University of Florida Journal of Law & Public Policy

Volume 19 | Issue 1 Article 7

2008

Don't Stop the Music: No Strict Products Liability for Embedded Software

Seldon J. Childers

DON'T STOP THE MUSIC: NO STRICT PRODUCTS LIABILITY FOR EMBEDDED SOFTWARE*

*Seldon J. Childers***

I. INTRODUCTION 126

II. HISTORY AND POLICY RATIONALES FOR STRICT PRODUCTS LIABILITY 129
A. *A Brief History of the Doctrine of Strict Product Liability* 130
B. *The Policy Rationales for Strict Products Liability* 134
C. *The Legal Background of Software and Tort Law* 140

III. THE ARGUMENT AGAINST EXTENDING STRICT PRODUCTS LIABILITY TO SOFTWARE 152
A. *Extending Strict Liability to New Realms is Inconsistent with Tort Reform* 152
B. *The "Stack" Problem Posed by Component Software Supplier Liability* 153
C. *Embedded Software is Not Inherently Different than Other Types of Software* 155
D. *Strict Liability Should Apply Only to Manufacturing Defects, and not to Software* 158
E. *Software—at the Current State of the Art—is an Essential but Unavoidably Unsafe Product Category Deserving of Exemption from Strict Liability* 161
1. Necessary Products Facing Potentially Industry-Swallowing Liability Should be Exempt from Strict Products Liability 162
2. Unavoidably Unsafe Products Like Software Should be Exempt from Strict Products Liability 167
3. The Software Industry is Immature and Should be Nurtured, Not Burdened with Strict Liability 172

IV. ALTERNATIVES TO STRICT PRODUCTS LIABILITY 175

---

* Editor's Note: This Note received highest honors in the *University of Florida Journal of Law & Public Policy* Open Writing Competition in Summer 2007.

** Juris Doctor Candidate, 2008, University of Florida Levin College of Law; software developer and management consultant, 1993-2004; jchilders98@gmail.com. The author would like to thank professor Lyrissa Lidsky for all her encouragement, assistance, and support.

A. *Ordinary Negligence* 176
B. *Design and Warnings Defects* 177
C. *Product Warranties (Express and Implied)* 178
D. *Professional Malpractice Toward Software Developers* 178
E. *Hardware-Based Products Liability* 179
F. *Misrepresentation (Negligent and Fraudulent)* 180
G. *Proof of Causation* 181

V. CONCLUSION AND RECOMMENDATIONS 182

I. INTRODUCTION

Joseph Birdsong cannot hear the music. On May 19, 2006, as a representative for his class, he filed a class-action suit against Apple, claiming that his hearing had been harmed due to his iPod's "defectively" high music volume levels.[1] The suit followed alarming media reports in early 2006 that Apple Computer's blockbuster iPod personal music device could cause hearing damage and even deafness when played at high levels of volume for long periods of time.[2] Apple responded quickly to the bad press. The firm made a new version of the iPod's internal software available to its customers, who could easily install the new software free of charge from the Internet. The improved software allows parents (or any user) to set limits on how loud the music player's volume control can be

1. Joseph Birdsong v. Apple Computer, Inc., No. 06-02280 at 11 (N.D. Cal. filed May 19, 2006). The *Birdsong* amended complaint does not allege that Birdsong or any of the class members were injured by the device. Rather, the complaint alleges that class members would not have bought the device in the first place had they been warned of the risk of hearing damage. Therefore, the claim would appear to be for economic losses and not for personal injuries. However, this Note engages in the hypothetical that the suit was founded in a personal injury action.

2. *See, e.g.*, Erika Morphy, *IPod User Sues Apple for Potential Hearing Damage*, TECH NEWS WORLD, Feb. 3, 2006, *available at* http://www.technewsworld.com/story/48666.html?welcome=1203802395 ("Two weeks ago, the non-profit House Ear Institute launched a consumer awareness campaign aimed at teens and young adults to highlight the dangers of improper use of [iPod] earbuds"); *Pete Townshend Warns IPod Users*, BREITBART.COM, Jan. 4, 2007, *available at* http://www.breitbart.com/article.php?id=D8ETS6S00&show_article=1 ("guitarist Pete Townshend has warned iPod users that they could end up with hearing problems as bad as his own if they don't turn down the volume of the music they are listening to on earphones."); Matthew Erikson, *Listen Carefully*, FORT-WORTH STAR-TELEGRAM, Dec. 9, 2007, at D5 ("Public health groups have made sundry warnings about the perils of listening to iPods at high volume via earbuds"); *Crank it Down*, COLUMBUS DISPATCH, 2007 WLNR 14759627, Aug. 1, 2007 ("with the popularity of iPods and other personal music players, the [hearing loss] problem is likely to get worse [for] children and teenagers especially").

set.[3] Still, if Birdsong and his class's hearing were found to have been damaged by the iPod's original internal software, and the software was found to have been improperly designed (i.e., defective), then the majority of commentators would extend the strict products liability doctrine to embedded software like that in the iPod and hold Apple liable for all damages, regardless of whether Apple had actually been negligent or not. Presumably, Apple could then seek indemnity from any component software developer who participated in development or whose software was used in the iPod. Alternatively, these component software developers might be jointly liable, regardless of whether the component developer even knew its code was used in the device.[4]

The *Birdsong* case is a herald of things to come. Software is everywhere. As one looks around a room today, one's gaze almost certainly falls on a myriad of electronic products, all powered by complex electronic software.[5] Just to name a few that might be within the reader's view right now: cell phones, cordless phones, cable set-top boxes, TiVO, stereo system, clock radio, digital camera, MP3 player, handheld game, all-in-one remote, refrigerator, microwave oven, electronic thermostat, security system, smoke detector, and of course, personal computers. As computer processors shrink in size, manufacturers find more ways to include them in otherwise ordinary consumer products, and where a computer goes, software follows. Automobiles increasingly manage every driving action from steering to braking through a central computer system.[6] Airplanes avoid colliding with each other as a result of complex software calculations. Many people have an even more intimate relationship with software: software-driven, life-sustaining medical devices are implanted in their bodies.[7] Because software has become such an intimate part of almost every device we rely on, it is practically certain that people will be injured and even killed by defectively designed software.

---

3. *See* Christian Toto, *Ear Pollution: High Decibels can Damage Hearing*, WASH. TIMES, Sept. 5, 2006, at B1 (stating "[Apple Computer] announced in March a software update for iPod and related gadgets allowing users to set their own volume limits . . . .").

4. This is the strict liability standard in several jurisdictions. *See infra* Part II.C.

5. *See, e.g.*, John G. Spooner, *Embedded Chips Swarming Consumer Goods*, CNET NEWS.COM, Apr. 10, 2001 (indicating that a microprocessor is associated with almost every part of an automobile, among other electronic products), http://news.com. com/2100-1040-255617.html.

6. *See id.*

7. *See, e.g.*, David Malakoff, *Software May Improve Utility of Implants for Deaf*, NAT'L PUB. RADIO, Oct. 17, 2005 (noting that Duke University researchers use software to make cochlear hearing implants more suitable for hearing speech and music) (Audio transcript, *available at* http://www.npr.org/templates/story/ story.php?storyId=4961269).

Judges and many legal academics have disagreed about how to resolve software products liability cases. Almost universally, judges have refused to apply strict products liability to software, usually by finding that software is not a "product."[8] Almost equally universal, academics have called for an end to what they perceive as a protectionist era for software and call instead for the application of strict products liability to the industry.[9] Academics contend that: (a) the software industry, featuring billion-dollar behemoth corporations, is no longer a "fledgling industry" in need of protection from strict liability;[10] (b) the incentives created by strict liability force software makers to take greater care when designing their products and thereby make consumers safer;[11] and (c) the difficulty and expense to plaintiffs of proving software defects justifies strict liability for software, just as it does for other types of mass-market commercial products.[12] These arguments are so common in academic legal literature

---

8. Usually, courts find that software is a service, not a product, for purposes of tort liability (although courts have found software to be good under the Uniform Commercial Code (U.C.C.). *See infra* text accompanying note 109).

9. For example, the Products Liability Design and Manufacturing Defects database concludes, "[S]trict liability may be applicable to the seller of off-the-shelf software, and certainly against the manufacturer of software-controlled equipment[.]" LEWIS BASS, PROD. LIAB.: DESIGN AND MFG. DEFECTS § 2:18, 4 (2d ed. 2005). *See also* David G. Owen et al., *Publications and Products Liability*, 141 PRODUCTS LIABILITY ADVISORY 1 (Nov. 2000) (stating "the commentators widely favor the application of products liability theories [for personal injuries caused by defective software]. . . . When defective software proximately results in personal injury, there appears to be no good reason not to apply normal products liability principles."). *See also* Frances E. Zollers et al., *No More Soft Landings for Software: Liability for Defects in an Industry That Has Come of Age*, 21 SANTA CLARA COMPUTER & HIGH TECH. L.J. 745, 771 (2005) (noting "We are unmoved by the argument that imposing strict liability will stifle innovation, especially because we are focusing on a segment of the industry—software that foreseeably causes physical harm when defective—rather than the entire software industry.").

10. *See, e.g.*, Zollers et al., *supra* note 9, at 746.

> [T]he software industry is no longer in its infancy. Its development has moved out of garages and into corporate offices. It has matured to become a dominant sector of the economy. Consequently, it is appropriate to consider liability for defective software in the same light as liability for defective automobiles, pharmaceuticals, and other products.

See also id. at 756 (remarking "the [software] industry is in a position to and should be made to absorb the cost of harm occasioned by defects.").

11. *See id.* at 770 (explaining "a strict liability standard increases the liability ex ante and will . . . [i]ncrease the incentives to manufacture and distribute [software] without defects that cause harm.").

12. *See id.* at 782 (reasoning "it [is] sound public policy to have a strict liability regime [for plaintiffs] . . . , rather than putting injured parties to the task of proving negligence.").

that they have been noted in the Restatement (Third) of Torts: Products Liability.[13]

This Note challenges academic orthodoxy and presents six reasons to exempt software from the doctrine of strict products liability. In Part II, this Note considers the history of strict products liability, the policy rationales undergirding the doctrine, and the legal background of tort liability as applied to software. Part III demonstrates why the policies do not make sense when applied to software. Two special considerations are discussed: liability for the "stack" (component software supplier liability), and the reasons why, for purposes of tort liability, embedded software should be analyzed as distinct from the rest of the device in which it is embedded, and no differently than any other non-embedded category of software. Part IV briefly describes each of the commonplace alternative theories for recovery that may be available to plaintiffs injured by commercial mass-market products containing defective software. Finally, Part V concludes with a recommendation for courts and legislatures.

II. HISTORY AND POLICY RATIONALES FOR STRICT PRODUCTS LIABILITY

Until the 1950s, the tort doctrine of strict or absolute liability was reserved for a narrow range of concerns like ultra-hazardous activities,[14] the keeping of wild animals, and the distribution of adulterated foods.[15] Strict liability is liability without fault; even if a person exercises reasonable care (or even superlative care), he remains liable for injuries caused as a result of his activity. So it is that a manufacturer of blasting caps can be held liable when children are injured, but a specific maker of the cap is not proven or even known.[16]

---

13. RESTATEMENT (THIRD) OF PRODUCTS LIABILITY § 19 cmt. d (1998) (noting "for purposes of strict products liability in tort[,] . . . . [n]umerous commentators have discussed the issue and urged that software should be treated as a product.").

14. *Fletcher v. Rylands* is the famous case that establishes absolute liability for ultra-hazardous activities. Fletcher v. Rylands, 1 L.R.-Ex. 265, 266 (1866).

15. *See* MacPherson v. Buick Motor Co., 217 N.Y. 382, 389 (1916) (removing privity requirement from duty for negligence; authored by Justice Benjamin Cardozo).

16. *See* Hall v. E. I. Du Pont de Nemours & Co., Inc., 345 F. Supp. 353, 372 (E.D.N.U. 1972) (holding that all makers of blasting caps were liable for injuries to children when no particular maker could be identified). *See also* Pittsburg Reduction Co. v. Horton, 113 S.W. 647 (Ark. 1908). A ten year old boy found discarded blasting caps and carried them home. *Id.* at 648. His father, familiar with blasting caps, saw him playing with them, but did not take them away from the boy. *Id.* About one week later, the boy carried them to school and traded them to a classmate. *Id.* But

A key 1963 decision[17] triggered the rapid extension of strict tort liability to product manufacturers for personal injuries (generally to consumers) caused by defective products. Strict products liability is the most controversial application of strict liability.[18] Part of the controversy undoubtedly arises because strict liability violates the core tort principle that liability should be based on fault, and results derived from application of absolute liability sometimes challenge fundamental notions of justice and fair play. The doctrine has produced harsh results at times, and tort reform efforts have focused more on narrowing the strict products liability doctrine than on any other area.[19]

This section presents a brief history of the development of and the major rationales for the modern strict products liability doctrine. The background explains why strict products liability should not be applied to software.

A. *A Brief History of the Doctrine of Strict Product Liability*

"It is to the public interest to discourage the marketing of products having defects that are a menace to the public."[20] While the strict products liability doctrine has roots in implied warranties of contract, Justice Roger Traynor first recognized the doctrine in his famous concurrence[21] in the case of *Escola v. Coca Cola Bottling Company*.[22] Traynor advocated extending the existing doctrine of strict liability for adulterated food products to all manufactured products. He cited the manufacturer's ethical responsibility to exercise control that the consumer lacks, the "risk spreading" ability of the manufacturer to pass the cost of insuring against

---

for the parents' having known of the danger and having allowed the boy to keep the caps, the manufacturer would have been held strictly liable when injuries occurred. *Id.* at 649.

17. Greenman v. Yuba Power Prod., Inc., 377 P.2d 897, 900 (Cal. 1963).

18. *See* RICHARD A. POSNER, ECONOMIC ANALYSIS OF LAW 182 (6th ed. 2003) (explaining "The most controversial area of strict liability today is liability, now called strict in most states, for personal injuries (mainly to consumers) caused by defective or unreasonably dangerous products.").

19. The significance of tort reform in this area is perhaps best exemplified by having received its own Restatement. The 1998 publication of the Restatement (Third) of Torts: Products Liability "largely abandons the doctrine of 'strict' products liability for design and warnings cases, which comprise the bulk of products liability litigation." DAVID G. OWEN, PRODUCTS LIABILITY LAW 248 (2005). *See also* Kathy Gill, *Tort Reform-State Recap*, ABOUT.COM, Feb. 10, 2005, http://uspolitics.about.com/od/healthcare/a/01_tort_reform.htm.

20. Escola v. Coca Cola Bottling Co., 150 P.2d 436, 440-41 (Cal. 1944) (Traynor, J., concurring).

21. Owen calls Justice Traynor's concurrence in *Escola* "perhaps the most renowned concurring opinion in all of American tort law . . . ." OWEN, *supra* note 19, at 253.

22. *Escola*, 150 P.2d at 436.

liability along to consumers of the product, and the social interest in driving unsafe goods from the market:

> It is evident that the manufacturer can anticipate some hazards and guard against the recurrence of others, as the public cannot. Those who suffer injury from defective products are unprepared to meet its consequences. The cost of an injury and the loss of time or health may be an overwhelming misfortune to the person injured, and a needless one, for the risk of injury can be insured by the manufacturer and distributed among the public as a cost of doing business. It is to the public interest to discourage the marketing of products having defects that are a menace to the public.[23]

It is significant that the defective product in *Escola* was a glass bottle, one of perhaps millions of identical units stamped out in an opaque, repetitive, and highly automated manufacturing process. In such a context, the difficulties facing plaintiffs attempting to prove negligence are manifest, the ability of the manufacturer to "spread" liability is at its pinnacle,[24] and the policy arguments for strict products liability are therefore at their strongest. The last sentence quoted above is also noteworthy: here, at the very genesis of the strict products liability doctrine, Traynor reasoned that defectively dangerous products should be driven from the market altogether—presumably under the heavy burden of strict liability.

Over the next decade and a half, the *Escola* concurrence proved persuasive, and in 1961, the American Law Institute (A.L.I.) adopted a draft rule recognizing strict liability for sellers of food products. Over the next few years, the draft was amended several times to include other products that were considered to be similar to food products. In 1963, the California Supreme Court decided the landmark case of *Greenman v. Yuba Power Products*[25] and extended strict tort liability to a power tool manufacturer (i.e., beyond the realm of food and "personal" items). The A.L.I. responded in 1964 through a new draft of section 402A of the Restatement (Second) of Torts, extending strict products liability to any

---

23. *Id.* at 440-41 (Traynor, J., concurring).

24. Coca-Cola is a sophisticated, established business well able to actuarially calculate the costs of potential liability for injuries and to easily distribute those costs into a small fraction of the price of each of the millions of units sold.

25. Greenman v. Yuba Power Prod., Inc., 377 P.2d 897, 901 (Cal. 1963) (explaining that a man injured by a shop saw, which his wife purchased for him, may recover for his injuries based on reasonable expectation of safety even without privity).

commercial manufacturer of a defective product.[26] Within just twelve years, forty-one states had adopted the section, frequently citing *Greenman* as a seminal case and citing the widespread influence of Traynor's concurrence in *Escola*.[27]

Due to section 402A's roots in food product safety, a doctrine founded on an implied warranty rationale,[28] the Restatement (Second) defined defectiveness not in terms of any particular feature of the product but as any product quality which defeats a consumer's reasonable expectation of safety—the so-called "consumer expectations test."[29] This test premises tort liability on the idea that a defectively dangerous product defeats the consumer's reasonable expectation that the product is safe for ordinary use. Section 402A is widely recognized as one of the most influential sections in any Restatement, and its influence has swept beyond the borders of the United States.[30]

The Restatement (Second) section 402A is a victim of its own success. Beginning in the 1970s, manufacturers and insurers began to push back against the growing tide of products liability litigation. A successful reform movement arose,[31] fueled in part by an intrinsic ambiguity bedeviling the otherwise popular consumer expectations test. For example, it could be possible that consumers simply expect a power saw to be dangerous to use. If so, what are the reasonable expectations for safety for such a product? The inherent ambiguity of the test allows judges and juries to reach almost any conclusion on the same facts and fails to provide much guidance.[32]

---

26. RESTATEMENT (SECOND) OF TORTS § 402A (1965).

27. *See* MARK A. GEISTFELD, PRINCIPLES OF PRODUCTS LIABILITY 16 (2006).

28. *See* RESTATEMENT (SECOND) OF TORTS § 402A, cmt. f (1965).

> The basis for the rule is the ancient one of the special responsibility for the safety of the public undertaken by one who enters into the business of supplying human beings with products which may endanger the safety of their persons and property, and the forced reliance upon that undertaking on the part of those who purchase such goods.

Id.

29. *See id.* cmt. g. (explaining that strict liability "applies only where the [defective] product is, at the time it leaves the seller's hands, in a condition not contemplated by the ultimate consumer, which will be unreasonably dangerous to him.").

30. It has heavily influenced legislation in the European Economic Community and Japan, among others. GEISTFELD, *supra* note 27, at 18.

31. *See* OWEN, *supra* note 19, at 24-25.

32. *See id.* at 299 (citing PROSSER & KEETON ON TORTS § 99, at 698 (footnote omitted)).

Authorities and commentators have found that this flexibility makes the test reflect a standard more like ordinary negligence than true strict liability because plaintiffs (by way of consumer "expectations") must persuade the jury that the manufacturer acted unreasonably in order to obtain a favorable verdict.[33] Others feel the flexibility makes the consumer expectations test pro-consumer.[34] Still others suggest that the consumer expectations test is fully analogous to the newer test for product defects found in the Restatement (Third) of Torts: Products Liability, which gained favor in some jurisdictions as a replacement for the consumer expectations test.[35]

In 1998, the A.L.I. published the Restatement (Third) of Torts: Products Liability, giving the products liability doctrine its very own Restatement. This new Restatement arrived in the midst of the political and legislative tort reform movement. The new Restatement explicitly rejects the consumer-expectations test[36] and instead adopts a risk-utility test, which requires a plaintiff to prove the existence of an alternative reasonable design (or warning) to establish that a product is defective. This test balances the manufacturer's reasonable efforts at achieving product safety against the cost of discovering and preventing certain injuries. The risk-utility test looks like ordinary non-strict negligence because it requires a manufacturer to exercise reasonable care when selecting a product design (and by definition to avoid unreasonably unsafe designs).

The current status of products liability generally, and strict products liability particularly, is in flux, subject to change, and somewhat confusing.[37] However, if a trend can be described at all, it is that tort law seems to be narrowing rather than extending the liability of manufacturers.[38] So it is puzzling that commentators continue to advocate for a major expansion of the harshest form of products liability—strict

---

33. *See* OWEN, *supra* note 19, at 299-301 (discussing how the consumer expectations test has become ambiguous and difficult to apply; reasonable expectations of consumers similar to principles relied upon to establish liability in non-strict liability cases).

34. *See, e.g.*, Halliday v. Sturm, Ruger & Co., Inc., 792 A.2d 1145, 1154 (Md. 2002) (finding the Restatement (Second) to be an important pro-consumer advance and rejecting the Restatement (Third)'s risk-utility test as a retrogression).

35. *See* James A. Henderson, Jr. & Aaron D. Twerski, *Achieving Consensus on Defective Product Design*, 83 CORNELL L. REV. 867, 872 (1998).

36. *See* RESTATEMENT (THIRD) § 2 cmt. g (1998) (noting that consumer expectations are not determinative of tort liability for design and warning defects).

37. *See* Henderson & Twerski, *supra* note 35, at 871 (stating "the rhetoric of products liability law is, undeniably, a mess.").

38. For a more complete discussion of the tort reform movement, see OWEN, *supra* note 19, at 24-25.

products liability—into the previously untouched realm of computer software, a peculiar result at odds with the legal development of law in this important area. It is unclear why these commentators wish to leapfrog the non-strict forms of products liability and to jump directly to the most onerous form of the doctrine.

B. *The Policy Rationales for Strict Products Liability*

The rationales for imposing strict liability for commercial products take two forms. The first is a set of moral arguments, based on fairness, positing that manufacturers are ethically responsible to innocent consumers who have been harmed because the consumers had a reasonable expectation that the manufacturer would supply a safe product.[39] The second group of rationales is based on economic arguments or efficiency.[40] For example, it is argued that manufacturers are best able to insure against losses and to spread the cost of such insurance among all the consumers who purchase their products, and that strict liability creates socially desirable economic incentives for manufacturers to produce safer products. As discussed below, these traditional policy rationales are not a good fit in the area of commercial computer software.

The idea that manufacturers are ethically required to be the gatekeepers of consumer safety is fundamental to tort-based products liability. Manufacturer attention to safety is seen as a moral imperative due to the inherent inequality of the relative positions of power held by manufacturer and consumer and the manufacturers' ability to control circumstances that the consumer cannot:

> Manufacturers have much greater control over product safety than consumers in many ways: the manufacturer, not the consumer, conceives and determines the balance of utility and safety in the product; the manufacturer alone determines how much quality control to use to prevent and screen out errors in production; the manufacturer has practical access to far greater safety information than consumers, and it alone determines how much and in what manner to share such information with the consumers who need it; and the manufacturer alone decides what promises about product

---

39. *See* David G. Owen, *The Moral Foundations of Products Liability Law: Toward First Principles*, 68 NOTRE DAME L. REV. 427, 468-69 (1993) (ethically, manufacturers owe a paternalistic duty to potential accident victims who are entitled to a fair measure of product safety).

40. *Id.* at 457 (the principles of utility and efficiency seek to deter accident-producing conduct that is on balance wasteful for society).

> safety to make to consumers to induce them to buy the product. In sum, the manufacturer's initial power over product safety—risk control—is enormous; by comparison, the consumer's initial control of product risk is almost trivial. Thus, there is a gross inequality in the initial distribution of risk control between the maker and the user.[41]

This moral argument portrays manufacturers as faceless entities with deep pockets and consumers as completely dependant on manufacturers to disclose potential product dangers. This scenario describes *Escola* perfectly, where consumers reasonably relied, without thinking, on a giant bottling company to produce a safe product. Presumably Coca-Cola wields sufficient market power to control every aspect of the manufacture and distribution of its cola product, if it so chooses, and can therefore be efficiently motivated by threat of liability to exercise such control through its contracts with its distributors and retailers. This rationale is not as compelling, however, in the case of manufacturers: (1) who produce socially desirable products that cannot, through reasonable effort be made any safer; (2) who face risks that are unforeseeable; (3) who wield significantly less market power than does Coca-Cola; or (4) who may not have any ability to dictate how their products are handled after being released into the stream of commerce.

The case of *Lechuga v. Montgomery* is frequently cited[42] for its enumeration of the policy justifications for strict products liability.[43] Some

---

41. *Id.* at 452.

42. Westlaw returns 68 citing references (as of Sept. 2007).

43. Lechuga, Inc. v. Montgomery, 467 P.2d 256, 261-62 (Ariz. Ct. App. 1970) (Jacobson, J., concurring) (stating that lessors of automobiles are just as strictly liable for product injuries as is the original manufacturer). The Restatement (Third) of Torts also enumerates seven policy rationales that justify imposition of strict products liability. The Restatement (Third) factors are more general than the *Lechuga* factors, and they are also well-represented by *Lechuga.* The Restatement (Third) factors are as follows:

> (1) the public interest in life and health; (2) the invitations and solicitations of the manufacturer to purchase the product; (3) the justice of imposing the loss on the manufacturer who created the risk and reaped the profit; (4) the superior ability of the commercial enterprise to distribute the risk of injury as a cost of doing business; (5) the disparity in position and bargaining power that forces the consumer to depend entirely on the manufacturer; (6) the difficulty in requiring the injured party to trace back along the channel of trade to the source of the defect in order to prove negligence; and (7) whether the product is in the stream of commerce.

of the justifications are as follows:

> (1) The manufacturer can anticipate some hazards and guard against their recurrence, which the consumer cannot do.
> (2) The cost of injury may be overwhelming to the person injured while the risk of injury can be insured by the manufacturer and be distributed among the public as a cost of doing business [(i.e., risk spreading)].
> (3) It is in the public interest to discourage the marketing of defective products.
> (4) It is in the public interest to place responsibility for injury upon the manufacturer who was responsible for its reaching the market.
> (5) That this responsibility should also be placed upon the retailer and wholesaler of the defective product in order that they may act as the conduit through which liability may flow to reach the manufacturer, where ultimate responsibility lies.
> (6) That because of the complexity of present-day manufacturing processes and their secretiveness, the ability to prove negligent conduct by the injured plaintiff is almost impossible.[44]

The first policy argument expresses the socially desirable objective that tort law should create economic incentives for manufacturers to design safe products. Implicit in this objective is the notion that (a) the particular risks are foreseeable, and (b) the products can be made safe, that is, they are not unavoidably unsafe. Automobiles are commonly cited as an

---

RESTATEMENT (THIRD) OF TORTS § 19 cmt. a, n (1998).

44. *Lechuga*, 467 P.2d at 261-62 (internal citations omitted). The seventh and eighth *Lechuga* rationales rely on decisions extending strict tort products liability by analogy to non-personal injuries to chattels, so called "economic losses." These cases are questionable law. For the interested reader, the last two rationales are as follows: (7) "the consumer does not have the ability to investigate for himself the soundness of the product," and (8) "the consumer's vigilance has been lulled by advertising, marketing devices, and trademarks." Santor v. A & M Karagheusian, Inc., 207 A.2d 305, 311 (N.J. 1965) (extending strict products liability to a case where a man purchased a carpet inferior to what he was promised); Nalbandian v. Byron Jackson Pumps, Inc., 399 P.2d 681, 681 (Ariz. 1965) (extending strict liability to a pump manufacturer who sold a defective pump and property damage resulted). Both cases analogize the contract warranty of fitness of purpose to encompass strict tort product liability for non-personal injuries. *Lechuga*, 467 P.2d at 261-62. Since 1965, many courts, including the U.S. Supreme Court, have rescued contract law from tort law by reaffirming the economic loss rule, which would forbid such verdicts. *See* E. River S.S. Corp. v. Transamerica Delaval, Inc., 476 U.S. 858, 867 (1986) (adopting as part of admiralty law the "economic loss" rule, which denies the purchaser of a defective product a tort action against the seller or manufacturer for purely economic losses sustained as a result of the product's failure).

example of a product category that has become much safer because of the pressure created by potential strict products liability.

But in the case of software, as with any disruptive new technology, the potential risks are not always obvious or foreseeable.[45] The cost to the manufacturer of acquiring the knowledge of risks must be efficient, i.e., economically reasonable. Thus, there is a point of diminishing returns where the costs of excessive testing for defects makes a product unmarketable.

The second policy argument requires an examination of the "state of the art": could the manufacturer have cost-effectively designed a safer product? If it is not cost-effective for the manufacturer to design a safer product, then the alternatives are for consumers to live with the risk or for the product to be removed from the market altogether. In the case of software, it is well known that software defects (bugs) are ubiquitous and possibly unavoidable.[46] If particular risks of injury from software are unforeseeable, and if current software technology is unavoidably unsafe, then the economic incentives posed by strict liability cannot rationally affect software producers' behavior. Imposing strict products liability might therefore over-deter software producers to the detriment of society which would lose access to socially desirable software products.

The second and third policy arguments are economic in nature: (1) the manufacturer can afford the cost of injury more than can the consumer can, and (2) it maximizes social utility to discourage potentially injurious defective products. This is the risk-spreading rationale, encompassing the body of Traynor's economic arguments in *Escola*.[47] Under these arguments, two assumptions must be met before the externality of strict liability can be efficiently imposed on a market. First, a mature market for a product must exist, one which is sufficiently sophisticated to actuarially calculate the cost of potential liability and include it in the product price, thereby distributing it among all consumers of the product.[48] Insurance sellers will not provide coverage to a manufacturer if such actuarial calculations cannot be made.[49] It seems obvious that, in order to obtain the

---

45. This Note discusses unavoidably unsafe products at length. *Infra* Part III.E.

46. *See, e.g.*, THOMAS J. CARTIN, PRINCIPLES AND PRACTICES OF ORGANIZATIONAL PERFORMANCE EXCELLENCE 61 (1999) (stating it is "a generally accepted fact that all software programs, when used, will contain bugs.").

47. Escola v. Coca Cola Bottling Co., 150 P.2d 436, 440-41 (Cal. 1944) (Traynor, J., concurring).

48. *See* GEISTFELD, *supra* note 27, at 154-55 (discussing types of products having the "actuarial characteristics" necessary for insurability).

49. *See id.*

data needed to make these calculations, the industry must have existed for long enough that there are a sufficient number of similar competitive products against which the instant product's design can be compared. Second, for strict liability to make sense, the net social cost of potentially uncompensated injuries plus society's desire to discourage the industry must exceed the sum of the excess benefits obtained by society from the industry plus society's desire to protect or nurture the market.[50] Society has encouraged socially desirable markets via protection from the most onerous forms of tort liability. In spite of what the proponents of strict products liability for software would argue, the software industry is too immature to afford strict products liability in tort and risks being over-deterred.[51] If the ubiquitous incorporation of software into almost every conceivable product benefits society in excess of the costs, including the cost of uncompensated injury, then this rationale must be discounted in favor of software producers.

The fourth policy argument is that the public interest requires placing liability for injury on the manufacturer who was responsible for creating the product that will eventually reach the market. In the software market, particularly embedded software,[52] it may be difficult, if not impossible, to identify which developer should be liable. Holding all participating developers liable raises the spectre of uncontrolled absolute liability. Furthermore, many software components are non-commercial in nature.[53] Non-commercial activities have generally been held outside tort and warranty strict liability,[54] further complicating the analysis.

> The insurability of a risk depends on the quality of data regarding the probability and the severity of loss. [While] [i]nsurers usually do not have difficulty collecting such data for construction or manufacturing defects .... [u]nforseeable risks pose a hard actuarial problem, making the provision of insurance much more difficult, if not impossible.

Id.

50. Expressed algorithmically: (PLu + S--) > (Ue + S++), where PLu is the probability and extent of uncompensated injury, S-- is the extent to which society disapproves of the product (i.e., "sin products"), Ue is the excess benefit or "utility" obtained by society over and above the actual cost of the product to consumers, and S++ is the extent to which society wishes to encourage the industry's growth.

51. *Infra* Part III.E.3 (developing the argument more fully).

52. *Infra* Part III.B (defining and discussing embedded software that is part of the "stack").

53. Non-commercial component include free and open-source software, in particular.

54. For example, the Restatement (Second) of Torts section 402A(1) begins "[o]ne who sells any product . . . ." Therefore, under section 402A, there is no liability without a "sale" (i.e., commerce).

Finally, this argument turns a manufacturer into an insurer. The idea is that it is more equitable for the product manufacturer, who stood to benefit from the sale of the product, to be responsible for the cost of injuries resulting from that commercial activity. This is efficient as long as (a) the manufacturer can afford to purchase insurance or to self-insure, and (b) insurance is available to the manufacturer. As discussed further in Part III.D.3, the vast majority of contemporary software developers are small firms or individuals. Additionally, liability insurance is likely to be unavailable or abnormally expensive due to the immaturity of the software industry. Inasmuch as placing liability on software producers is therefore likely to over-deter socially desirable products, it would be inefficient for society to do so.

The fifth *Lechuga* policy argument is that retailers and distributors should share liability for products subject to strict liability. In addition to manufacturers, retailers and other non-manufacturing participants in the stream of commerce may also be strictly liable for defective products.[55] While the trend seems to move toward limiting this extension of strict liability to the original manufacturer, most jurisdictions still recognize this form of broad strict liability. Even in jurisdictions that have implemented tort reform in this area, retailers and distributors may still be reachable even without fault when the original product manufacturer is unreachable[56] or insolvent.[57] In the case of software, it seems self-evident that holding distribution participants such as Internet Service Providers (ISPs) strictly liable for injuries caused by defective software is a luxury that society cannot currently afford.[58]

The sixth policy argument is perhaps the most common in the literature discussing strict products liability. Some plaintiffs will find it extremely difficult to prove causation in cases involving complicated products. Consequently, strict liability works like a form of *res ipsa loquitor*,

---

55. *See* OWEN, *supra* note 19, at 958-61.

56. *See, e.g.*, Malone v. Schapun, Inc., 965 S.W.2d 177, 185-86 (Mo. Ct. App. 1997) (holding the "innocent seller statute" did not apply to retailer when the court had no jurisdiction over manufacturer and distributor because the plaintiff had earlier settled with them).

57. *See, e.g.*, Marcon v. Kmart Corp., 573 N.W.2d 728, 730 (Minn. Ct. App. 1998) (finding 0% liability assigned to the retailer and 100% to the manufacturer; however, the retailer was held strictly liable when the court found that the manufacturer was bankrupt).

58. For example, consumers can download replacement software for their iPod which is not from Apple. Suppose this software played music too loud and damaged hearing. The original "manufacturer" is in reality a group of one or more hobbyists and not a commercial enterprise able to self-insure against tort judgments. Should the innocent plaintiff then have recourse against the ISP (i.e., analogizing the ISP to a fault-free retailer)? *See* iPodLinux Project, Main Page, http://ipodlinux.org/Main_Page (last visited Sept. 18, 2007).

relieving the plaintiff of the burden of proving how or when negligence occurred.[59] But, as we shall see in Part III.C, this rationale is most persuasive when considered in the context of manufacturing defects, and not as persuasive when it comes to design or warning defects. It is true that modern software is astonishingly complex.[60] Still, the process of creating software has no analogue to a traditional manufacturing process, and software defects do not appear in "one in a million" configurations which occur in cola bottles. Finally, in spite of the technical complexity of software, it is not a foregone conclusion that proof of negligence is impossible or even necessarily very difficult.[61]

C. *The Legal Background of Software and Tort Law*

There is very little law on tort liability for software and even less in the area of software products liability. The Second and Third Restatements of Torts offer guidance that is at best ambiguous and at worst conflicting. To the extent that software can be considered a "good," the Uniform Commercial Code (U.C.C.)[62] lends product warranty law to the discussion. Available case law discussing losses caused by software has focused primarily on economic damages and has applied ordinary negligence principles, generally casting software as a service.[63] Proponents of strict

---

59. "One argument favoring strict liability is that the unavailability of evidence in some cases renders the plaintiff unable to establish what may well have been the defendant's actual negligence." RESTATEMENT (THIRD) OF TORTS § 17 cmt. a (2005).

60. *See* Uniform Computer Information Transactions Act (UCITA) (2002) § 403 cmt. 3a (2002) [hereinafter UCITA] ("[software can contain] over ten million [interoperating] lines of code or instructions . . . . [c]ontrasted with a commercial jet airliner that contain[s only] approximately six million parts. . . .").

61. *Supra* Part IV.G (discussing the evidentiary issue in more detail). One interesting development in the software community is a form of unsolicited peer review. So called "white hat hackers" spend their time testing software and publicly reporting potential security flaws. *See* Wikipedia, *White Hat*, http://en.wikipedia.org/wiki/White_hat (defining a white hat hacker) (as of Sept. 18, 2007, 13:50 EST). It is not difficult to imagine that, as software with the potential to cause human injury becomes even more ubiquitous, similar community-based peer review may assist plaintiffs with discovery of defects.

62. There have been attempts to codify the law of software under the U.C.C. in what was first called U.C.C. Article 2b, but which later became the UCITA.

63. If software is a service, then ordinary negligence applies. In many cases, courts have avoided the issue of whether software can give rise to a negligence claim by focusing not on the product but rather on the installation and services provided by the software vendor. *See, e.g.*, Data Processing Servs. Inc. v. L.H. Smith Oil Corp., 492 N.E.2d 314 (Ind. Ct. App. 1986); Micro-Managers, Inc. v. Gregory, 434 N.W.2d 97 (Wis. Ct. App. 1988); *see also* Note, *Computer Software as a Good Under the Uniform Commercial Code: Taking a Byte Out of the Intangibility*

products liability would prefer courts to consider software as a product, bringing products liability law into play. The battle therefore begins with the question of what *is* software: service, good, product, intellectual property, or something else?

The issue of whether tort law considers software to be a product is the threshold question for establishing whether products liability exists. If software is not a product, then the strict products liability doctrine simply does not apply. The distinction is critical: if software is considered a service (and not a product), then established law requires that only ordinary negligence applies and plaintiffs must prove fault.[64] It is not surprising, therefore, that there has been substantial advocacy in the literature for commercial mass-market software to be considered a product under tort law.[65] This section considers how the law defines a "product" and whether that definition can be extended to computer software.

But what is software? One cannot touch it, see it, smell it, or taste it. One cannot move it upstairs, downstairs, across the street, or even across a desk. Yet, it can travel the globe in an instant. One cannot put it in one's hand, a box, or a truck. It takes up no space whatsoever, but it can have more parts than a 747 jetliner.[66] There is neither manufacturing lead time nor any raw materials needed to create it; numerous copies of software can be made instantly for no additional cost based on need. Unlike other manufactured products, software is capable of reproducing itself.[67] Software is created using programming languages (i.e. words). A programmer at work is indistinguishable from a novelist at work. Like a written novel, a software program is wholly the product of the human

---

Myth, 65 B.U. L. REV. 129 (1985); Comment, *The Warranty of Merchantability and Computer Software Contracts: A Square Peg Won't Fit in a Round Hole*, 59 WASH. L. REV. 511 (1984).

64. *See* RESTATEMENT (THIRD) OF TORTS § 19 cmt. a (1998) (stating "[o]nce the era of strict products liability in tort arrived in the early 1960s, liability turned primarily on whether what the defendant distributed was, or was not, a product."). The reporters' note discussing the section states that "only when the complained-of injury was allegedly caused by a defect in something within this Section's definition of 'product' should the defendant manufacturer or seller be strictly liable for the harm caused." *Id.* cmts. a, n.

65. There seems to be nearly universal agreement that software designed on specification for a single customer is a service, not a product. This is an interesting conclusion; however, if software is a service when provided to one customer, what about its nature changes when it is later provided to many other customers so that it is then a product? It would seem the mass-market distinction is not so much about any intrinsic quality of the software at all, but rather only about policy objectives when more customers could potentially be at risk. For a full discussion, see RESTATEMENT (THIRD) OF TORTS § 19 cmt. d (software that was developed specifically for a customer is a service).

66. *See* UCITA, *supra* note 60, § 403 cmt. 3a (noting that modern software is more complex than a commercial airliner).

67. Such software is commonly known as a virus.

mind. If software is stolen, the original owner might not even notice and will continue to enjoy its use. If programmer does notice, recourse is available under copyright law, which also protects the novelist's work. It is no wonder that the distinction over whether software is a good, a product, a service, or something else has given the legal community fits since the introduction of the first commercial software product. Software is a gossamer ship on an ethereal ocean.

Existing case law supports the conclusion that software is not a product. Diligent investigation has revealed no cases where judges have held that software was a product for purposes of tort products liability.[68] The Restatement (Third) concludes the same, stating in the section 19 comments that "[regarding products liability for software,] there are no cases on point on the facts."[69] It is significant that judges, when considering the facts of the cases before them, have so far held the line and refused to find tort product status for software.[70] Still, several cases are repeatedly cited in the literature as "pre-cursors" for software products liability.

A series of cases against aeronautical chart makers[71] established that a printed flight map can be a "product" subject to strict liability.[72] In *Aetna Casualty & Surety Co. v. Jeppesen & Co.*, a plane crashed on the landing approach when the pilots relied on a Jeppesen & Co. chart. The chart was defective because it featured an illustration not drawn to the same scale as all other Jeppesen charts. An expert witness testified that the pilots could have been cognitively confused by the use of the non-standard scale. The court found Jeppesen strictly liable, relying on the products liability section of the Restatement (Second) of Torts section 402A.[73]

This case, and a series of aeronautical chart cases which followed, figure prominently in the argument for software strict liability for two reasons. First, proponents analogize aeronautical charts to software: both

---

68. *See* Owen et al., *supra* note 9 (stating "While the commentators widely favor the application of products liability theories [to software], the case law is nonexistent.").

69. *See* RESTATEMENT (THIRD) OF TORTS § 19 cmt. d (1998).

70. *See, e.g.*, James v. Meow Media, Inc., 90 F. Supp. 2d 798, 818 (W.D. Ky. 2000) (analyzing video game negligence as a speech issue under the First Amendment).

71. For a thorough discussion of the *Jeppesen* series of cases (and other similar cases), see David L. Abney, *Liability for Defective Aeronautical Charts*, 52 J. AIR L. & COM. 323 (1986).

72. *See* Aetna Cas. & Sur. Co. v. Jeppesen & Co., 642 F.2d 339, 341-43 (9th Cir. 1981). Subsequent cases established that the Ninth Circuit really meant "strict"; in one case, Jeppesen merely reprinted publicly available FAA data which turned out to be inaccurate, and the court still held him liable. Brocklesby v. United States, 767 F.2d 1288, 1295 (9th Cir. 1985).

73. *See Jeppesen*, 642 F.2d at 343; *see also* RESTATEMENT (SECOND) OF TORTS § 402A (1965).

are purely functional, intangible, not implicative of the First Amendment, and historically not considered "products."[74] Second, an interesting case from the Ninth Circuit, in dicta, drew an analogy between aeronautical charts (subject to strict products liability) and defective computer software.[75]

This dicta emerged from *Winter v. G.P. Putnam's Sons*[76]—not an aeronautical chart case but a so-called "information" case. In *Winter*, mushroom enthusiasts were badly poisoned[77] after relying on an inaccurate guide book and sued the book publisher under a strict products liability theory. The plaintiffs argued that strict liability should apply to books that provide instruction for inherently dangerous activities, or in the alternative, that the mushroom guide was analogous to aeronautical charts because it was intended for use while engaging in a hazardous activity like flying an airplane.[78] The circuit court found the plaintiff's strict liability argument unpersuasive and "decline[d] to expand products liability law to embrace the ideas and expressions in a book."[79] But, in a remarkable bit of dictum, the court drew a mild analogy between defective computer software and aeronautical charts. The court noted that "[c]omputer software that fails to yield the result for which it was designed may be [analogous to aeronautical charts]."[80]

This lukewarm comment has fertilized a substantial and vigorous crop of theories under which software would be recognized as a product for strict liability purposes. Most commonly, the theorists argue that the *Winter* court meant that software, like the charts at issue in the aeronautical cases, are "information" products. Others would go further and rely on the dictum for the proposition that the court meant that software, like the charts in the cited cases, should be subject to strict products liability.[81]

In spite of its popularity in law reviews, *Winter* ultimately has been a disappointment to proponents of strict products liability for software for three reasons. First, the facts in *Winter* are unusual and the dicta analogizing software to aeronautical maps is so attenuated that it seems

---

74. *See* Zollers et al., *supra* note 9, at 763-64.

75. Winter v. G.P. Putnam's Sons, 938 F.2d 1033 (9th Cir. 1991).

76. *Id.* at 1036.

77. As a result of their culinary misadventure, both plaintiffs became critically ill and had required liver transplants. *Id.* at 1033.

78. *Id.* at 1035-36.

79. *Id.* at 1036.

80. *Id.*

81. The argument is so popular that it is recognized in the Restatement (Third) of Torts. RESTATEMENT (THIRD) OF TORTS § 19 cmt. d (1998).

almost an afterthought. Second, in the sixteen years since *Winter*, no reported case, including in the Ninth Circuit, has undertaken to follow, enhance, or expand upon the *Winter* court's dicta. Finally, the *Winter* court never explained its proposed analogy between aeronautical charts and software. Perhaps in 1991, the *Winter* court thought about flight mapping software of the sort Jeppesen[82] currently sells rather than software categorically when it made the analogy.[83] In any case, the amazing panoply of software today bears little resemblance to a laminated card with a map and chart printed on it. A laminated chart is designed to keep pilots' critical information close at hand for their direct use. Conversely, the software driver in an iPod is designed to translate digital audio information and relay it to the sound hardware—all behind the scenes without any human participation. Beyond noting that both items are "functional," which is far too broad a measure for comparison—a hammer is functional, it seems difficult to identify any closer similarities. At best, *Winter* is an ambivalent "precursor" to strict liability for software. Lacking any new legal development in the sixteen years since *Winter*, and considering the substantial changes in the technological landscape since that time, the dicta may be destined to fade into obscurity.

The Restatements of Torts have done little to clarify the status of software as either software or product. The Second and Third Restatements of Torts largely conflict regarding their respective definitions of "product" for products liability purposes, and the difference is particularly stark in the case of software.[84] Still, both Restatements of Torts have been used to argue for strict products liability for software. Some commentators think the Restatement (Second) provides a broad and flexible definition that might be extended to encompass software as a "product."[85] Contrarily, others think the Restatement (Third) provides a narrower definition that explicitly excludes intangibles, including

---

82. *See* Making Every Mission Possible, http://www.jeppesen.com/wlcs/index.jsp (last visited Sept. 19, 2007).

83. Arguably, map software might implicate the same policy concerns as do laminated maps. *But see* Jennifer L. Phillips, *Information Liability: The Possible Chilling Effect of Tort Claims Against Producers of Geographic Information Systems Data*, 26 FLA. ST. U. L. REV. 743, 765-66 (1999) (arguing that makers of personal G.I.S. mapping systems are perhaps improperly deterred by "the oppressive weight of strict liability" laid upon chart makers, and rather that the novel technology should be encouraged).

84. *See* RESTATEMENT (SECOND) OF TORTS § 402A (1965); RESTATEMENT (THIRD) OF TORTS § 19 (1998).

85. *See* David W. Lanetti, *Toward A Revised Definition Of "Product" Under The Restatement (Third) of Torts: Products Liability*, 55 BUS. LAW. 799, 807-08 (2000).

software.[86] The flexibility of the Restatement (Second) was not obvious in its initial drafts, which originally considered only foodstuffs to be products for the purposes of strict liability. The flexibility comes instead from the lack of any definition–the Restatement (Second) does not explicitly define a product and instead just offers examples in the section's comments.[87] As the final draft of § 402A was completed in response to concerns from various constituencies, the comments were extended to include a non-exclusive list of examples of what would be considered products.[88] The complete definition was left to be fleshed out by judges in the case law following its publication.[89] Even though software is not listed among the comments' examples, nor directly addressed by case law, the Restatement (Second)'s vague but expansive definition of a product as an item "expected to reach the ultimate user or consumer" favors the theory of recognizing software as a product.[90]

The Restatement (Third) recognized the uncertainties created by this lack of a formal definition of "product" and added an explicit definition of it in section 19. The Restatement (Third) defines a "product" as:

> tangible personal property distributed commercially for use or consumption. Other items, such as real property and electricity, are products when the context of their distribution and use is sufficiently analogous to the distribution and use of tangible

---

86. *See, e.g.*, Zollers et al., *supra* note 9, at 775. "To the extent that [the definition of product in the Restatement (Third) would preclude the inclusion of software in it], we urge courts not to adopt the definition and, instead, focus on the policies underlying strict liability . . . ."

87. In later drafts, the definition was expanded to include "intimate bodily use" personal items. Subsequent debate resulted in a further expanded definition. RESTATEMENT (SECOND) OF TORTS § 402A (1965).

88. *Id.*

> The rule stated in this Section is not limited to the sale of food for human consumption, or other products for intimate bodily use, although it will obviously include them. It extends to any product sold in the condition, or substantially the same condition, in which it is expected to reach the ultimate user or consumer. Thus the rule stated applies to an automobile, a tire, an airplane, a grinding wheel, a water heater, a gas stove, a power tool, a riveting machine, a chair, and an insecticide. It applies also to products which, if they are defective, may be expected to and do cause only "physical harm" in the form of damage to the user's land or chattels, as in the case of animal food or a herbicide.

RESTATEMENT (SECOND) OF TORTS § 402A cmt. d (1965).

89. *See* Lanetti, *supra* note 85, at 812-13.

90. *See, e.g., id.* at 816.

> personal property that it is appropriate to apply the rules stated in this Restatement.[91]

The same section also notes that "[s]ervices, even when provided commercially, are not products."[92]

While software is distributed commercially for use or consumption, the requirement for tangibility[93] arguably eliminates software from the category of "products."[94] Therefore, proponents of strict liability for software advocate for either ignoring the word "tangible," interpreting "tangible" to include software, or relying instead on the Restatement (Second)'s more favorable definition.[95] The Restatement (Third) offers some flexibility and would allow treating software as a product if software is found to be "sufficiently analogous" to tangible personal property.

On balance, the Restatement (Third) would seem to take the position that software should not be treated as a product, or perhaps, to take no position at all on the issue. The comments to section 19, which contain the definition of a product, specifically discuss the possibility of strict software products liability by noting the *Winter* dicta, the aeronautical maps cases, and the U.C.C.'s treatment of software as a good.[96] But the comments also note the dearth of case law support for the proposition that software should be treated as a product.[97]

Although the U.C.C. governs contract and not tort law, many commentators argue that software should be defined as a product by analogizing to the treatment of software as a "good" under the U.C.C. Customized and unique software aside, the preponderance of case law and literature supports the theory that courts consider mass-market commercial software to be a good under the U.C.C. despite its inherent intangibility.[98]

---

91. *See* RESTATEMENT (THIRD) OF TORTS § 19(a) (1998).

92. *Id.* § 19(b).

93. *See id.* § 19(a). The Restatement (Third)'s definition allows an item to be a product if it is sufficiently analogous to the distribution and use of tangible personal property. This is not an easy solution for proponents of strict products liability for software. The argument for analogizing software to the distribution and use of tangible personal property has proven so difficult to make that proponents prefer the alternatives discussed above.

94. *See* Lanetti, *supra* note 85, at 817 (explaining "[t]his shift of computer software from the tangible to the intangible realm arguably eviscerates any contention that software qualifies as a U.C.C. 'good' because an intangible article is not 'movable.' To assume that software would automatically qualify as a 'product' therefore appears both premature and short sighted.").

95. *Id.* at 817-18.

96. RESTATEMENT (THIRD) OF TORTS § 19 cmt. d; *see infra* note 98.

97. RESTATEMENT (THIRD) OF TORTS § 19 cmt. d.

98.

> Tangibility is not the critical issue for UCC application; the important

The plain language of the U.C.C. defines a good as "all things (including specially manufactured goods) which are movable at the time of identification to the contract for sale other than the money in which the price is to be paid . . . ."[99]

The key distinction for the U.C.C., therefore, is movability–if an item is "movable," it is a good.[100] Many commentators recognize that software is not inherently "movable" but believe this distinction is hair-splitting. They argue that judges should freely analogize intangibles to commercial goods when policy dictates such a consideration under the U.C.C.[101]

But in spite of the mixed case law and commentary supporting the theory of software-as-good, the question of tangibility continues to perplex tort analysts. Do the bits and bytes that make up a software program,

---

> characteristics of a good are movability, transferability, and identification at the time of sale. The fact that a computer program cannot be seen or felt should not preclude UCC coverage, as the UCC does not make those qualities the test for exclusion. The type of intangibility meant to be excluded from Article 2, that of choices in action, is different from the type of intangibility characteristic of software. That program instructions are intangible does not rule out UCC applicability, as programs can be identified, moved, transferred, and sold in the same manner as other pieces of personal property classified as goods.

Bonna Lynn Horovitz, *Computer Software As A Good Under The Uniform Commercial Code: Taking A Byte Out Of The Intangibility Myth*, 65 B.U. L. REV. 129, 151-52 (1985). *See also* 68 AM. JUR. 3D *Proof of Facts* § 4, at 333 n.2, David Polin's 2005 Amjur article "Proof of Manufacturer's Liability for Defective Software," which lists a number of cases reinforcing judicial recognition of software as a good under the UCC. The support for software-as-good under the U.C.C. stems from the desire to impute the normal "goods" warranties to software (i.e. merchantability and fitness). But pro-software-liability advocates argue that the U.C.C. definition should extend to torts. The argument proceeds: if software is a "good" under the U.C.C., then it could be considered a "product" under tort liability law.

99. U.C.C. § 2-105 (1998).

100. As my old contracts professor liked to say, "If you can put it in a truck, it's a good." (referencing the UCC's movability test). Professor Jeffrey Davis, Gerald A. Sohn Research Scholar, U.F. Levin College of Law.

101. *See, e.g.*, Jean Braucher, *When Your Refrigerator Orders Groceries Online and Your Car Dials 911 After an Accident: Do We Really Need New Law for the World of Smart Goods?*, 8 WASH. U. J.L. & POL'Y 241, 247 (2002).

> Even in the twenty-first century, lawyers find it hard to give up certain vestiges of formalism. Thus, in making the determination whether software is goods, some commercial lawyers want to debate whether copies of software are "tangible." This approach can be tied to the Article 2 definitional requirement of a movable thing. Even if software is not a "thing," the courts may apply Article 2 by analogy.

Id.

wholly the result of human intelligence, have an independent physical existence from the media upon which the software is delivered? The argument goes that a software program plus the physical CD on which it is delivered equals a good.[102] The contrary argument, however, likens the CD to the cardboard box in which the CD is packaged.

Both are "containers" which "hold" the software. Both can be discarded after the software is "installed." Neither is required to use the actual product which is the software program itself. Finally, the "CD-as-good" classification completely falls apart when one considers software that is purchased online and downloaded directly into the consumer's computer. In this situation, what, exactly, has "moved"?[103] Has the software moved or is it only electrons that have moved?[104] Commentators realize that the new realities for the distribution of software by direct

---

102. *See* Multi-Tech Sys., Inc. v. Floreat, Inc., No. 01-1320 DDA/FLN, 2002 U.S. Dist. LEXIS 4644, at *9 (D. Minn. Mar. 18, 2002) (quoting Advent Sys. Ltd. v. Unisys Corp., 925 F.2d 670, 675 (3d Cir. 1991) (noting that computer software is a "good" within the meaning of the UCC when the software code exists on a disc or other medium that is "tangible, moveable, and available in the marketplace."). However, other cases suggest that software should be considered a "good" even though it is intangible. *See, e.g.*, Micro Data Base Sys., Inc. v. Dharma Sys., Inc., 148 F.3d 649, 654 (7th Cir. 1998) ("[W]e can think of no reason why the UCC is not suitable to govern disputes arising from the sale of custom software . . . ."); Advent Sys. Ltd., 925 F.2d at 676 ("The importance of software to the commercial world and the advantages to be gained by the uniformity inherent in the U.C.C. are strong policy arguments favoring [the] inclusion" of software transactions within Article 2).

103. *See* Raymond T. Nimmer, *Article 2B: Proposals for Bringing Commercial Law into the Information Age*, *in* FOURTH ANNUAL INSTITUTE FOR INTELLECTUAL PROPERTY LAW, at 675, 680 (PLI Patents, Copyrights, Trademarks, and Literary Property Course Handbook Series No. 532, 1998) ("The underlying property rights . . . cause[] differences in contracting practices between the information world and the goods world. The differences are enhanced by the Internet and online services that allow transfer of information without using any tangible objects.").

104. Is software movable? The facile answer, and the one traditionally relied on by proponents of strict products liability for software, is that software exists on physical media (CDs, DVDs, floppy disks). To say the answer is facile is not to say that it is without merit. A close analogy can be drawn between software and music CDs. Both contain intangible "products": a CD has a recording of a human voice. Absent the CD, does the recording exist and if so, where exactly? Curiously, these existential challenges have never seemed to have much relevance when it comes to music. No one would argue that Nora Jones's new release was not a product. Of course, electrons have been considered a good when it comes to electricity (and a product under tort law too). RESTATEMENT (THIRD) OF TORTS: PRODUCT LIABILITY § 19 cmt. d (1998). But, electricity is a completely fungible natural phenomenon. Software is wholly the result of unique intellectual effort, a true intangible. Other than the fact that software relies on electricity to carry out its function, the two things seem non-analogous.

download from the Internet to the user's computer or device may preclude it from being considered a "good" under the current U.C.C. definition.[105]

Another issue is that computer software is protected by intellectual property doctrine.[106] Intellectual property law protects intangibles like ideas, designs, and methods. Software's status as intellectual property rather than as chattel would seem to make it clearly intangible, similar to a song, a graphic design, a television episode, or a poem. Like each of these things, software resembles intellectual property more than tangible goods: software is typically licensed, not sold, and it can be reproduced in near unlimited fashion at no or low cost. If other kinds of intellectual property are not "goods," it would seem that software should also not be a "good" for purposes of strict liability.[107] In fact, the conundrum over software-as-good is partially responsible for sparking a proposed addendum to the U.C.C. to handle software-related transactions. The addendum was originally titled Article 2B, but is now known as the Uniform Computer and Information Technology Act (UCITA) of 2002.

UCITA defines software as information rather than a U.C.C. good.[108] Remarkably, UCITA goes so far as to contend that defect-free software does not exist.[109] The language in the comment is interesting and invokes

---

105. *See* Lanetti, *supra* note 85, at 817 ("This shift of computer software from the tangible to the intangible realm arguably eviscerates any contention that software qualifies as a U.C.C. 'good' because an intangible article is not 'movable.' To assume that software would automatically qualify as a 'product' therefore appears both premature and short sighted."). *See also* Lee Kissman, *Revised Article 2 and Mixed Goods/Information Transactions: Implications for Courts*, 44 SANTA CLARA L. REV. 561, 566-67 (2004) ("The 'facts and circumstances' that affect the scope of Revised Article 2 consist of continuously emerging technology transactions that seem to defy classification in terms of the Article 2 definition of 'moveable goods.'").

106. Primarily software is protected by copyright, but more recently has been protected by so-called "business method" patents. Copyright law defines software as a "literary work." *See* 17 U.S.C. §§ 101, 102(a)(1) (2002); Apple Computer, Inc. v. Formula Int'l, Inc., 562 F. Supp. 775 (C.D. Cal. 1983); Tandy Corp. v. Personal Micro Computers, 524 F. Supp. 171 (N.D. Cal. 1981).

107. The author concedes that software, unlike most other examples of intellectual property, has the ability to cause injury. This line of thinking takes us squarely into consideration of the policy issues.

108. *See* UCITA, *supra* note 60, § 102(a)(35). ("'Information' means data, text, images, sounds, mask works, or computer programs, including collections and compilations of them."). The trend seems to be moving in the direction of considering software as something other than a "good." For example, Revised Article 2 excludes "information" from the definition of goods. *See* U.C.C. § 2-105 (1) (2005). It goes on to define computer software as information. *See* U.C.C. § 2-103(1)(m) (2003).

109. *See* UCITA, *supra* note 60, § 403 cmt. 3(a). ("It is often literally impossible or commercially unreasonable to guarantee that software of any complexity contains no errors that might cause unexpected behavior or intermittent malfunctions, so-called 'bugs.' The presence of minor errors is fully within common expectations.").

"common expectations,"[110] perhaps in an attempt to influence tort law by codifying the principle that consumers expect software to include defects.[111] The UCITA's point of view has not been persuasive. First, and perhaps unsurprisingly, critics claim that UCITA was designed to give software companies a free pass in regards to contract liability by putting "common" defects beyond warranty recourse.[112] Second, UCITA has not been widely adopted, and three states have even adopted anti-UCITA laws.[113] Finally, there is substantial confusion regarding whether software embedded in a device is different in kind from UCITA software.[114]

Commentators favoring strict liability for software would like to extend the case law allowing software to be a "good" under the U.C.C. to software-as-good under tort law for product liability. They downplay UCITA, citing the consumer backlash[115] and that Act's lackluster adoption by the states. Still, in leaping over the UCITA to get to the U.C.C., the commentators largely fail to address the most important point, which makes any analysis of the U.C.C. altogether suspect: the U.C.C. governs contract law. Inasmuch as the parties are in privity of contract such that the U.C.C. governs, contract warranties apply. When a party not in privity

---

110. *Id.*

111. I.e., if consumers *expect* bugs in software, then a bug that causes an injury cannot be said to defeat consumer expectations, which is the test for product defect used in most jurisdictions.

112. Michael Traynor, while president of the A.L.I. said that the "enactment of UCITA, as it now stands, would not be a beneficial development for the law." *See* Michael Traynor, *Feb. 4 Letter to A.L.I. Members on UCITA*, A.L.I. REP. (A.L.I., Philadelphia, PA), Winter 2003, *available at* http://www.ali.org/ali/R2502_03-Letter.htm.

113. *See, e.g.*, Cem Kaner, *Software Engineering and UCITA*, 18 J. MARSHALL J. COMPUTER & INFO. L. 435, 437-43 (2000) (noting that critics of UCITA include twenty-four attorney generals, the American Intellectual Property Association, and the Software Engineering Institute). In addition, three states have adopted *anti*-UCITA legislation. *See* IOWA CODE § 554D.104(4) (2002); N.C. GEN. STAT. § 66-329 (2002); W. VA. CODE § 55-8-15 (2002).

114. *See* Braucher, *supra* note 101, at 250.

> The drafters of Revised Article 2 attempted to include embedded software within its scope by explicit, technical language, but they were stymied by the fact that it becomes ever harder to distinguish embedded and non-embedded software. If much depends on any given test, the product could be re-engineered to take advantage of whatever legal regime is preferable. Thus, attempting a distinction would likely drive engineering decisions in undesirable ways.

Id.

115. *See, e.g.*, Jeffrey P. Cunard & Jennifer Coplan, *Selected Topics in E-Commerce and Internet Law: 2001*, 112 PLI/NY 241, 354 (2001) (explaining "there appears to be an increasing backlash against UCITA, and efforts in many states to adopt it may be stalled absent further changes").

sustains an injury and a negligence or products liability claim is made, tort law comes into play. The courts have vigorously defended the boundaries between contract and tort law, not least in the key strict products liability cases themselves.[116] A court may never directly use U.C.C. terminology in a tort case (e.g., by calling products "goods") and may never consider applying U.C.C. law to decide the outcome. Therefore, even if the U.C.C. defined software to be a "good," the utility of analogies between goods and products is questionable regarding what they may offer in relation to uncharted areas of tort law like strict products liability for software.[117] Furthermore, the erosion of the idea of the "movability" of software and the inherent intangibility of software raise legitimate questions whether the U.C.C. would even consider software to be a "good" in the first place.[118]

Ultimately, the U.C.C. and the UCITA have raised more questions than answers for proponents of software strict products liability. Both works are plain creatures of contract law, and to the extent they have been persuasive, it is in the realm of commerce and not that of personal injury. The concept of strict or absolute liability is foreign under contract law, where economic efficiency is the primary objective, and the privity requirement bars absolute liability. Tort law is a different animal indeed, and strict liability is tolerated in the presence of an impressive variety of tort doctrines[119] that operate to contain and limit liability.

In summary, regarding whether software may legally be considered a "product" for purposes of strict tort liability: (a) there is no case law on point, (b) the Restatements are at best ambiguous to the definition, (c) software is considered to be intellectual property and is protected by intellectual property law like other non-product intangibles, and (d) the analogy to software-as-good under the U.C.C. grows increasingly suspect as the nature of software evolves. The weight of authority, including case law, finds software is not defined as a "product" for purposes of tort products liability. This schism between the current law and what strict

---

116. *See, e.g.*, Greenman v. Yuba Power Prods., Inc., 3777 P.2d 897, 901 (Cal. 1963) (quoting Ketterer v. Armour & Co., 200 F. 322, 323 (S.D.N.Y. 1912); Klein v. Duchess Sandwich Co., 93 P.2d 799, 804 (Cal. 1939) (reasoning that "the remedies of an injured consumer . . . ought not to be made to depend 'upon the intricacies of the law of sales,' and the warranty of the manufacturer to such consumer should not be made to rest solely on 'privity' of contract.").

117. *See, e.g.*, RESTATEMENT (SECOND) OF TORTS § 402A cmt. m (1965) ("[I]t should be recognized and understood that the 'warranty' [justifying strict products liability] is a very different kind of warranty from those usually found in the sale of goods, and that it is not subject to the various contract rules which have grown up to surround such sales.").

118. If software is not a good in the first place, what benefit is there to drawing the analogy?

119. One example is the economic loss rule. For a discussion of the economic loss rule, see generally E. River S.S. Corp. v. Transamerica Delaval, Inc., 476 U.S. 858 (1986).

liability advocates would favor therefore requires a major shift. The imposition of strict products liability would have to be founded on policy arguments sufficiently strong to overcome a wholly novel extension of the doctrine to the arena of non-product intangibles.

III. THE ARGUMENT AGAINST EXTENDING STRICT PRODUCTS LIABILITY TO SOFTWARE

A. *Extending Strict Liability to New Realms is Inconsistent with Tort Reform*

The last decade has witnessed an explosion of tort reform and not least in the specific area of strict products liability.[120] A consensus of sorts has emerged that strict products liability has grown too powerful, too broad, or perhaps too inelegant in its application. For better or worse, legislatures across the country are busily engaged in narrowing the doctrine's application.[121] More than a trend, the movement resulted in the creation of a separate Restatement (Third) of Torts: Products Liability, published by the A.L.I. in 1998, which greatly restricted the application of strict products liability. Given the social and political forces arrayed against the strict products liability doctrine, it would seem perverse, or at least contrarian to now advocate extending the reach of strict products liability to an entirely new area of intangible software that has historically been off-limits to the doctrine.

---

120. *See, e.g.*, American Tort Reform Association's Tort Reform Record (July 10, 2006), *available at* http://www.atra.org/ files.cgi/7990_Record_7-06.pdf (listing tort reform legislation passed in every state); *see also* National Association of Mutual Insurance Companies, Tort Reform: An Overview of State Legislative Efforts to Improve the Civil Justice System, http://www.namic.org/reports/tortReform/default.asp (listing state tort reform efforts); Stephen Labaton, *Bush's Calls for Tort Overhaul Face Action in Congress*, N.Y. TIMES, Feb. 3, 2005, at A16.

> In his State of the Union address on Wednesday evening, [President] Bush once again urged lawmakers to rewrite the tort law rules to do away with what he has called frivolous and costly lawsuits, which he has repeatedly said have become a significant drag on the economy . . . there is broad bipartisan support for some pieces of Mr. Bush's tort law plan . . . .

Id.

121. *See* sources cited *supra* note 120.

B. *The "Stack" Problem Posed by Component Software Supplier Liability*

Within the last decade, the nature of software has changed dramatically. A contemporary software programmer does not begin with a *tabula rasa*; he avoids "re-inventing the wheel" by first assembling a "kit" of software components pre-written by other developers. These software components are freely available online in both commercial and non-commercial forms. The product the programmer ultimately delivers will therefore be an amalgam of the programmer's work layered on and interwoven with the work of many others. The program, in turn, will be installed into a context where it is designed to work in concert with yet other applications—all designed to function interdependently. Therefore, in a single device like an iPod, cell phone, or automobile, there may be software written by dozens, hundreds, or even thousands of individual developers and firms. This Note refers to the group of layered, interdependent software operating within a particular device as "the stack."[122]

Generally, the strict products liability doctrine has been extended to reach suppliers of a product's component parts. Makers of component parts (such as automobile tires) have been held strictly liable for defects in the final product along with the finished good manufacturer, even though the finished good manufacturer makes all the design and manufacturing decisions, and even though the component supplier may have had no idea how its component product would be used.[123] A question

---

122. Consider the many layers of software involved just in an iPod's audio subroutines. First, all of the software in the iPod must run on an operating system such as some derivative of the popular and freely available open-source Linux system. Next, the operating system provides a hardware interface layer of software that talks to software on the audio chip. The application layer, or user-interface software, communicates with the operating system's interface software to direct it to play a music file. There might be a separate software program to handle each different type of audio, video, or image file. Each of these layers and programs could have been written by one or more different developers working for different firms, yet all are expected to work together seamlessly. Because software occurs in layers we think of a stack of software in a particular device.

123. The law in this area is not settled and liability for component parts suppliers varies from jurisdiction to jurisdiction. For a case demonstrating application of strict products liability to a component parts supplier; *see, e.g.*, Speer v. Wheelabrator Corp., 826 F. Supp. 1264, 1265 (D. Kan. 1993) (finding that the manufacturer of a multi-purpose push button electrical switch can be liable for failure to warn the purchaser of the switch for use in a Tumblast machine that metallic dust produced by the machine, which can cause the button to malfunction by sticking). Generally, courts have held component parts makers strictly liable when there is a defect in the component part itself rather than a defect arising out of the manner in which the finished good manufacturer used the part. Zaza v. Marquess & Nell, Inc., 675 A.2d 620, 629 (N.J. 1996). There is no bright-line rule.

therefore arises that was not addressed by commentators calling for an extension of strict products liability to software: what is the nature of liability for component software providers? There are millions of software components available on the Internet—some commercial (used after paying a license fee), and some non-commercial (some entirely free or free but with usage restrictions). Software developers liberally rely on these components, and the component software provider usually has no specific idea who is using its code or for what purpose.

It is undeniable that society reaps enormous benefits from component software. "Large" software companies like Apple are able to deliver inexpensive yet high-quality, software-enabled products like the iPod only because software components are so freely available. This is largely because Apple can avoid the cost of "re-inventing the wheel" with each product. It is also probably true that the ready availability of high-quality component software contributes to overall software product safety: programmers with interests in highly specific areas create components addressing those areas, and the functionality and reliability of the component is consequently quite rich. Any extension of strict products liability to software component suppliers would likely cause those suppliers to respond by (a) removing all non-commercial components from the web[124] and (b) restricting and controlling components that were previously easily and freely available. It is difficult to calculate the extent of lost social productivity of such a move or the increased cost imposed on all software-based devices, but it is fair to say it would greatly slow technological progress and probably cause an enormous detriment to society. Perhaps, it would even have the effect of reducing the safety of software products overall.

Compare id. (noting that a manufacturer of a quench tank to be used in a coffee bean decaffeinating system is not required to equip the quench tank with safety devices), *with* DeSantis v. Parker Feeders, Inc., 547 F.2d 357, 361 (7th Cir. 1976) (noting that a supplier of component parts for a cattle feeder can be liable for failing to provide a cover for the trough and auger parts of the feeder).

124. Non-commercial component suppliers would not necessarily be immune just because tort products liability requires that a supplier be inherently commercial or that a product must be sold. Many component suppliers cannot easily be declared commercial or non-commercial. They are probably best described as quasi-commercial. An example would be a component developer whose software was available for no cost, but who accepts donations on his web site. Another example is a developer who offers a free version of his component with fewer features but hopes to sell upgrades to an expanded-feature commercial version. Also, many large governmental agencies and firms (like Apple, Sony, IBM, and even Microsoft) are participating in the collaborative nature of code sharing and making code available without cost for use by other developers. Is such an action by a commercial entity considered commercial or non-commercial? What if the entity has a stake in the adoption of a certain technology or standard, and the software they are providing at no cost helps to achieve such a strategic goal?

C. *Embedded Software is Not Inherently Different than Other Types of Software*

Should a physical device (like a cell phone or iPod) that is powered by embedded software be considered one product or two?[125] Is the software inside the device a service, a product, or something else? Existing tort law provides no clear answers. But this is a critical question, perhaps *the* critical question, facing contemporary products liability for software. As software becomes a more ubiquitous part of every product we buy or use, it will become even more tempting to consider the internal software as just another component part of the product. This unification would essentially eliminate the question of whether software can be considered a product and open the door to hold manufacturers of software strictly liable for product defects in the same manner that manufacturers of any other kind of product are held strictly liable for product defects, regardless of negligence.

On one hand, the software inside a device might be considered an intrinsic part of the device–a component, like a button or screw. In that case, if the device itself were defective, the software would have no separate functional existence and consequently no separate analytical meaning. A traditional products liability analysis would then ensue. For example, if the software powering the ABS braking system in a Jaguar is defective, and the law defines the embedded software as an intrinsic part of the automobile, then Jaguar Corporation may be strictly liable for injuries caused when the software fails to operate properly. This section offers four reasons why software should not be considered an intrinsic part of the physical device and therefore should be analyzed separately when injuries occur as a result of defective embedded software. Finally, I will advocate for the elimination of any categorical legal distinction between "traditional" and "embedded" software.[126]

First, the physical, tangible part of a device containing embedded software is manufactured in a traditional assembly process. In contrast, the

---

125. "An example of embedded software, is the software that controls the anti-lock braking system in cars." Charles Shafer, *Scope of UCITA: Who and What are Affected?*, *in* UNIFORM COMPUTER INFORMATION TRANSACTION ACT: A BROAD PERSPECTIVE (Stephen Y. Chow et al., Co-chairs, 2001).

126. The author is not alone. *See also id.*; Philip Koopman & Cem Kaner, *The Problem of Embedded Software in UCITA and Drafts of Revised Article 2 (pt. 1)*, 43 U.C.C. BULLETIN, Release 1, 1, 2 (2001). Previous drafts of Article 2 attempted to draw a distinction between software embedded in goods subject to Article 2 and non-embedded software subject to UCITA. *See id.* Release 2, 1, 6.

embedded software is intangible and its production cannot be compared to any traditional manufacturing process.[127] Still, commentators argue that because intangibility is not a barrier to software's classification as a "good" for the U.C.C.'s purposes,[128] tort law should also consider software to be a "product" in spite of its inherent intangibility.[129] However, the policies and purposes of tort law differ fundamentally from those of contract law. Under tort law, the fundamental differences between tangible and intangible products require separate legal treatment.

Second, in U.C.C. cases where software providers have been sued under implied warranty theories for installing defective business software, courts have developed a "hybrid" model where the software plus the hardware is considered a good.[130] However, courts have refrained from creating a bright-line rule, instead making the determination on a case-by-case basis. Proponents of strict products liability for embedded software will almost certainly analogize hybrid U.C.C. goods and services cases to software embedded in a physical device, asserting that tort law should consider such a scenario to be a "hybrid" product. This analogy fails to be entirely compelling because the U.C.C. cases are split on this issue, and even the most favorable examples require case-by-case analysis. This strategy hardly seems to be stable footing upon which to establish a major expansion of tort liability. Embedded software, quickly becoming the most common form of software, is not marketed to end-user consumers like traditional mass-market software.[131] A simplistic analogy to hybrid goods and services products, however tempting as a quick fix, is inadequate. The policies behind the issue require more careful examination.

Third, to the extent that the UCITA has been adopted or has been found persuasive anywhere—a highly debatable point—it is recognized as

---

127. *See* Part II.C (discussing the tangibility argument and its relevance).

128. *See* discussion, *supra* Part II.C.

129. *See generally* Kaner, *supra* note 113, at 435.

130. *See, e.g.*, Youngtech, Inc. v. Beijing Book Co., Inc., No. L-3799-04, 2006 WL 3903976, at *5 (N.J. Super. Ct. App. Div. Dec. 29, 2006) (citing Dreier Co., Inc. v. Unitronix Corp., 527 A.2d 875,879 (N.J. Super. Ct. App. Div. 1986)) ("The judge correctly applied the UCC to this matter. The sale of a computer system involving both hardware and software is a 'sale of goods' even if there are incidental service aspects of the transaction; therefore, the UCC applies."). In this case, and others like it, a court looks at the form of payment (here, a single fixed price for all hardware, software, and services together), and concludes that the entire sale is a U.C.C. "good." Of course, this case and cases like it feature U.C.C. and not tort issues.

131. *See* Zollers et al., *supra* note 9, at 769 (explaining "[a]dmittedly, software embedded in medical equipment, airplanes, and air traffic control systems and used to monitor nuclear power plants is not mass-marketed in the same way that automobiles and other consumer products are . . . ."). That is, when consumers purchase a product containing embedded software, they do not usually even think about the software and may not even be fully aware that it exists.

attempting to deal with the issue of embedded software as distinct from other kinds of software.[132] Proponents of software-as-good wish to keep embedded software squarely under Article 2's umbrella and away from seller-favorable UCITA laws. The confusion generated by the debate has led to recognition that manufacturers of embedded software might easily choose more favorable law just by modifying their products in minor ways that enable semantic classification under UCITA rather than the U.C.C. For example, a manufacturer might provide the necessary software in a separate download service conducted when the consumer gets the device home, rather than by providing it pre-installed at the time of sale.[133]

Finally, it is difficult to escape the conclusion that all software is, in its most essential form, embedded. After all, traditional mass-market software runs on a device–a personal computer. What is it about a laptop or desktop computer that is fundamentally different than any other kind of device powered by a computer processor? Initially, embedded software was software that could not be changed[134] and was an intrinsic part of a device. However, most modern devices now permit their internal software to be changed with updates and upgrades or replaced with competitive, open-source, or hobbyist versions.[135]

Upon reflection, it would seem that the concept of embedded in regards to software is a distinction without a difference, at least perhaps for the purposes of tort law. Worse, the situation is likely to become even more confusing as embedded software becomes yet more ubiquitous and continues to take on newer forms difficult now to imagine. Tort law can elect to reject the tortured semantic pathways contemplated by contract law, escaping the confusion inherent in multiple legal definitions of software by categorically rejecting any definition of software as a product, whether embedded or traditional, for purposes of strict products liability.

---

132. *See* Kaner, *supra* note 113, at 483.

133. For a complete discussion of this issue, see Kaner, *supra* note 113, at 483 (concluding that replacement software for an automobile fuel injection system is a UCITA transaction, and a full examination of the embedded software problem). *See generally also* Braucher, *supra* note 101, at 241.

134. So called "firmware" used to be fixed in a computer's hardware (hence "firm"). For more information, see *Firmware, in* TELECOM GLOSSARY 2000 (2000), *available at* http://www.atis.org/tg2k/.

135. *See, e.g.*, Aaron Weiss, *The Open Source WRT54G Story*, WI-FI PLANET, Nov. 8, 2005, http://www.wi-fiplanet.com/tutorials/ article.php/3562391 (noting "[t]he story of the Linksys Wireless-G Router (model WRT54G) and how you can turn a $60 router into a $600 router is a little bit CSI and a little bit Freaks & Geeks.").

D. *Strict Liability Should Apply Only to Manufacturing Defects, and not to Software*

The Restatement (Third) of Torts: Products Liability categorizes product defects into three areas: manufacturing defects, design defects, and warnings defects.[136] Of these, only the category of manufacturing defects is subject to strict products liability under the Third Restatement.[137] A manufacturing defect occurs "when the product departs from its intended design [in a manner which causes an injury]."[138] A defectively manufactured product has no ready analogue in the world of software. Nor are the policy rationales behind making manufacturers strictly liable for defects that occur in the manufacturing process analogous to software.

A commonly cited policy rationale for imposing strict products liability on software is shifting the burden of proving negligence from the plaintiff to the manufacturer.[139] When a product is complex, and when the manufacturing process is cloaked in trade secrets and confidential and poorly documented steps, it is difficult and expensive for plaintiffs to prove when and how negligence might have occurred. Strict products liability greatly simplifies the plaintiff's burden of proving his case, by eliminating the requirement that the plaintiff prove the element of negligence. In this way, strict products liability resembles the doctrine of *res ipsa loquitor* and offers similar policy justifications.[140] Therefore, a plaintiff must prove only that the product was defective in such a way as to cause injury.[141] To show a defect, the plaintiff must prove that the

---

136. *See generally* RESTATEMENT (THIRD) OF TORTS: PRODUCTS LIABILITY § 1 cmt. a (1998) (discussing the three types of defects and the requirement of defect).

137. *See* GEISTFELD, *supra* note 27, at 71 (discussing the strict liability doctrine of the Restatement (Third)'s manufacturing defect category).

138. *See* RESTATEMENT (THIRD) OF TORTS: PRODUCTS LIABILITY § 2(a) (defining "manufacturing defects").

139. *See* GEISTFELD, *supra* note 27, at 72-74 (discussing the *res-ipsa loquitor* rationale for the policy of strict liability for manufacturing defects).

140. *See* RESTATEMENT (THIRD) OF TORTS § 2 cmt. a ("In many cases manufacturing defects are in fact caused by manufacturer negligence but plaintiffs have difficulty proving it. Strict liability therefore performs a function similar to the concept of res ipsa loquitur, allowing deserving plaintiffs to succeed notwithstanding what would otherwise be difficult or insuperable problems of proof."); DAVID G. OWEN, PRODUCTS LIABILITY LAW 284 (Thomson West 2005).

141. *See, e.g.*, Phipps v. Gen. Motors Corp., 363 A.2d 955, 958 (Md. 1976) (noting "the requirement of proof of a defect rendering a product unreasonably dangerous is a sufficient showing of fault on the part of the seller to impose liability without placing an often impossible burden on the plaintiff of proving specific acts of negligence.").

received product departed from the intended design[142] during manufacturing. This is a lighter burden to some extent because original design specifications are readily discoverable as are other products in the line without the defect. Therefore, the *res ipsa* policy is most compelling in the area of defects occurring during the manufacturing process, which are inherently difficult to prove. It is not as compelling in the area of design or warnings defects, which are by their nature much easier to demonstrate.

The *res ipsa* policy behind strict products liability was first conceived in the context of the so-called "exploding bottle" cases.[143] In those cases, glass cola bottles, under pressure and subject to a myriad of stresses during the manufacturing and recycling steps, would occasionally explode spontaneously, injuring bystanders or drinkers with flying shards of glass.[144] There was no practical way for a plaintiff to trace the provenance of a single bottle back to the factory and obtain evidence about what happened at the exact date of manufacture of that bottle. In all likelihood, tangible evidence of what a worker may or may not have done at any particular time never existed at all. The evidentiary problem was exacerbated by the fact that, in the process of malfunctioning, the bottle destroyed itself, making any demonstration of the pre-existing defect plainly impossible. Therefore, when a product defect occurs in a single product out of thousands or millions of properly functioning products—a so-called "one-in-a-million" accident—the plaintiff's evidentiary burden of proving negligence is improbably high.[145] In cases like these, it seems clear that it is unfairly burdensome to require the plaintiff to produce evidence of negligence, and the application of *res ipsa loquitor* in the form of strict products liability appears therefore to be appropriate.

---

142. *See* GEISTFELD, *supra* note 27, at 72, 72 n.6 (citing Pouncey v. Ford Motor Co., 464 F.2d 957, 960 (5th Cir. 1972) (involving expert testimony over whether the "plaintiff's car had a fan blade with an excessive number of metallic impurities in the steel")).

143. *See* Escola v. Coca Cola Bottling Co., 150 P.2d 436, 437-38 (Cal. 1944) ("original" modern products liability case).

144. For example:

> Plaintiff testified that after she had placed three bottles in the refrigerator and had moved the fourth bottle about eighteen inches from the case "it exploded in my hand." The bottle broke into two jagged pieces and inflicted a deep five-inch cut, severing blood vessels, nerves and muscles of the thumb and palm of the hand.

Id. at 438.

145. *See* Thomas A. Cowan, *Some Policy Bases of Products Liability*, 17 STAN. L. REV. 1077, 1087 (1965) ("[C]ourts . . . realize the great expense that plaintiffs would have to undergo to prove negligent manufacture of one out of a million products of the defendant").

Products subject to liability in tort have historically been tangible goods, assembled in a discrete manufacturing process. Software is intangible. It is an entirely different kind of product than those contemplated by the *res ipsa* policy behind strict products liability, having no discrete or even analogous manufacturing process. Because of its intangible nature, there is no such thing as a "one-in-a-million" accident in the creation of software. Software is created in a process more similar to the writing of a novel than anything else. There are no raw materials, no machines, no manufacturing plant, no delivery trucks, no crates or pallets, and no assembly-line workers. Once the prototype of the first model of a software program is finished, it is immediately ready for distribution. The bottom line is that a plaintiff will never be asked to prove where negligence may have occurred in the software manufacturing process, because there is no such process.

Therefore, most injurious software defects will be of a design-defect type that has the evidentiary helpful quality of reproducibility. Because each unit of a device contains the same software, it is easy for the plaintiff to demonstrate the defective behavior.[146] Demonstration of negligence will not turn on any issue of negligence that may or may not have occurred during the software's manufacturing process, leaving only analysis of the larger (and much simpler to prove) issue of what the manufacturer should have considered in the original software design.[147] The plaintiff can subpoena all the design documents and communications between managers, designers, and software engineers, making expert analysis of the complex inner workings of the iPod's software potentially unnecessary (and beside the point). Therefore, because the plaintiff's burden of proving negligence is not unreasonably high, software design defects should not be held to a standard of strict products liability, but rather to an ordinary negligence standard. This is the approach the Restatement (Third) would seem to take by applying a risk-utility test to such design defects.

In conclusion, the *res ipsa* policy of strict products liability is most useful for situations where "one-in-a-million" manufacturing defects occur, and proof of a manufacturer's negligence is unreasonably difficult to obtain. Because there is no "manufacturing" process when building software (nor even any reasonable analogue), and it is improbable that anything resembling a "one-in-a-million" defect could occur, the *res ipsa*

146. For example, a plaintiff's expert could just play any iPod with the highest volume selected and measure its decibel output.

147. For example, the questions of: did they know of the defect? Or, should they have known of the defect?

policy behind strict products liability is not appropriate for analysis of software-based injuries.[148]

E. *Software—at the Current State of the Art—is an Essential but Unavoidably Unsafe Product Category Deserving of Exemption from Strict Liability*

The Restatement (Second) defines a special category of products as "unavoidably unsafe" and exempts them from strict liability.[149] Similarly, some high-risk industries, such as those for cheeseburgers,[150] vaccines, and human blood are considered to be so valuable to society that they have earned extraordinary judicial and legislative protection from strict products liability.[151] Society values these commercial activities highly enough that it protects them from the economic externalities of liability for certain product-related injuries.[152] But what do such industries have in common with each other and with software? Perhaps it is that, like other exempted industries, the fledgling software industry holds out the promise of dramatically improving the quality of human life, relieving suffering, and increasing the effective wealth of every person on the planet. Still, at first

---

148. Additionally, the incredibly vibrant and growing practice of community and peer review in the software industry makes the plaintiff's job even easier.

149. RESTATEMENT (SECOND) OF TORTS, § 402A cmt. k (1965). Extended discussion of this comment follows.

150. In 2004, federal legislation shielding fast food makers from obesity-related lawsuits almost passed. *See* Kate Zernike, *Lawyers Shift Focus from Big Tobacco to Big Food*, N.Y. TIMES, Apr. 9, 2004, at A15. At least a dozen states have subsequently passed similar legislation. *See, e.g.*, FLA. STAT. ANN. § 768.37 (West 2007).

151. Protection varies from state to state and is not absolute. Generally, however, the protection takes the form of exempting a product from strict products liability, permitting plaintiffs access to the normal negligence-based alternatives. *See, e.g.*, 42 U.S.C. § 300aa-15 (2007) (limiting the compensation that can be awarded under the National Vaccine Injury Compensation Program). At first, the argument over human blood was whether blood was a "product" or not. But, plaintiffs who (among other things) produced their hospital receipts showing specific amounts for blood received in measurable product units, successfully satisfied some courts that a commercial market in fact existed (at least in some key cases). Legislatures were thereby forced to confront the policy issues directly, and in response quickly created special statutory protection for the blood bank industry. *See* GEISTFELD, *supra* note 27, at 76-80.

152. The author refers to broad categories of products exempted on the basis of social utility. Society's valuation of these products is exemplified by statutory exemptions exist in various jurisdictions for all sorts of products including drinking alcohol, cheap handguns, products containing asbestos, and above-ground pools with vinyl bottoms. For a comprehensive test, see RESTATEMENT (THIRD) OF TORTS: PRODUCTS LIABILITY § 2 cmt. d, Reporter's Note § IV(D) (1998). It is also worth noting that tobacco products (i.e., cigarettes) almost made it into a protected category in the Restatement (Third) but this exemption was narrowly voted down due to the impending tobacco litigation. For more, see A.L.I. Proceedings 209-10 (1997).

glance, one might conclude that these product categories have nothing substantive in common except legislative exemption from strict liability. After all, blood and vaccines are critical medical products, whereas fast food is an issue of personal responsibility.[153] However, these industries share at least four compelling qualities as discussed below. The software industry also shares these qualities and therefore deserves similar judicial or legislative protection from strict products liability.

First, all four groups of products have the uncommon potential to cause widespread harm. This makes them especially susceptible to strict products liability and makes the economic cost imposed by the doctrine particularly large for these industries relative to other products. Second, the exempted products all have qualities that make it inherently difficult, if not impossible, to achieve safety beyond a certain point, so-called "unavoidably unsafe" products.[154] Third, the products are seen to be crucially important or vitally necessary to society in some way. Fourth, each product category delivers benefits to society far in excess of the actual aggregate cost of actual injury. The following two sections explain these qualities, and make the case that software shares them and should likewise be exempt. Finally, this Note examines the software industry's relative immaturity and argues that an industry with such uncommon promise should be nurtured and shielded from strict products liability.

1. Necessary Products Facing Potentially Industry-Swallowing Liability Should be Exempt from Strict Products Liability

The authors of one article advocating strict products liability for software[155] grant a remarkable concession, admitting that "[i]f software can help relieve suffering and enhance the quality of life, we would want to create an environment where its development can flourish."[156] In other words, if the social utility created by software outweighs its risks, the economic burden created by strict liability should not be placed on software. It is noteworthy that categorically exempt products—like blood, vaccines, asbestos, and fast food—include the potential for widespread

153. Yet, what about other products where personal responsibility is implicated? These products have not seen swift legislative exemption as has fast food. For example, you might ask the tobacco companies about personal responsibility.

154. The Restatement (Second) of Torts defines such products this way: "[t]here are some products which, in the present state of human knowledge, are quite incapable of being made safe for their intended and ordinary use." RESTATEMENT (SECOND) OF TORTS, § 402A cmt. k.

155. Zollers et al., *supra* note 9, at 745.

156. *Id.* at 772.

harm and consequently face an uncommonly large potential liability for tort damages.

For example, until sometime in the late 1960s, over one-third of blood transfusion recipients acquired hepatitis.[157] The economic burden created by these victims, if allowed to recover from the fledgling network of blood banks, could have drastically limited or even eliminated the blood bank industry entirely. In a 1995 class-action lawsuit filed by hemophiliacs who had been infected with HIV, Judge Richard Posner noted that the "$25 billion in potential liability (conceivably more)" could "hurl the industry into bankruptcy" and potentially swallow "a major segment of the international pharmaceutical industry . . . ."[158]

In the case of vaccines, where injuries might not be discovered for decades or even generations, millions could be harmed, and the potential damages could dwarf even the DES (Diethylstilbestrol) drug cases. Even the threat of such unlimited liability could have socially expensive ramifications when vaccine producers abandon such high-risk products. This reaction by vaccine producers has occurred, and it is not surprising, therefore, that vaccines have garnered exemptions from strict liability.[159] Fast food is consumed by many, and regular consumption is known to have adverse health effects. If, as has been suggested, obese persons might become plaintiffs against these fast food companies, the potential liability would seem to be incalculable. The fact that these high-risk products have been exempted from strict liability—precisely because of their potential to cause widespread injury—is significant. It demonstrates that society values these products in excess of their potential cost, and it realizes that the burden of liability without fault could severely curtail or completely destroy these necessary industries.[160]

---

157. Worse yet, almost half of American hemophiliacs (some 10,000) and about 29,000 others were infected with AIDS via transfused blood between 1970 and 1985, when the first reliable commercial AIDS test became available. *See* GEISTFELD, *supra* note 27, at 77-78; Michael J. Miller, Note, *Strict Liability, Negligence and the Standard of Care for Transfusion-Transmitted Disease*, 36 ARIZ. L. REV. 473, 513 (1994).

158. *In re* Rhone-Poulenc Rorer Inc., 51 F.3d 1293, 1298, 1300 (7th Cir. 1995).

159. Vaccines have been exempted even where potential injury is not only foreseeable but is anticipated. The classic case is that of the rabies vaccine, cited as an "outstanding example" in comment *k*. The vaccine "not uncommonly leads to very serious and damaging consequences when it is injected[, but] . . . 'since the disease itself invariably leads to a dreadful death, both the marketing and use of the vaccine are fully justified, notwithstanding the unavoidable high degree of risk which they involve.'" *See* GEISTFELD, *supra* note 27, at 78-79.

160. It may not be immediately obvious why society should have a stake in protecting the oft-disparaged fast food industry. But on closer analysis, compelling reasons emerge. First, the fast food industry as a whole employs a vast number of Americans. This industry provides menu entry-level and unskilled jobs. Second, the industry provides a source of readily available, low-cost food

Embedded software is ubiquitous and a defect can be propagated nearly instantaneously as thousands or millions of users download a new version. Software, therefore, could face enormous and potentially industry-swallowing liability as the result of a single injury-causing defect.[161] Despite this potential nightmare scenario, the actual cost to society of real uncompensated injuries resulting from defective software appears to be very low, if not nonexistent. People are not currently being injured by defective software with any measurable frequency. The argument for extension of strict products liability to software based on the potential for injury is something of a bogeyman. The real social cost of software appears to be very low at present—too low to justify slowing or reducing of the social benefits provided by software.

Calculating the benefit conferred to society by software is difficult to do with any precision. Yet, it appears that the benefit is enormous, and well in excess of the direct cost to society.[162] Almost every area implicated in the quality-of-life measurement is also substantially affected by the

---

to a large segment of the population. Third, the industry currently thrives under razor-thin margins. If the potentially vast liability for obesity, heart disease, diabetes, and other diet-related illnesses is assigned to the fast food industry, it is likely that either the industry would shrink dramatically, or the costs of fast foods would increase to the extent necessary to afford uncommonly large settlements. The benefits accruing to society by having the fast food companies in the economy offering low-salary jobs and low-priced goods outweigh the harm done to the minority of consumers who over-indulge. As we have seen in the case of blood, even if the number of those injured is large, an industry may be considered valuable enough to overlook the harm. While the personal responsibility issue is also implicated in the liability exemptions for fast food, it is not sufficient. Just ask the tobacco companies.

161. Imagine the number of potential plaintiffs represented by Birdsong's class in the iPod litigation. Should Apple be found liable for hearing damage, possibly to anyone who ever listened to an iPod device, it is likely that the dollar value of the awards could dwarf the gross receipts from the relatively inexpensive device (a recipe for bankruptcy). While such a result might be preferred in the case of a device that society wishes to discourage altogether, it is distinctly to be avoided in the case of a "quality of life" enhancing device like the iPod.

162. *See, e.g.*, Michael R. Maule, *Applying Strict Products Liability to Computer Software*, 27 TULSA L.J. 735, 755 (1992).

> [I]ncreased reliance on computer technology has given humankind a glimpse at a technological revolution that will rival the industrial revolution in its impact on society. . . . Additionally, the continued growth of computer technology has important political effects [perhaps] one of the more significant consequences [of ubiquitous software availability. Essentially, c]omputers have improved our lives in many ways.

Id.

development and use of computer software.[163] Furthermore, for the next foreseeable period of time, software is likely to continue to increase our quality of life far beyond current levels.[164] The vigorous political response to the "offshoring" threat to U.S. programming industry indicates a widespread recognition that the industry is of critical importance to the economy. The software boom has arguably created more well-paying white collar jobs than any other sector. The export of American intellectual property in the form of software,[165] or enabled by software,[166] is a multi-billion dollar industry which grows each year. But as significant as these economic indicators are, they pale in comparison to the promises that embedded software offers for the quality of life, health, and longevity of every human on the planet. Inexpensive devices powered by software are making their way into every aspect of life, reducing human work load, increasing safety, and increasing health and longevity.

The conclusion is that software has a substantially high present value to society, exceeding the benefits conferred by other "ordinary" types of

---

163. For example, consider the awe-inspiring developments that software offers the disabled. "[J.B.] Galan, injured in a diving accident six years ago, is paralyzed from the shoulders down. But thanks to Project Archimedes, he excels at using computers for complex tasks. Writing letters, designing Web pages, and using telephones and electronic mail are [now] part of his everyday routine." David Salisbury, *Total Access Despite Disabilities*, STANFORD TODAY, Mar./Apr., 1996, *available at* http://www.stanford.edu/dept/news/stanfordtoday/ed/9603/9603nsmf.html. "A sensor implanted in a paraly[z]ed man's brain has enabled him to control objects by using his thoughts alone. The experimental set-up allowed the man, who has no limb movement at all, to open e-mail, play a computer game, and pinch a prosthetic hand's fingers." *Brain Sensor Allows Mind-Control*, BBC NEWS, July 12, 2006, *available at* http://news.bbc.co.uk/2/hi/health/5167938.stm. For an example of a software-powered robotic prostatectomy device, see Pioneer and Leader in Laparoscopic Radical Prostatectomy Web Site, http://www.krongrad-urology.com/ (last visited Sept. 19, 2007).

164. Imagine the potential impact of a few simple examples: traffic-regulating software, embedded in automobiles, that not only makes travel less stressful but also reduces injury and saves lives. Robotic technology offers the possibility of creating more labor savings around the home, replacing human workers in high-risk occupations, and greatly increasing available leisure time. Medical robots allow remote surgeons to perform complex surgeries. Simple medical procedures may soon be performable by completely automated robotic devices, and therefore, much less expensive and lower-cost. For an example of a prototypical robotic fire hose that can enter a burning building (no human needed) and put out a fire, see Snake Robot to the Rescue, Innovations Report, http://www.innovations-report.com/html/reports/energy_engineering/report-67839.html (last visited Sept. 19, 2007).

165. Consider as one example the worldwide impact of Microsoft Windows and the applications that run on it.

166. Consider the delivery of American music, news, magazines, movies, and television and content that is delivered by software, e.g., onto a cell phone screen, which is a very popular feature in Europe.

goods. And, even though the potential for injury from software appears inevitable at some point, injuries caused by defective software today are very rare. Society is realizing tremendous benefits from software far in excess of any real costs. The imposition of strict liability might slow or impede the industry or even deter useful products altogether—the exact opposite of the result that society prefers.[167]

Like blood, vaccines, asbestos, fast food, and other socially desirable products, software should be exempted from potentially industry-swallowing liability in the form of strict products liability. For the present, at least, the software industry is responsible for creating some of the most dramatic improvements ever seen in the quality of human life and promises to continue this trend into the foreseeable future. Society should nurture such an industry as long as the aggregate benefits exceed the total costs.

---

167. The story of the vaccine industry is particularly alarming in this regard.

> Vaccines have been a powerful tool in improving our well-being and longevity. Vaccines have been developed to control the following diseases: polio, whooping cough, measles, rubella, mumps, diphtheria, tetanus, influenza, pneumococcal and meningococcal infections, and hepatitis. Four decades ago, polio was a dreaded disease that afflicted 57,000 Americans; in 1984, there were four cases. . . .
>
> Yet, despite this spectacular accomplishment, the number of drug companies producing vaccines has declined sharply. "Between 1965 and 1985, the number of U.S. vaccine manufacturers shrank by more than half; . . . [a]nd only two major companies . . . were still investing heavily in vaccine research." Within the 1980s, the number of firms producing vaccines for five serious childhood diseases declined from thirteen to three.
>
> Fear of liability was a major reason for this retreat. . . . The profit per dose is low, and yet the perceived liability per dose is high. . . . [O]ne scholar has asserted that blame for this perception must be assigned to the shift in tort law from negligence . . . to strict liability (with its emphasis upon alleged defects in the product itself).
>
> Certainly there are instances that critical commentators can cite to buttress their claim that modern tort law is a major culprit in drying up the number of vaccine manufacturers. Bendectin, a morning-sickness drug, was voluntarily withdrawn from the market after a flood of litigation . . . . [T]he manufacturer, Merrell Dow, . . . gave up . . . because of adverse publicity and the $18 million annual cost of legal fees and insurance that approximated the $20 million in sales.

John P. Wilson, *The Resolution of Legal Impediments to the Manufacture and Administration of an Aids Vaccine*, 34 SANTA CLARA L. REV. 495, 505-06 (1994).

2. Unavoidably Unsafe Products Like Software Should be Exempt from Strict Products Liability

Blood, vaccines, and fast food all have qualities that make them unavoidably unsafe. For blood, the prevalence of blood-borne illnesses, the direct injection of blood into the patient, and the high-pressure circumstances under which most transfusions occur add up to a recognizably high-risk situation. Vaccines are similar in that they are injected directly into a patient's veins, have systemic effects, and are also generally composed of particles of deadly illnesses. Fast food is by definition food that is low-cost, of low quality, and high in fat and sugar—a recipe for health problems.[168] Products which cannot reasonably be made any more safe are exempted by both Restatements of Torts.

In the Restatement (Second), comment *k* to section 402A plainly identifies the category, using medical devices and vaccines as examples:

> k. *Unavoidably unsafe products.* There are some products which, in the present state of human knowledge, are quite incapable of being made safe for their intended and ordinary use. These are especially common in the field of drugs . . . . Such a product, properly prepared, and accompanied by proper directions and warning, is not defective, nor is it *unreasonably* dangerous . . . . It is also true in particular of many new or experimental drugs as to which, because of lack of time and opportunity for sufficient medical experience, there can be no assurance of safety, . . . but such experience as there is justifies the marketing and use of the drug notwithstanding a medically recognizable risk. The seller of such products, . . . properly prepared and marketed, . . . is not to be held to strict liability . . . merely because he has undertaken to supply the public with an apparently useful and desirable product, attended with a known but apparently reasonable risk.[169]

This comment has been very influential,[170] but its reliance on pharmaceutical examples has led some commentators to erroneously conclude its application is limited to drugs and healthcare-related

---

168. For an entertaining yet frightening look at the problem, see SUPER SIZE ME (Samuel Goldwyn Films, LLC 2004). For legal analysis, see generally Richard C. Ausness, *Tell Me What You Eat, and I Will Tell You Whom to Sue: Big Problems Ahead for "Big Food?,"* 39 GA. L. REV. 839 (2005).

169. RESTATEMENT (SECOND) OF TORTS, § 402A cmt. k (1965).

170. *See* 2 MADDEN & OWEN ON PRODUCTS LIABILITY § 22:3.

products.[171] However, the plain language of comment *k* as well as the contemporaneous scholarship of Dean Prosser and his key A.L.I. advisors demonstrate the clear intention to apply the exemption generally to useful products, which can not reasonably be made any safer, and not just to pharmaceuticals.[172]

The Restatement (Third) does not directly address unavoidably unsafe products in the manner of comment *k*. However, by requiring plaintiffs to prove that a manufacturer failed to adopt a "reasonable alternative design"[173] that would eliminate the danger, it effectively excludes liability for injuries caused by unavoidably unsafe products. After all, if inherent risk is unavoidable, then by definition, no reasonable alternative product design exists and the existing design cannot be described as defective. No design defect, no liability.

Case law solidly supports the proposition that products that are unavoidably unsafe by their nature should be exempt from strict products liability.[174] For example, an oft-cited Florida Supreme Court case, *Radiation Technology, Inc. v. Ware Construction Co.*[175] describes a rationale for exempting unavoidably unsafe products:

> [The consideration for exemption from strict liability] balances the likelihood and gravity of potential injury against the utility of the product, the availability of other, safer products to meet the same need, the obviousness of the danger, public knowledge and expectation of the danger, the adequacy of instructions and warnings on safe use, and the ability to eliminate or minimize the danger without seriously impairing the product or making it unduly expensive. Thus, an unsafe product, whether it be characterized as inherently dangerous or unavoidably dangerous, would not necessarily be an unreasonably dangerous product.[176]

---

171. *See* GEISTFELD, *supra* note 27, at 76 (discussing problematic interpretation by some courts of "unavoidably unsafe product" as limited to pharmaceutical products).

172. *See* Owen et al., *supra* note 9, at 344 (discussing the development of comments i, j, and k and their applicability to unreasonably unsafe products doctrine).

173. *See* RESTATEMENT (THIRD) OF TORTS, § 2(b) (1998) (providing that a product "is defective in design when the foreseeable risks of harm posed by the product could have been reduced or avoided by the adoption of a reasonable alternative design . . . , and the omission of the alternative design renders the product not reasonably safe . . . .").

174. Radiation Tech., Inc. v. Wave Constr. Co., 445 So. 2d 329 (Fla. 1983).

175. *Id.*

176. *Id.* at 331.

The discussion of "utility" in *Radiation Technology* exemplifies the policy rationale that it is economically inefficient to impose strict liability on manufacturers of socially desirable, but inherently dangerous, products. Furthermore, economic analysis of products liability tort law suggests that it is possible that manufacturers have no ethical responsibility to protect consumers from unavoidable risks.[177] Influential commentator David G. Owen has posited that "[u]tility theory, as truth and equality, [] suggests that manufacturers are not morally accountable for generic dangers[.]"[178] He argues that society would be harmed by a rule which holds manufacturers liable for "unavoidable" or "undiscoverable" risks.[179] Such a rule would "overdeter" manufacturers and cause them to avoid introducing socially desirable, beneficial products until such a time as the risks can be quantified.[180] Finally, imposing a duty requiring strict manufacturer liability for unknowable risks would probably have little effect on the manufacturer's behavior and therefore, would be ineffective as an incentive.[181]

A critical question then is whether embedded software is unavoidably unsafe at current levels of technology. As discussed, the UCITA[182] clearly states that the manufacture of defect-free software, at the current state of human knowledge, is impossible:

> [A] popular operating system program . . . contained over ten million lines of code or instructions. In a computer, these instructions interact with each other and with code and operations

---

177. RESTATEMENT (THIRD) OF TORTS § 2(b) (1998).

178. *See* Owen, *supra* note 39, at 484.

179. *Id.* at 483.

180. *Id.* at 484.

181.

> An argument sometimes proposed in favor of such a duty is that liability for such risks may serve instrumentally as an incentive for manufacturers to increase their levels of research to discover dangers at the threshold of knowledge, which may result in a net benefit for society. If liability for failing to warn of unknowable dangers really did have this result, and if the resulting benefits to potential victims really did exceed the detriments to other consumers and to shareholders, then utility and efficiency would support a rule of liability in such cases. But manufacturers should be investing optimally in research under a negligence standard of accountability, based on knowable risks alone, and extending liability to risks that cannot reasonably be discovered probably will not cause most manufacturers to do more research than they already think is best.

Id. at 483-84.

182. *See* UCITA, *supra* note 60, discussed *supra* Part III.B.

> of other programs. This contrast[s] with a commercial jet airliner that contained approximately six million parts, many of which involved no interactive function. . . . It is often literally impossible or commercially unreasonable to guarantee that software of any complexity contains no errors that might cause unexpected behavior or intermittent malfunctions, so-called "bugs." The presence of minor errors is fully within common expectations.[183]

This comment, especially the last sentence, is remarkable. Defect-free software would itself be unexpected! The UCITA argues, therefore, that software having the potential to injure is unavoidably unsafe (or unavoidably defective).[184] Of course, as discussed above, there are problems with relying on the UCITA for any tort analysis. First, and most importantly, the UCITA is founded in contract law, not torts. Second, the UCITA is controversial and not widely adopted,[185] and yet the comment cited above resonates. Surely every human who has worked with computers in any capacity is familiar with commonly occurring software defects or "bugs." Programmers would also certainly agree with the UCITA statement. Additionally, it might be true that most people would agree that it is more remarkable that software-based products work as well as they do, rather than the reverse. The UCITA's comment, therefore, may simply be describing the situation as it exists today.

To conclude that software development is not unavoidably unsafe, it is necessary to assume that software developers possess: (a) the tools and technologies necessary to achieve "safe" products and (b) have the ability to foresee the particular dangers posed by software or by devices incorporating software. These assumptions are unsound. In the last twenty-four months, literally hundreds of books instructing software developers in general programming techniques (not counting texts on specific programming languages), have been published.[186]

A typical guide designed for software programmers instructs that software design should not be referred to as "software engineering" but rather as "software research."[187] Engineering, it explains, such as bridge

---

183. UCITA § 403 cmt. 3(a).

184. *Id.*

185. *See* discussion of UCITA, *supra* Part II.C.

186. Just search Amazon for "programming" and witness the vast number of results.

187. WHIL HENTZEN, THE SOFTWARE DEVELOPER'S GUIDE 121 (3d ed. 2002).

building, is based on a standard set of tools, technologies, and processes.[188] If any of these three core components change during an engineering job, overruns and difficulties are foreseeable and expected.

Imagine if the bridge-building engineers were required to adopt new, untested tools and components for each project. Programmers always confront this situation: "With software projects, the technology changes [for] every project, the tools change [for] every project, and the process is in a continuous state of flux."[189]

The analogy of software design to traditional engineering demonstrates the difficulties that software developers face in designing any product, much less one that pushes technological boundaries. Another way to state the situation: the software development process will have matured when a consistent and widely accepted set of tools, technologies, and processes are available and change only infrequently. Such is not the case now, nor will it be in the foreseeable future.

No official count of the number of programming languages is currently available. One web site purports to list 8512 current programming languages as of 2006.[190] Programming languages are incorporative; that is, a certain programming language is itself written in another, more basic, programming language.[191] That language can in turn be used to develop other, more complex languages.[192] The number of possible combinations of languages used to iteratively create other languages is growing exponentially. Defects can and do arise at any level of all these incorporated technologies, not just at the most superficial level of effort where the programmer is writing the code used for the instant application.

---

188. *Id.* Furthermore, engineering is based on principles like physics and architecture, which do not change very often, and when they do change, enormous disruption is expected. Software design, on the other hand, is based on core principles that can change radically *within each year.*

189. *Id.*

190. *See* The Encyclopedia of Computer Languages, Murdoch University, Australia, *available at* http://hopl.murdoch.edu.au/ (lat visited Sept. 19, 2007).

191. The concept of a programming "language" is itself in flux. There is not even agreement as to what a programming language *is*.

192. For example, the currently popular "AJAX on Rails" framework (a type of meta-language) is made up of several other languages: Javascript, XML, and Ruby on Rails. Ruby on Rails is written in Ruby, which is in turn written in C++. C++ is an extension of the C language, which is written in assembly code. C language varies depending on which type of processor is used. Javascript exists within an Internet browser. Therefore, each browser offers its own version of Javascript, not entirely compatible with the other browser versions. Worse yet, different revisions of each browser (for example, Microsoft's Internet Explorer versions 4, 5, 5.5, and 6) all support different versions of the Javascript language, each unique within that type and version of browser. *See, e.g.*, Wikipedia, *JavaScript*, http://en.wikipedia.org/wiki/Javascript (describing history and features of JavaScript) (as of Mar. 3, 2008, 16:15 EST).

So, even while the quality of the available languages and tools is increasing, the complexity of inter-relatedness is increasing even faster.

If it is impossible for the average developer who relies on these ever-changing tools and technologies, through reasonable effort, to completely eliminate defects from software, it follows that embedded software is to some extent unavoidably unsafe. Therefore, it would be perverse to hold a non-negligent developer strictly liable for injuries caused by those defects. As applied to software, strict products liability would attempt to compel a result over which the software engineer lacks control. Such a result would defeat a major policy rationale for strict products liability: to provide an incentive for products engineers and manufacturers to develop safe products. Certainly, software developers whose products will be used in hazardous or dangerous conditions, such as makers of implantable medical devices, have a duty of reasonable care to ensure that their products are as defect-free as possible, and to test them thoroughly. This is a traditional negligence standard. It falls short of imposing liability without fault on those developers—a standard which cannot possibly create the incentives for which the policy is designed.

The current state of software technology is changing so quickly that it is impossible for developers to anticipate and design against all risks. Moreover, software is inherently unstable (i.e., developers cannot avoid "bugs"). Because embedded software is becoming ubiquitous, and can be rapidly installed in many places, potential liability is huge. Yet, society considers this kind of software very important, if not critical. Therefore, software is entitled to an exemption from strict products liability based on its nature as unavoidably unsafe, yet socially necessary.

3. The Software Industry is Immature and Should be Nurtured, Not Burdened with Strict Liability

The concept that nascent industries deserve some protection from tort liability has an impressive pedigree. The development of strict products liability doctrine was delayed at the beginning of the industrial revolution, largely to give the new industrial technologies time to "get off the ground" and to mature before subjecting them to the additional externalities imposed by tort liability.[193] Recognizing this, many of the law review

193. "During the nineteenth century, courts . . . were concerned that any expansion of liability beyond the contractual relationship would expose manufacturers and other product sellers to excessive liability, thereby disrupting product markets to the detriment of society." *See* GEISTFELD, *supra* note 27, at 9; *see, e.g.*, Michael Rustad & Lori E. Eisenschmidt, *The Commercial Law of Internet Security*, 10 HIGH TECH. L.J. 213, 259-62 (1995) (stating the defenses in tort law developed

articles calling for strict products liability for software announce, as if it were fact, that the software industry has now matured to the point where it can afford to, and should, bear the cost of an unanticipated injury.[194]

One economic explanation for giving new industries the kid-glove treatment is that industries that produce or have the potential to produce massive social benefits far in excess of their "real" cost should be nurtured. Nurturing includes being treated more delicately as far as legal burdens and government regulations are concerned. Furthermore, when industries are nascent, they often cannot afford to bear the burden of injuries caused by products which can be inherently unstable due to the intrinsic nature of immature technologies.[195] Finally, in the early years of industrial development, the potential for injury can be high but actual injuries relatively low, due to the slow rate of adoption of the new technology.[196]

Another explanation for treating promising new industries differently when it comes to strict products liability is that immature industries cannot readily obtain insurance against risks. A key rationale for holding manufacturers strictly liable for injuries caused by their products is the "insurance" or "risk-spreading" rationale—product manufacturers can efficiently pass on the cost of insuring against injury to consumers in a diluted fashion, whereby the manufacturer increases the price of each product sold by a small amount to cover the cost of liability.[197] Thus, in the aggregate, the users of the product effectively self-insure against risks, an economically efficient outcome. This has proven to work well in many cases involving products in mature industries where risks are easily foreseeable.

However, such strategy is meaningless when the manufacturer cannot obtain insurance. Insurers are unwilling to insure immature or developing industries where potential liability is very large and insufficient statistical data exists to perform necessary actuarial analysis.[198] The majority of

---

to protect industrialization (citing MORTON J. HOROWITZ, THE TRANSFORMATION OF AMERICAN LAW 1780-1860, at 99-161 (1977)).

194. *See* Zollers et al., *supra* note 9, at 771-72.

195. These immature technologies are perhaps, unavoidably unsafe. Refer to detailed discussion of this issue, *supra* Part III.D.

196. For example, in 1930, there was only a fraction of the number of automobiles in use in the United States, compared to the number today.

197. *See* OWEN, *supra* note 19, at 486.

198. *See* GEISTFELD, *supra* note 27, at 155.

> Unforeseeable risks pose a hard actuarial problem, making the provision of insurance much more difficult, if not impossible. Good actuarial data depend upon

devices containing embedded software (music players, cell phones, personal mapping tools) are altogether new types of devices, that inherently lack any history of use or information about "normal" levels of injury. Without such data, insurers are unwilling and unable to provide reasonably-priced insurance to manufacturers.[199] If software developers cannot obtain reasonably-priced insurance (or any insurance at all) against product-related injuries, then such costs cannot be passed to the consumer and the risk-spreading policy objective is moot. In this case, between the software manufacturer and the consumer, the consumer may actually be in a better position to insure against injury than is the manufacturer.

The software industry is immature and is in that sense fragile.[200] The articles advocating strict products liability for software rely on a highly simplistic metric to gauge the maturity of the industry.[201] Generally, the pro-strict liability arguments proceed by pointing out a few examples of large software firms,[202] noting the gross annual sales for the industry and concluding "maturity" based on the industry's pure economic value (i.e., dollars of sales),[203] as if the ability to pay damage awards alone defines an industry's level of maturity. However, the reality is that the software industry is composed in vast numbers by small firms, individuals, and hobbyists. The very few large software firms are the exception rather than the rule. Therefore, imposing across-the-board strict products liability for

---

> a large sample size. To determine the likelihood that a warning is defective . . . , the relevant sample for statistical purposes cannot involve the number of products within a given product line, since each has the same warning. The statistical sample must instead involve a large number of different [warnings], which must either come from different product lines or different manufacturers of similar products. Data involving different product lines or manufacturers may not be sufficiently similar in the relevant respects, complicating the actuarial analysis . . . . The magnitude of the liability exposure, coupled with the difficulty of estimating the likelihood of liability, makes it extremely difficult to insure against unforeseeable risks.

Id.

199. *Id.*

200. Most of the articles calling for the extension of strict products liability to software admit that advances in technology may slow as a result, suggesting that the industry is perilously sensitive to changes in its cost matrix. This reflects a quality of immaturity for an industry which has not developed, for example, the ability to protect itself through political lobbying. *See, e.g.*, Maule, *supra* note 162, at 755 (remarking "[t]he imposition of strict products liability may have a chilling effect on the creation of new advances in computers.").

201. *See id.*

202. Such large software firms include, for example, Microsoft and IBM.

203. Although proponents also cite the size of a very few software firms as evidence. *See* Zollers et al., *supra* note 9, at 770-72.

software would put the smaller firms and individuals, who lack the actuarial expertise needed to calculate the value of their potential liability, out of the market altogether.

Ability to pay is an insufficient metric for calculating the relative maturity of an industry. From an economic standpoint, an industry's maturity is signaled by several factors: (a) the industry's stage in its lifecycle (beginning, middle, end), (b) the standardization of elements within the industry (tools, processes, standards, semantics, contract terms, pricing), (c) the number of competitors (many, few), (d) the "normal" profit margins (above-average, average, low), and (e) the barriers to entry for new competitors (low, medium, high). Mature industries are indicated by a late position in the industrial lifecycle, high standardization of elements within the industry, few and stable competitors, low profit margins, and high barriers to entry. Examples include: the automotive, petroleum, home appliance, and carbonated beverage industries. In contrast, the software industry is at an early stage in the industrial lifecycle; has little or no standardization of elements (programming languages, professional standards, testing tools, or contract terms);[204] it features legions of competitors and a lot of turnover (i.e., the number of programming companies and individuals in the industry); maintains above-average profitability; and offers extremely low barriers to entry for new firms.[205]

Society prefers to shield promising new industries from the burdens of regulation and strict liability placed on more mature manufacturers. Because, by any credible measure, the software industry is still in its infancy, it should also be shielded from the onerous cost of strict products liability against which it cannot insure.

IV. ALTERNATIVES TO STRICT PRODUCTS LIABILITY

> [T]he rhetoric of products liability law is, undeniably, a mess. With a plethora of available doctrines—e.g., negligence, strict liability, and express and implied warranties of merchantability—courts have

---

204. There is no licensing authority for programmers, no standards embodied in law, and no minimum educational requirement.

205. Barriers to entry are an economic concept representing the cost for a new competitor to enter the market. Since all a programmer needs is access to a computer in order to enter the market, the barriers to entry are extremely low (compare to the cost of opening a retail store, offering telephone service, or building a power plant).

been at sea trying to determine how the standard for defective design fits into these doctrinal theories.[206]

Retaining the status quo by refraining from extending the doctrine of strict products liability to software will not, as some commentators suggest,[207] leave plaintiffs with a paucity of alternatives to recover for their injuries. Plaintiffs will continue to have all avenues of recovery that the law currently provides. The second and more important point is that these current options are numerous and legally fertile. Injured plaintiffs are under no threat of denial of recourse against software developers should courts and legislatures properly decline to extend strict liability to the industry.

A number of alternative theories and doctrines exist under which an injured plaintiff may recover against a software developer or the maker of a device containing embedded software. This section reviews the most common alternatives and briefly discusses each in the context of software.[208]

A. *Ordinary Negligence*

Plaintiffs have prevailed against software-seller defendants on the basis of traditional negligence by showing that the software provider failed to exercise a duty of reasonable care in providing the software to the plaintiff.[209] In *Invacare Corp. v. Sperry Corp.*,[210] the court held that the defendant software provider "was negligent in . . . recommending the [software] and . . . it knew, or in the exercise of ordinary care, it should have known, that the systems were totally inadequate . . . for plaintiff[.]"[211] In its opinion, the court specifically noted that liability was premised on ordinary negligence and "does not give rise to a new tort of 'computer malpractice.'"[212]

---

206. James A. Henderson, Jr. & Aaron D. Twerski, *Achieving Consensus on Defective Product Design*, 83 CORNELL L. REV. 867, 871 (1998).

207. *See* Zollers et al., *supra* note 9.

208. In addition to the doctrinal remedies described in this section, there may also be state-law based remedies available to plaintiffs. Frequently, states have enacted legislation governing business practices generally and products liability in particular which may provide additional causes of action for consumers injured by defective software. For examples, see commentary to RESTATEMENT (THIRD) OF TORTS § 19.

209. *Id.*

210. Invacare Corp. v. Sperry Corp., 612 F. Supp. 448 (N.D. Ohio 1984).

211. *Id.* at 453.

212. *Id.*

Still, plaintiffs who suffer property damages or physical injury caused by defective software can sue the developer or provider based on ordinary negligence notwithstanding other claims such as remedies on the contract or as provided by consumer protection laws. Furthermore, in cases where a software developer fails to exercise due care, the developer's liability exists independent of any contractual disclaimer of warranty. Finally, plaintiffs may have a cause of action founded in statutory negligence wherever defective software violates jurisdictional consumer protection law.[213]

B. *Design and Warnings Defects*

The Restatement (Third) of Products Liability permits claims against manufacturers based on injuries allegedly caused by design defects or warnings defects.[214] Both doctrines require a plaintiff to prove the existence of a reasonable alternative design (or warning).[215] This is a non-strict liability standard that amounts to a failure of due care on the part of the manufacturer (and hence, resembles the ordinary tort of negligence).[216] If software is to be considered a product for purposes of tort liability, then the Restatement (Third) doctrines of design and warnings defects (and not manufacturing defect) should apply. In the case of embedded software, physical instrumentalities of the device may have a defective design or fail to properly warn against the risk of injury.[217]

---

213. *See* Jim Prince, *Negligence: Liability for Defective Software*, 33 OKLA. L. REV. 848, 854 (1980).

214. *See* OWEN, *supra* note 19, at 332-52.

215. *Id.*

216. *Id.*

217. This is the main approach taken by Birdsong in the class action against Apple.

> Apple has produced and sold [iPods] . . . which are inherently defective in design and are not sufficiently adorned with adequate warnings regarding the likelihood of hearing loss and specifically the onset of noise induced hearing loss, a condition which has no cure or treatment. Not only are the [iPod]s defectively designed, but the earphones, commonly referred to as "ear buds," that are packaged with each [iPod] are likewise defective in design and do not contain adequate warnings.

Complaint at 1, Birdsong v. Apple Computer, Inc., No. 06-02280, at *1 (N.D. Cal. May 19, 2006). Birdsong does not allege hearing loss in his complaint, rather that he would not have purchased the iPod altogether if he had known of its allegedly dangerous qualities. *See id.*

C. *Product Warranties (Express and Implied)*

Warranty law governs the relationship between the product— seller and consumer and is associated with promises arising out of sales transactions ordinarily conceived as part of the law of contract.[218] Still, warranty law arose from common law tort and was, in fact, the progenitor of strict products liability as conceived in section 402A.[219] Today, versions of the U.C.C.'s Article 2, as adopted by the states, define express and implied warranty rights and obligations.[220] Claims for breach of the implied warranty of merchantability have been described as analogous to the theory of recovery for the sale of defective products under section 402A of the Restatement (Second) and "sometimes [can be a] powerful products liability law claim" that "truly is a form of 'strict' liability."[221] Proof that a product is defective under tort law generally will establish that a product is not merchantable, giving rise to a recovery under the implied warranty.[222]

The U.C.C. allows product sellers to disclaim the warranty of merchantability, albeit with some restrictions. Most modern software disclaims the implied warranties under so-called "shrink-wrap" licenses, which have been upheld.[223] However, implied warranties cannot be disclaimed in all cases, and a particular plaintiff may still have a claim in spite of disclaimers depending on the law in the plaintiff's jurisdiction.[224]

D. *Professional Malpractice Toward Software Developers*

In cases where software is found not to be a product, but rather to be a service, plaintiffs may pursue a tort cause of action founded in malpractice by proving a breach of a duty of reasonable care of a similar professional. It is a sign of the immaturity of the software industry that,

---

218. *See* GEISTFELD, *supra* note 27, at 10.

219. *See* Owen et al., *supra* note 9, at 146.

220. *Id.* at 147.

221. *Id.* at 166-67.

222. *Id.*

223. *See* ProCD, Inc. v. Zeidenberg, 86 F.3d 1447, 1449 (7th Cir. 1996). This is considered the leading case upholding the validity of shrinkwrap licenses, in which the court held that a shrinkwrap license is enforceable as a general matter unless its terms are objectionable on grounds applicable to contracts in general, such as unconscionability.

224. *See* John M. Conley, *Tort Theories of Recovery Against Vendors of Defective Software*, 13 RUTGERS COMPUTER & TECH. L.J. 1, 4 (1987) (arguing U.C.C. remedies are sufficient for plaintiffs injured by defective software). Note that the *Birdsong* complaint alleges violation of both the implied warranty of merchantability as well as the implied warranty of fitness for a particular use as defined in California's Civil Code section 1791.

regardless of a recognized need, professional standards have not developed, and therefore, no commonly-accepted definition of a "professional computer programmer" exists.[225] Understandably, courts have refused to recognize a new tort of computer malpractice.[226] However, some courts indicate their willingness to consider an elevated standard of care for professional programmers. For example, in *Diversified Graphics, Ltd. v. Groves*,[227] the court decided that a computer consultant could be held to a professional standard of "degree of skill and care" which was a matter of fact to be decided by the jury.[228] Furthermore, "professional persons *and* those engaged in any work or trade requiring special skill must possess a minimum of special knowledge and ability as well as exercise reasonable care."[229] Additional cases show a brief trend in the late 1980s where courts began recognizing an elevated professional standard for software developers.[230]

Therefore, a cause of action may lie in proof that a software developer failed to meet a minimum standard of skill or failed to exercise professional reasonable care in the development of the software. Still, there is no licensing authority for programmers, no standards embodied in law, and no minimum educational requirement. However, as the industry matures, it may become possible for courts to begin to shape the outlines of a professional malpractice standard for computer programmers.

E. *Hardware-Based Products Liability*

Because embedded software must work with a physical component (the device) to cause injury, a separate cause of action may lie for manufacture, design, or warnings defects.[231] In the *Birdsong* complaint, the plaintiff alleges a defective design of the iPod itself (presumably including the software), and separately, a defective design of the "ear bud" style

---

225. *See* Patricia Haney DiRuggiero, Note, *The Professionalism of Computer Practitioners: A Case for Certification*, 25 SUFFOLK U. L. REV. 1139, 1140 (1991).

226. *See* Analysts Int'l Corp. v. Recycled Paper Prods., Inc., No. 85 C 8637, 1987 U.S. Dist. LEXIS 5611, at *6 (N.D. Ill. June 19, 1987); *see also* Chatlos Sys., Inc. v. Nat'l Cash Register Corp., 479 F. Supp. 738, 740-41 n.1 (D. N.J. 1979).

227. Diversified Graphics, Ltd. v. Groves, 868 F.2d 293 (8th Cir. 1989).

228. *Id.* at 296.

229. *Id.* (citing LeSueur Creamery, Inc. v. Haskon, Inc., 660 F.2d 342, 348 (8th Cir. 1981)).

230. *See, e.g.*, Data Processing Servs., Inc. v. L.H. Smith Oil Corp., 492 N.E.2d 314, 319-20 (Ind. Ct. App. 1986) (computer programmers "hold themselves out to the world as possessing skill and qualifications in their respective trades or professions [and] impliedly represent they possess the skill and will exhibit the diligence ordinarily possessed by well informed members of the trade or profession.").

231. *See* OWEN, *supra* note 19, at 332-52.

headphones.[232] The "ear buds" are inserted directly into the listener's ears and form a complete seal, which reduces external noise and enhances the music sound quality.[233] If Birdsong can demonstrate that the design of the ear bud headphones is unreasonably dangerous, then a recovery based solely on the hardware design would be available; the software would not be implicated at all.[234] Even if the ear bud headphones were provided to Apple by a component manufacturer, Apple would remain liable as the final assembler.[235] Insofar as the hardware is implicated in the injury, separate causes of action for design, manufacturing, or warnings defects may exist.[236] Furthermore, a strict liability claim for manufacturing defects remains available to plaintiffs when the hardware component of a particular product fails to work as designed due to a problem in the manufacturing process.[237]

F. *Misrepresentation (Negligent and Fraudulent)*

Plaintiffs injured by devices containing embedded software may have a cause of action for misrepresentation, including in some jurisdictions, a strict liability action not requiring any proof of defect at all.[238] Claims of misrepresentation can be fraudulent (intentionally deceitful) or negligent.[239] In either case, the outcome will turn on the actual communications made by the product seller—particular words, either written or spoken.[240] Fraudulent misrepresentation requires proof of several elements (varying by jurisdiction), including intent to deceive, also called *scienter*.[241] Negligent misrepresentation replaces the scienter requirement with a duty of reasonable care.[242]

The Restatement (Second) of Torts section 402B describes the elements of a claim for strict products liability in tort for

---

232. *See supra* note 1.

233. *Id.*

234. *Id.*

235. Although Apple might implead the component manufacturer of the headphones.

236. *See* OWEN, *supra* note 19, at 332-52.

237. For example, if the battery in a particular iPod were miswired at the factory, and as a result over-loud volume was played and damaged the plaintiff's hearing.

238. *See* Owen et al., *supra* note 9, at 111-12.

239. *See* OWEN, *supra* note 19, at 112-15.

240. *See id.* at 115.

241. *See id.* at 121.

242. *Id.*

misrepresentation, which has been adopted in a minority of states.[243] Under this doctrine, even a non-negligent seller is liable for a consumer's injuries if the consumer justifiably relies on the seller's affirmative, material, false, factual communications.[244] Furthermore, the consumer's reliance on the communication must proximately cause the injury.[245] Therefore, a manufacturer's description of its product as "completely safe" can give rise to such liability when the product injures the consumer.[246]

G. *Proof of Causation*

Proving, without legions of experts, defects in products containing software is not impossible. Advocates of strict products liability for software express concern about the inherent difficulty to search a massive program's source codes and to identify with specificity a particular bug or defect.[247] But plaintiffs need not find the precise line of code where the defect occurred.

The plaintiff's burden of proving a design or manufacturing defect is much lower than this standard. In fact, the Restatement (Third) specifically addresses the burden of proof issue:

> [W]hen circumstantial evidence supports the conclusion that a defect was a contributing cause of the harm and that the defect existed at the time of sale, it is unnecessary to identify the specific nature of the defect and meet the requisites of § 2. Section 3 frees the plaintiff from the strictures of § 2 in circumstances in which common experience teaches that an inference of defect may be warranted under the specific facts, including the failure of the product to perform its manifestly intended function.[248]

Furthermore, as technology has advanced, plaintiffs have acquired access to a plethora of tools that reduce their cost of proving defects.[249]

---

243. Support for the doctrine of strict products liability for misrepresentation has been found in perhaps as many as eighteen states. *Id.* at 136 n.8. It has also been reaffirmed by the Restatement (Third) of Torts: Products Liability § 9 cmt. b (1998).

244. *See* OWEN, *supra* note 19, at 126.

245. *Id.* at 141.

246. *See* Hauter v. Zogarts, 534 P.2d 377, 381 (Cal. 1975) (the "Golfing Gizmo" case).

247. Of course, the plaintiff's concern that such a defect would be impossible to find echoes the defendant's assertions that such bugs are inherently impossible to guard against.

248. RESTATEMENT (THIRD) OF TORTS: PRODUCTS LIABILITY § 2 cmt. b (1998).

249. For example, debuggers and reverse engineering tools are more robust than they have ever been.

Typically, defendant software manufacturers' documents (such as e-mail) regarding product design, testing, troubleshooting, and experiences with other customers are all available via discovery in searchable electronic format.[250] Most defects in embedded software are reproducible once identified and can be demonstrated in the courtroom.[251]

Furthermore, an enormous worldwide community of programmers is instantly available online for expert assistance. Not just passive resources, these communities of programmers frequently and proactively conduct community "peer reviews" of commercial software and document shortcomings.[252] Finally, while it is true that software tools and standards are in a constant state of flux, it is also true that some common testing standards are emerging out of the maelstrom. Plaintiffs can reasonably expect a software developer defendant to produce evidence of testing, particularly of products that implicate the potential for human injury.[253]

Just ten years ago, commentators probably questioned whether software was a "black box," which was impenetrable by an ordinary consumer injured by such defective software. But the introduction in 1998 of the Restatement (Third)'s more flexible evidentiary standards, the increasing availability of low-cost methods of proving design defects in software, and the emergence of some consensus on testing standards provide plaintiffs with tools that alleviate the evidentiary challenges posed by defective software.

V. CONCLUSION AND RECOMMENDATIONS

Judges may hold the line against an inappropriate application of strict products liability to software simply by continuing to recognize that software is not a product for the purposes of tort products liability. Such a course would be wholly consistent with prior case law; the current momentum in tort reform, which leans against such a major extension of products liability; the intangible nature of software; and the policies behind strict products liability, which would be perverse if applied to software at this stage in the industry's lifecycle. Still, this is not a perfect solution.

---

250. *See, e.g.*, FED. R. CIV. P. 34(b) (2008).

251. For example, Birdsong could measure the decibel output of a set of iPod ear bud headphones.

252. This trend is seen clearly in the "white hat" movement where ethical hackers test commercial software for security defects. *See* Wikipedia, *supra* note 61.

253. For but a single example, *see* Wikipedia, *Unit Testing*, http://en.wikipedia.org/wiki/Unit_testing (defining Unit Testing) (as of Sept. 19, 2007, 11:30 EST).

Refusing a products liability claim altogether may deny some deserving plaintiffs, in some eccentric cases, access to recourse for injuries caused by negligently designed software. Furthermore, despite software being wholly a product of the human mind, when embedded into the otherwise mundane devices which permeate our lives, it begins, more and more, to resemble a commodity. It is also becoming more and more difficult to separate the hardware from the embedded software.

Finally, software, unlike other forms of intangible intellectual property, has the unique ability to cause physical injury. To date, the courts have yet to face a case involving a software-based injury that shocks the conscience.[254] However, at some point in the future, the courts will inevitably face such a case. Therefore, judges may reconsider the issue and decide that software can in some circumstances be defined as a "product."

If products liability suits are to be permitted against software (by allowing software to be a "product"), courts should adopt the Restatement (Third)'s categorization of defects. However, plaintiffs should be limited to the negligence-based categories of design and warnings defects except in cases where true unique manufacturing defects have occurred and caused injury, typically in the physical or hardware part of the device.[255] Plaintiffs' lawyers will at some point mistakenly analogize "bugs" to manufacturing defects, but judges should focus on the policies underlying the distinction and recognize that a bug is more appropriately considered a design defect, not a manufacturing defect. A bug is not a "one-in-a-million" defect like a weakened glass cola bottle. A bug is more like a lower quality screw selected to hold a saw blade in place, which is a design defect.[256] Further, courts should recognize that, at the current state of the art, software is, to some extent, unavoidably unsafe.

Hence, providers should be held to a just standard of reasonable care. This requires them to exercise prudence in testing and to make a reasonable attempt to anticipate errors and the potential for injury. Developers of software with the obvious potential to cause injury[257] should, for example, be expected to provide evidence of software testing procedures used in the development of the software and document

---

254. One example might be a case where a software developer who knew of the risk, perhaps through prior warnings from consumers, but failed to make changes which would have prevented subsequent injuries from occurring.

255. The author regretfully predicts that it will become more and more difficult to separate the software and hardware components from each other, however.

256. The discussion of the basis for determining that a software bug is a design rather than a manufacturing defect is a topic for a completely separate article.

257. A few examples are software which identifies drug interactions, tells a diabetic when to take insulin, initiates a smoke alarm, or deploys an air bag.

consideration of user safety issues. This will effectively take strict products liability off the table and require plaintiffs to prove elements, including: (a) that the software manufacturer should reasonably have foreseen the danger; (b) that the manufacturer had the ability, through reasonable care, to eliminate the defect or that a safer design should reasonably have been adopted; and (c) that a risk-utility analysis shows that it would have been efficient for the manufacturer to expend the effort required to identify and eliminate the defect.

Legislative protection for software is needed[258] and should be included in ongoing tort reform efforts. Appropriate legislation should formally recognize (a) that product liability suits against software are to be decided under the Restatement (Third); (b) that software defects may only be categorized as design or warnings defects (i.e., bugs are not manufacturing defects); and (c) that social policy and the state of the art of software require that the industry be exempt from strict products liability. Plaintiffs' lawyers might take comfort in the recognition of software as a product (allowing claims arising from design and warnings defects) even if strict liability is not available for suits arising from software. The drafters of the Restatement of Torts (Third) should also codify that manufacturing defects are not applicable to computer software (whether embedded or otherwise).

The software industry must be allowed the opportunity to survive its infancy and adolescence and to deliver on its promise of "a technological revolution that will rival the industrial revolution in its [beneficial] impact on society."[259] Regardless of the potential risk posed by embedded software, and in spite of some near-misses (such as the year 2000 bug crisis), software has not created undue social costs of uncompensated injury. Given this fact, and given the enormous benefit society realizes from the software industry, courts should be in no hurry to extend strict products liability to software and legislatures should respond appropriately to protect this immature industry.

---

258. *See* James A. Henderson, Jr., *Why Creative Judging Won't Save the Products Liability System*, 11 HOFSTRA L. REV. 845, 845 (1983) (stating "Especially in cases involving allegedly defective product designs, courts have encountered difficulties trying to reach results that are consistent with each other and with underlying notions of public policy.").

259. Maule, *supra* note 162, at 755.