Computer and Information Science

www.ccsenet.org/journal.html

Vol. 2, No. 4

November 2009

Strict versus Negligence Software Product Liability

Farhah Abdullah

Department of Law

Universiti Teknologi MARA (UiTM)

Dungun Campus, Dungun 23000, Terengganu, Malaysia

Tel: 60-1360-48211 E-mail: farha523@tganu.uitm.edu.my

Kamaruzaman Jusoff (Coresponding author)

TropAIR, Faculty of Forestry, Universiti Putra Malaysia, 43400 UPM Serdang, Selangor, Malaysia

Tel: 60-3-8946-7176 E-mail: kjusoff@yahoo.com

Hasiah Mohamed

Universiti Teknologi MARA (UiTM)

Faculty of Computer and Mathematical Sciences

Dungun Campus, Dungun 23000, Terengganu, Malaysia

Tel: 60-19989-7963 E-mail: hasia980@tganu.uitm.edu.my

Roszainora Setia

Universiti Teknologi MARA (UiTM)

Language Academy

Dungun Campus, Dungun 23000, Terengganu, Malaysia

Tel: 60-1-9919-9909 E-mail: roszainora@tganu.uitm.edu.my

Abstract

The law of products liability in tort is designed to maintain a reasonable balance between the inevitable social costs and the benefits of innovative product technologies. Technological development must be supported not only for the best interests of the public but also the side effect namely product defect into one of the following: (1) manufacturing defect; failures to correctly implement safety measures from the design; and (2) design defects: failures of the design itself to exhibit socially acceptable levels of safety. Software has been described as an artifact with fundamentally different properties than other engineered artifacts. This article will discuss several issues such as identifying the features of high technology products which lead to difficulties in applying traditional tort notions to them.

Keywords: Software, Strict liability, Negligence, Damages, Risk

1. Introduction

Software is a relatively new technological artifact to reach the consumer market. It has offered many technological possibilities and also, the potential for personal injury (Stephen, 1998). Such products have already been involved in cases of personal injury and death (Leveson & Turner, 1993). When a consumer is injured and the common law of products liability is invoked, two different legal standards are available: negligence and strict liability. Software is the new tool; its nature is totally different from the application of traditional legal and engineering tools and defect classifications used to determine the standard of liability.

The modern law of products liability developed from roots in negligence and warranty causes of action. In the 1960s strict products liability in tort developed in response to the perceived inadequacies of negligence and warranty causes of action when applied to products of modern complexity involved in personal injury (S. L. Birnbaum, 1980). It was to be based on proof of product defect rather than proof of fault. Once a product defect was proven to have caused injury to the person,

damages could be awarded. But, for a negligence case, unreasonable conduct must be proved and the injury may be merely economic in nature. Fault (unreasonable conduct) must be proved, the injury may be merely economic in nature, and a product need not be the instrumentality of the injury (it could be a service) (Ravi Belani, n.d.).

Here are some selected examples of the following list of software errors that resulted in recalls of medical equipment (Armour & Humphrey, 1993a), namely incorrect match of patient and data, incorrect readings in an auto analyzer, faulty programming provided false cardiac FVC values, faulty programming caused pacer telemetry errors, incorrect software design caused lockup of cardiac monitor, incorrect calculations, failure of central alarm in arrhythmia monitor, table top moved without a command, detector head could hit the patient, algorithm error caused low blood pressure readings, and over infusion due to programming error. The reasonableness of human conduct presents many arguable issues, and this raises expected costs for the plaintiff in prosecuting a successful case (Keeton & Dobbs, 1984).

The common "limitations of liability" and bold statements negating the manufacturer's responsibility for the behavior of the product have no bearing on a products liability case (Ravi Belani, n.d.). In order to apply, the strict products liability standard, personal injury must be involved. Mere dissatisfaction with the product or economic damages will not support a case; neither will the provision of services. Thus, if a software defect threatens the person or property of a customer or a third party, the injured party is entitled to bring a strict liability claim against the supplier notwithstanding contractual disclaimers, limitations of remedies, and limited warranties. The virtue of a strict liability action from the claimant's perspective is that there is no need to prove that the supplier was negligent. If the product was defective and it caused the claimant's loss, strict liability applies (Armour & Humphrey, 1993b).

There is some debate about whether software is a product or a service. There is growing evidence, however, that courts will consider software a product. The instrumentality causing the injury must be considered a “product”. Thus, the first critical legal inquiry may be whether a product is involved in the injury. Next, the defect must be classified. The term “defect” has evolved to encompass two general categories: defects in “manufacture” and defects in design. Both categories have risen to the same legal label: strict products liability. However, the defect in design has been a controversial category respecting its proper inclusion under the strict products liability rubric which should be subject to a negligence standard. This section will continue to discuss on the problem of vendor liability in respect of a critical issue of whether software is a “Product” or a “Service”.

As computer technology evolves, more powerful computer systems and software are available to more individual users and businesses (Gemignani, 1980). As a result, software vendors are likely to face increasing exposure to lawsuits alleging that software did not perform as expected. The consequences of such lawsuits to software vendors could be catastrophic.

Several examples illustrate the extent of the potential liability faced by software vendors. For instance, in the Therac 25-Accidents (Nancy Leveson & Turner) an error ("bug") in a computerized therapeutic radiation machine caused it to administer incorrect dosages. Two people were killed and several others were seriously injured (Zammit & Savio, 1987). In another case, a construction company alleged that a bug in a spreadsheet program caused the company to underbid a $3 million contract. The company sued the manufacturer of the program for $245,000, claiming it had lost that amount as a result of the incorrect bid (Blodgett, 1986). Finally, in Scott v. White Trucks (1983), the defendant's truck was equipped with computer- controlled anti-lock brakes. After the brakes failed and the truck crashed, the driver of the truck brought a product liability action alleging defects in the software.

Few software product liability cases have been litigated, so little directly applicable judicial guidance is available to pinpoint steps a software vendor should take to minimize liability. However, by understanding the legal theories upon which a hypothetical plaintiff may rely in a suit against a software vendor, it is possible to identify and understand where the risk of liability lies.

2. Threshold issue: Is software a “product" or a "service"?

Some authors like Chris and Angel believed that the classification of software into products and services is illogical (Reed & Angel, 2000). Their late 1990’s argument was based on s.61 of the Sale of Products Act (SOGA) 1979 which defines "products" includes all personal chattels other than things in action and money, and in Scotland all corporeal moveables except money; and in particular "products" includes emblements, industrial growing crops, and things attached to or forming part of the land which are agreed to be severed before sale or under the contract of sale;. However, software producers might argue that software recorded in the form of tangible medium supplied does not covered under the said SOGA 1979 (Reed & Angel, 2000).

A threshold issue is whether computer software is a "product" or a "service." One reason this is important is that sales of products, but not of services, are subject to the damages and warranty provisions of the Uniform Commercial Code (UCC) unless the context otherwise requires, the Article applies to transactions in products. Another reason is that manufacturers of products, but not providers of services, may be subject to suit under a strict liability theory of tort (Bimbaun, 1988). A review of relevant cases shows that courts have used various analytical approaches to decide this issue. Software is

generally licensed and not sold, and the UCC by its terms applies to sales. However, when courts have found that software is a "product," they have not allowed the fact that software is licensed to insulate vendors from liability under the UCC.

3. Contractual liability

It is vital to distinguish between the two different elements of software supply i.e. the license of intellectual property and the development and/or supply of a copy of the software. According to Reed & Angel (2000), the former has one real contractual risk is that a third party may possess intellectual property rights which are superior to those of the licensee. The nature and extent of this risk and the drafting of suitable provisions to control is not the concern. Liability will arise either from the express terms of the contract or from those implied by law (Reed & Angel, 2000).

Breach of contract can be subdivided into (Reed & Angel, 2000) namely,

(a) Contracts of sale or supply; contracts to provide services; and license contracts.

(b) Product Liability, for physical injury or property damage caused by a defective product.

(c) Negligence claims for physical injury or property damage,

(d) Negligence claims for financial loss, divided into two, namely the consequential losses because the software is unusable; and losses caused by reliance on information, produced by the software and addressed to the human mind.

4. Strict liability vs. negligence

Under a strict liability interpretation, a person who is harmed in some way by a software failure would have the right to obtain damages either from the manufacturer of the software or the institution operating the software when the error occurred (eg: a patient suing a hospital because of an x-ray machine software malfunction). Under the negligence interpretation of liability, the victim would need to prove that the manufacturer of the software failed to develop and test its product well enough to the point where it was reasonably confident that the product was safe to operate, or that the operator of the software failed to use the software correctly or grossly failed to interpret the software's findings correctly ("Strict Liability vs. Negligence,").

Under current law, strict liability principles are not applicable to doctors and hospitals, although strict liability is being applied more frequently these days to manufacturers of medical software. It is often difficult to prove negligence, as it can be very difficult to prove where in the chain of production the defect occurred. Potential sources of a defect include ("Strict Liability vs. Negligence,") software manufactures, equipment manufacturers, program distributers, programmers' consultants, companies using the software, and software operators.

4.1 Applying strict liability

Courts have not addressed the issue of whether software is a product for the purposes of applying strict liability in the same way that they have addressed the issue of whether software is a product for the purposes of applying the UCC. Certain cases, however, may be relevant. In one series of cases, courts have held that information is a product and that strict liability therefore applies to it. First, in Saloomey v. Jeppesen & Co. (1983) navigational charts were found to be a product, not a service. In this case, inaccuracies in the charts caused a fatal airplane crash, and the decedents' administrators brought a wrongful death suit against the publisher of the charts (Saloomey v. Jeppesen & Co. 1983). The court said that since the charts were mass- produced and it was likely that purchasers substantially relied on them without making any alteration to them, the publisher had a duty to insure that consumers would not be injured by the use of the charts (Saloomey v. Jeppesen & Co. 1983).

Second, in Brocklesby v. United States (1985) the court held a publisher of an instrument approach procedure for aircraft strictly liable for injuries incurred due to the faulty information contained in the procedure. Strict liability applied because the product was defective, even though the publisher had obtained the information from the government (Lawrence B. Levy & Bell). In contrast, other cases have not applied strict liability to information contained in books. For example, in Cardozo v. True (1977) a woman was poisoned when she ate an exotic ingredient called for by a recipe in a cookbook. The court in this case held that the information contained in the book was not a product for the purposes of applying strict liability, and, therefore, the seller of the book was not liable on that theory for failure to warn of the nature of the ingredients used in the recipe. These cases may be relevant because software usually either contains information or assists in the generation or manipulation of information.

There are strong policy reasons to apply the UCC to sales of software. For example, software sales would then be subject to a concise and statutory, predictable to both vendors and users (Rodau, 1986). It would also be consistent with the parties' expectations; although they typically are licensees, especially when the same program is made available to multiple users, software users frequently regard themselves as purchasers of products (Beckerman-Rodau, 1986). Finally, since the UCC provides for such measures as warranties, disclaimers, and limitations of liability (L. N. Birnbaum, 1988), the UCC provides a mechanism to allow the parties to allocate the risk of product performance (Lawrence B. Levy & Bell). However, if software is considered a "product" in order to apply the UCC, it increases the likelihood that it will be

characterized as a product for other purposes, such as the application of strict product liability law. Software vendors may be held liable for damages caused by their products under various contract and tort theories.

4.2 Contract theories of liability

Often a contract (such as a development or license agreement) exists between the injured user and the software vendor. If so, it will usually be the first place to which the user will turn for a theory of liability under which it may recover against the vendor for damages resulting from defects in the software. For example, the user may claim that the defect breached the terms of a software warranty contained in the contract. The elements and defenses of such a case would follow the normal rules of local contract law. In addition, if the transaction is determined to be a sale of products (Rodau, 1986), the provisions of UCC Article 2 will apply to the interpretation of the contract and the recoverable damages.

Moreover, a court may hold a vendor liable under a warranty theory for statements about the software made outside the formal agreement. "Unless the context otherwise requires, this Article applies to transactions in products (“Uniform Commercial Code”, 1989). In addition, if a court determines that the transaction was a sale of products subject to the UCC, in the absence of a warranty disclaimer it may hold the vendor liable for breach of warranties implied by that statute. One such warranty is the implied warranty of merchantability, which requires that the software be reasonably fit for the general purpose for which it is sold (Levy, 1988). Another warranty is the implied warranty of fitness for a particular purpose, which applies when (1) the vendor knew of a particular purpose for which the software was required; and (2) the vendor knew that the user relied on the vendor's skill and judgment to furnish a suitable program. "Unless excluded or modified ... , a warranty that the products shall be merchantable is implied in a contract for their sale if the seller is a merchant with respect to products of that kind." and "Products to be merchantable must be at least such as ... are fit for the ordinary purposes for which such products are used ...." (“Uniform Commercial Code”, 1989).

Under a contract theory, a plaintiff may recover the difference between the market value of the software as delivered and the contract price of the software (“Uniform Commercial Code”, 1989). Plaintiffs may also be able to claim additional damages such as consequential damages, incidental damages, lost profits, or other losses that the sellers should have reasonably anticipated, such as the loss to the construction company in completing the contract mentioned above (“Uniform Commercial Code”, 1989). The possibility of large damages from these additional categories poses a great threat to software vendors faced with a breach of contract claim.

Since these are contractual claims, the vendor frequently can draft its contracts to substantially limit its exposure (Lawrence B. Levy & Bell). However, in Pennsylvania, privity of contract is not required to recover for breach of UCC warranties of merchantability and fitness for a particular purpose under Sections 2-314 and 2-318 of the Pennsylvania Uniform Commercial Code. Holding that Section 2-318 should be co-extensive with the state's doctrine of strict product liability, a Pennsylvania court allowed a purchaser to recover damages from a computer manufacturer for breach of implied warranty even though the purchaser had actually purchased the computer system from a reseller who had purchased it from the computer manufacturer. The disclaimers and limitations in the manufacturer's contract with the reseller did not apply to the reseller's customer (Spagnol Enter., Inc. v. Digital Equipment Corp. 1989) there may be limits to this ability to minimize liability.

4.3 Tort theories of liability

Since it may be possible for vendors to draft agreements that limit their exposure to contract damages for defective software (Lawrence B. Levy & Bell), injured users also look to a variety of tort theories for recovery. Contractual liability disclaimers may not be effective in limiting vendor liability to suits brought under tort theories. In addition, injured third parties may rely on tort theories, whose applicability is not predicated upon the existence of a contractual relationship with the vendor.

4.4 Misrepresentation

One tort theory involves claims that the vendor fraudulently misrepresented the capabilities of the software (Lawrence B. Levy & Bell). In order to prevail under this theory, the plaintiff must show that it was damaged because (1) the vendor misrepresented a material fact concerning the software, and (2) the plaintiff justifiably relied on this misrepresentation. Plaintiffs apparently have had some success under this theory (Reece, 1987). For example, in Laurie Financial Corp. v. Burroughs Corp.(1985), a vendor claimed that its computer system, including software, would be suitable for a prospective customer's business needs, when in fact the system was not (Reece, 1987). The court held the vendor liable to the customer on a fraudulent misrepresentation theory. It found that the customer justifiably relied on the vendor's misrepresentations because the customer had no previous experience with the particular system involved and had not made an independent investigation of the system's capabilities (Reece, 1987).

The defendant's misrepresentation usually must be intentional for the plaintiff to recover (Reece, 1987), but some jurisdictions recognize a cause of action for negligent misrepresentation whereby (Harper Tax Servs Inc. v. Quick Tax Ltd 1987). Courts may allow some "puffing" in marketing efforts, but seem to be more inclined to find misrepresentation when the vendor makes misstatements concerning facts that are exclusively within the vendor's knowledge (Conley,

1987). Under New York law a claim of negligent misrepresentation is stated only in special circumstances and upon certain allegations including the existence of a nexus of intimacy between the parties approaching that of privity and closer than that of ordinary buyer and seller.

A fraudulent misrepresentation claim is especially threatening to software vendors because under this theory, a plaintiff may sue when it suffers damages solely to its intangible economic interests (such as business reputation), rather than personal injuries or damage to tangible personal property. In the case of Computer Sys. Eng'g, Inc. v. Quantel Corp (1984), held that fraud claim upheld on appeal where purchaser neither knew nor could have known of defects in software.

4.5 Negligence

A second tort theory is that the vendor was negligent in developing the software. The plaintiff must show that the vendor had a duty to use a specific standard of care, usually "reasonableness," and that the vendor breached that duty. There are many actions or omissions that might constitute such a breach. These might include, for example, a failure to (1) write or test the program properly, (2) correct significant bugs in the program, (3) warn of limitations in the program, (4) instruct users how to operate the program (Computer Sys. Eng'g, Inc. v. Quantel Corp. 1984), or (5) provide adequate security for the system. The standard of care in each instance will depend on the specific circumstances. In addition, as technology evolves, there is the possibility that courts will hold vendors liable for less obvious breaches of duty such as the failure to insure that the software does not contain any hidden destructive programs (known as viruses) (Midgely v. S.S. Kresge Co. 1976). Finally, the plaintiff must show that the vendor's breach caused the plaintiff's injury, and that the plaintiff suffered damages from its injury (Branscomb, 1990). Recoverable damages usually are restricted to bodily injury or property loss (Zammit & Savio, 1987).

For a variety of reasons, plaintiffs have had less success claiming damages from software vendors under this theory than under other theories (Invacare Corp. v. Sperry Corp. 1984) Demonstrating that a vendor's conduct is unreasonable is difficult and expensive. Moreover, the defenses of assumption of risk and contributory negligence are available to vendors. For example, vendors have successfully defended on the grounds that the buyer was negligent in using defective data or in hiring incompetent operators. As a result, it may be difficult to prove that the plaintiff's injury was caused by the vendor's breach of duty, such as supplying defective hardware or software, rather than by the plaintiff's own error, such as use of incorrect data. Finally, some courts hold that a negligence action for economic loss is barred if a contract exists between the parties (Reece, 1987).

In the American case of Scott v District of Columbia (1985), where the claimant alleged false arrest and wrongful imprisonment against the police, who had relied on a warrant erroneously issued by a computer system, her suit failed because the officers who arrested her had not been negligent in relying on the computer. It would be legally irrelevant that a computer was involved. But first we must identify the type of damage suffered by the claimant. For instance, the noise from computer printer constituted a nuisance or an information retrieval system contained defamatory material as the same noise from a typewriter or defamation in a book.

4.6 Strict Liability

"Product" is defined in section 1(2) as "any products or electricity" including components. "Products" are defined in section 45 as including "Substances:, and it also defines "substance" as "any natural or artificial substance" (Reed & Angel, 2000). The Directive also defines "product" as any moveable. What about software which is installed by copying it onto the purchaser's system from a medium (e.g. a tape or disk) ownership of which is transferred to the purchaser (Reed & Angel, 2000). Section 5(1) in the Act also limits liability to death or personal injury or damage to property which is ordinarily intended for private use or consumption and was so intended to be used by the claimant by referring to Section 5(1) and (3). Thus, if software is purchased by a business and used solely within the business, claims can only be expected from outsiders who are injured by the business's activities where a cause was a defect in the software. Section 2 (1) requires that the damage be caused by a 'defect' in the product, and this is defined in section 3.A product is defective if it does not provide the level of safety that persons generally are entitled to expect. Once a claimant has established that he has suffered damage, he has a choice of defendants. In addition, Reed and Angel criticized the Consumer Protection Act 1987 as limited whereas scope and the majority of claims in respect of software will be for financial loss.

Another tort theory that might be applicable is strict liability. It is the theory often used in cases involving defective consumer products, and it would only apply if software is characterized as a product (Lawrence B. Levy & Bell). For strict liability to apply to the manufacturer of software, the user must have used the product in a reasonable fashion and the product must have reached the user without substantial change.Thus far efforts to find strict liability as to the services themselves have entirely failed (Bimbaum, 1988). If the user is injured while using the product, the user need show only that the product caused the injury, as stated by the court in *Scott v. White Trucks,* a plaintiff "without fault on his part" may recover under this theory if he "proves ... that the product was defective when it left the hands of the manufacturer." (Scott v. White Trucks 1983) and that the product was sold in a defective or unreasonably dangerous condition. The alleged defect could be a defect in the design or manufacturing of the software, or it could simply be a failure to warn of hazards.

An important feature of the strict liability theory is that it renders legally irrelevant the issue of whether the vendor acted reasonably (Zammit & Savio, 1987). By preventing the vendor from presenting exculpatory arguments, this theory in effect forces software manufacturers to guarantee the safety of their products (Zammit & Savio, 1987). The strict liability theory also has an effect on potential defendants and on recoverable damages. If it is applied, everyone in the chain of distribution of the product may be liable for the plaintiff's damages. However, users are not generally compensated for economic loss under a strict liability theory, but only for personal injury or property damage.

To date, courts have been reluctant to apply the theory of strict product liability to computer software. A Pennsylvania court, however, held that a purchaser may recover for a breach of statutory warranties against a remote manufacturer for purely economic loss (Frank, 1987; Spagnol Enters., Inc. v. Digital Equipment Corp.) Proponents have offered several arguments in favor of doing so. They argue that the software manufacturer should be held liable because it is in the best position to prevent defects, and can spread the risk of liability through insurance or by increasing the cost of the product; that strict liability assures compensation of the victim; that negligence is too difficult to prove; and that the application of strict liability will deter the manufacturer from producing defective software (Lawrence B. Levy & Bell). It is partially these types of considerations that have led courts to apply strict product liability to certain types of information (James, 1988).

On the other hand, the application of strict liability theory might hinder the development of software. Consider again the medical program that regulates the amount of radiation exposure for cancer patients (Lawrence B. Levy & Bell). It is likely that clinical judgments and heuristic rules for making decisions, rather than fixed engineering and mathematical principles, would be used to develop the software (Lawrence B. Levy & Bell). The software developer is therefore not necessarily able to eliminate defects, any more than a medical practitioner can guarantee positive results from radiation treatments. Nevertheless, under the strict liability theory, the software manufacturer might be held liable for large personal injury damages if the radiation treatments do not help, or injure, the patient. Moreover, everyone in the chain of distribution may be liable, including the hospitals and medical practitioners that used the program (Lawrence B. Levy, & Bell, S. Y).

In the coming years, we can expect to see each of these theories tested in the courts as the number of damage claims alleging defective software increases. The prudent software vendor should be cognizant of these risks and take actions to limit its exposure. We now turn to these preventive actions.

5. Malaysian approach

Technology in IT runs so quickly before an ex ante law upon software engineers could be drafted to minimize the number of accidents. The Government agency such as Multimedia Department will impose regulations on the software industry when the software involved is safety critical by raising the cost of software (AR. Rizal, 2004). If government regulation is to be imposed, it should be limited to software whose failure might endanger lives or cause serious damage.

5.1 The Sale of Goods Act 1957

Under the Malaysia's the Sale of Goods Act 1957, section 3 defines products in a very limited and narrow definition. It only applies to every kind of moveable property other than actionable claims and money; and includes stock and shares, growing crops, grass and things attached to or forming part of the land which are agreed to be severed before sale or under the contract of sale; a person is said to be "insolvent" who has ceased to pay his debts in the ordinary course of business, or cannot pay his debts as they become due, whether he has committed an act of bankruptcy or not.

5.2 Consumer Protection Act 1999

Whereas under the Consumer Protection Act 1999. "products" means products which are primarily purchased, used or consumed for personal, domestic or household purposes and includes products attached to or incorporated in, any real or personal property, animals, including fish, vessels and vehicles, utilities and trees, plants and crop whether on, under or attached to land or not, but does not include chooses in action, including negotiable instruments, shares, debentures and money.

While "services" in section 3 in Consumer Protection Act 1999 includes any rights, benefits, privileges or facilities that are or are to be provided, granted or conferred under any contract but does not include rights, benefits or privileges in the form of the supply of products or the performance of work under a contract of service. If we compare the two (2) provisions under Consumer Protection Act, section 71 : Prohibition on exclusion from liability means The liability of a person under this Part to a person who has suffered damage caused wholly or partly by a defect in a product, or to a dependent of such a person, shall not be limited or excluded by any contract term, notice or other provision.

Moreover, section 62 of The Sale of Goods Act 1957: Exclusion of implied terms and condition as to where any right, duty or liability would arise under a contract of sale by implication of law, it may be negatived or varied by express agreement or by the course of dealing between the parties, or by usage, if the usage is such as to bind both parties to the contract, it gives two conflicting views on the part of the liability of the software programmer. Thus, an exact and

appropriate policy recommendations for software liability laws should be made so that a distinction can be made between safety critical and normal software applications as different liability rules will induce different levels of care, any software liability statute or doctrine should differentiate between regular and safety-critical applications such as exacting levels of care should be demanded from programmers whose failures could result in the injury or lose of a human being.

For regular (non-safety-critical) applications, a negligence standard should be imposed. Guidelines for the standard should be drafted by computer professional organizations. In these cases, the burden of proof would be the consumer to show that the company did not meet the guidelines that would discourage lawsuits (AR. Rizal, 2004). Problem regarding the software industry is its high level of innovation. So, to impose strict liability on software engineers would expose individual programmers as well as companies to a floodgate of litigation. The fundamental nature of software engineering makes bug-free software unattainable and the unique characteristics of software that make error-free software should not be expected. Negligence standard should be tailored to the concept of software engineers as professionals. Now, software engineers' skill demands professional responsibility. As their product can be accessible by anyone who rely on their skill, they should be held to high standards of duty such as an auditor whose degree of care and skill are expected of their level of expertise of that profession. Nevertheless, the government should draft regulations on software projects that deal with safety – critical-software. Strict liability should be used in litigation dealing as well as regulations on the software projects by having authorized certificate of programmers and the government performed testing before release to the public.

The authors agree that strict liability is more appropriate to be used as opposed to negligence as software programmers and engineers are not professional having own professional body although their contribution is enormous and the possibility of failure results will emerge.

6. Conclusion

As society increasingly relies on software to perform critical functions in everything from manufacturing to life support systems, the risk that an error in a software program will lead to economic loss, property damage, or personal injury increases. Prudent software developers will be cognizant of these risks and will take steps to minimize their exposure to this type of liability. One of the solutions is to suggest that the syllabus of the Information Technology Law and Cyber Law subjects are not merely discussed ways to apply the new software technology but also adequate methods to make it work safely. Determining how far a software developer should go in this effort requires balancing the degree of exposure to software product liability against the adverse impact as product liability cases in tort are not subject to contract or license disclaimers of liability (Henningsen v. Bloomfield Motors,Inc., 1960), if any, of the steps required to limit this exposure on the software vendor's ability to market and sell its software. This liability issue is still evolving. For the time being, applying strict liability principle is better than negligence as the latter is harder for the consumers to prove.

References

Abdul Rahman,Rizal (2004). *Liability of Software Producers and Users (Handout)*, Malaysia : National University of Malaysia : Masters in Law.

Armour, J., & Humphrey, W. S. (1993a). Software product liability. *Software Engineering Institute, TR CMU/SEI-93-TR-13, ESC-TR-93-190*, 3.

Armour, J., & Humphrey, W. S. (1993b). Software product liability. *Software Engineering Institute, TR CMU/SEI-93-TR-13, ESC-TR-93-190*, 7.

Beckerman-Rodau, A. (1986). Computer Software: Does Article 2 of the Uniform Commercial Code Apply? *Emory Law Journal, 35*, 853.

Birnbaum, L. N. (1988). Strict products liability and computer software. *Computer/Law Journal, 8*, 135-156.

Birnbaum, S. L. (1980). Unmasking the Test for Design Defect: From Negligence [to Warranty] to Strict Liability to Negligence. *Vand. L. Rev., 33*, 593.

Blodgett. (1986). *Suit Alleges Software Error,* A.B.A. J., Dec. 1, at 22.

Branscomb, A. W. (1990). Rogue computer programs and computer rogues: Tailoring the punishment to fit the crime. *Rutgers Computer and Technology Law Journal, 16*, 1.

Brocklesby v. United States 767 F.2d 1288 (9th Cir. 1985).

Computer Sys. Eng'g, Inc. v. Quantel Corp., 740 F.2d 59, 65-66 (1st Cir. 1984).

Consumer Protection Act 1999.

Contra Invacare Corp. v. Sperry Corp., 612 F. Supp 448, 454 (N.D. Ohio 1984), at 396.

Frank (1987), *Tort Adjudication and the Emergence of Artificial Intelligence Software,* 21 SUFFOLK U. L. REV. 623, 647.

Gemignani, M. C. (1980). Product liability and software. *Rutgers Computer and Technology Law Journal, 8*, 173.

Harper Tax Servs., Inc. v. Quick Tax Ltd., 686 F. Supp. 109, 113 (D. Md. 1988).

Henningsen v. Bloomfield Motors,Inc., 161 A.2d 69 (NJ 1960).

Keeton, W. P., & Dobbs, D. B. (1984). Prosser and Keeton on torts. *St. Paul, MN: West Publishing*.

Lawrence B. Levy, & Bell, S. Y. Software Product Liability: Understanding and Minimizing The Risks. *Journal*. Retrieved from http://www.law.berkeley.edu/journals/btlj/articles/vol5/Levy/html/text.html.

Leveson, N. G., & Turner, C. S. (1993). An investigation of the Therac-25 accidents. *IEEE computer, 26*(7), 18-41.

Midgely v. S.S. Kresge Co. (1976), 55 Cal. App. 3d 67, 127 Cal. Rptr. 217.

Nancy Leveson, & Turner, C. S. An Investigation of the Therac-25 Accidents. *Journal*. Retrieved from http://courses.cs.vt.edu/~cs3604/lib/Therac_25/Therac_1.html.

Ravi Belani, Charles Donovan, Howard Loo, & Jessen Yu,(n.d.) Strict Liability vs. Negligence. *Journal*. Retrieved February 25, 2005 from http://cse.stanford.edu/class/cs201/projects-95-96/liability-law/Liability&Neg.html.

Rodau (1986), *Computer Software: Does Article 2 of the Uniform Commercial Code Apply?,* 35 EMORY L.J. 853, 857-60.

Reece (1987), *Liability for Defective Computer Software,* COMPUTER L. REP. 853, 855.

Reed, C., & Angel, J. (2000). *Computer law* (Fourth ed.): Blackstone London.

Saloomey v. Jeppesen & Co.,707 F.2d at 671-77 (2d Cir. 1983).

Scott v District of Columbia (1985) 493 A 2d 319.

Scott v. White Trucks 699 F.2d 714 (5th Cir. 1983).

Spagnol Enter., Inc. v. Digital Equipment Corp., 568 A.2d 948, 390 Pa. Super. 372 (1989).

Stephen, R. S. (1998). *Classical and Object-Oriented Software Engineering W/ Uml and C++*: McGraw-Hill, Inc.

Strict Liability vs. Negligence. *Journal*. Retrieved from http://cse.stanford.edu/class/cs201/projects-95-96/liability-law/Liability&Neg.html.

The Sale of Goods Act 1957.

Uniform Commercial Code (1952). Article 2.

Zammit & Savio (1987). *Tort Liability For High Risk Computer Software,* 23 PLI/PAT 373, at 391.