The Y2K Problem: Proposed Statute to Guide Triers of Fact in Determinations of Negligence

William D. Horgan
The Y2K Problem: A Proposed Statute to Guide Triers of Fact in Determinations of Negligence

William D. Horgan[*]


Table of Contents:

I. Introduction

II. The Y2K Problem

A. The Miscalculation Itself

B. The Problem's Scope

III. The Inadequacy of Remedies Other Than the Common Negligence Tort

A. The Inadequacy of Actions in Contract

B. The Inadequacy of Product Liability Actions

C. The Inadequacy of Fraud Actions

D. The Inadequacy of Negligent Misrepresentation Actions

IV. The Inadequacies of the Common Negligence Tort

V. The Brief Rise and Fall of the Tort of Computer Malpractice
"It was August 13 '54, I was five years old, depending where you're counting from."[***]

I. Introduction

{1} Following the coming new year, the Y2K Problem[1] will create problems worldwide. While the exact extent of its harm is open to debate,[2] there is no disagreement over its inevitability. In fact, some computer-related companies (including the makers of Norton Anti-Virus,[3] and Quicken for Windows[4] have already been sued for damages arising from allegedly non-Y2K-compliant products.[5] While various actors at all levels of business and government will be subject to legal liability for such malfunctions,[6] this article will examine the legal liability of software producers and engineers under current remedial theories. Software manufacturers are a logical choice for this examination because they will likely be ultimately liable for most Y2K errors. For instance, if an individual sues his bank for miscalculation of interest arising from Y2K errors, the bank will, in turn, bring an action against the maker of the software responsible for such calculations. This scenario will repeat itself in many contexts.

{2} Most non-tort remedies will not adequately protect plaintiffs harmed by non-Y2K-compliant software, as discussed later in this case note. The traditional tort negligence action will often be the primary basis for these claims. Negligence law, however, frequently under-compensates plaintiffs seeking recovery for damages. Software companies will be held to the usual "reasonable actor" standard in assessing liability. In light of the extensive modern reliance on computer systems and the resulting immense damages from malfunctions, an elevated standard of care similar to a medical malpractice standard is appropriate. Case law has not developed this higher standard, despite some early indications that it would. Thus, a statutory standard is needed. Furthermore, because triers of fact across the country will arrive at varying standards of conduct for defendants in identical situations, a uniform standard of conduct is in order. From a policy standpoint, this uniform statutory standard of care would not only impose an elevated standard of care on a professional industry relied upon in nearly every facet of modern life and provide uniformity and clarity to juries in their determination of negligence, but it would also allow software producers to assess their potential liability from a common elucidated standard.

{3} This article will explain the Y2K problem, address the failure of remedies other than the common negligence tort to adequately compensate those harmed by non-Y2K-compliant software, describe the problems with the common negligence tort, such as the lack of an appropriately high standard of care and a lack of uniform application nationwide, in light of the judicial rejection of the computer malpractice tort, which could correct those two problems, assert that statutory guidance is therefore necessary in this matter, examine industry standards upon which to base such a statute, and propose and evaluate a substantive statute designed to hold all software producers to a uniform elevated standard and to create a rebuttable presumption of negligence upon its violation.

II. The Y2K Problem [7]
A. The Miscalculation Itself

{4} The Y2K Problem is borne from the use of only the final two digits of a year, rather than the entire four, to signify the calendar year in many business software application programs written in the past two decades. [8] Thus, the years 1900 to 1999 were represented with the numerals "00" to "99" inclusively. This abridgement saved computer programmers time, effort, limited disk space, and processing memory. [9] Conserving the latter was a particularly strong rationale, for, in the 1960's and 1970's a megabyte of memory could cost tens of thousands of dollars. [10] Most of the computer systems created using this shorthand have not survived to today, but as later systems arose they were made compatible with the earlier versions, thereby perpetuating the condition. [11] Miscalculations occur when such programs must recognize dates before 1900 or after 1999. [12] If the problem has not been addressed, the computer will interpret the date abbreviated "00" to be 1900, when it is in fact 2000, and make identical interpretations for 2001, 2002, etc. [13] For example, on January 1, 2000, a computer attempting to determine the age of an individual born in 1984 would take the current year ("00" to the computer), subtract the year of birth ("84"), and conclude the person's age to be "-84". The computer may conclude that the person at issue is yet to be born. This particular example would not occur until the year 2000, but various malfunctions have already occurred. For instance, the Philadelphia Police Department became a victim of the Y2K Problem well before 2000. [14] As early as the mid-1980's, when virtually no one had heard of the impending complications, the Department's software strangely began deleting the records of active parolees. [15] The software was supposed to delete only the records of parolees whose parole periods had expired. Instead, it considered anyone whose parole was slated to end after 2000, as having already ended in the early 1900's and thus deleted their files. [16]

{5} In other instances, a computer network that scheduled patients' appointments for over seventy health care facilities shut down when an appointment for January 2000 was entered, a state prison computer ordered the incorrect early release of some prisoners it thought had already served their sentences, and a corned beef company destroyed recently made product under the mistaken assumption that it was almost a century old. [17] Some Y2K-induced injuries are already the subject of lawsuits. For example, in Produce Palace International v. Tec-America Corp., the plaintiff alleged that a cash register system installed in 1995 lacked the ability to process credit cards expiring on or after the year 2000. [18] The cash registers presumably interpreted a credit card expiring in 2002 as having expired in 1902, and thus rejected it. [19]

B. The Problem's Scope

{6} The Y2K Problem is most prevalent in mainframe systems because they generally hold older data and programs than personal computers ("PCs"). [20] Any computer system that performs date calculations is vulnerable. [21] However, and this clearly includes PCs. When one starts a PC, certain read-only software (called "BIOS" for Basic Input/Output System) controls the startup process, including management of data such as the time and date. [22] Most common PC-operating systems like DOS, Windows, and OS/2 will not be affected by the Y2K Problem as they can calculate dates beyond 2000. [23] In PCs, thus, the problem should be relatively limited. External software problems, however, are potentially more vulnerable. Any external software program relying on date calculations in its performance is subject to the Y2K Problem, as in the Philadelphia Police Department example above. [24] Vulnerable components will either miscalculate dates or simply shut down. [25] Fixing the Y2K Problem in these contexts is a simple task, requiring the line-by-line examination and revision of all computer code within the system or the issuance of a Y2K-compliant software upgrade. [26] Date fields must be converted from the common six-digit format (dd/mm/yy) into an eight-digit format (dd/mm/yyyy). [27] The cost to fix each line of code is estimated at one dollar. [28] This seemingly simple task is daunting, however, in scope. For example, the Prudential Insurance Company of America estimates it must correct 125 million lines of code. [29] AT&T, Federal Express, and General Motors must each examine 500 million lines. [30] Their estimated costs are easily calculable.

{7} The Y2K Problem extends beyond the realm of mainframes, PCs, and software. Electronic control
devices known as "embedded systems" operate from computer chips hidden within thousands of products. Billions of them are in use every day. If an embedded chip relies on date calculations in its functions, the Y2K Problem could arise. Automatic teller machines, car ignition control systems, parking meters, trains and buses, hotel room keys, coffeemakers, clocks, refrigerators, air traffic control systems, and air-conditioning systems are just some of the myriad of examples. Most embedded chips will not malfunction on January 1, 2000, either because they do not recognize dates, or in some cases, because it does not matter if the system does not recognize the year 2000. Susan E. Thomas, a Y2K consultant for Unisys Corporation, estimates that only four to six percent of embedded chips will experience problems. Even with such a small percentage of malfunctions, the billions of these devices operating on earth will create a large number of failures. In fact, embedded chips, like hardware and software, have already malfunctioned in a similar date computation context. As it became December 31, 1996, production came to a halt at New Zealand Aluminum Smelters when embedded chips ceased functioning; they had not been programmed to recognize the 366th day of the year resulting from the added leap day. The resultant lack of temperature regulation damaged equipment costing over one million dollars to repair. While this article is primarily concerned with software vendor liability, the embedded chip manufacturer will find itself in a similar, if not identical, position.

The Y2K Problem will go unfixed to some degree. Gartner Group, Inc., an information technology research firm, has estimated that more than eighty percent of companies will experience some Y2K-related malfunctions and that thirty percent of companies will experience malfunctions in critical systems that will substantially impact the company's possibility of survival. Business Week has reported that between 1998 and 2001, the Y2K Problem could cost U.S. business $119 billion in lost economic output.

The most commonly-cited estimate of worldwide cost to repair the Y2K Problem in this context lies between $300 billion and $600 billion. Others have estimated the repair cost at $1.6 trillion. The U.S. government is variously estimated to spend between four and ten billion dollars; the states as a whole around two billion. These costs are just to fix the problem; resultant litigation costs may reach $1.4 trillion.

III. The Inadequacy of Remedies Other Than the Common Negligence Tort

A central tenet of this article is the inadequacy of the common negligence tort remedy in future Y2K litigation. It is that inadequacy that prefaces the call for a uniform, elevated standard of care in Y2K negligence cases. Resort to that under-compensatory standard, however, is necessitated largely because remedies outside the realm of negligence torts are equally, if not more, inadequate as negligence actions, and cannot be so easily altered to provide adequate protection. Simply put, most non-negligence causes of action are quite immutably undercompensatory, and negligence causes of action are less undercompensatory and less immutable. Before one can propose a new standard, one must examine the weaknesses of the current standard.

A. The Inadequacy of Actions in Contract

The foremost issue in any Y2K litigation based on contract is whether the relevant contract is a transaction involving "goods," for, if it is, the Uniform Commercial Code ("U.C.C.") will apply. The U.C.C. defines a good as "all things which are movable at the time of identification to the contract." Hardware is easily identifiable as such, but software is a trickier question. Courts have generally determined software to be a "good" as defined in the U.C.C.; thereby determining that it thus falls under its purview. The U.C.C. then becomes the logical tool for enforcing contract rights. Under the U.C.C., a plaintiff injured by breach of contract has various grounds for recovery, but all will generally fail in their attempt to properly compensate for losses incurred.
Under U.C.C. Section 2-602, a buyer has the right to reject non-conforming goods upon delivery. Of course, most, if not all, buyers are precluded from this option because upon delivery of the goods they had no knowledge of their nonconformity. When the problem becomes evident, it is then almost certainly too late to exercise rights under this "perfect tender rule."

U.C.C. Section 2-608 allows a buyer to revoke acceptance of nonconforming goods within a reasonable period of time if the nonconformity substantially impairs the value of the goods and the goods were accepted without discovery of the nonconformity. Successful revocation of acceptance entitles the buyer to a refund. Here again, time precludes most recovery. Revocation of acceptance must occur within a reasonable time after the buyer knows or should have known of the nonconformity.

Vendors have a strong argument that, in light of widespread societal recognition of the Y2K Problem in recent years and a particularly pervasive recognition among those who rely on computers for business purposes, every buyer should have been aware of potential Y2K noncompliance long ago and all reasonable time periods to revoke acceptance have lapsed.

In addition to rejection of the goods or revocation of their acceptance, the U.C.C. provides actions for breach of express and implied warranties. Section 2-313 provides for express warranties arising from promises, affirmations of fact, descriptions of the goods, or samples or models that become a part of the basis of the bargain. Whether language creates an express warranty is generally a question for a trier of fact. Any contract for the sale of goods also automatically includes an implied warranty of merchantability (if the seller is a "merchant" of such goods), which is a promise that the goods are "fit for the ordinary purposes for which such goods are used." Also, if at the time the contract is entered the seller "has reason to know of (1) any particular purpose for which the buyer requires the goods and (2) that the buyer is relying upon the seller's skill or judgment" in his selection, then an implied warranty of fitness for a particular purpose will also attach. Warranties protect buyers by awarding them the difference between the value of the goods as accepted and the value the goods had they been as warranted, along with incidental and consequential damages.

A software company, however, can easily avoid liability for breach of warranty. The U.C.C. itself instructs sellers how to legally disclaim implied warranties. An implied warranty of merchantability is disclaimed merely by use of language that mentions merchantability and is conspicuous if written. An implied warranty of fitness for a particular purpose is disclaimed by written conspicuous language. Even the simple use of the phrase "as is" is sufficient to disclaim all implied warranties under the U.C.C. Express warranties generally cannot be disclaimed but it is simple enough not to make them: a manufacturer need only ensure that it makes no promises (in the form contemplated by Section 2-313) of Y2K-compliance in the sale of its product.

Sellers further reduce buyers' potential remedies through direct contractual limits on remedies and damages. The U.C.C. states that a contract may substitute its remedies for those of the U.C.C. and may limit remedies to "return of the goods and repayment of the price or to repair and replacement." Consequential damages also may be limited; another obstacle to adequate recovery is then put in place.

Taken singularly, any one of the limits on U.C.C. liability described above would probably not pose too large a problem overall. Most software vendors, however, have prudently limited customers' rights under the purchase contract. The end-user license agreement for Microsoft's word processing software, Word 97 is a good example. It reads in relevant part:

By installing, copying or otherwise using the software product, you agree to be bound by the terms of this [contract].

Microsoft's and its suppliers' entire liability and your exclusive remedy shall be, at
Microsoft's option, either (a) return of the price paid, if any, or (b) repair or replacement of the software product [if it is] returned to Microsoft with a copy of your receipt.[68]

To the maximum extent permitted by applicable law, Microsoft and its suppliers disclaim all other warranties and conditions, either express or implied, including, but not limited to, implied warranties of merchantability . . . [and] fitness for a particular purpose . . . with regard to this software product . . .

To the maximum extent permitted by applicable law, in no event shall Microsoft or its suppliers be liable for any special, incidental, indirect, or consequential damages whatsoever (including without limitation, damages for loss of business profits, business interruption, loss of business information, or any other pecuniary loss) arising out of the use of or inability to use the software product . . . . In any case, Microsoft's entire liability under any provision of this [contract] shall be limited to the greater of the amount actually paid by you for the software product or U.S. $5.00.[69]

{18} Thus, consolidated, the result is a severe, if not total, limit on a plaintiff's ability to recover under the U.C.C. Microsoft has conditioned a purchaser's use of the product upon acceptance of an agreement that delineates the meager remedies of repair and replacement, disclaims all warranties to the law's fullest extent, and limits damages on the whole and those consequential in nature.[70]

{19} Perhaps the largest hurdle for a plaintiff contemplating a contract suit is one that need not be mentioned in the contract itself - the statute of limitations. An action on a contract governed by the U.C.C. must be brought within four years after the cause of action has accrued.[71] Section 2-725(2) states that, "[a] cause of action accrues when the breach occurs, regardless of the aggrieved party's lack of knowledge of the breach. A breach of warranty occurs when tender of delivery is made . . . ."[72] The timing of the buyer's knowledge of the good's defect is of no consequence. Much computer equipment suffering from Y2K-related defects was sold long enough ago to bar claims under this limitation.[73] After all, a buyer of defective software in 1987, for example, was very likely unaware of the Y2K Problem in 1991 when the statute of limitations on his contractual claim expired.[74]

{20} These exculpatory clauses and the nature of the U.C.C. statute of limitations create an undercompensatory regime for plaintiffs seeking to recover on Y2K-related claims.[75] Software purchasers must incur time, money, and opportunity costs on a large scale in any attempt to prevail upon an action founded on one of these well-drafted agreements.[76]

**B. The Inadequacy of Product Liability Actions**

{21} Products liability law also provides inadequate remedies to plaintiffs in the Y2K context. The basic statement of strict liability in tort is stated in Section 402A of the Second Restatement of Torts, which reads in relevant part:

(1) One who sells any product in a defective condition unreasonably dangerous to the user or consumer or to his property is subject to liability for physical harm thereby cause to the ultimate user or consumer, or to his property, if

(a) the seller is engaged in the business of selling such a product, and

(b) it is expected to and does reach the user or consumer without substantial change in the condition in which it is sold.[77]
Most states have adopted this provision by statute or judicial decision. 

A close reading of this section reveals that a seller is subject to liability for physical injuries; there is no mention of manufacturer liability for purely economic loss. 

In fact, while a small number of jurisdictions have allowed recovery of consequential economic damages, the vast majority of cases disallow such recovery in strict liability tort actions likely upon the outdated notion that contract law is sufficient to compensate harmed parties. This "economic loss doctrine" will be inapplicable in the Y2K malfunction cases that involve personal physical injury or property damage, while a cause of action in product liability may lie. But, in many Y2K malfunction cases plaintiffs will seek recovery of solely economic damages, such as lost profits. Thus, strict liability product actions are essentially unavailable to those injured by non-Y2K-compatible software.

C. The Inadequacy of Fraud Actions

Fraud and intentional misrepresentation are tort recovery actions often used to sidestep contractual limits on recovery. Like products liability actions, the standards for fraud actions vary somewhat among the states. Generally, a plaintiff in an intentional fraud action must prove the following: (1) the seller made false representations of fact; (2) for the purpose of inducing the buyer to enter into the contract; (3) with knowledge of their falsity or a reckless disregard for the truth; (4) that were material to the bargain; and (5) and reasonably relied upon by the purchaser. The concealment or nondisclosure of a material fact can form the basis for a fraud action. Fraud damages typically include direct damages, any consequential economic loss the plaintiff can prove with reasonable certainty, and because this is an intentional tort, punitive damages.

The main impediments to adequate plaintiff recovery on fraud grounds is the simple difficulty of proving a defendant's knowledge of a statement's falsity and the probable lack of intentionally false statements of Y2K-compatibility. A party misinformed of the Y2K-compliance of purchased software must leap the high hurdle of proving not only that a defendant actually made a statement that was material and false, but also that the defendant knew of, or acted in reckless disregard for, the statement's falsity. Also, in light of the recent growing awareness of the Y2K Problem, it is quite unlikely that software makers (among the most Y2K-aware parties) would purposely falsely represent the Y2K-compatibility of their software; such activity is simply too dangerous from a business standpoint.

D. The Inadequacy of Negligent Misrepresentation Actions

In states that have them, negligent misrepresentation statutes are often identical to or closely resemble Section 552 of the Restatement (Second) of Torts, which reads in relevant part:

One who, in the course of his business, profession or employment, or in any other transaction in which he has a pecuniary interest, supplies false information for the guidance of others in their business transactions, is subject to liability for pecuniary loss caused to them by their justifiable reliance upon the information, if he fails to exercise reasonable care or competence in obtaining or communicating the information.

At first glance, negligent (as opposed to intentional) misrepresentation may appear to be a viable cause of action against software vendors, largely because plaintiffs need not prove the defendant's deceptive intent. A further examination of the concept reveals its undercompensatory nature, however. Plaintiffs may not recover for pecuniary losses. Also, the supplier of the information must supply the information in the course of his business. Most courts construe this to mean that the actor's business must be that of supplying information, not merely that the actor supplied the false information in the course of any business. Software providers are generally supplying a product, not information. Also, the information must be supplied to guide the recipient in relation to third parties. While negligent representations about Y2K compatibility may affect a plaintiff's relationship with third parties, those representations are not meant to
IV. The Inadequacies of the Common Negligence Tort

{26} The above inadequacies of non-negligence causes of action will likely push Y2K plaintiffs toward negligence actions. While negligence actions may under current law provide a more likely means to recover damages, proceeding under such a theory is also quite problematic.

{27} First, actions in negligence are subject to strong defenses. The aforementioned economic loss doctrine will often bar or greatly inhibit recovery. In addition, defendants will assert that, because the plaintiff's software has yet to malfunction, they have suffered no harm and have no standing to seek recompense. Intuit, Inc., the maker of Quicken software, successfully asserted this notion, among others, in having the complaint against it dismissed in a class action Y2K case. [95] Strictly speaking, this will not bar recovery, only postpone it, but it still harms plaintiffs because they must: 1) incur the additional cost of filing an amended complaint likely subject to the same attack; 2) pursue a less appealing but more immediate cause of action; or 3) wait until harm has befallen them to file suit. In addition to standing issues, defendants in Y2K cases will likely point to contractual negligence liability disclaimers. Persons not recognized as professionals can generally disclaim their liability in negligence; whereas, professionals generally may not. [96] Courts have been loathe to label software vendors as professionals, [97] and to thus allow such disclaimers from them. Plaintiffs then bear the burden of proving these disclaimers unconscionable, a very difficult task indeed. [98]

{28} Second, and more important to the purposes of this article, the traditional negligence tort will hold software creators to the usual "reasonable actor" standard, a standard problematic in the Y2K context. This standard will vary from case to case and from jury to jury, in light of the evidence introduced as to the appropriate level of care. What is found to be negligent in one case may be found to be not negligent in another. Triers of fact have no set standard to guide them. A singular standard of care particular to software engineers in the Y2K context is needed. This new standard will not only hold software vendors to an appropriate care level based upon industry standards, but will also provide a uniform standard within and among jurisdictions. This new standard should be flexible, however, it must not delineate activity that is per se negligent. Rather, it must account for variations of fact by delineating certain activities which must be performed when ensuring Y2K-compatibility of software products and provide that failure to perform these activities creates a rebuttable presumption of negligence. Lastly, this standard must be statutorily promulgated; judicial reticence to adopt one necessitates such. If properly researched, drafted, and issued, such a statute will hold all software engineers to a uniform standard of care commensurate with a professional actor having the level of expertise expected of an individual employed in the relevant field. The tort of computer malpractice can serve as a model for this proposed statute because like other malpractice actions, it considers industry standards in its analysis of negligence.

V. The Brief Rise and Fall of the Tort of Computer Malpractice

{29} Though not characterized as such, the first inklings of a computer malpractice tort arose in the 1977 case of F & M Shaefer Corp. v. Electronic Data Systems Corp. [99] Plaintiff F & M Shaefer Corporation and defendant Electronic Data Systems had contracted for Electronic Data Systems in relevant part to develop and to install a modern computer system for F & M Shaefer. [100] F & M Shaefer filed suit following repeated malfunctions of the system. [101] The U.S. District Court for the Southern District of New York determined that a special relationship existed between the plaintiff and defendant in this case based upon F & M Shaefer's necessary reliance on Electronic Data Systems's technically complex knowledge of an arena of which Shaefer had no knowledge. [102] Therefore, as is prevalent in malpractice cases, the court ruled that
the statute of limitations should begin to run only upon the cessation of this continuing professional relationship. [103] While, no court ever reached the issue of professional duty in this case due to its eventual settlement. [104] the idea that heavy reliance on a computer company's technical knowledge may create in them a heightened standard was proffered.

{30} The first overt attempt by a plaintiff to characterize a defendant's conduct as computer malpractice came in a similar vein in Triangle Underwriters, Inc. v. Honeywell, Inc. [105] Plaintiff, Triangle Underwriters, Inc. alleged that a computer system consisting of both hardware and software purchased from defendant Honeywell, Inc. failed to perform and sought relief on grounds of negligence, fraudulent inducement, breach of contract, and breach of warranty. [106] The court held the contract and negligence causes of action were time barred. [107] In an attempt to circumvent the same statutes of limitations that barred those claims, Triangle Underwriters, Inc. additionally alleged that Honeywell, Inc. failed to supervise and correct deficiencies in the system and wrongfully withdrew their support personnel, characterizing these actions as "computer malpractice". [108] subject to the "continuous treatment" doctrine recognized in malpractice actions. [109] The continuous treatment doctrine, briefly mentioned above in F & M Shaefer Corp., dictates that the statute of limitations in medical malpractice actions begins to run at the end of continuous treatment or the physician-patient relationship because a patient is not expected to interrupt a continuing course of treatment "by serving a summons on the physician or hospital superintendent . . .". [110] The Second Circuit Court of Appeals recognized that New York courts had extended the continuous doctrine outside the medical realm. [111] but refused to extend it to the tort of computer malpractice, finding that, "there is wholly lacking in the case at bar that professional relationship upon which application of the doctrine, [of continuous treatment] in any context, depends." [112] This court also emphasized the element of reliance between the parties, perhaps providing for computer malpractice actions in cases of substantial reliance. [113] While, Triangle Underwriters, Inc. was probably more concerned with overcoming their limitations problems than with the characterization of their claims as computer malpractice, the significance of this case lies in its recognition of the tort and in its indirect support of F & M Shaefer Corp.'s theory that increased responsibility may accompany increased reliance.

{31} Later that same year, the U.S. District Court for New Jersey addressed the issue of a computer malpractice claim, albeit briefly. In Chatlos Systems, Inc. v. National Cash Register Corp., plaintiff, Chatlos Systems, Inc., filed suit following the purchase and installation of an allegedly faulty computer system from defendant, National Cash Register Corp. [115] The court rejected Chatlos Systems, Inc.'s computer malpractice claim in its first footnote (and similarly, did not address it in the opinion's body) upon the reasoning that the general technical complexity and importance to the business world of National Cash Register Corp.'s endeavors did not justify an increase in their potential liability exposure. [116] The court also based their rejection upon the "absence of sound precedential authority." [117] By 1980, no court had recognized the tort of computer malpractice. The undercurrent of reliance as a factor had nevertheless provided one rationale for the action. [118]

{32} In 1986, the door was opened to impose malpractice liability on computer engineers in Data Processing Services, Inc. v. L.H. Smith Oil Corp. [119] when Indiana's Fourth District Court of Appeal found computer engineers were analogous to doctors or lawyers. [120] There, Data Processing Services, Inc. and L.H. Smith Oil Corp. orally agreed that Data Processing Services, Inc. would create accounting software for Smith. [121] After Smith failed to make timely payments, Data Processing Services, Inc. sued and L.H. Smith Oil Corp. counterclaimed for breach of contract. [122] The Fourth District Court of Appeals affirmed the lower court's award of damages of $33,000 in favor of L.H. Smith Oil Corp. [123] on the basis that Data Processing Services, Inc. held itself out as possessing skill and qualifications in its respective profession and breached its implied representation that it would perform with the skill and diligence ordinarily possessed by other well-informed members of their possession. [124] This idea was not unique. [125] Quite unique, however, was the court's position that "[t]he situation here is more analogous to a client seeking a lawyer's advice or a patient seeking medical treatment for a particular ailment than it is to a customer buying seed corn, soap, or cam
The court's direct analogy between computer engineers and doctors and lawyers, the latter of which are clearly actors subject to professional malpractice, pushed slightly ajar the door to computer malpractice claims. This idea, coupled with the substantial reliance rationale explained above, forms the basis for the call for recognition of the computer malpractice tort.

{33} The next relevant case in this area was Diversified Graphics, Ltd. v. Groves. [127] While views conflict on the importance of this case to the evolution of computer malpractice tort, [128] the court's notions of professional standards are relevant to this article. Plaintiff Diversified Graphics, Ltd. and Ernst & Whinney, of whom defendant Groves was chairman, contracted that Ernst & Whinney would advise Diversified Graphics, Ltd. on the purchase and installation of a computer system. [129] Ernst & Whinney selected a vendor from which Diversified Graphics, Ltd. purchased a computer system. [130] After the system proved difficult to operate and inadequate to perform Diversified Graphics, Ltd.'s necessary functions, Diversified Graphics, Ltd. filed suit against Ernst & Whinney asserting various claims including negligence. [131] Following a jury verdict amounting to $82,500, Ernst & Whinney appealed upon the grounds that the district court should have held it to an ordinary standard of care rather than to the professional standard of care it applied. [132] The district court instructed the jury with pattern instructions used in actions against physicians. [133] The Eighth Circuit decided that Ernst & Whinney had been appropriately held to a professional standard of care, on the grounds that a negligence action based upon a client-consultant relationship generally gives rise to a professional standard, of care. [134] In so doing, the court discussed two important concepts. First, the court stated that "[t]he degree of skill and care . . . required of a professional is a question of fact for the jury." [135] Ernst & Whinney had contended that Diversified Graphics, Ltd. failed to adequately define the applicable professional standard but the court found that Diversified Graphics, Ltd.'s expert had appropriately elucidated applicable standards. [136] Obviously, if the appropriate standard of care was statutorily elucidated and uniform, this entire analysis would be unnecessary. Second, and relatedly, the court placed much emphasis on standards adopted by the American Institute of Certified Public Accountants. [137] It found that ample evidence existed to establish the defendant's failure to meet those standards. [138] While the lack of a comprehensive licensing scheme for software engineers is often cited as a reason for denial of malpractice liability, [139] applicable industry standards can provide a suitable alternative basis upon which to craft a common standard of care. Despite this application of a professional standard, this case was not deemed a malpractice action. [140] Two significant features of malpractice actions, however, occurred in this case: "[f]irst, an elevated standard of care was applied;" second, Diversified Graphics, Ltd., acting in tort, recovered economic loss damages. [141]

{34} Following Data Processing Services [142] and Diversified Graphics, [143] legal scholars and practitioners might have foreseen the coming of a computer malpractice tort analogous to medical or legal malpractice and based upon ideas of increasing reliance, the necessity of a common standard of care, and the availability of industry standards, even in the absence of a licensure requirement. Such a standard would provide the very benefits this article advocates: a higher standard of care, uniformity of application, and predictability and efficiency in and out of the courtroom. Courts, however, remained unwilling to build such a cause of action upon this foundation. [144] largely because of the precedential notion that a lack of a licensing scheme precludes imposition of professional liability. It is this unwillingness that necessitates a call for statutory guidance. A statute seeking to delineate a uniform standard of care must be based on the benchmark looked to in individual cases -- industry standards.

VI. An Examination of Some Basic Standards

{35} Industry standards for Y2K-compliance amply describe what a product must do to be deemed Y2K-compliant. For instance, several federal agencies have issued a regulation defining Y2K-compliant with respect to information technology:
The information technology accurately processes date/time data (including, but not limited to, calculating, comparing, and sequencing) from, into, and between the twentieth and twenty-first centuries, and the years 1999 and 2000 and leap year calculations, to the extent that other information technology, used in combination with the information technology being acquired, properly exchanges date/time data with it.\[145\]

Most definitions of Y2K-compliance appropriately focus on this end result - the ability of software or hardware to perform certain calculations. These standards generally do not define, nor address the testing that software must undergo to ensure its Y2K compliance. Because a technical review of available software testing procedures is beyond the scope of this article, and largely beyond the knowledge of the common person, this article will examine, with more particularity than the above-quoted definition, what software must be able to do so as not to malfunction, and then posit that failure to appropriately test one's product to ensure its capability to perform these basic functions constitutes presumptive negligence.

\{36\} The British Standards Institution has issued a document entitled, A Definition of Year 2000 Conformity Requirements.\[146\] Its definition of Y2K compliance contains four rules. First, no value for the current date, whatever that may be, will cause any interruption in operation. Second, date-based functioning must operate consistently for all dates before, during, and after the year 2000. Third, in all data storage and interfaces, the century in any date must be delineated either explicitly (four-digit year displays) or by unambiguous algorithms. Fourth, the year 2000 must be recognized as a leap year.\[147\] Put simply, the British Standards Institute's definition of Y2K conformity, thus, contemplates software that can correctly recognize the current date, correctly perform any date-based calculation, and accurately depict the date to users. The ability to recognize and to perform calculations based on dates before and after January 1, 2000, is predictably a dominant theme in Y2K conformity standards.

\{37\} Other standards reach the same basic conclusion through more specific language concerning software's practical operation. GTE has also issued proposed criteria for what they term "century compliance."\[148\] In the above vein, GTE notes that Y2K-compliant software must be able to "correctly handle all representations and manipulation of dates . . . between [January 1, 1900] and [December 31, 2050]."\[149\] Within its criterion that no value for a current date may cause interruptions in normal function, however, GTE additionally lists several critical midnight crossings that may cause problems, including the midnight crossings that will lead into January 1, 2000, the fundamental millennium transition, and February 28, 2000, the first leap day of the new millennium.\[150\] Other potentially troublesome dates include January 10, 2000, the first seven-character date; October 10, 2000, the first eight-character date; December 31, 2000, the 366th day of the year; and January 1, 2001, the exit from the year 2000.\[151\]

\{38\} A software program's ability to continue normal operations during and after these critical midnight crossings, and to compute and recognize dates and durations is usually tested through simulation of the crossings.\[152\] Software testers simulate the crossing by making the software "think" (usually through setting the hardware's date) that the current time is just before midnight on December 31, 1999.\[153\] Optimally date-dependant data should be previously entered into the software. The computer is then allowed to run through its simulated crossing and its ability to continue functioning and to correctly recognize and process dates and durations is observed.\[154\] The date-dependant data previously entered is evaluated to ensure its intact survival of the transition.\[155\]

\section*{VII. A Statutory Proposal}

\{39\} Any statute purporting to regulate the standard of care in negligence actions based on non-Y2K-compatibility must simply state a basic level of care. The level of care must be such that failure of the reasonable software engineer to adhere to it quite clearly would be negligent. It must be founded upon basic
In negligence actions based on non-Y2K-compatibility of any software product, excluding embedded software or embedded chip technology, a software manufacturer marketing a software product on or after January 1, 1996 will be presumed negligent in the manufacture of its product if:

(1) (a) the manufacturer, prior to marketing the software product:

(i) fails to test its ability to perform any critical midnight crossing by simulation of the crossing; or

(ii) tests its ability to perform any critical midnight crossing by simulation of the crossing, the software fails to operate properly upon making the test crossing, and the manufacturer markets the product; and

(b) the software fails to operate properly upon making the crossing.

(c) For purposes of this section,

(i) failure to operate properly when making a midnight crossing includes but is not limited to failure to operate, failure to properly interact with other software or hardware components, miscalculation, and data loss.

(ii) critical midnight crossings include the crossings into the dates of January 1, 2000; February 29, 2000; and December 31, 2000; or

(2) (a) the manufacturer, prior to marketing the software product,

(i) fails to test its ability to recognize dates before, after, and upon January 1, 2000 by simulation of current date environments lying before, after, and upon January 1, 2000; or

(ii) tests its ability to recognized dates before, after, and upon January 1, 2000 by simulation of current date environments lying before, after, and upon January 1, 2000; the software fails to properly recognize any date upon testing; and the manufacturer markets the product; and

(b) the software fails to properly recognize any date before, after, or upon January 1, 2000.

(c) For purposes of this section, failure to properly recognize dates includes but is not limited to failure to properly compute the current date, failure to recognize that any date exists, and failure to properly determine in which
(3) (a) the manufacturer, prior to marketing the software product,

(i) fails to test its ability to calculate durations lying wholly before, wholly after, and crossing January 1, 2000 by test calculation of each of the above three types of durations in simulated environments occurring before, after, and upon January 1, 2000; or

(ii) tests its ability to calculate durations lying wholly before, wholly after, and crossing January 1, 2000 by test calculation of each of the above three types of durations in simulated environments occurring before, after, and upon January 1, 2000; the software fails to properly calculate any such duration upon testing; and the manufacturer markets the product; and

(b) the software fails to properly calculate any such duration.

(c) For purposes of this section, failure to properly calculate durations includes but is not limited to failure to calculate the proper number of time units between two dates.

{40} Four important points arise within the introductory paragraph of the statute. First, this standard of care applies to software manufacturers who market their product on, or after January 1, 1996. The date of development, manufacture, or initial marketing of the product is irrelevant. If a software product is being marketed after January 1, 1996, the statutory standard of care will apply. For example, a producer that completes manufacture of its product in 1992, but does not market it until 1997, falls within the ambit of this statute, as does a producer who first markets a product in 1992, and continues to market that product beyond January 1, 1996. Second, the date of January 1, 1996 has been selected for imposition of the standard, on the basis that, as a whole, the informed public had, by then, at least heard of the Y2K Problem. Thus, software manufacturers should also be expected to account for the potential problem. Third, this statute presupposes no finding of negligence as a matter of law in the event of its violation; it imposes a rebuttable presumption of negligence upon those failing to adhere to it. The burden of proof then shifts to the defendant to proffer evidence of non-negligence sufficient to overcome the presumption. In light of the very basic standard of care delineated, this burden-shifting is reasonable. Fourth, this statute expressly excludes embedded software and embedded chip technology producers. While, embedded software producers find themselves in a similar legal position in the Y2K context, the potential for holding such actors to an identical standard of care is beyond the scope of this article.

{41} Section One of the statute pertains solely to critical midnight crossings. If a software manufacturer fails to test its product's ability to perform any critical midnight crossing, and the software subsequently fails to operate properly upon making the crossing, the manufacturer shall be presumed negligent. The testing, of course, must occur prior to any marketing of the product occurring after January 1, 1996, and must consist of simulation of the crossing environment. The myriad of software programs precludes a more specific description of the required testing procedures; some variance must be allowed for application-specific, crossing simulations. In addition, one will be presumptively negligent under subsection (ii) if tests reveal Y2K-related operation errors but the manufacturer markets the product anyway. A non-exhaustive list of examples of failure to operate properly can be found in subsection (c)(i). Subsection (c)(ii) lists three dates which should be crossed into during testing. January 1, 2000, as the primarily problematic millennium
crossover, is vital to any basic standard of care. The other two dates, February 29, 2000, used to recognize that 2000 is a leap year, and December 31, 2000, the 366th day of the year, are less vital. Their removal would likely not compromise the effectiveness of the statute, however their inclusion can only add to it. [157]

{42} The language in Sections Two and Three is patterned upon the language in Section One. Section Two covers testing of proper recognition of dates lying before, after, and upon January 1, 2000, while Section Three addresses the testing of proper duration calculations. Again both sections impose liability not only for failure to test through environment simulation, but also for marketing a software product that has failed such tests. Both subsections (c) include non-exhaustive lists of failure to properly operate. The length of subsection (3)(a)(ii) necessitates explanation of its content. When testing for Y2K compliance, a software manufacturer must ensure its software can calculate durations between dates lying wholly before, wholly after, and across January 1, 2000. But software must be able to do this regardless of the current date--whether it be before, after, or upon January 1, 2000. Therefore, a software manufacturer must test its product's ability to perform each of the three types of calculations in each of the three simulated current date environments.

VIII. Conclusion

{43} A uniform statutory standard of care in the Y2K context for software manufacturers is necessary. Society's increasing reliance, in business and non-business settings, upon computer industry actors; the specialized, professional nature of software producers' education and day-to-day work; and judicial reticence to adopt such a standard, provide rationales for this promulgation. More important, however, is the orderly, predictable, and efficient administration of justice in this oncoming surge of litigation, which serves both as a rationale for and result of the proposed statute. Protracted expert testimony as to the applicable standard of care would be eliminated. Juries would not be asked to comprehend and to sort through highly-technical testing procedures to determine which procedures a reasonable software manufacturer would have conducted, and then decide whether the particular defendant before them had conducted them. The simple language of the statute can lead them to the same conclusion without technical, industry-specific testimony. The language's simplicity also facilitates the creation of associated jury instructions. Furthermore, a statutorily elucidated standard of care allows for predictability during negotiation and litigation. Software manufacturers need not guess as to what the applicable standard of care will be. They can determine with reasonable certainty whether a presumption of negligence will arise against them, and they can accordingly plan their daily operations, settlement strategy, and trial focus. Plaintiffs too can readily ascertain the evidence necessary to cause such a presumption to arise. Lastly, verdict uniformity will result. With an identical standard of care applied throughout any jurisdiction subject to the statute, similar factual scenarios should create similar results. There should be few, if any circumstances, wherein, identically-situated software manufacturers find themselves differently situated after appealing to a trier of fact. For these reasons, a statute holding all software manufacturers to a uniform standard of care for Y2K compatibility for their products is an enactment past due.

William D. Horgan

[*] J.D., Florida State University College of Law, 1999; B.S., Criminology, Florida State University, 1996. I wish to thank the faculty, students, and staff of the College of Law, in particular Professor Steven G. Gey, without whose guidance and knowledge this article would not have been possible, and, above all, my best friend Maddy.

[**] NOTE: All endnote citations in this article follow the conventions appropriate to the edition of THE


[1]. The Y2K Problem is also known as the Millennium Bug, the Year 2000 Problem, the Century Date Change Problem, and the Century Rollover. See Jeff Jinnett, Legal Issues Confronting the Federal Government and the State Governments Due to the Year 2000 "Millennium Bug" (last modified Oct. 18, 1999) <http://www.llgm.com/FIRM/article4.htm>; John K. Halvey & Thomas A. Unger, Year 2000 Software Compliance: Legal Issues for Outsourcing Customers, 3 STAN. J.L. BUS. & FIN. 117 (1997).


[7]. In addition to the Y2K Problem, there are other potential computer malfunctions related to date processing in a limited number of computers. One is the Year 2000 Leap-Year Problem which arises because only every Fourth Century (including 2000) begins with a leap year (1700, 1800, and 1900 were not leap years). Another problem could result on August 22, 1999, when some global positioning system locating devices will strangely think it is January 6, 1980. See Marc S. Mayerson, Insurance Recovery for Year 2000 Losses (last modified June 23, 1998) <http://www.2k-times.com/y2k-p140.htm>.


Today one megabyte of data can be stored on a diskette costing less than a dollar. See Leslie J. Nicholson, Y2K: A little programming glitch that's causing a worldwide panic, TALLAHASSEE DEMOCRAT, Jan. 24, 1999, at 1A, 6A.

[12]. See id. See also Leon Kappelman & James Cappel, A Problem of Rational Origin That Requires a Rational Solution, 47 J. SYS. MGMT. 6 (July 1996) (for a detailed recitation of the problem).
[13]. See Nicholson, supra note 10, at 1A.
[14]. See Leslie J. Nicholson, Y2K predictions are all over the map, Tallahassee Democrat, Jan. 24, 1999 at 6A.
[15]. See id.
[16]. See id.
[17]. See Mark E. Konrad, Countdown to 1/1/00: Solving the "Year 2000 Problem," 6 NEV. LAW. 14 (June 1998).
[19]. See id.
[21]. See id .
[23]. See id.
[24]. See Nicholson, supra note 14, at 6A.
[25]. See id.
[26]. See id.
[27]. See Jinnett, supra note 8.
[28]. See Ashdown, supra note 6, at 299.
[30]. See Clayton, et. al, supra note 22, at 758; Konrad, supra note 17, at 14. The magnitude of the repair needed is also illustrated by the Pentagon's Year 2000 Management Plan, which has identified 25,000 vulnerable systems; 3,143 of them are "mission critical." Approximately 1,000 F-16 fighter airplanes are together just one of the 3,143 mission critical systems. See M.J. Zuckerman & Anthony DeBarros, Avoiding Digital Disaster: U.S. Gets Serious About the Year 2000 Computer Glitch, USA TODAY, Dec. 17, 1997, at 1A, 2A.
[31]. Nicholson, supra note 10, at 6A.

[32]. See id.

[33]. See Olmsted, supra note 2, at 64-65.

[34]. See Nicholson, supra note 10, at 6A.

[35]. See id.

[36]. See Konrad, supra note 17.


[38]. See id. at 188-89.


[43]. See Konrad, supra note 17, at 14; Monica, supra note 37, at 184. The Gartner Group has estimated federal government repair costs at $30 billion. See Zuckerman & DeBarros, supra note 30, at 2A.

[44]. See Konrad, supra note 17, at 14.

[45]. See Monica, supra note 37, at 185; U.C.C. § 2-102 (1990).


[50]. Id.

See id.

See id. § 2-608.

See Monica, supra note 37, at 186.


See id. at (2).

See id.

See id.


See id.

End-user License Agreement for Microsoft Software (Word 1997).

Id.

Id.

See id.


Id. at (2).


See U.C.C. § 2-725(2) (1990). (The U.C.C. does contain one exception to its statute of limitations for warranty actions. Section 2-725 allows actions for breach of express warranty to be brought after the four year limitations period if the underlying contract extends the express warranty to future performance. Prudent contract drafting by software vendors will essentially rule this out).

[76]. See id.

[77]. Restatement (Second) of Torts § 402A (1977).

[78]. See John S. Allee, Product Liability § 2.01 (1996). New Jersey, California, New York, and several other states have adopted their own versions. See id.

[79]. See Restatement, supra note 77.


[81]. See Allee, supra note 78, § 11.07[2]; see, e.g., Waggoner v. Town & Country Mobile Homes, Inc., 808 P.2d 649 (Okla. 1990) (denying damages in product liability case where injury to the product only results in no economic loss); Flintkote Co. v. Dravo Corp., 678 F.2d 942 (11th Cir. 1982) (holding no award of consequential damages for economic loss resulting from faulty traveling ship unloader).

[82]. See Perlman, supra note 75, at 396 n.78; see also supra part III.A.

[83]. See Monica, supra note 37, at 188.


[85]. See id.

[86]. See Clayton, et al., supra note 22, at 787; Monica, supra note 37, at 190.

[87]. See Warren Freedman, The Business Tort of Fraud and Misrepresentation § 1.2 (1990) (citing Lindberg Cadillac Co. v. Aron, 371 S.W.2d 651 (Mo. Ct. App. 1963), where defendant car seller found liable for failure to inform plaintiff buyer of known cracked engine block).

[88]. See Conley, supra note 84, at 671.

[89]. Restatement (Second) of Torts § 552 (1977).

[90]. An exact determination of the viability of state claims is precluded by the potential variation between state negligent misrepresentation statutes and section 552, but the general similarities between the two afford a very reasonable comparison.

[91]. See Restatement (Second) of Torts § 552, cmt. b (1977).


[93]. See id.


See Perlman, supra note 75, at 397.

See infra part V.

See, e.g., Farris Eng'g Corp. v. Service Bureau Corp., 406 F.2d 519, 521 (3d Cir. 1969) (limiting choice of law used to interpret contract); AMF, Inc. v. Computer Automation, Inc., 573 F. Supp. 924, 930 (S.D. Ohio 1983) (finding that "it strains credibility to hold that a business like AMF was not, or should not have been, aware of the language disclaiming warranties (or any other provisions in the 1977 contract."); Office Supply Co. v. Basic/Four, Corp., 538 F. Supp. 776, 788 (E.D. Wis. 1982) (finding that warranty limitation provisions were valid because "plaintiff was aware of the warranty limitations before the contract was signed . . .").


See id.

See id.

See id.

See id.


See id. at 740-41

See id. at 741, 744-46.

See id. at 741, 745-46.

See id. at 744.


See Triangle Underwriters, 604 F.2d at 744-45.

See id. at 745.

See MacKinnon, supra note100, at 1077.


See id. at 740.

See id. at 741 n.1; see also MacKinnon, supra note 100, at 1074-75.

See Chatlos Systems, 479 F. Supp. at 741 n.1. In opposition to this reasoning stands the statement by William Prosser that new and nameless torts are being recognized constantly, and the progress of the common law is marked by many cases of first impression, in which the court has struck out boldly to create a new cause of action, where none has been recognized before . . . The law of torts is anything but static, and the limits of its development are never set. W. Prosser, Law of Torts § 1, at 3-4 (4th ed. 1971) (quoted in
MacKinnon, supra note 95, at 1092-93 n.146).

[118] See MacKinnon, supra note 100, at 1077.


[121] See id. at 316.

[122] See id. The exact procedural history of this case is more protracted.

[123] See id. at 322.

[124] See id. at 319-20.

[125] See Graziano, supra note 119, at 181.


[127] 868 F.2d 293 (8th Cir. 1989).

[128] Compare Conley, supra note 84, at 698-99 (concluding that the decision signaled a step towards recognizing the tort) with Perlman, supra note 75.


[130] See id. at 294-95.

[131] See id. at 295.

[132] See id. at 295, 297.

[133] See Conley, supra note 84, at 683.


[135] Id.

[136] See id. at 296-97.

[137] See id.

[138] See id. at 297.


[141] See Diversified Graphics, 868 F.2d at 297; Conley, supra note 84, at 686-87.

868 F.2d 293 (8th Cir. 1989).


See id.


See id.

See id.


Of course, a software program's ability to process dates and durations lying wholly before the millennium can be tested without such simulation.


See id.

See id.

See id.

This scenario also may lead to fraudulent or negligent misrepresentation claims.

A reasonable alternative statute might include only January 1, 2000 as the critical midnight crossing. Failure to prepare for this crossing should create a rebuttable presumption of negligence. Failure to test one's ability to cross the other two dates could be reasonably left as evidence of negligence (not giving rise to a burden-shifting presumption).

Related Browsing

Copyright 1999 Richmond Journal of Law & Technology