AFRICAN JOURNAL OF EDUCATIONAL MANAGEMENT

ISSN [illegible] VOL. 2 NO. 2 JUNE 2008

AFRICAN JOURNAL OF

EDUCATIONAL MANAGEMENT

ISSN 0975 – 0065 Volume 12, No. 2 June, 2009

A PUBLICATION OF THE DEPARTMENT OF
EDUCATIONAL MANAGEMENT,
UNIVERSITY OF IBADAN

TABLE OF CONTENTS

CONTRACTUAL LIABILITIES IN SOFTWARE TRANSACTIONS: AN OVERVIEW

M.A. Araromi
Faculty of Law
University of Ibadan, Ibadan
Phone number: 08052236247
Email Address: marcdexa@yahoo.com

Abstract

Software technology has taken an important position in the information technology environment. Basically, computing equipment may be useless without putting the softwares to drive it in place. The development of software has therefore taken a centre stage in the information technology market. The importance of computer in the society cannot be underestimated, as virtually all organizations, trades and professions cannot operate efficiently without having something to do with computers. Most of these organizations, trades and profession sometimes need customized softwares to drive their computers. No doubts, software development seems to be a money making ventures, especially for experts in this field. However, this juicy advantage may turn into a daunting nightmare for unguarded software developer who may not understand the legal terrain in which he operates. It is on this note that the probable liabilities software developers may incur in the process of plying his trade are discovered and discussed in this work.

Introduction

It is important to know that information technology has had a far reaching impact in our society. It can be considered as one of the instruments of modernization. The introduction of computer in the early twentieth century has brought about a land slide development in the information technology world. It is trite that with use of computers many old things are now being done in a new way and many novel things are also introduced through the invention of computers. In the words of Bainbridge D. (2004): computer has had a permeative effect in virtually every professional, commercial and industrial activity, and many organizations and establishments would find it difficult, if not impossible, to function without relying heavily on computers. Many

activities are now being carried out these days with the aid of computer technology. These activities vary from accounting, computation, weather forecasting, electronic mailing, electronic commerce, banking, diagnosis, publishing, record keeping to data analysis, etc. Computer can also be used as an electronic store house which helps to stock avalanche of information in the computer memory and other electronic and magnetic storage media.

Computer hardware, though representing the physical hard parts or components of the computer, may constitute a mass of useless substance if the necessary software to drive it to work is not installed into the memory of the computer. Software refers to a set of programs or instructions that enables the computer to perform specific tasks or functions. In other words, software makes the hardware to work.

Software can be of two categories based on the types of function they perform. These are the Operating Systems (system software), which controls the working of the computer, and the Application Software which addresses the multitude of tasks for which people use computers. An Application Software can also be of two forms; either a programming software or an application (A.B Adepoju, 2005). The term 'computer software' includes computer programmes, data bases, computer files, preparatory design materials, all manner of works stores digitally and accessible by computer and associated printed documentation such as manuals for users (Bainbridge D., 2005). The scope of the software to be treated in this work will only cover software programmes. There are different types of contract which can relate to computer software. These include *inter alia* contract for the sale or lease of software; contracts licensing software; contracts for the maintenance of software (which may be referred to as support contracts); distribution agreements between manufacturers and distributors of software; bureau services contracts, where one party which has computer software supplies computer services or facilities to a party which does not have its own software (Rowland D. and Macdonald E., 2000).

Another important differentiating factor in software is the difference between the bespoke software, i.e. software written for a particular user, and the software made en-masse and is purchasable off the shelf by prospective users. This latter software is referred to as standard software. However, this standard software may be modified

to some extent to meet the needs of the individual user. This division may aid in considering whether a contract for the supply of software should be regarded as a contract for the sale of goods or the supply of services (or something else).

There are apparently possible ways in which software can be protected under the law. These are by patent, under copyright and by contract. We are mostly concerned with the third method of protection of software because it is so far the surest way to protect the software developer's interest in the software (Michele Rennie M.T., 1991). It is not unusual that parties to a contract will not always find it smooth with each other as far as meeting specification and observance of the terms of the contract are concerned. So there is bound to be conflicts between the parties in situations like this. This may especially be true of software development contracts. One particular difficulty with the software development contract is identifying exactly what the acquiring party requires. In the words of Staughton L.J in the case of **Saphena Computing Ltd. V Allied Collection Agencies Ltd** (1995) FSR 616, he said:

> Just as no software developer can reasonably expect a buyer to tell him what is required without a process of feedback and reassessment, so no buyer should expect a supplier to get his programs right first time. He, too, needs feedback on whether he has been successful. This is why the buyer needs to run acceptance tests using typical business transactions to ensure that each works correctly. Inevitably, though, some will not. This may be the supplier's fault but it is equally possible that the buyer may have got his requirements wrong, have expressed them badly or unwittingly have used terms which were open to different interpretations. Whatever the cause, the programs have to be modified and then retested until the correct result is achieved.

In essence, what the above points signify is that an acquirer may want particular software tailored to certain specifications, i.e. a bespoke software, unlike off the shelf software. What he needs do is to approach a developer of software telling him his needs and specifications. It is the duty of the supplier or the software engineer or developer to develop software to meet those specifications. If after the

delivery of the software, the acquirer is not satisfied with it, disputes may naturally arise. However, software has been described not to be a commodity which is delivered once and for all, but one which will necessarily be accompanied by a degree of testing and modification (Rowland D. and Macdonald E., 2000). It is possible that an acquirer not satisfied with the delivered software may for the first time of such delivery bring legal action against the supplier on the ground that the delivered software does not confirm with specifications and therefore not fit for the purpose for which it is intended to be used. This is not an advisable first approach style, rather the software has to be subjected to modification and testing at intervals before the final delivery. Thus, the selection of software is the most important part of the planning of a computer system; this is because it is the software that performs the basic functions for which the system is acquired (Reed Chris, 1993). It is therefore important that the selection process of software, especially bespoke software, should start with the user defining in writing his needs and requirements. This document can be referred to as the functional or requirements specification and is a crucial document. The document will serve as a guide or a standard bearer for the user to be able to compare with the software delivered to him by the software developer. In other words, the document will serve as an initial blueprint for the software developer and the benchmark upon which his work will be measured. It is therefore important that the user makes his intention clear to the developer. Thus, in the case of Micron Computer Systems Ltd V Wang (U.K) Ltd (1990) unreported, 9 May, the plaintiff's claim included, *inter alia*, that the computer system they bought from the defendant did not provide 'transaction logging'. It was observed per Steyn J. as follows:

> The acknowledged absence of a transaction logging facility is not in reality a fault in the system which was sold. Micron can only complain about its absence if Micron can establish a contractual term, express or implied, of an actionable representation, to the effect that the system included such a facility. In order to make good its case on transaction logging, Micron must therefore establish that they made known to Wand that they required such a facility.

The judge went further to hold that Micron (the plaintiff) had not made its requirement for transaction logging clear to Wang (the defendant), and accordingly Micron's claim was bound to fail. It should however be observed that the subject mater of this suit was that of a system. The same decision will also be reached if it were software.

After all said and done, it is crucial at this stage to treat the liability for defective software. One peculiar way of acquiring software is by licence. The licence sets out what the acquirer could, and could not, do with the software. Software licensing has been described as a vehicle by which the acquirer is given rights to use the software (Rowland D. and Macdonald E., 2000). This phenomenon has also provided an appropriate approach to the exploitation of software by the developer. Granting software licence to an acquirer confers certain rights exercisable on the software by the acquirer, which if not for the licence such acts could constitute illegality or encroachment upon the intellectual property right of the developer, which is protectable under copyright law. Software licence may be exclusive. This means that the developer cannot make a further grant of licence to another person where such a licence is granted to an acquirer. This is especially true of bespoke software. Such software is customized that anybody else cannot acquire it later. Granting licence in a software contract of this nature means that the property, or the proprietary interest, in such software belongs to the developer who retains the source code of the software but only gives the prepared and readily usable programme stored in a medium to the acquirer, thus granting the acquirer licence to use the software based on agreed terms of the contract. In another way, the software developer may make the grant of the licence non-exclusive in which case he would have the opportunity to exploit his work or the software maximally for economic gains.

The crux of software contract is that in most cases the ownership in the software belongs to or is retained by the developer, except there is an express term of the contract transferring the ownership of the software to the acquirer.

In negotiating terms for the formation of software contract, it is not unusual that there may be exaggerated claims as to the performance and specification of software and carrying out of obligations under the contract. This is especially true if the understanding by the parties of the terms of the contract differs. In

some cases, it may be difficult to know whether the correspondence that flowed between the parties in form of letters setting out the client's requirements or the software developer's recommendations constitutes part of the contract, or the terms of the contract are to be restricted to the ones contained in a formal document constituting the final agreement between the parties. In order to escape from this bottleneck, it suffices if the parties can include in their formal written agreement that the terms, or the entire written agreement, constitute the whole contract between the parties.

A successful software contract can be achieved only if there is a full co-operation between the software developer and the client from the period of the development of the software till the installation. The required co-operation was emphasized in the case of **Anglo Group Plc Vs Winther Browne & Co. Ltd** (2000) 72 Con LR 118. Here the client did not want a bespoke system and a standard package was delivered which inadvertently required that the client's other software systems would have to be modified to fit with the standard system. Thus, a full co-operation between the parties was required, especially because the client did not have a full technical knowledge of a computer professional. The judge in this case laid down some implied terms that may be imputed into such a contract thus:

It was an implied term that:

- the purchaser communicates clearly any special needs to the supplier;
- the purchaser takes reasonable steps to ensure that the supplier understands those needs;
- the supplier communicates to the purchaser whether or not those precise needs can be met and if so how they can be met. If they cannot be met precisely the appropriate options should be set out by the supplier;
- the supplier takes reasonable steps to ensure that the purchaser is trained in how to use the system;
- the purchaser devotes reasonable time and patience to understanding how to operate the system;
- the purchaser and supplier work together to resolve the problems which will almost certainly occur. If such co-operation is not present it is likely that the purchaser will not achieve the desired results from the system.

One can safely conclude that the implied terms enumerated by the learned trial judge are equally important in contracts for the development of software. Active co-operation of the parties to the contract is required to enable the parties achieve results. Often time, software is an abstract phenomenon - it is conceptual in nature, and a vivid description of the requirements to be met in the software needs to be drawn. The development of such software needs some intermittent inspections and trials in order to meet with specifications. It is important therefore that the parties co-operate effectively to achieve their desired goals.

Time of delivery of software is also another important term in software contract. There may be a prior agreement between the client and the software developer as to the time such software should be delivered, thereby terminating the contract by way of completion. It is not unusual that software development may extend beyond the agreed time of delivery; this occurs especially when the developer is striving to ensure that the software agree with specifications and special needs of the client, which may require intermittent checking and test-running of the software. In a situation like this, the client may agree to extend the time of delivery of the software. The extension of time may be evidenced in writing with the new date being firmly stated as a condition. It is not unusual to find provisions in contracts for late delivery and late payment. The contract might provide for a specific amount of money to be paid by the defaulter on a time base. This form of predetermined amount of money is referred to as liquidated damages and it is quite different from a penalty. Liquidated damages are a genuine pre-estimate of loss suffered as a result of the breach. Penalty on the other hand is meant to punish the erring party and it is always out of proportion of the damages suffered by the innocent party. The court will often times refuse to enforce a penalty. It should be noted that time of payment, unlike time of delivery which is often termed as a condition, is usually treated as being a warranty unless the contract states otherwise or the circumstances suggest a different interpretation (D. Bainbridge, 2004).

It has been stated above that software by its very nature, especially bespoke software, cannot be boldly delivered as error free work by the software developer at once. Such software often contains errors that need to be traced and corrected within a reasonable time.

This issue was considered by the court in the case of **Saphena Computing V Allied Collection Agencies** (1995) FSR 616. Here a contract was terminated on the basis of a breach of contract because of error in the programs. The Court of Appeal held that it would not be a breach of contract to deliver software that might initially have errors in it. The court stated that software was not a commodity that could be handed over once and for all that it would usually require testing and further modifications within a reasonable time. This 'reasonable time' will only operate where there is no prior agreement between the parties as to when the corrections are to be effected.

Sometimes error correction services may be available with the programme which may be at a separate charge. Support or consultancy services may be available with the more complex programs. Such services are usually separately chargeable and may constitute a term of a software developing contract. The support services often cover the complexities of implementing the program and integrating it into the client's operations, as well as ascertaining the cause of operational difficulties, which may be as a result of hardware or software malfunction [Edwards Chris et al, 1990].

At this juncture, it is imperative to give a brief exposition on the ownership of software or programs written by a developer because this often raises some legal questions. In the absence of any agreement to the contrary, the ownership of software written by a freelance staff of an orgnaisation or a software contractor *prima facie* belongs to the freelance programmer or the software contractor, as the case may be. The organisation employing the services of a programmer may want to own the copyright of a written program or software for its full exploitation or it may want, on the basis of business strategy, to prevent its competitors from accessing the program to avoid stifling its competitive prowess. It is essential therefore that such organisation includes in its contract for development of software provisions for determining the ownership of copyright. In the absence of clear provision of ownership in the contract, the freelance employee continues to have the ownership of the program and may thus grant licence to others for its use.

It is plausible that there may be some uncertainty as to the precise terms of a contract and there is a limit as to how much the courts may be willing to imply. Uncertainty in the terms of a contract

may render the purported contract void and unenforceable. This may not be palatable for the parties that might have committed some resources in actualizing or performing the supposed contract. Thus, taking to litigation in such a situation may not give the best results, especially where the hands of the court are tied in implying or imputing terms into the contract, as the court is not expected to dictate terms in a contract, which must be left to the judgments of the parties to the contract and must be expressed in clear and definite language. This can be distilled from the words of HH Judge Richard Seymour QC in the case of **Co-operative Group (CWS) Ltd v International Computers Ltd** (2003) EWHC 1[TCC).

> If satisfied that parties did indeed intend to enter into a binding agreement and sought to do so, it is no part of the function for the court to seek to frustrate that intention. At the same time it is no part of the function of the court to impose upon the parties a contract which they did not, objectively, make for themselves.

However, the court may have to put into consideration previous cause of dealing between the parties to provide some clues as to the precise scope of the parties' rights and obligations under the contract. Wherever the terms of a contract are not certain, it could be hazardous for the party who has committed his resources towards performing such supposed contract where such a matter is litigated and the court decides there is no contract in existence. This may be true where the terms considered as important by the parties have not been objectively agreed or certain. In some cases, the party who has expended his resources may be entitled to claim *quantum meruit*, this may be possible if the other party has agreed to or at least acquiesced in the claimant carrying out the work. Be that as it may, litigation will not be the best option for the parties because it proffers 'winner takes all' solution which may truncate the prior good intention of the parties to enter into a productive contract for a mutual benefit of both parties.

Contractual Liability in Software Contract

A contract cannot confer enforceable rights or impose corresponding obligations arising under it on any person, except parties to it. Hence, only parties to a contract may enforce it. This concept is known as privity of contract. In exceptional circumstances, contracts may be

assigned or novated so that they become enforceable between persons who are not parties to the original contract. The same legal situations as stated above are applicable in software contracts. Software contracts may take two forms, depending on their contents; these are contracts for the supply of goods or contracts for the provision of services (Edwards Chris et al, 1990). In contract for the supply of goods, there is an implied condition that the goods will be of merchantable quality. The term 'merchantable quality' has been described to mean merely that the goods will be reasonably fit for the purpose for which such goods are usually supplied but does not extend to fitness for any unusual or particular purpose. On the other hand, no such condition is implied in contracts for provision of services, though the conditions that the services provided will be provided with reasonable skill and care and in a timely manner can be implied in such contracts (Edwards Chris et al, 1990). The English law implies strict conditions of quality in contracts for the supply of goods than it does in the contracts for the provisions of services. However, strict obligations attached by statutes to contract for the supply of goods will only apply to software supply contracts if the software can be described as "goods". Moreover, there have been constant academic debates as to the real nature of software contracts. The practice, however, is to treat each case with its merits, i.e. whether such a contract is a contract for the supply of goods or provision of services will depend on the facts of each case. Thus, if software is provided as a component of a contract under which computing equipment and other goods are also provided, such a contract can be categorized as a contract for the supply of goods. On the other hand, if a software developer is engaged to write software to a particular specification, the software is most likely to be categorized as a product of a contract for the provision of services. Consequently, the English strict condition of quality in contracts for the supply of goods may not apply.

Conditions, Warranties and Other Terms in Software Contracts

A condition is a term of a contract that goes to the root of such contract and its breach can give the injured party an option to treat the contract as terminated. Warranty, on the other hand, may be only subsidiary to the main purpose of the contract; even though it is an important term of the contract in its own respect, it can only give the injured party a

right to damages. 'Innominate or intermediate term' was evolved by the court in recent times and it is a hybrid between a condition and a warranty, a breach which could lead either to damages or to repudiation, depending on the effects of the breach. Thus if the breach is so devastating as to deprive the injured party of substantially the whole benefit as was intended from the contract, the remedy would be repudiation for the contract, otherwise it would be damages (Sagay I.E., 1997). It must be noted that the same legal regime as it is applicable to other regular contracts is also applicable to software contract. The terms of contract as expressed above will obviously apply in software contracts. It is however a common practice in the information technology industry, especially in the developed world, like the U.K, for suppliers to seek to limit or entirely exclude their liability for damages for breach of contract. In such a situation, breach of a warranty which usually gives rise to a claim of damages will be of no value to the injured party, whereas the right to treat the breach as condition which entitles the injured party either to withhold payment or to reclaim payment already made is a more effective remedy. Under the common law rule, the commonest exclusions or limitation of liability clause can be linked to a breach of description or quality of software, thus it is particularly common to exclude all liability for loss consequential on a breakdown or malfunctioning of the software or equipment it is used to drive (Reed Chris, 1993). The exclusion clause under the common law rule is strictly interpreted. The agreement under which it reflects must be contractually binding on the buyer. It is most easily effected if it is contained in a written contract signed by "the buyer".

Aside the distinction between conditions, warranties and innominate terms, there is the possibility that a subsidiary agreement, either between the parties to the contract or between one of them and a third party, may be a collateral contract related to, but independent of, the principal contract. A collateral might arise if a user before demanding for an equipment from a manufacturer asks the software house if particular software developed by the software house will be suitable for running on the equipment. If the software house gives its assurance of the quest and the user relied on this in buying the equipment, the only parties to the contract of purchase of the equipment will be the equipment manufacturer and the user. Therefore, the user cannot complain to the equipment manufacturer

that he relied on the faulty recommendation of the software house for purchasing the equipment. However, a court may input a collateral contract as being in existence between the user and the software house. In such a circumstance, although the user could not set aside the equipment purchase contract, it might then be able to recover from the software house the losses incurred in reselling the equipment basing its claim on a breach by the software house of its collateral contract as to compatibility between the equipment and the software required.

Repudiatory Breach of Software Development

Apart from a breach of condition which goes to the root of a contract and a breach of warranty which gives the innocent party an opportunity not to repudiate the contract but only to obtain damages for the breach, there is also another breach which may occur in software contract which is known as repudiatory breach. This breach occurs where a party either expressly or impliedly refuses to be bound by his obligations under the contract, or takes a voluntary step which disables him from performing the contract in accordance with its terms (Edwards Chris et al, 1990). For instance, if a software house was contracted to develop an application for the working of a particular or a unique machine disposed of the machine that it makes it impossible to develop such software any longer, such a software house could be said to have repudiated the contract. In such a case, the other party who has requested for the services of the software house to develop the software may affirm the contract and continue to request for its performance, or he may choose to treat the contract as terminated. In either case, the client has additional right to claim damages, unless the contract included a provision which effectively excused the software house from all liability for damages in any event (Edwards Chris et al, 1990).

Termination due to Frustration

Frustration occurs whenever the law recognizes that without the fault of either party a contractual obligation has become incapable of being performed, because a change of circumstances makes it legally, physically or commercially impossible to fulfil the contract.

In other words, a contract may be discharged by frustration. Frustration occurs when some underlying fundamental fact or condition on which the contract has been based changed or ceased to exist. This common law rule is also applicable in software contracts. This is especially true where there is accidental loss or destruction of the equipment on which the software to be made is to run, especially where such an equipment is unique.

Standard of Care in Software Development

Though there are standards of care for doctors, lawyers, architects and other professionals, but no generally accepted standard has yet been established in development of softwares. Standards of good practice in the software industry continue to change by the day and this makes it difficult to identify at a particular point in time a reasonable required standard. However, proper system of working, quality assuring software product and diligence in creation of the software are necessary to avoid liability for negligence.

The Consumer Protection Act 1987 of the U.K. passed in compliance with an European Economic Community Directive on defective products has created a new strict liability for certain classes of damage caused by defective products. For this purpose, the U.K Act defined product "any good or electricity". There has been no consensus as of damage as to whether software falls within this definition. Be that as it may, the approach to be adopted in treating software cases or contractual matters will depend on individual view of the facts of each case, i.e. each case must be treated on its own merits, as earlier stated. It is suggested that software development contract should not be generally placed under the burden of strict liability, as doing so will mar the interest of developers in venturing into this exercise when they weigh their accruing benefits vis-ą-vis the apparent risks. This may stifle growth in this area of technology and it can spell doom to technology development of many nations, especially the developing jurisdiction, considering the fact that supply of software is apparently a new phenomenon compared to other products. The issue of bespoke software is another stead where strict liability can be considered compared to on the shelf software. When making a bespoke software, the developer must act with care and diligence to ensure accuracy in carrying out the biddings of the user. Though it is not always the case

that software will accurately meet specifications at first attempt, but the developer must make sure that at every delivery such software will not cause any damage to the user's hardwares or data. Rather than importing strict liability into software contracts, the liability of the developer should be base on 'reasonable standard of care' expected of a developer, and this should go in line with the standard of practice of software development required in the trade which changes with time.

Conclusion

The importance of software and its development cannot be underestimated in the world of technology. Many computing equipment are operating on the power of driving softwares which make such equipment mass of empty figures without the availability of the required softwares to spur them into actions. Many computer applications cannot be carried out if the necessary softwares are not in place to facilitate them. The use of computer in virtually every establishment cannot be over emphasized, even in educational institutions. Many schools have adopted the use of softwares for registration of students; recording and computing students' results, provision of students' transcripts, etc. Hence, manual way of collecting data and computation is frizzling out. Therefore, computer software has in no small measure promoted technology development which has given the society at large a soaring profile in the area of advancement. As such, there have been constant demands for softwares and software-related services in this technology-driven world, so much so that such demands are fast out running physical computation equipment. As investments in softwares and software-related services are increasing, so is the need to safeguard the intellectual property rights of the software developers and/or owners. Also, there is a dare need to protect the software developers from unsolicited claims on their accrued profits from their software development exercise. As Christ Edwards et al, (1990) puts it: just as investments in software, and returns on these investments, are increasing, so is the risk of claims arising and converting potentially profitable investment into a loss-making liability. Success in software marketing can no longer be hinged upon marketing skill, but also on the risk management skills. Therefore, those who develop and market software need to be abreast with the constant changes in laws relating to information generally, and trade

practices in that area. Unlike the later, changes in the law may be unexpected, and may have unexpected consequences for those involved with information technology (Chris Edwards, et al 1990). Though a good law does not operate retrospectively, there may be constant need for software developer to be conversant with his legal environment, as this will avoid his being caught unawares.

References

Adepoju A.B (2005). Computer and Computer Use Made Easy for Beginners, 1st Ed., Atman Publisher Ltd, Osun State

Bainbridge D. [2004). Introduction to Computer Law, 5th Ed. Pearson Education Ltd, England.

Edwards Chris et al, (1990). Information Technology and the Law 2nd Ed, Macmillan Publishers Ltd, Great Britain.

European Union Council Directive of 25th July 1985 on Product Liability.

Michele Rennie M.T. (1991). Computer Contracts Handbook, 2nd Ed. Sivect and Maxwell, London.

Reed Chris (1993). Computer Law, 2nd Ed., Blackstone Press Ltd, London.

Rowland, D. and Macdonald, E. (2000). Information Technology Law, 2nd Ed. Cavendish Publishing Ltd, London Sydney.

Sagay I.E. (1997). Nigerian Law of Contract, Ed. Spectrum Books Ltd.

Sales of Goods Act (1979) of the U.K.

Supply of Goods and Services Act 1982 of the U.K.