melville, [1911-1916] MacCC 168, Corelli v Grey, [1913] TLR 570

8 So much so that almost all cases, particularly, those dealing with non-literal copyright infringement discuss at length the manner in which computer programming takes place and the various components of a computer program

9 Laddie Prescott & Vitoria, supra note 4, 1604

10 These sub tasks are known as ‘modules’ in programing jargon

11 Velasco, supra note 1, 245. The requirements stage determines the use that the program can be applied to by the user, for instance, word processing (eg. Microsoft Word), spreadsheet use (eg. Microsoft Excel), Graphics and Media Applications Tool (eg. Adobe Photoshop), Video Player (eg. Quicktime), etc
12 Id. This stage is known as the ‘functional specification’
13 stage. To illustrate, a user of a word processing program
14 such as Microsoft Word may want the program to
15 automatically correct spelling and grammar, perform
16 formatting functions, interface with printing hardware,
17 create tables and charts as and when required, etc
18 Each function is termed as a 'sub-routine' which is
19 essentially a smaller program within the larger program
20 Velasco, supra note 1, 246
21 This process of writing code has two elements – the
22 programmer writes out code in high level programming
23 language, which resembles highly simplified human
24 languages such as English, but is specialised for the use of
25 programmers with highly rigid rules of syntax. This aspect
26 of the program is also known as source code, and is
27 converted into a machine readable binary format through
28 compilers that perform this function. The source code in
29 machine readable format which can be processed by a
30 computer is also known as object code
31
32 Laddie Prescott & Vitoria, supra note 4, 1607
33 Ibid, 1606-1607
34 Stanley Lai, The law of copyright in computer software in
35 the United Kingdom (2000), 26-7 (hereafter "Lai")
36 Three major decisions taking this view have been Whelan
37 Associates v Jaslow Dental Laboratory, 797 F. 2d 1222;
38 Lotus Deep Corp v Paperback Software, 740 F. Supp 37;
39 and Computer Associates Intl v Altai, 982 F. 2d 693
40 See Nichols v Universal Pictures, 45 F. 2d 119, at 121
41 982 F. 2d 693 (2nd Circ., 1992)
42 For instance Horgan v Macmillan, 789 F.2d 157
43 (recognizing that a book of photographs might infringe
copyright in ballet choreography); Twentieth Century-Fox v MCA Inc,
44 715 F.2d 1327 (motion picture and television series); Sid &
45 Marty Krofft Television Prods Inc v McDonald's Corp, 562
46 F.2d 1157 (television commercial and television series);
47 Sheldon v Metro-Goldwyn Pictures Corp, 81 F.2d 49 (play and
48 motion picture); Stewart v Abend, 495 U.S. 207 (1990)
49 (recognizing that motion picture may infringe copyright in
50 book by using its 'unique setting, characters, plot, and
51 sequence of events')
52 See Steven Englund, Idea, process or protected expression?
53 Determining the scope of copyright protection of computer
54 programs, Michigan Law Review, 866, 1990, 88; Marc
55 Kretschmer, Copyright protection for software architecture:
56 Just say no!, Columbia Business Law Review 823, 1988
57 (hereafter “Kretschmer")
58 Menell Peter S, An analysis of the scope of copyright
59 protection for application programs, 41 Stanford Law
60 Review, 1045, 1989, 1080 (hereafter “Menell”)
61 Weinreb Lloyd L, Copyright for functional expression, 111
63 Answered in the affirmative, as discussed previously
64 101 US 99 (1879). The case dealt with whether copyright
65 could subsist in a book describing a book-keeping system.
66 The court held that although the book could be copyrighted,
67 there could be no copyright in the art or the idea itself, i.e.,
68 the book-keeping system that was described. Copyright
69 extended to the plaintiff’s description of book-keeping, but
70 not the system itself
71 On the basis of this test the Court of Appeals upheld the
72 substantial similarity finding of the lower Court
73 Melville Nimmer and David Nimmer, Nimmer on
74 Copyright, §13.03 [F] [1] (13-122) (hereafter “Nimmer”); Computer Associates v Altai, 982 F. 2d 693, at 705. See also
75 Menell, supra note 24, 1074, 1082; Kretschmer, supra
76 note 23, at 837-39; Thomas M Gage, Whelan Associates v
77 Jaslow Dental Laboratories: Copyright protection for
78 computer software structure—what’s the purpose? Wisconsin
80 “… each subroutine is in itself a program, and… may be
81 said to have its own idea”, per Walker J, Computer
82 Associates v Altai, 982 F. 2d 693, 705
83 Velasco, supra note 1, 268
84 Indeed the 9th Circuit never went beyond the extrinsic test,
85 after the application of which there were no more triable
86 issues remaining
87 960 F. 2d 1465, 1475
88 Velasco, supra note 1, 268
89 Ibid, 269
91 Ibid, 60
92 45 F. 2d 119. “Upon any work… a great number of patterns
93 of increasing generality will fit equally well, as more and
94 more of the incident is left out. The last may perhaps be no
95 more than the most general statement of what the play is
96 about, and at times might consist only of its title; but there
97 is a point in this series of abstractions where they are no
98 longer protected, since otherwise the playwright could
99 prevent the use of his ‘ideas’, to which, apart from their
100 expression, his property is never extended.” Per Learned
101 Hand J, 121
102 In Gates Rubber v Bando Chemical Co, 9 F. 3d 823 (10th
103 Cir. 1993), which followed and applied Computer Associates v Altai, the following six levels of abstraction
104 were said to be generally applicable to computer software:
105 (1) the main purpose; (2) the program structure or
106 architecture; (3) modules; (4) algorithms and date
107 structures; (5) source code; and (6) object code
108 740 F. Supp. 37, 61
109 The Doctrine of Merger postulates that where a given idea
110 can be expressed in only one or a few given ways, the
111 expression of the idea and the idea itself ‘merge’, thereby
112 excluding copyright protection of the merger. See Baker v
113 Selden, 101 US 99 (1879), Herbert Rosenthal Jewellery
114 Corp v Kalpakian, 446 F. 2d 738
115 The Doctrine of scenes a faire postulates that where there
116 are certain well known and standard expressions of a well
117 known idea which reside in the public domain, these are
118 excluded from the realm of copyright. These expressions
119 flow naturally from the work’s theme rather than from the
120 author’s creativity. This doctrine has traditionally been used
121 in relation to works with factual or historical themes. See
122 Nimmer, supra note 30, §13.03 [F] [3] (13-130); Hoehling v
123 Universal City Studios, 618 F. 2d 972 (9th Cir., 1980); Landsberg v Scrabble Crossword Game Players Inc, 736 F.
124 2d 485 (1984)
125 740 F. Supp. 37, 61
126 982 F. 2d 693 (1992). Due to the similarity of the two
127 formulations, much of the criticism applicable to the test
laid down in the Altai case is equally applicable to the
Paperback Test, and the same will be discussed at that stage
982 F. 2d 693 (1992)
47
Ibid, 707. “At the lowest level of abstraction, a computer program
may be thought of in its entirety as a set of
individual instructions organized into a hierarchy of
modules. At a higher level of abstraction, the instructions in
the lowest-level modules may be replaced conceptually by
the functions of those modules. At progressively higher
levels of abstraction, the functions of higher-level modules
conceptually replace the implementations of those modules
in terms of lower-level modules and instructions, until
finally one is left with nothing but the ultimate function of
the program.... A program has structure at every level of
abstraction at which it is viewed. At low levels of
abstraction, a program's structure may be quite complex; at
the highest level it is trivial”
Ibid., 707-710.
49
In the context of computer program design, the concept of
efficiency is akin to deriving the most concise mathematical
formulation or formulating the most succinct mathematical
computation. In the context of computer program design, the concept of
efficiency is akin to deriving the most concise mathematical
formulation or formulating the most succinct mathematical
computation. In the context of computer program design, the concept of
efficiency is akin to deriving the most concise mathematical
formulation or formulating the most succinct mathematical
computation. In the context of computer program design, the concept of
efficiency is akin to deriving the most concise mathematical
formulation or formulating the most succinct mathematical
computation. In the context of computer program design, the concept of
efficiency is akin to deriving the most concise mathematical
formulation or formulating the most succinct mathematical
computation. In the context of computer program design, the concept of
efficiency is akin to deriving the most concise mathematical
formulation or formulating the most succinct mathematical
computation. In the context of computer program design, the concept of
efficiency is akin to deriving the most concise mathematical
formulation or formulating the most succinct mathematical
computation. In the context of computer program design, the concept of
efficiency is akin to deriving the most concise mathematical
formulation or formulating the most succinct mathematical
computation. In the context of computer program design, the concept of
efficiency is akin to deriving the most concise mathematical
formulation or formulating the most succinct mathematical
computation. In the context of computer program design, the concept of
efficiency is akin to deriving the most concise mathematical
formulation or formulating the most succinct mathematical
computation. In the context of computer program design, the concept of
efficiency is akin to deriving the most concise mathematical
formulation or formulating the most succinct mathematical
computation. In the context of computer program design, the concept of
efficiency is akin to deriving the most concise mathematical
formulation or formulating the most succinct mathematical
computation. In the context of computer program design, the concept of
efficiency is akin to deriving the most concise mathematical
formulation or formulating the most succinct mathematical
computation. In the context of computer program design, the concept of
efficiency is akin to deriving the most concise mathematical
formulation or formulating the most succinct mathematical
computation. In the context of computer program design, the concept of
efficiency is akin to deriving the most concise mathematical
formulation or formulating the most succinct mathematical
computation. In the context of computer program design, the concept of
efficiency is akin to deriving the most concise mathematical
formulation or formulating the most succinct mathematical
computation. In the context of computer program design, the concept of
efficiency is akin to deriving the most concise mathematical
formulation or formulating the most succinct mathematical
computation. In the context of computer program design, the concept of
efficiency is akin to deriving the most concise mathematical
formulation or formulating the most succinct mathematical
computation. In the context of computer program design, the concept of
efficiency is akin to deriving the most concise mathematical
formulation or formulating the most succinct mathematical
computation. In the context of computer program design, the concept of
efficiency is akin to deriving the most concise mathematical
formulation or formulating the most succinct mathematical
computation. In the context of computer program design, the concept of
efficiency is akin to deriving the most concise mathematical
formulation or formulating the most succinct mathematical
computation. In the context of computer program design, the concept of
efficiency is akin to deriving the most concise mathematical
formulation or formulating the most succinct mathematical
computation. In the context of computer program design, the concept of
efficiency is akin to deriving the most concise mathematical
formulation or formulating the most succinct mathematical
computation. In the context of computer program design, the concept of
efficiency is akin to deriving the most concise mathematical
formulation or formulating the most succinct mathematical
computation. In the context of computer program design, the concept of
efficiency is akin to deriving the most concise mathematical
formulation or formulating the most succinct mathematical
computation. In the context of computer program design, the concept of
efficiency is akin to deriving the most concise mathematical
formulation or formulating the most succinct mathematical
computation. In the context of computer program design, the concept of
efficiency is akin to deriving the most concise mathematical
formulation or formulating the most succinct mathematical
computation. In the context of computer program design, the concept of
efficiency is akin to deriving the most concise mathematical
formulation or formulating the most succinct mathematical
computation. In the context of computer program design, the concept of
efficiency is akin to deriving the most concise mathematical
formulation or formulating the most succinct mathematical
computation. In the context of computer program design, the concept of
efficiency is akin to deriving the most concise mathematical
formulation or formulating the most succinct mathematical
computation. In the context of computer program design, the concept of
efficiency is akin to deriving the most concise mathematical
formulation or formulating the most succinct mathematical
computation. In the context of computer program design, the concept of
efficiency is akin to deriving the most concise mathematical
formulation or formulating the most succinct mathematical
computation. In the context of computer program design, the concept of
efficiency is akin to deriving the most concise mathematical
formulation or formulating the most succinct mathematical
computation. In the context of computer program design, the concept of
efficiency is akin to deriving the most concise mathematical
formulation or formulating the most succinct mathematical
computation. In the context of computer program design, the concept of
efficiency is akin to deriving the most concise mathematical
formulation or formulating the most succinct mathematical
computation. In the context of computer program design, the concept of
efficiency is akin to deriving the most concise mathematical
formulation or formulating the most succinct mathematical
computation. In the context of computer program design, the concept of
efficiency is akin to deriving the most concise mathematical
formulation or formulating the most succinct mathematic
Treaty and other initiatives in software copyright in the United Kingdom, 1 Intellectual Property Quarterly, 35, 1998

85 LAI, supra note 18, 25
86 [1999] Masons C.L.R. 157
88 The court followed the decision in Cattic Components v Hill & Smith, [1982] R.P.C. 182
89 Attridge Daniel J M, Copyright protection for computer programs, European Intellectual Property Review, 563, 2000, 567
90 The relevant portion of Article 1(2) reads as follows – “Protection in accordance with the Directive shall apply to the expression in any form of a computer program. Ideas and principles which underlie any element of a computer program ... are not protected by copyright under this Directive”
91 MANU/SC/0256/1978
93 MANU/SC/0256/1978, at para 46
94 Id
95 Though Nichols v Universal Pictures was never referred to in the judgment, the facts in both cases were quite similar. Nichols too involved the alleged violation of a play by a movie, both of which involved a common theme of love between two persons coming from families of different religious-cultural backgrounds
96 Ibid, paras 59-62
97 [2003] 47 S.C.L. 445 (Cal)
98 Two recent cases endorsing this ‘look and feel’ approach are Raja Pocket Books v Radha Pocket Books, 1997(40) DRJ 791 (Del) and Zee Telefilms v Sundial Communications, 2003 (27) PTC 457 (Bom)
99 FE Engineering and Consultancy Ltd v LG Cable, MANU/DE/1636/2002, para 14
100 1997(40) DRJ 791 (Del), para 17
101 AIR 2002 Del 279
102 Ibid. para 29. The court states in paragraph 82 of the judgment – “…if the idea is developed into a concept fledged with adequate details, then the same is capable of registration under the Copyright Act…”
103 Id