Intellectual Property in Software: Insights for Indian Business

Mary Mathew †
Department of Management Studies, Indian Institute of Science, Bangalore 560 012
and
Malati Hegde
ECE, Indian Institute of Science, Bangalore 560 012
and
Gopi Garge
ERNET, ECE, Indian Institute of Science, Bangalore 560 012

Received 7 June 2004

Amongst the challenges that actors in the software side of information technology are experiencing, none is more contentious than the issues it raises for intellectual property. The controversy pits administration against faculty, scientists against humanists and academic values against financial interests, employees against management, and software developers against corporate houses. The conflicting global intellectual property practice in the case of software has resulted in multiple schools of thought, each providing their way of how it must be managed. In this paper, three areas of concern in intellectual property practice, business, technology and legal in nature, are addressed. The paper aims to describe these concerns and provides clarity regarding the polarization of schools of thought in intellectual property practice. Through a better understanding of the global and domestic situation, the authors hope to motivate a note of concern amongst those with a weak intellectual property focus in the Indian software sector.

Keywords: IP in software, IP ownership, TRIPS Agreement, patent, copyright, open source, closed source

The degree to which actors who produce software give importance to software protection in terms of patents does differ within the software industry. A recent Marks & Clerk survey revealed that only one third of UK technology companies have an executive at board level with responsibility for intellectual property rights. While no firm would supply goods without systematically collecting payment, many are less diligent when it comes to intellectual property. Consequently, competitors are getting away with using their software inventions free of charge. Intellectual property needs to be addressed at the management level and throughout software development.

†Email: mmathew@mgmt.iisc.ernet.in
Whilst Indian corporate software organizations relentlessly toil at acquiring global software projects, retaining customers through enterprise wide solutions and meeting project deadlines with excellence in quality and project management skills, the issue of intellectual property in software is put either on the back burner, or on no burner. Time has come for countries like India, that have software services as a front-runner in terms of internal economic growth, to look deep into intellectual property issues associated with this sector. The fear that one would be ‘left behind with the wrapper’ in the long term is apparent in current day Indian software business.

Software and intellectual property protection must be understood in the background of business, technology and legal practice worldwide (Figure 1). Each element in the figure triggers concerns in software and intellectual property practice. Technology refers to the nature of software as a technology and the difficulty in defining intellectual property in its context. Business practice refers to the way software business in India is conducted and its repercussions on software and intellectual property management. Legal practice refers to the domestic and global laws and harmonization efforts as well as the problems in the patent office.

The four specific areas of concern that heighten the current confusion in software and intellectual property, stem directly from these elements are: (i) the current practice of software business in India (Business), (ii) the nature of software (Technology), (iii) the inherent diplomacy in TRIPS Agreement (Legal), and (iv) challenges with the patent office (Legal/organizational).

Fig. 1—Source of concerns for software and intellectual property protection
The Current Practice of Software Business in India

Supplying the global market with software solutions is not difficult for India as it has emerged as a competent software services leader in various areas of focus, be it operating system platforms, applications, network management or application sectors such as banking, insurance, health care and manufacturing to name a few. The global software market pull is gigantic. Software trends in the world indicate that it will continue to be an industry of the future with worldwide IT spending increasing in the Americas, European Medicines Agency (EMEA) and Asia Pacific regions. While the estimated increase in spending was about 10.6% in 2001-2002, the compounded annual growth rate is expected to be 12.4% in 2001-2006, with USA itself increasing its spending by US$113 billion in the next five years.\(^3\) Hence, the software business in India is predicted to see a growing trend in software export revenue, characterized by the tier 1, tier 2 companies, MNC back-ends, small start-ups and focused service players\(^3\) and in the volume of software trading.

While Indian software service is expected to grow, the mix and nature of software services is likely to undergo changes. India is considered to be leading, above China, Russia and Ireland in areas of current IT industry size, cost structure for service delivery, quality of service delivery, regulatory environment and culture for services. It has the people advantage with a large skill base of knowledge professionals and English language as premium strength. All of these are also identified as future strengths\(^3\). In the year 2001-2002, Indian software services saw the highest export revenues in custom application development and maintenance (Table 1).

<table>
<thead>
<tr>
<th>Service lines</th>
<th>US $ billions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software and services</td>
<td>4.95</td>
</tr>
<tr>
<td>IT consulting</td>
<td>0.05</td>
</tr>
<tr>
<td>System integration</td>
<td>0.15</td>
</tr>
<tr>
<td>Custom application development and maintenance</td>
<td>2.65</td>
</tr>
<tr>
<td>Packaged software support installation</td>
<td>0.30</td>
</tr>
<tr>
<td>Application outsourcing</td>
<td>1.75</td>
</tr>
<tr>
<td>Network infrastructure management</td>
<td>0.05</td>
</tr>
<tr>
<td>IT enabled services</td>
<td>1.49</td>
</tr>
<tr>
<td>R&amp;D services</td>
<td>1.21</td>
</tr>
<tr>
<td>Product development and design</td>
<td>0.30</td>
</tr>
<tr>
<td>Embedded software</td>
<td>0.91</td>
</tr>
<tr>
<td>Total</td>
<td>7.65</td>
</tr>
</tbody>
</table>

With US$7.65 billion worth software being handed over abroad, there is a need to understand the nature of software trade in the context of intellectual property.

In all activities related to IT consulting, systems integration, custom application development and maintenance, packaged software support installation, application outsourcing, network infrastructure management, IT enabled services, R&D, software product development and embedded software, the deliverable is known. Generically speaking, the deliverable is 'software code' which is handed over to the client who asks and pays for it to be developed in one or more of the above-mentioned areas.
Indian organizations can use various models of intellectual property transfer during the process of handing over of the software code. When intellectual property is handed over to a client in return for a project fee relating to one or more of the software service activities mentioned in Table 1, the following happen:

(i) Complete Transfer of Intellectual Property Ownership to the Client
This mode refers to the handing over of code in open source, embedding algorithms, architecture, design and documentation in entirety to the client. The copyright is transferred to the client and the Indian software organization takes ownership of managing infringement cases of the client.

(ii) Shared Intellectual Property Ownership with Client
The handing over of code in open source, including algorithms, architecture, design and documentation to the client; however, the copyright is shared. The software developer or organization may reuse code in other projects, but infringement is jointly managed.

(iii) No Transfer of Intellectual Property Ownership
This is the case where the code in closed source together with ‘need to know’ documentation is handed over to the client. The copyright is held with the Indian software organization and tracking infringement is the problem of the Indian software organization developing the code.

While the first mode of intellectual property transfer is more common than the others in India, it is true that there is a lack of empirical evidence proving this. The copyright form of intellectual property is commonly used here. Patents are not used for various reasons and this necessitates the question, why did Indian software organizations agree to hand over code and the ownership of their code to their foreign clients?

In the past two decades when software business picked up in India, the intellectual property handing over process has been in the above mode. The business sector has grown used to this practice. To revert now to a situation where code transfer must be actively negotiated is tough, and the sector is bound to lose clients in the process. One draws a parallel to the Indian Pharmaceutical sector that practised its business, developing process technologies rather than focusing on drug development and products until the TRIPS Agreement put a stay on this business pattern. Now, in the pharmaceutical sector, there is a shift to R&D for drug development, a departure from the earlier focus of process patents. However, many pharmaceutical organizations had to close sectoral operations to do the changeover or even fade out. Similarly, Indian IT has been liberal with handing over of code in the first form (i). In the case of software internal awareness strategies towards R&D and its commercialization is essential if intellectual property ownership must belong to India.

An attempt to understand the nature of the technology gives some insight into why this possibly happened. However, if India now has to emerge as a software
leader, it is inevitable that its software intellectual property portfolio must be owned and displayed or generated and marketed. This calls for a new approach to managing software organizations and their business focus, a major change in management of basic and applied R&D initiative starting from the top management.

The Nature of Software

Classification of software is done in many ways. One popular method is to slot it into software that is related to the operating system and software that is related to the applications that run on any operating system. Hence, software that makes up the operating system (OS) is a critical platform on which any application runs efficiently. Operating systems are made for devices, for example, the computer, the mobile phone, the microwave to name a few. Irrespective of device, applications are always operating system dependent, and hence written or encoded for operating systems. The trend however is to make applications operating system or platform free in many cases. Software development occurs both at the operating system and applications levels. The critical point to note is that irrespective of whether one is developing code for OS and / or applications software, the development process has the same nature, structure and flow. They follow the same rules for software development. Software is further examined with a view to understanding intellectual property that emerges during development.

Few authors have focused on the nature of software. Software has been classified as follows in terms of its appearance:

—product in its own right, e.g. for use with general purpose computers,
—component of a product, e.g. of high-comfort telecommunication terminals,
—component of a system, e.g. for image-processing in a computer tomograph,
—component of an installation, e.g. for the control of railway traffic or of a power station,
—an instrument for the manufacture of products, systems and installations, e.g. in factory automation

The software must be looked at as a product and its components. It is looked at also as part of a larger automation system. The components of software during its product development cycle should be viewed in two ways: (i) the development stages of software (Figures 2 and 3), and (ii) the forms in which software is transferred to a client (Figure 4)

(i) The Development Stages of Software

The software development cycle can broadly be captured in a three-stage evolution cycle, namely, problem, process and output (PPO) (Figure 2).

Software in general can be defined as follows:

Software refers to the method (algorithm), programs (code) and related documentation, associated with executing a function on a computer system/device, for example, the control, functioning and operations of computer hardware (or any other device).
The product development cycle typically starts with a problem definition. The problem identification phase or the selection of the problem itself is a phase that is specific to the type of industry. For example, in the communications industry, software may be developed ‘to manage the band-width usage to the Internet’. In a statistical analysis software package, an extended feature can become the defining software problem and may address, for example, ‘reporting of results in a printable table format with options to chose this format’. Once such a problem is identified, the solution to the problem is either researched for prior art in the
area or developed. At times, the prior art prescribes the solution and at other times the prior art is limited, indicating that more time will have to be spent on problem solving. When the problem requires solving through self-developed algorithms, it requires a domain expertise with a good mathematical background and high logical competency.

Process is the means by which the solution to the problem is arrived at and made available as a product. The solution will require the use of available, derived or new methods (algorithms). Problem solving methods are the basis of such a solution. They are often derived from a formal solution that is the result of the application of a theoretical and experiential treatise on the problem. An algorithm can be defined as follows:

An algorithm is a theoretical flow and logical sequence, of problem solving methods (step by step), procedure, or a model of an event or turns of events, mathematical in concept and expressed in the form of a computer program, aimed at performing the function that solves the problem at hand.⁵

When a solution is available, the algorithms for the software are finalized. The interpretation of the solution in the form of the software system is now needed. This is termed as the software architecture. This system is further translated into machine-readable inputs through the development of code. This code is then linked together to form one set of binaries that will work collectively to achieve the desired objective (Figure 3).

The output is the software that executes the functional solution. At this point, the software is tested and validated for functionality and performance. It is typically available as a set of binary programs meant to run on a specific hardware and software platform. These binary programs (binaries) are provided either on a media (CD ROMs, DVDs, floppy disks) or through an online access point for downloading.

According to the Figure 3, once the problem is identified, a series of algorithms are researched and a master algorithm arrived at. Relating the master algorithm to the problem at hand and the operating system in which the software must function, the design-systems-architecture and the design-software-architecture are worked out. The design-systems-architecture addresses the platform in which the software will be nested and active. The design-software-architecture addresses the design of the computer program that will embody the algorithms and the design-systems-architecture. The computer program or code, written for a platform or OS, makes the software interoperable for that platform taking care of the systems-architecture. An objective is to make the software platform free at this stage, i.e. capable of running on any operating system. Computer program can be defined as:

A computer program is an expressive listing of the order of events, routines, loops, symbolic language and other pertinent information, put in tangible form,
(such as on a disk, hard drive, CD, print out, in a book), and required to control the functioning of and operations of computer hardware in order to solve the problem at hand.

In Figure 3 the term libraries are used. These are sets of computer programs that are useful utilities pre-written by other authors and plugged into currently written computer program. Such libraries are catalogued and inserted when common critical functions are repetitive and hence the developer need not re-write that part of the code. It is very common for software developers to maintain such libraries in knowledge repositories and share them with others who need these sub-functions to avoid duplication of work and save time. Libraries of this nature are often a part of the design-systems-architecture and design-software-architecture, other functions that are a part of applications and operating systems. Hence a software developer calling in libraries into the software currently being developed is using intellectual property of the person who wrote that library. Depending on the restrictions posed by such an author, the library may be freely called in or called with permission of its author.

(ii) The Forms in Which Software is Transferred to a Client

Viewed from one perspective if software must solve a problem at hand, then the software team must work on algorithms, design-systems-architecture, design-software-architecture and then develop the associate computer program to make the blueprint run on a device such as the computer. Hence, if the first three stages are supplied to a software developer then, developing the computer program or code in any language is not difficult and just requires logical and hierarchical thinking and familiarity with the software language. This is an important understanding where intellectual property is concerned. If the foreign client provides the algorithms, design-systems-architecture, design-software-architecture and the Indian software organization must just develop the computer program based on the given intellectual input by the client, then that can be a justified reason why the Indian organization copyrights the code, and hands over the code to the client along with the copyright. The Indian organization also expresses in computed language the intellectual requirements of the client. The intellectual effort is skewed towards the early stages of computer code development.

In this three-stage evolution cycle of a software product, intellectual property gets generated through PPO. However, intellectual property has greater probability of getting generated at the early stages of the cycle. The early stages as seen from Figure 3 are identifying the algorithms and master algorithm, design systems architecture and design software architecture. Writing of code is based on these early stages. Code demonstrates the intellectual property in the functional software.

In the event, the Indian software organization solves the problem at hand from the stage of the algorithm itself, then, there is a fair chance that the
intellectual property will be held by the Indian organization even though the computer program that embodies all the early stage developments of the software is handed over to a foreign client. When computer programs are transferred to foreign clients, it is done in two ways (Figure 4).

There are three ways in which computer programs or code can be handed over to the client: in the form of source code, object code and executable code. Source code is the native form of the program developed using programing languages to solve a particular problem at hand. However, to make this program run on a particular operating system on a computer device, it requires to be further compiled into an executable binary. This executable binary is a different ‘form’ of 0/1 form, referred to as binaries, a language that translates the computer program into electrical pulses that the computer device can readily understand. This executable binary code will include:

- References to the operating system’s library of routines that will be accessed during the run time of the program such as the math libraries, network socket libraries, etc.
- References to additional libraries that the software program may be based upon such as packet capture libraries, packet classification libraries, etc.
Inclusion of object code generated from a third party source code (neither belonging to the operating system, nor derived in any manner from the original source code) in the executable binary code such as peripheral drivers, etc.

Software is handed over to a client in the form of the raw source code and/or the executive binaries. Once converted into executable binaries, the source code is unseen, and very difficult, but not impossible, to retrace by a third party.

When Indian software organizations develop their software for a client who pays for the development of code, then the above mode of handing over of source code and executive formats of the code separately is the practice at the end of the programming phase. Further, often the Indian software organization engages in working the software in the clients’ environment, and maintains the software for a given period of about five years.

On the other hand, if the Indian software organization desires to sell the software to many buyers, say in shrink wrapped form, in super markets or such places of marketing, or such forms of distribution, then it has an option of selling the software only in the form of binaries or executive code without the source code. The software runs its business life cycle in the market and fades out of the market when someone else re-does the functioning of the software in an improved or radically innovative manner.

The reason why a foreign client will demand both the binaries and the source code is because a client must always know what is embedded in the computer program. The source code can only show evidence of expression of the algorithm, the design-system-architecture and the design-software-architecture. Further, if after a period of time, the client wants to improve upon the computer program and intends to give it out to some third party to do so (not the original Indian software developer), then the source code and manual help in reworking and further development of the computer program. However, in this process the third party can have access to the code of the Indian software developer. It is for this reason that customer retention and long term relationships with foreign clients are critical to Indian software organization, else their code will be distributed all over the globe over a period of time and the history of transaction will be unknown to the Indian software organization. In this manner, when retention of clients does not occur, the copyrighted computer program of the Indian organization can pass on to other software developer organizations.

Inherent Diplomacy in TRIPS Agreement

Since 1883, the patent harmonization process is in perpetual transition. Smoothening of conflicts due to differential intellectual property policies and practices between countries has become critical. The process was initiated a century ago and remains in a transient state as of 2004. Besides the inter-country differences, generic intellectual property (IP) law when applied equally to all industrial sectors is bound to pose a
problem. The conflict is readily seen between Pharmaceutical and Information Technology (IT) sectors, with manufacturing sectors falling somewhere in-between. While twenty years of life for a patent is relevant for pharmaceutical patents, the same may not be good for software patents. The speed with which an invention gets generated and loses its market life, is certainly perceived as a business issue, and a missed factor in intellectual property law formulation exercises. *Sui generis*, Latin for ‘of its own kind’, multi-sectoral international patent guidelines are not an easy target to reach unless uni-sectoral lobbying is initiated at a faster pace. International lobbying management must receive greater emphasis in management books. Since the consensus is a method of decision-making that is difficult to practice to perfection, Indian software organizations together with their government have much to prepare for prior to such inter country consensus decisions on software and intellectual property. The best case scenario of the ‘harmonization arrived’ state would be when sectoral definitions of intellectual property are outlined and guidelines available, and then again technology will continue to blur the boundaries of sectors. The transient period, however, will cause conflicts in business practice and this is very apparent.

A noticeable characteristic of the transient period is the issue of overdiplomacy. Policy makers tend to be over diplomatic in the process of decision making, bringing into policy formulations subtle flexibility in articles. When viewed in comparison, actual contradictions in articles are noticed. Such contractions produce multiple ways of doing business. Such a path once embedded in international business is very difficult to retrace. The TRIPS Agreement is one significant attempt at harmonization. Where software is concerned, the confusion in TRIPS Agreement rests in articles 10 and 27.

*Article 10–Computer Programs and Compilations of Data*

1. Computer programs, whether in source or object code, shall be protected as literary works under the Berne Convention (1971).

2. Compilations of data or other material, whether in machine readable or other forms, which by reason of the selection or arrangement of their contents constitute intellectual creations, shall be protected as such. Such protection, which shall not extend to the data or material itself, shall be without prejudice to any copyright subsisting in the data or material itself.

*Article 27–Patentable Subject Matter*

Subject to the provisions of paragraphs 2 and 3, patents shall be available for any inventions, whether products or processes, in all fields of technology, provided that they are new, involve an inventive step and are capable of industrial application (5) subject to paragraph 4 of Article 65, paragraph 8 of Article 70 and paragraph 3 of this Article, patents shall be available and patent rights
enjoyable without discrimination as to the place of invention, the field of technology and whether products are imported or locally produced.

The articles can be confused to imply that if the computer program can be called a field of technology, which information technology is anyway, then the computer program can be patentable and becomes an invention, with an inventive step and industrial application (from Article 27). Hence, if experts feel it does, it ceases to merit just a copyright protection and can have a patent protection. On the other hand if one focuses only on Article 10 then one only uses copyright. Given this diplomatic manner of presentation of intellectual property guidelines of the computer program, the variations in business practice across countries are many and these variations are further plagued with issues related to the nature of software itself. There is much difference between the way Europe and USA practices intellectual property for software. An Indian software organization doing business in USA preferably patents software and one doing the same business in Europe need not patent the software but must copyright it! This is so because there is a difference with US and European software protection practices, where the former permits a patent and the latter promotes copyright, although a large number of software patents are filed in Europe too. This confused practice can be a difference in philosophy towards intellectual property that has been taken care of within the TRIPS Agreement.

**Challenges with the Patent Office**

If the software developers opt to patent the software, then, there are a series of problems that the patent examiner in the patent office is bound to face. The controversy arises over proof of novelty and non-obviousness. This is more a problem of the patent office than that of the developer. There is much to be learnt from the experience of countries such as Britain in this context.

In 1994, the public forum held at the UK Patent Office noted about software that:

“The central test for patentability was inventiveness and much greater emphasis should be laid on this aspect of the examination process. Patent Office examiners should be firmer and less inclined to give the benefit of the doubt. The size of the problem of the ‘missing prior art’ was far too large and in any event much of the prior art was in the form of machine code which had never been analyzed or described.”

The chances of finding this prior art during a patent search are bleak. Access to such prior art is impossible as vast amounts of codes are in unorganized circulation, or under wraps in restricted circles, hence not available to the Patent Office for its assessment of inventiveness when an application arises. Grover reports the view of the IP Committee of the British Computer Society as follows:

“The difficulty of examining applications for novelty increases the cost of obtaining a patent, and also puts at risk those who disclose
their inventions in reliance on patent protection which many prove invalid for lack of novelty on challenge after grant”.

As early as 1994, the British Forum found a situation that is similar to current day Indian software business in that much of the software industry tended to be small firms of 1 to 100 people for whom the copyright appears to be the better option. It is possible to prevent copying and infringement, but it is not possible in the case of patents as per the views of small organizations.

Much has been written about the ‘criteria for inventiveness’7, but these criteria are very weak. Besides, there is a problem of defining inventiveness itself in the context of software. If patents are hence ‘handed out for an idea, which any normal person would consider obvious, developers would be living in a quagmire of worthless intellectual property and be inhibited from working’. Making the criteria for inventiveness stringent would on the other hand clear only the most extraordinary ideas namely industrial projects.

Summing up the experience of Britain, the problems are with: (i) proving inventiveness of software through source code, (ii) problem of accessing prior art, (iii) escalating costs, (iv) delay in award of patents, and (v) unnecessary exposure of the inventor to competitors.

Amongst these, prior art is a critical one. The Indian patent offices must begin the process of compiling prior art. One example of such an exercise is the project on software and intellectual property in the network and database management areas sponsored by the Ministry of Communications and Information Technology, Department of Information Technology, Government of India, New Delhi, under the guidance of their Intellectual Property Cell. The project was executed by the Indian Institute of Science, Bangalore, and a repository of software in the above-mentioned domains was developed. The repository has softwares that are not patented, but are in the public domain. This also constitutes prior art. Such repositories are databases that the patent office can use for their prior art search if updated constantly. The repository titled SALIS is an example of what can be done by many research institutions in specialized domain areas to assist the Indian Patent Office with its processing of software protection. Yet another activity is one of databasing any copyrighted software that has been registered in India, under the Ministry of Human Resources Development. Access to such a database is important for the patent office assessing the potential for patenting software.

Inevitable Outcome: Polarized Schools of Thought

Many times an author mentions that there is a dilemma within a businessperson when confronted with the following question: ‘will you patent software’?

When the European industry was asked whether it favours such an extension of patent protection for computer software, many experts nearly always stated one of the answers4: ‘Why not?’ and ‘Do you really want to go so far?’
There is no consensus. European industry is not yet fully sure what such an extension of software patent protection would really mean. Some authors clearly state that software patents could become the kiss-of-death for many software developers, because it is becoming impossible to write a program without a serious risk of falling foul of some patent—frequently an undeserved and opportunistic one. The threat is also grave for many smaller businesses in Europe. They could easily be threatened by out-of-the-blue demands for patent licensing fees for their software, or for some process on their Internet site.

Software in itself is an intellectual property and its development involves intellectual effort. As output, software has a chance to be novel, non-obvious having utility as a characteristic. In USA, there is a clear position of patents and copyrights and less of oscillation.

The emerging polarized schools of thought where software is distributed are:

—Patents vs copyright
—Open source vs closed source
—Freedom to reuse any code in full vs no permission to reuse code even in part

The trade secret method of protecting software is very useful too. It is the one that is stealthily used and is successful when critical functions are embedded into the computer program, and distributed in closed source.

Much has been written about patenting software. So much so that authors have bifurcated the patent community from the software community, implying that those who want software patents must address and explain solutions to inventiveness and prior art to the software community.

Bringing to light the nature of software, Paley states that:

“Copyright and patent law pull software apart to protect it. They inconsistently treat software like a book for one purpose and like a machine for another. Copyright law prevents others from copying the words comprising software as though they were text and the images on the screen as though they were pictures. Software's nature is intrinsically a hybrid combining both text and action. Software's text is its literal expression, while software's action is its nonliteral expression. Courts have found both forms of expression to be copyrightable. Software text consists of a complex set of instructions written in a formal computer language, Judicial Acceptance of the Notion of Duality of Expression. Courts have recognized the twin modes of literal and the nonliteral software expression. Courts have granted separate copyright protection to nonliteral elements, such as a program's overall organization, the structure of its command system, and its presentation of information on the screen.”

The polarized schools of patent vs copyright have much to do with the TRIPS Agreement. A copyright for the computer program does not protect the
algorithms, design-system-architecture and the design-software-architecture, but the code per se. Software developed by authors where the early stage development holds the key to the software may require a patent protection if inventiveness can be proved. Software that drives a camera, satellite launch or runs a smart house, for example, requires R&D for algorithm development, prior to the stage of code writing. Software that drives ATMs and banks and such applications too may require a greater protection. Software that runs secure network systems and high security data may require stronger closed source protections. However, there is a school of thought that open source (OS) works better in such security network situations, but more empirical evidence must be gathered for this to be clearly declared as the better route. One also notices that both corporations and a community of developers develop software. The community of software developers that develops code for the sheer pleasure of programing puts it back into public repositories that one and all can use. They use open source methods of transfers and work in areas of code development that are related to the OS and mass use applications such as word processing, presentations, compiling, multi-media rendering, etc.

Proponents argue that patents provide the necessary incentives to innovate at a time when more inventions are computer-related. Critics claim that such intellectual monopoly hinders innovation because software giants use them to attack fledgling competitors. As software is often built on the achievements of others, writing code could become a legal hurdle, if code per se is patented. There is going to be no apparent convergence amongst these two schools of thought.

While the patent vs copyright is a conflict within the legal systems of intellectual property that of close source vs open source is more related to the nature of the technology itself. The polarized schools of open source vs closed-source on the other hand stem from the value that software must be always kept open-source for a user to verify prior to use. Hence, a user has the right to know what is in the code, and hence the distribution of closed source ‘alone’ is unfair. The conflict is visible between a community of developers and business corporations.

According to Kogut, open source software development is a production model that exploits the distributed intelligence of participants in Internet communities. This model is efficient because of two related reasons: it avoids the inefficiencies of a strong intellectual property regime and it implements concurrently design and testing of software modules. The hazard of open-source is that projects can fork into competing versions. Open-source communities consist of governance structures that constitutionally minimize this danger. Because open source works in a distributed environment, it presents an opportunity for developing countries to participate in frontier innovation.

On the other hand, the development of software in certain areas involves R&D
investment with a return on investment (ROI) model. The development of software can also happen without the investment of much finance and R&D, and an absent ROI model when developed in the community mode. Both forms of software development are prevalent in today’s global market. While the latter is the community approach to software development, the former is the corporate approach to software development.

Those who need to use the closed source strategy because they need to turn around their R&D investments will continue to do so. Those that feel the need to let go with open source will continue to do so. Again there is little hope of convergence for these two schools of thought.

The inherent nature of software itself makes the need to reuse software in its component form (sub-sets of computer program or code) again and again. This makes certain sub-sets of code critical. Within the business organization, knowledge repositories are used for this purpose. However, the degree to which a developer will permit the reuse of code by a third party without permission of the author is a personal philosophy and similar to the closed source vs open source conflict mentioned above. Over time will emerge clarity regarding the classification of critical libraries that are strategically left reusable and open owned by all, and those that are closed and protected by its inventor. Corporations themselves do a cleaning job periodically letting go certain bits of code to the public domain when they feel the need to, and there is no strategic importance to keep it closed source.

The nature of licence when software code is transferred to a client varies from case to case and organization to organization. Business transactions of this type are governed by the legal clauses in the licence agreement. Licences are written in a large number of ways and range from being restrictive to non restrictive. Both the open and closed source schools of thought have their own licences that have third party restrictions in terms of code reuse or is non restrictive in terms of code reuse. Hence, the schools of thought also cause a difference in the way license itself is written and a large plethora of licence types have emerged over time. This complicates the situation further.

Conclusions
An intellectual property protection typology that best fits ‘software’ has had no consensus for over a decade in India. Intellectual property is the right of the inventor, and the degree to which an inventor and an invention can be proved in software will determine the type of protection that will be used. The polarized schools of thought that have emerged in today’s software world of patent vs copyright, closed source vs open source, free reuse vs restricted reuse, is wrought with confused conflict between sociology and economics. It has emerged as a war between the entities beyond the limits of any legal court. They are, the ‘developer community’ and the ‘business corporations’. As is the case with most schools of thought, the eclectic school, uses the best of both, and any one who
understands the conflict well, learns to get the best of both in managing intellectual property in software. It is also true that many from the software developer community who prefer to use the open source route are employed by the business corporation, which does not necessarily support open source. This results in conflict for those developers who believe in open source when their top managements are working with closed source software patents.

There is at this juncture no clarity about convergence. The community developers and the business corporations are likely to take paths that do not converge. The TRIPS Agreement is not likely to come out with a separate perspective on software in the near future. The nature of software in itself is complex and thus precipitating possible confusion regarding the path the intellectual property developer can use. This leaves the software developer with the option to make a business decision about the intellectual property route that can be tread prior to software transfer, loan or sale, rather than a philosophical one. These routes remain patent, copyright, trade secret, licence to no licence or free software. Considering that software has a short run business cycle time, then at times it will only be the licence written for the software that determines how others can use it. Patenting incremental changes in software will be an error, and patents must be strategically decided, and copyright is anyhow a default. Practices of prior art gathering will have to become an activity related to the patent office if Indian patent examiners have to be in a position to award software patents to a computer program that is ‘not code per se’.

The tendency for India to keep its intellectual property, whether in the form of copyright, patent or trade secret, will be largely determined by its actual role in early phases of software development. Hence, if Indian organizations develop new algorithms to solve problems and develop on their own, the design and architecture that goes with the algorithm, then there is greater chance of Indian ownership of intellectual property.

Newer methods of protection of self-written code are, however, emerging. These technology-based methods can become superior in course of time. According to information security software organizations, the Source Code Vault solution enables secure, high-performance code sharing and exchange over the Internet providing newer marketing strategies yet code protection. Thus virtual organizations confidently leverage offshore development of code, outsourcing and other collaborative development projects without opening the entire enterprise's network to subcontractors and other outsiders. This, hence, reduces the risk of code theft. Such technology-based solutions complement the legal solutions that top management in organizations has today to manage intellectual property of software.

References

1  Thompson D F, Intellectual property meets information technology, Educom Review, 34(2) 1999, 14-21
3 NASSCOM, The IT industry in India, *Strategic Review*, New Delhi, 2003
4 Koerber A, What do software patents mean to European industry? *Software Patents in Europe*, December 2000
5 Mathew M, Intellectual property and software, presentation at ECE, IISc, supported by Ministry of Communications and Information Technology, (MCIT), New Delhi, 2001
8 Lea G, Software patents: Will Europe roll over for the multinationals, *The Register*, October 2000
12 Open source’s local heroes, *The Economist*, 369(8353) 2003, 5