Software Defect Classes and No-fault Liability

Clark Savage Turner
Debra J. Richardson

UCI-ICS Technical Report No. 99-17
Department of Information and Computer Science
University of California, Irvine, CA. 92697-3425

April 5, 1999

Software Defect Classes and No-fault Liability

Clark Savage Turner
Debra J. Richardson

Department of Information and Computer Science
University of California, Irvine, CA. 92697

Technical Report No. 99-17

April 5, 1999

Abstract

Software is increasingly used to control systems that can cause harm to the consumer. There is consensus among commentators that many software systems are "products" for purposes of the law of products liability. However, strict products liability in tort has yet to be applied to software products. The day is certainly coming.

The *Restatement of the Law, Third, Products Liability*, explains that the legal standard for defect in design is based in negligence, while the standard for defect in manufacturing is strict. Expected costs for software developers are higher for defects in manufacture because "due care" provides no defense under the strict standard. The mere existence of the defect that causes injury will trigger liability for damages. Software defects have not yet been characterized relative to these categories.

The necessary characterization of defects depends on the existence of an algorithm that can distinguish product design from manufacture; deliberate engineering decisions from nondeliberative activities in implementing the design. Solution is sought by analogy to working algorithms used in traditional engineering domains. Due to some properties fundamental to software, no such algorithm can be found. Neither legal nor software engineering notions of defect present workable alternatives.

Introduction

Software[1] is a relatively new technological artifact to reach the consumer market. It has made possible a new array of technological possibilities with its speed, efficiency and general applicability. It is increasingly being tasked to control products that are very valuable to society, but have the potential for personal injury. Such products have already been involved in cases of personal injury and death.[2]

When a consumer is injured and the common law of products liability is invoked, two different legal standards are available: negligence and strict liability. These standards have different incentive structures for the parties and may result in very different outcomes for similar facts. The application of each standard to the proper case is very important to support the overall goals of the tort liability system. The applicable legal standard is determined relative to the category of the particular defect under consideration. The proper legal categorization of product defect is therefore critical to proper operation of the law of products liability.

Software as a product may be fundamentally different from traditionally engineered products.[3] If this is true, to what extent will the nature of this new product affect the application of traditional legal and engineering tools and defect classifications

[1] For a definition of "software," see generally Schach, CLASSICAL AND OBJECT-ORIENTED SOFTWARE ENGINEERING, 4TH ED., McGraw-Hill, 1999; Gemignani, *Product Liability and Software*, 8 Rutgers Computer & Technology L. J., 173 (1981).
[2] Leveson, Turner, *An Investigation of the Therac-25 Accidents*, IEEE Computer, Vol. 26, no. 7, July 1993.
[3] Hamlet, *Are We Testing for True Reliability?* IEEE Software, 21, July 1992.

used to determine the standard of liability? This is the central question for this paper.

Organization of this Paper

In Part 1, the basics of the law of products liability are summarized, and emphasis is put on concepts critical to analysis of defects classification. In Part 2, the software product is introduced and realistically assessed for applicability of strict products liability.

Part 1: Strict Products Liability

The modern law of products liability developed from roots in negligence and warranty causes of action. In the 1960's *strict products liability in tort* developed in response to the perceived inadequacies of negligence and warranty causes of action when applied to ***products*** of modern complexity ***involved in personal injury***.[4] It was to be based on proof of product defect rather than proof of fault. Once a product defect was proven to have caused injury to the person, damages could be awarded. On the other hand, for a negligence case, unreasonable conduct must be proved and the injury may be merely economic in nature. The reasonableness of human conduct presents many arguable issues, and this raises expected costs for the

[4] For a good historical view of the development of strict products liability, see generally Birnbaum, *Unmasking the Test for Design Defect: from Negligence [to Warranty] to Strict Liability to Negligence*, 33 Vanderbuilt L. Rev. 593 (1980).

plaintiff in prosecuting a successful case.[5] Defendants have a lower expected cost under this legal standard because they have a chance to win more cases by proof that their conduct was reasonable.

To reiterate, In order to apply the strict products liability standard, personal injury must be involved. Mere dissatisfaction with the product or economic damages will not support a case; neither will the provision of services. The instrumentality causing the injury must be considered a "product." Therefore, the first critical legal inquiry may be whether a product is involved in the injury. Next, the defect must be classified. In the defect classification many critical legal and technical issues arise.

Product "Defect"

The term "defect" has evolved to encompass two general categories: defects in "manufacture" and defects in "design." Both categories have given rise to the same legal label: strict products liability. However, the defect in design has been a controversial category respecting its proper inclusion under the strict products liability rubric.[6]

Courts and commentators have, over time, recognized that the nature of the design defect more properly involves a negligence analysis. This is justified on the grounds

[5] See generally, PROSSER AND KEETON ON TORTS, Fifth Edition, section 99(1), page 695, West Publishing, 1984

[6] The argument has been that design defects should be subject to a negligence standard. See note 4, *supra*..

that the element of human intention is essential to any design inquiry: the defect is *preventable.*[7] The manufacturing defect correctly entails a strict liability analysis since its essence is inadvertence in product construction: it is *inevitable* that such defects occur to some degree.[8] The *Restatement of Products Liability 3d,*[9] *section 2,* recognizes this in its definition of the categories of product defect:

> **Categories of Product Defect**
>
> **A product is defective when, at the time of sale or distribution, it contains a manufacturing defect, is defective in design, or is defective because of inadequate instructions or warnings. A product:**
>
> **(a) contains a manufacturing defect when the product departs from its intended design even though all possible care was exercised in the preparation and marketing of the product;**
>
> **(b) 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 by the seller or other distributor, or a predecessor in the commercial chain of distribution, and the omission of the alternative design renders the product not reasonably safe; [...]**

Notice that the manufacturing defect depends on a failure to satisfy "intended design" and that due care (negligence analysis) is explicitly excluded from consideration. Any application of this definition depends on a determination of intended design for comparison to the product itself. Defect in design entails the

[7] See generally, Owen, *Defectiveness Restated: Exploding the "Strict" Products Liability Myth*, 1996 U. Ill. L. Rev. 743.
[8] *Id*.
[9] Restatement Third, Torts: Products Liability, American Law Institute, 1998.

failure to consider and adopt a reasonable alternative design that would have made the product reasonably safe, a basic negligence analysis.[10]

A continuing role for the strict liability standard

Some might say that strict liability in tort fades in importance since design litigation is now seen to be based in negligence. However, to the extent that the strict liability standard is genuinely advantageous to plaintiffs, there will be pressure from the plaintiffs' bar to characterize product defects as defects in manufacture whenever possible.[11]

The ultimate social importance of the strict liability standard itself is demonstrated by the legal treatment of medical devices and pharmaceuticals. Unfettered innovation in these areas of manufacture is considered so very important that the design liability standard is raised well above ordinary negligence for the plaintiff's case.[12] However, even for such an important class of products, the manufacturing defect will still result in strict liability.[13]

[10] These ideas have been discussed long and hard in cases and literature. See, for example, note 7, *supra*.
[11] Note that defects in manufacture can always be characterized simultaneously as defects in design, since any good design should anticipate the possibility of manufacturing defect and plan to avoid or mitigate harmful effects.
[12] See generally, Restatement Third, Products liability, section 6 (c).

Determination of the Standard of Liability: Determination of Defect Category

Commonly understood meanings of the terms "design" and "manufacture" have not proved sufficient to distinguish the categories of defect. The essential characteristics of the two categories have been defined and developed over time by common law decisions. Briefly, several basic characteristics that distinguish these defect categories are shown below in table form.[14] The classification of defects is given across the top of the table and the conceptual dimensions that have been used to distinguish them are given in the rows.

	Design	**Manufacture**
Standard used for comparison	external, a social standard for risk-utility decisions	internal, the manufacturer's own standard is considered
Degree of human intention	conscious decision of the design engineers	inadvertent, a "mistake"
Avoidability of the danger	avoidable by proper risk-utility consideration	unavoidable
Defect "visibility"	visible part of functionality, a planned characteristic of the product	latent, not known before the accident (or QC would have rejected!)
Consumer participation in risk reduction	sometimes consumer found "best" risk avoider	not possible because defect is latent

FIGURE 1.

[13] See *Id.*, section 6 (b) (1).
[14] See generally the discussion and cases cited in *Id.*, comments to section 2.

Standard Used for Comparison: Design is judged "defective" by a social standard. The Court or jury must decide whether the design intention reasonably balances social risks and utility. In contrast, a manufacturing defect is found by comparison of the product to the manufacturer's own technical standards.[15] If the product is defective by internal technical standards, it is more dangerous than it was designed to be![16]

Degree of Human Intention: Design defects may be avoided by a socially responsible risk-utility consideration during design. Manufacturing defects cannot be eliminated this way, they are not the result of "consideration" of alternatives at all, but are failures in the process of construction of the product.[17]

Avoidability of the Danger Design defects involve conscious decisions of the design engineers. Manufacturing defects are not the result of conscious decisions but of inadvertence. The manufacturer knows that a certain amount of imperfection results from the construction process, regardless of the care taken in quality control.[18]

Defect Visibility: Design features define the product's functionality. Thus, any defect in design is a consciously chosen characteristic for the product. In this sense, the defects are "known" and "visible" to anyone who understands the product.

---

[15] See generally, Prentis v. Yale Mfg. Co., 421 Mich. 670, 365 N.W.2d 179 (Sup. Ct. MI, 1984).
[16] See *supra*, note 5.
[17] See generally *id*.
[18] Restatement of the Law, Torts, Products Liability, section 2, page 16.

Manufacturing defects are not seen or known since they are latent. They are unplanned "features" of the product.

Consumer Participation in Risk Reduction: Some design features include necessary risks in their beneficial use.[19] In such cases, consumers must participate in risk reduction in order to enjoy the product's benefits. Manufacturing defects are latent and consumers cannot generally participate in risk reduction.[20]

Practical Application of the Law

As shown in the *Restatement Third*, Courts seek "intended design" as the marker to determine whether there is a manufacturing defect. Caselaw exhibits two basic ways that Courts actually determine this marker:[21]

1. design specifications
2. deviation from the norm.[22]

[19] Consider the knife. Should manufacturers of knives be held liable when a consumer gets cut by the knife? The cutting is the feature the consumer desires from the product, and is expected to use it with care to minimize the chance of accident. This is a very simplified analysis but does make the point.
[20] For an interesting discussion of the issues, see Henderson & Twerski, *Closing the American Products Liability Fontier: The Rejection of Liability Without Defect*, 66 NYU L. Rev. 1263 (1991).
[21] See, for example, the cases cited in the Restatement Third, Products Liability, section 2 in the Reporter's Notes, comment c, Manufacturing defects.
[22] This test for manufacturing defects was so named by Justice Traynor in Traynor, *The Ways and Meanings of Defective Products and Strict Liability*, 32 Tenn. L. Rev. 363, 367 (1965).

1. Design specifications as an expression of intended design

When the Court searches for the manufacturer's intended design, a natural starting point is internal design documentation for the product. After all, the manufacturer often uses such documentation in its own efforts at quality control. In this sense, these documents can exhibit the manufacturer's intended design: a precise definition of what the manufacturer intended to produce.

Ideal design specification documents contain a complete, consistent, correct, unambiguous, comprehensible expression of the product design. If such design documents were always on hand, if product features were always traceable to their counterparts in the specification, and if the specification counterpart could be used to answer the question, "is the specification satisfied," they would indeed be sufficient to reliably make the desired distinction between design and manufacturing defects.[23]

Though there are cases where design documentation can be used to unambiguously determine whether intended design had been properly executed, ideal documentation is a chimera.[24] Real design documentation often does not provide

[23] Have an expert compare the alleged product defect to the design specifications for discrepancies.

[24] Requirements always contain conflicts such as speed and safety, efficiency and cost, etc. "It is quite impossible for any design to be the 'logical outcome of the requirements' simply becuase, the requirements being in conflict, their logical outcome is an impossibility." Pye, THE NATURE AND AESTHETICS OF DESIGN, Van Nostrand Reinhold Company, 1978.

enough information to make an unambiguous comparison to arbitrary product features.

2. Intended design in the manufacturing norm

If the specification is missing, if it is not comprehensible, if it is incorrect, inconsistent or ambiguous, it cannot reliably be used to divine "intended design" so that the individual product feature may be evaluated against it. Some other way is needed to determine whether a manufacturing defect exists in such cases. The "deviation from the norm" test compares the product in question to a number of others from the same production run.[25] If the given product defect is found in all the others, then it is said to be one of design intention. This fact gives rise to the descriptive term `generic' defects, referring to those defects that affect the entire product line (by design).[26] If the product feature in question does not appear in the majority of others, the defect is likely one in manufacturing: an unintentional failure to execute the design properly for that individual product.

This test is easy to understand and its genius lies in the ability to infer intended design from some sample of products - without reference to the design specifications at all. Simple and practical, it circumvents many known problems of a test to the design specifications.

[25] See *supra* note 22.
[26] The term, "generic" as applied to product defects is discussed in Henderson, *supra,* note 20.

Part Two: The Software Product

Software is increasingly used to control physical machines capable of causing [and contributing to causing] personal injuries. Nuclear power plant shutdown systems, medical therapy systems, avionics systems, and automobile ignition and antilock braking systems are some common examples.[27] Personal injury and death has already resulted from the use of such computer controlled systems.[28] Caselaw has yet to demonstrate the applicability of strict products liability to software systems.[29]

The Kind of Software Considered in this Paper

Potentially dangerous software systems can:

1. directly contribute to risk of injury by use of an embedded software control system[30] in the context of a physical machine; and,
2. indirectly contribute to the risk by providing inaccurate information or advice[31] to other decisionmakers.

---

[27] See generally the *Risks forum*, usenet: comp.risks. The archives contain volumes of examples.
[28] See *supra* note 2.
[29] But one federal court has stated that software is a product for purposes of products liability law in the context of the aeronautical charts line of cases. See, Winter v. G.P. Putnam's Sons, 938 F. 2d 1033 (9th Cir. 1991).
[30] Often the basic actions must be made so quickly as to eliminate the possibility of human intervention, thus, automatic decisionmaking must be implemented in support of the safety goals of the system.

Examples of the first type of software system include automated nuclear plant shutdown systems, medical linear accelerator systems, and antilock braking systems where system requirements do not allow time for human intervention. Examples of the second type of software system include medical expert systems and automobile mapping display systems where humans must make the ultimate decisions. This work is only concerned with the first type of software, embedded software that controls a physical machine capable of inflicting personal injury.

Software Products Liability

1. Software as a "product" for purposes of products liability

Legal discussions about software and products liability question whether software is really part of a "product" or more in the nature of a "service" performed for a client. If all software is classified as the provision of a service, any strict products liability analysis ends because the case does not involve a product.[32] Though this issue has not been settled by the courts, there is a general consensus among commentators that certain software systems are considered products for the

[31] For example, artificial intelligence expert systems, hospital record systems, computerized warning devices.

[32] The negligence standard applies to the provision of services. If software is indeed seen as service oriented, the attendant professional negligence issues will come to the forefront as pressure from the plaintiff's bar rises. For a general discussion of such issues, see, Kaner. *Software Negligence and Testing Coverage,* Software QA Quarterly, Vol. 2, No. 2, 1995 for a discussion of software professional negligence issues.

purpose of products liability.[33] Software that is part of a hardware system (already a product) that is mass produced for the consumer market is the primary example. The class of software systems considered here falls within this group.[34]

The Scope of the Definition of "Software" as Used in this Paper

The analysis of "software," as the term is used in this paper, will focus on the software design and the source code parts of the software product. Source code is the implementation of the software design solution to a problem in a programming language.[35]

2. Software Defects in "Manufacture"

There is a conceptual difficulty with the term "manufacturing." That term, as it is commonly understood, does not appear to describe many aspects of software development and production.[36] One software researcher explains that the only

---

[33] See generally Miyaki, *Computer Software Defects: Should Computer Software Manufacturers Be Held Strictly Liable For Computer Software Defects?* (Comment) 8 Computer & High Technology Law Journal 121 (1992).

[34] It is an integral part of a physical product. One might also argue that the software is a "component" of a larger product system as in Wolpert, *Product Liability and Software Implicated in Personal Injury*, Defense Counsel Journal, October 1993, 519, 523.

[35] The source code is written by a human being and then translated (by another program) into an isomorphic machine readable form. The machine readable form of the code is used in the computer to bring about the desired result or functionality. The computer is a general purpose machine and cannot perform useful functions without a properly written program. Thus the program may also be considered a necessary *component* of the programmed computer product. The *Restatement of Torts*, section 5 explains that strict liability may be had for defective *components*. This is an additional way to explain strict liability for software even if it is not formally considered a "product."

[36] Recall the difficulty with patentability of software algorithms made difficult by the commonly used term "mathematical algorithms"? The term "algorithm" was used in a

"manufacturing" that occurs during software production is program compilation and diskette reproduction.[37] A legal commentator uses a similar analysis and concludes that the only possible "manufacturing" defects for software occur due to physical diskette or tape flaws, eliminating the possibility of such defects in the code itself.[38] Such analyses depend on the common definition of "manufacturing" as descriptive of the legal essence of the defect.

Other legal commentators do not find the common usage of the term dispositive to their conclusions. They would find manufacturing defects in programmer errors in carrying out the instructions of the software designer.[39] This is a very different view of what constitutes a manufacturing defect in a software product.

As has been shown in the first section, common law courts have discussed the essential characteristics of the categories of product defect in terms other than the labels of "design" and "manufacture."

Software Flaws that are in the Nature of Manufacturing Defects

There is no argument about the nature of a physical defect in a diskette or other physical media: it is a product manufacturing defect in the traditional sense. No engineer designed such a thing as a product feature. However, when the defect is

---

different sense in software than it was in the law, and the law finally caught up to the domain specific meaning.

[37] Parnas, et. al., *Evaluation of Safety-Critical Software*, 33 Communications of the ACM number 6, page 636, 638 (June 1990).

[38] See *supra* note 33, page 532.

not perceived as "physical," when it appears to be a defect in "logic," the legal categories are not clear.

Consider this class of software flaw: the failure of a programmer to satisfy the software design when constructing ***source code*** *using a compiled, high level programming language.*

1. Basic Characteristics of a Defect in Manufacture

In the previous part, fundamental characteristics of defects in manufacture are contrasted with fundamental characteristics of defects in design. Recall, as shown in figure 1, fundamental characteristics of the defect in manufacture are:

1. the flaw itself is discovered by comparison to the internal design standard of the manufacturer;
2. the essence of the defect is inadvertence, not inadequate engineering intention;
3. the danger due to the defect is not avoidable by improved engineering analysis;
4. the defect is not known beforehand, it is latent;
5. the consumer is generally powerless to avoid the danger due to the defect because it is latent.

The standard for defining a manufacturing defect is an internal one. Correctness of software source code utilizes an internal standard, ideally the software design specification. Notice that risk-utility considerations carefully done by the software designers (during design) cannot eliminate these sorts of coding flaws, human

[39] Brannigan & Dayhoff, *Liability for Personal Injuries Caused by Defective Medical*

programmers can and will make mistakes.[40] Also notice that these flaws are certainly unintentional, programmers are trained and will generally work to produce correct code. Software programming flaws share the characteristic of latency, for if they had been found, software quality control is responsible to reject the product and the flaw must be corrected. Finally, the software consumer or user doesn't normally have a chance to minimize risks due to such flaws because the flaws are latent. Overall, this sort of flaw shares all the common characteristics of a manufacturing defect.

2. Legal Definition of a Defect in Manufacture

The defect in source code considered above fits the general characteristics of a manufacturing defect in products liability, but does it fit the formal legal definition? Recall the salient part of the *Restatement* definition:

> **... when *the product departs from its intended design* even though all possible care was exercised in the preparation and marketing of the product; ...** (emphasis is mine).

The essence of this definition is a departure from intended design.

The failure of a programmer to correctly satisfy the software design exhibits a `departure from intended design' by definition. The essence of the defect is the

---

Equipment, 7 Am. J. Law & Medicine No. 2, 123, 125 (1981).

difference between the designer's intention and the programmer's construction of the code product to satisfy that design intention.

Application of the Law to the Software Product

Recall that there are two ways for Courts to divine "intended design" - comparison with the design specifications and comparison with the production norm.

1. Manufacturer's software design specifications

Similar to other engineered products, the software product involves human decisions that exhibit engineering tradeoffs:[41] reliability versus safety, cost versus safety, performance versus safety, etc. These decisions are ideally recorded in software product specifications.[42]

The state of the art for software design specifications

Software design specifications that are not "complete" cannot be counted on to express full design intention, they are only a partial expression. Inconsistent specifications can exhibit opposing design decisions depending on what part of the specification is consulted. Incorrect specifications cannot be relied upon to exhibit the true intention of the designers. Ambiguous specifications can be interpreted to

[40] See generally, Parnas, Clements, *A Rational Design Process, How and Why to Fake It*, IEEE Transactions on Software Engineering, Vol. SE-12, No. 2, 251 (1986).
[41] See generally Leveson, *Safeware*, Addison-Wesley, 1995.

give differing views of design intention with equal authority. Any of these problems with a software design specification interfere with its use as an "oracle" to answer questions about whether the product satisfies its specifications. Software research explains that the problems of consistency, correctness, completeness and ambiguity are serious and continuing ones for software products of nontrivial size and complexity.[43] These practical and omnipresent deficiencies with software specifications become problematic for a Court whose task is to determine whether a software product departs from its design intention.

2. Deviation from the norm

The problems discussed above are not new for engineers in general.[44] Fortunately, the deviation from the norm test can often be applied to determine whether the product departs from its specifications *without reference to the written design specifications*.

Surprisingly, the software flaws considered here may be cast as "generic," that is, the defects will appear in all copies of the product. The flaw is present in the software source code and then compiled into the executable. Copies are made from that executable, therefore, the same flaw appears in all copies of the software product. Thus, software exhibits a completely new class of product defect: *the*

---

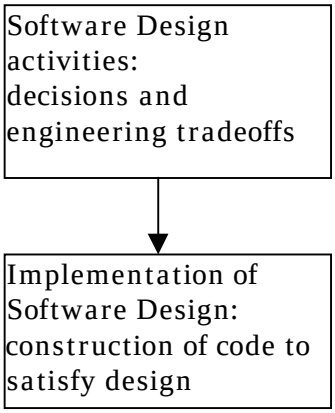
[42] The use of software design specifications is commonly taught in software engineering courses, but no standard methods or models have yet emerged.
[43] See generally, Jaffe, *Completeness, Robustness, and Safety in Real-Time Software Requirements Specifications: A Logical Positivist Looks at Requirements Engineering*, Dissertation, University of California, Irvine, UMI, 1988.
[44] Petroski, TO ENGINEER IS HUMAN, Vintage Books, NY., 1992.

generic manufacturing defect.[45] Notice that the deviation from the norm test fails to distinguish these defects from those of design, there are no "norms" to compare to if all the copies are exactly alike.

Design and Construction of Software Code

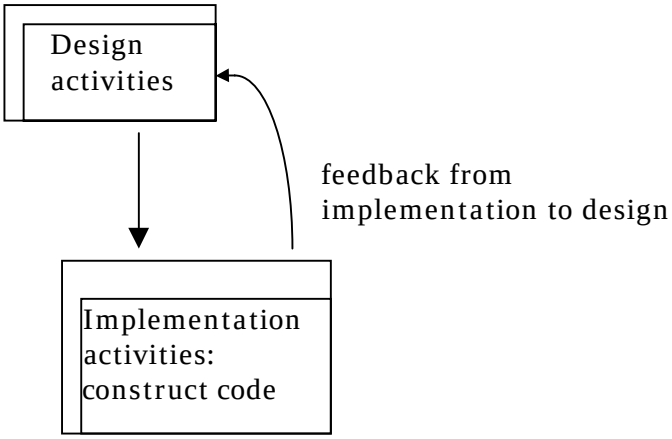
Similarly to all manufactured products, software production is often discussed in terms of discrete stages where distinct activities occur.[46]

Basic view:



However, more realistic views of the software process exhibit feedback loops showing that these stages are not really discrete, but intertwined.[47]

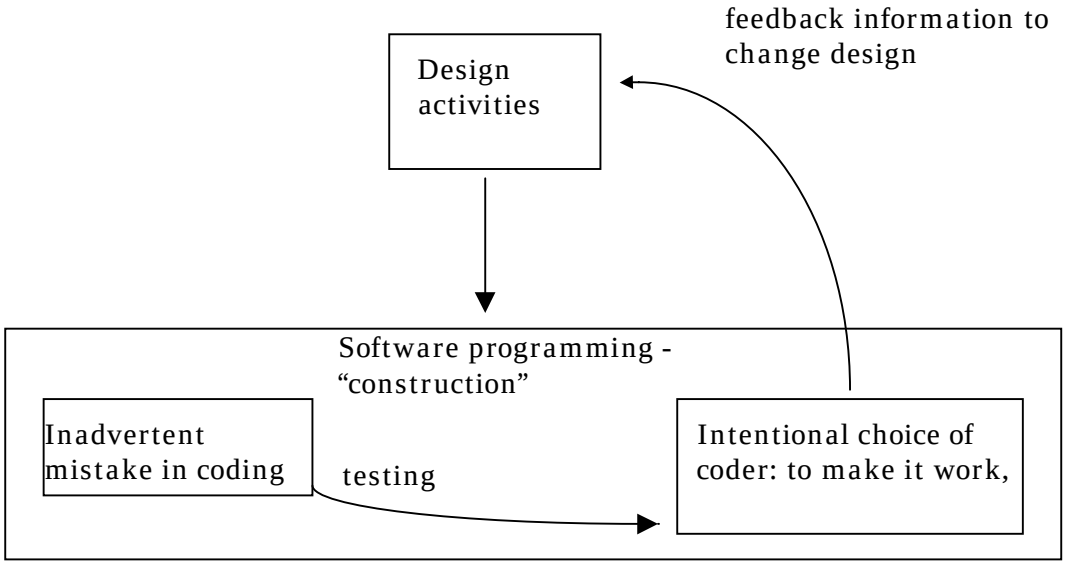
[45] It is interesting to note that another commentator has thought about a similar problem: the "inadvertent design defect." See Henderson, *Judicial Review of Manufacturers' Conscious Design Choices: The Limits of Adjudication*, 73 Columbia Law Review 1530, 1543 (1973). The term "generic manufacturing defect" was suggested by another researcher in this area, Cem Kaner, during our discussions about software defects.
[46] Most evident in the "waterfall model" of the software process, Schach, *supra* note 1.
[47] For a wonderful discussion of this issue, see generally Parnas, *supra* note 41.



Nothing is really new here. This is true for automobile manufacturing, too. There is certainly feedback from the assembly line, especially during the early stages of production, where construction to the given design proves problematic and the designers must rework the design to accommodate production and physical reality.[48] *This mixing of design and implementation activities turns out to be more extensive during software development than for traditional manufacturing of physical artifacts*, and it goes to the heart of our ability to distinguish design intention from construction (implementation) activities in software code.

[48] See generally Petroski, *supra*, note 45.

Here is a more detailed view of the software coding process:



Notice that the feedback loop from programming to design results from broad categories of intentional decisions that are possible, necessary, and occur frequently during the programming activity for software.[49]

The scale and extent of this design activity during construction (programming) for the software product is of a degree and on a scale that is new to engineers.[50] If major design is done concurrently with construction, then the two activities may merge as an activity and the dividing line between them becomes very murky or vanishes.

[49] This is explained in Parnas, *supra,* note 41. Note also that the design choices that occur during construction are easily distinguished from the inadvertent manufacturing defects *for physical products* by the deviation from the norm test.

For many traditionally engineered products, such as automobiles, the medium of design specification is logical description, drawings, models and other ways of capturing design intention so that the product may be constructed in a physical medium, characterized by physical constraints. The design specification can be used to construct a product within acceptable "tolerance" and be said to meet that specification.[51] There is a workable dividing line between the design and construction of the product in that the physical medium is normally distinguishable from the medium of design. And in difficult cases, the deviation from the norm test can be used to place a given defect in a class reasonably well.[52]

For software, the medium of design and the medium of construction are the same. The software coder, in the general sense, is only as constrained as the designer was in the construction of the product. With automobiles and many other traditional physical products, the construction of a particular product is heavily constrained by physical laws, by tooling, by training, by the parts made available by the management in the plant, etc.[53] These constraints are, for the most part, either missing or not as prevalent in software construction. Consider the following chart comparing software products to automobiles:

---

[50] See generally, Parnas, *supra*, note 38.
[51] Notice also that for physical systems characterized by "tolerance," overdesign, or design strength may be used to increase safety. Software cannot be easily characterized by "strength" to increase safety factors.
[52] This test can be used to determine the defect category "well enough" for social and engineering purposes. No arguments have been made that this is a bad way to decide, and it has been in use for some time. See generally Traynor, *supra* note 22.
[53] For example, the guys on the Ford Taurus line cannot just decide to build an Oldsmobile, and they cannot decide to build a boat that day. However, the software programmer is generally as free as the designer!

	SOFTWARE	AUTOMOBILE
MEDIUM of design	Logic	Logic, drawings, ...
MEDIUM of construction	Logic (must be "correct")	Physical (must meet "tolerance")

The coding (software construction) activity is seen as one where both implementation (construction, inadvertent mistakes) and design (intentional choices) activities take place side by side. Since the deviation from the norm test fails for software products, the only possibility for distinguishing a manufacturing from a design defect is to find design intention in the design specifications and documentation.

The Software Design Specification

As seen above, the nature of the software product dictates that design activities necessarily accompany construction activities, recorded and intermingled in the code. Ideally, the design decisions made (or changed) in code are then recorded in some design document through the feedback loop shown above.[54] In the end, after the fact, this would result in a more "complete" set of design documents. However, what is the real incentive structure to such ideal maintenance of the design documents? What is the possibility of creating such ideal sets of design documents for software? There are substantial factors that militate against such an ideal

document capable of distinguishing design intention from inadvertent coding mistakes in the software [code] product:[55]

1. it is expensive
2. it is time consuming
3. it is difficult, maybe impossible
4. many incentives for incomplete, ambiguous design documents[56]

The first 2 reasons may go without much explanation, it takes time and resources to accomplish such ideal documentation for a complex product. The difficulties of creating such ideal documentation are known and active areas of research in the software engineering community. Even the possibility of creating truly ideal design documentation is in doubt among commentators in the field of software engineering,[57] though methods for improvement are of great interest and the subject of large efforts.[58] Even if such ideal documentation could be created (or approached through application of great resources), the realities of the market must be addressed: the limitations of finite amounts of time and other resources.

---

[54] Or maybe in the comments of the source code!
[55] See Parnas, *supra,* note 41.
[56] In response to some discussions I had with a fellow student of software engineering, Arthur Reyes, he explained that in response to the definitions of product defect he would like to create a design document that is always satisfied, so that he could pull all cases against his product into the negligence realm. This is possible by writing specifications that are much less detailed, or at a high level of abstraction, or maybe even "trivial" in the sense that most any product would fit the specification.
[57] The main arguments against such ideal documentation may be found in Parnas, *supra*, note 41.
[58] Certainly the formal methods proponents believe that they have much to add to the value of design documentation, including preciseness and ability to create products consistent with

Conclusions and Consequences

Injuries involving software products will be dealt with under the law of products liability. However, in real cases, we need a method to distinguish the types of defect in order to properly support the goals of the law of products liability: a reasonable balance of product benefit against product risk. This paper shows that there is currently no reliable method by which we can distinguish a manufacturing from a design defect in the software code product.

The obvious solution is to relegate the software product to a pure negligence standard for any defects. This has already been suggested for other reasons.[59] Though this path may be difficult,[60] it certainly is possible and has the appeal of uniformity. However, this path may produce a fundamental anomaly in the common law. Is software, in general, of higher social value than medical devices and pharmaceuticals? Pharmaceuticals and medical devices are subject to the strict liability standard for manufacturing flaws even though the design standard is much different than ordinary negligence in recognition of such high social value. In fact, if software products were subject to a pure negligence standard a strange incentive structure is set up: incorporate more software with more responsibility for safety in any product and enjoy freedom from strict liability for inadvertent coding defects![61]

---

design documents. See generally, Schach, *supra*, note 1 for pointers to some work in this area.

[59] See, for example, Miyaki, *supra*, note 33.

[60] See Kaner, *supra* note 32.

[61] This is an alternative that produces more uncertainty in the field of safety, according to software safety experts. See generally, Leveson, *supra* note 42.

This is in direct opposition to the fundamental goal of products liability: an incentive towards safer products for society.

In conclusion, we see that embedded, real-time software products present new challenges to the law of products liability to form rules that support the basic goals of a safer society with balanced concern for promotion of innovation. Discussions of the core issues must take place now, involving the main stakeholders: software engineers and personal injury attorneys. These discussions must take place before the first cases arise so that all involved may have the benefit of a thorough discussion of the issues that will confront them, and not be stuck making decisions that may prove to be irrational (or oppose basic social goals) in the long run.