Software as Product: The Technical Challenge to Social Notions of Responsibility

Clark Savage Turner, J.D., Ph.D.

Associate Professor, CSC

Cal Poly State University

The Big Picture

Engineering and social notions of defectiveness and responsibility are challenged by the unique nature of the software product!

The law *will* be applied to software

- technical explication necessary
  - software won't "fit" because of its *essential* nature!

Roadmap

- Legal background - legal risk management
  - defect classifications
- Hypothesis - software defect classification
- Software - nature of code defects
- No rational way to classify code defects
- Solution by software engineering progress?
- Conclusions

Terminology

- *Design* - intention or “plan” for a product
- *Safety-critical* - capable of causing or contributing to personal injury (or property damage).
- *Software* - nontrivial, safety-critical, mass marketed
- *Specifications* - requirements, design
- *Design specifications* - same as above
- *Specification sufficiency* - ability of specifications to contain all intentional decisions for code construction
- *Product* - artifact with dangerous potential sold on mass market (contrast with service)

Innovation by Design

- *Homo Faber*: Man, the maker
  - design projects from the known into unknown, possible worlds
  - promise and optimism about benefits to humans
- New artifacts alter arrays of potentialities
  - inevitable social *costs* in new risks
  - ***someone always pays*** **the inevitable costs**!
    - *who* pays has consequences in the market

Social Progress

- Social desire for safety and predictability
  - conflicts with free technical innovation
- Social desire for technical innovation
  - conflicts with safety and predictability
- Society protects / advances its own welfare
  - one way: social notions of responsibility in tort
    - balance risks and benefits of innovative technology
      - common law goal is to optimize social welfare

Tort Law

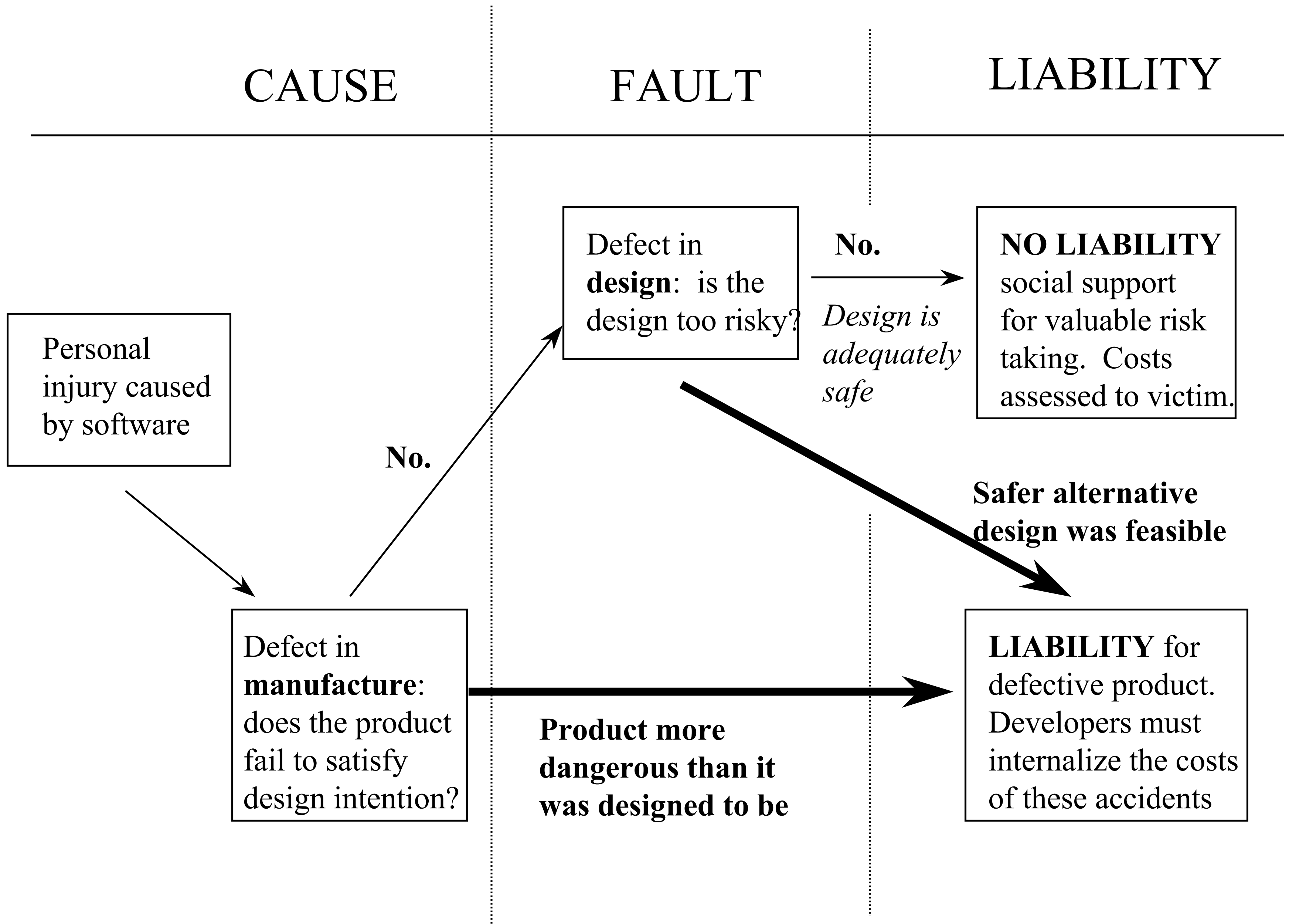
- Social obligations orthogonal to contract
    - common, “judge made” law
        - dynamic, self correcting
        - requires deterministic algorithm that halts
- Purpose: allocate costs of technical progress
    - ***sacrifice victim’s interests***
        - where social progress depends on technical progress
    - ***industry “pays its way”***
        - where social goals are not advanced

Tort Law Meets Risky Artifacts of Design

- "Products" - potentially dangerous artifacts sold to *remote customers*
  - must involve personal injury (or prop damage)
  - inapplicable to pure "services" (malpractice)
- General Rule of Products Liability in Tort: "*One … who sells … a defective product is subject to liability for harm … caused by the* ***defect***. [Res99]

Defect Classification [Res99]

1. Defect in “Manufacture”
   - if product “departs from intended design”
     - internal, technical standard: descriptive (correctness!)
       - risky “mistakes” are not socially beneficial
       - <u>strict standard - “due care” irrelevant</u>
2. Defect in “Design”
   - if design safety is not (socially) defensible
     - external, social standard: normative
       - risky “intention” may bring social benefit
       - <u>negligence standard - “due care” is central</u>


CAUSE
FAULT
LIABILITY
Personal injury caused by software
Defect in **manufacture**: does the product fail to satisfy design intention?
No.
Defect in **design**: is the design too risky?
No.
Design is adequately safe
NO LIABILITY social support for valuable risk taking. Costs assessed to victim.
Safer alternative design was feasible
Product more dangerous than it was designed to be
LIABILITY for defective product. Developers must internalize the costs of these accidents

Distinguishing Defect Class

- Find design intention (engineering question)
  - establishes legal standard: is due care relevant?
    - expected costs to parties can be determined
      - this is a BIG deal
        - who worries about this?
  - legal techniques:
    1. compare to "design specifications"
    2. "deviation from the norm" test
        - independent of designer's specifications!

Enter Software Products

- Innovative artifacts present new risks
  - increasingly used in avionics, nuclear, medical
    - example: Therac-25 medical linac [LT93]
      - 6 massively overdosed
    - no technical solution expected [Lev95]
- No legal precedent yet, but software *will* soon face a products' suit
  - software considered a "product"
    - **disclaimers ineffective**!

My (common) Hypothesis

- Rational classification of code defects by "stage of production" analogy:
  - software design => design intention
  - software code => product construction
    - hypothesize a different analogy?
- Question: can software engineers rationally identify the class of arbitrary code flaws?
    - The answer is NO! (I had to write my Ph.D. work up as a failure.)

Related Work

- Legal research is divided
  - code as design [Wol93]
  - coding mistake as manufacturing defect [BD81]
    - difficulty in software defect classification footnoted
- Software research appears divided
  - [Ham92] and others call code "design"
  - [Bro95] says code "construction" of product
    - note concern with satisfaction of specifications

Overview of Argument

- Code construction issues
- Defects of each class exist in code
  - can we identify the class of an arbitrary defect?
    - operationalize social risk management by tort law
- Extant tests fail to distinguish rationally
  - research seems to offer partial solutions
    - but are they solutions to the right problem?
- Difficulty is *essential,* not accidental

Reality and Code Construction

- One product built and copied identically
  - code and fix
  - waterfall model: discrete stages of production
- Inevitable intertwining [SB82]
  - specifications not self contained
  - pressure on coders to deliver working code
  - code *inevitably contains design* decisions
- Spiral model [Boe88]

Defects in Software Products

- Code has potential for either kind of defect:
  - *manufacture*: failure to satisfy design intention
    - "**x := y * 5"** instead of intended "**x := y + 5**"
  - *design* : intention expressed [only] in code
    - clear whenever specification is insufficient
- Where is "design intention" for code?
  - objective: specifications
  - subjective: coder's mind

Apply Current Tests to Distinguish Defect Class

1. *Deviation from the norm* test
   - **fails**: no deviations at all!
     - *NEW CLASS* - “generic manufacturing defects”
2. Comparison to *specifications*:
   - **fails**: specification insufficiency
     - might “work” for many flaws
     - won’t work for arbitrary flaws
       - specification completeness, consistency and correctness?

Example from Therac code

```
var := 0;
while (activity) do
        var := var + 1;
endwhile;
```

It Won’t Work

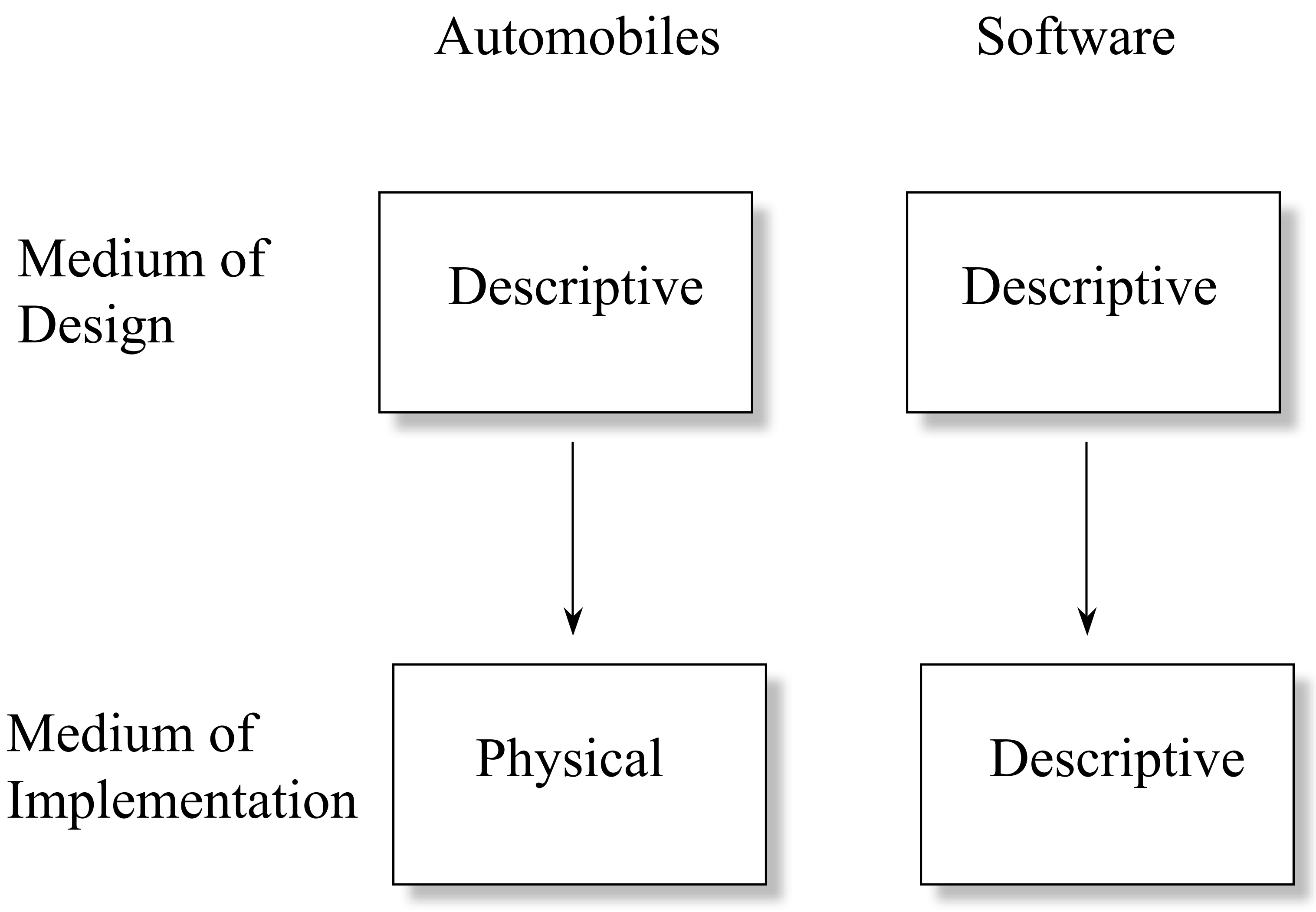
- Specification insufficiency not new [Pet92]
  - “generic manufacturing defects” *are* new
  - *but we must* focus on specifications
- Better software tools and methods to *satisfice?*
  - is software engineering fundamentally a process of “experimentation?”

Software Engineering Progress

- Software research makes progress
  - progress in specification sufficiency:
    - post hoc rationalization [Par86]
    - design standards [Gamma]
    - formal specifications [Fisher]
- Progress is helpful, but for *this* problem?

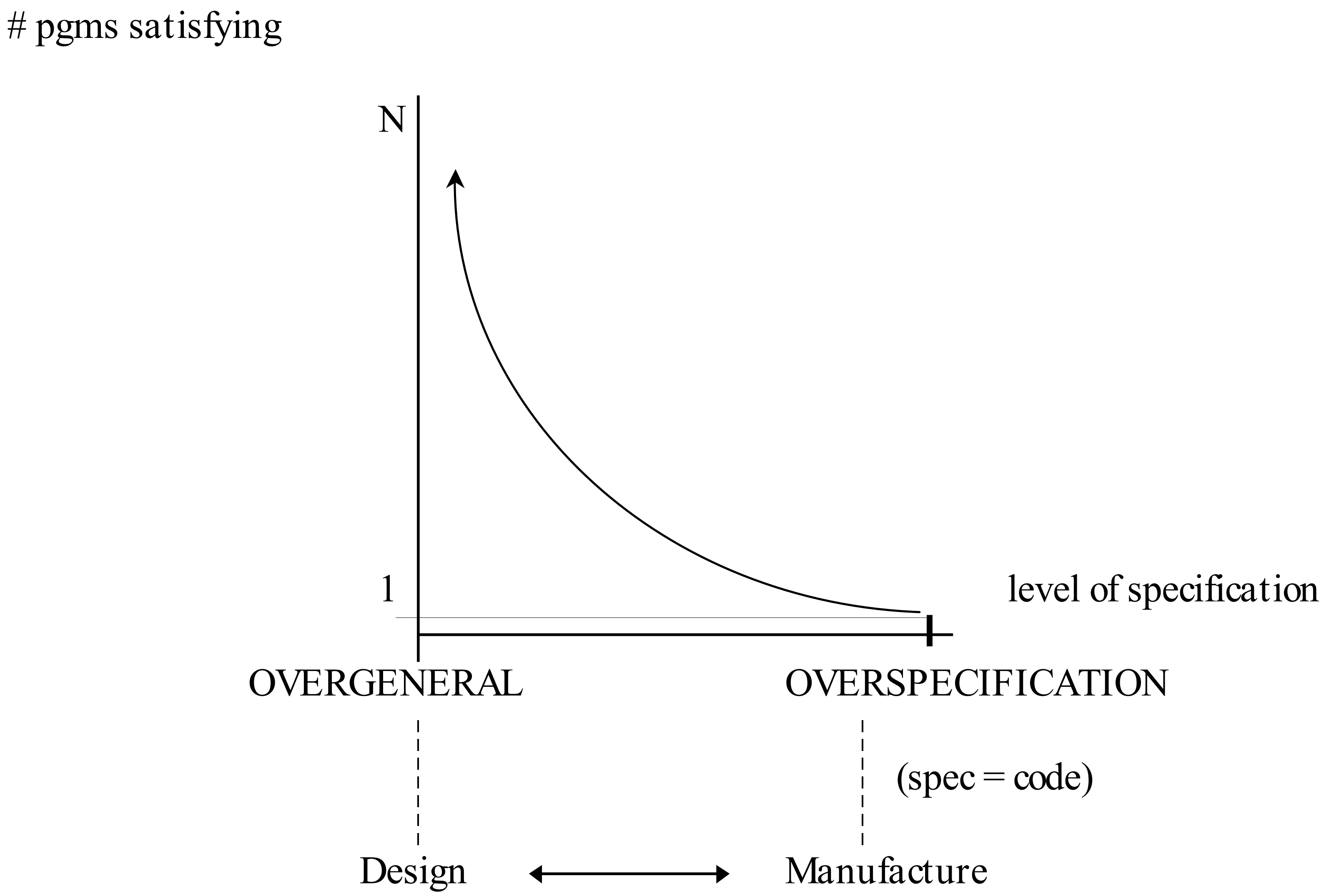
Essential Problems with the Specification Approach

- Software unique among risky products:
  - medium of design = medium of implementation
    - ***requires*** that coders be skilled in manipulation of a design medium.
    - ***enables*** coders to make major design decisions
      - the medium is not constrained like for automobiles
        - “easter eggs”
    - recall ***pressure*** on coders!


Automobiles
Software
Medium of Design
Descriptive
Descriptive
Medium of Implementation
Physical
Descriptive

Is *Any* Distinction Rational?

- Software specification sufficiency a mirage?
  - fix code, then vary level of specification detail
    - range: overgeneral to overspecified
      - note effect on tests to specifications
      - what is the "ideal" level of detail?
      - notice the strange incentive structure set up!
  - code / specification distinctions are subjective
    - inadequate to apply important social objectives through the classification of software defects



# pgms satisfying
N
1
level of specification
OVERGENERAL
OVERSPECIFICATION
(spec = code)
Design
Manufacture

Conclusions

- Software *will* face products liability law
- Courts *must* classify defects
  - the *only* standards *subjective* relative to code!
    - due to the essential nature of the software product
- Rational classification not possible
  - with current social/engineering notion of defect

Implied Conclusion

- Software engineering is inherently risky: it is a process of *experimentation*
  - we eventually find out what states the system might take that involve injury

Social Experimentation [MS89]

- Hypothesis: safe for intended purposes
- Population: users, passengers, patients, etc.
- Levels of experimentation
  - *lab*: counterexamples "fixed"
    - high control, low generalizability
  - *field*: possible lesson for state of the art [Pet85]
    - low control, high generalizability
- We experiment to make progress

Software "Manufacturing" (Implementation?) Defect

- Hypothesis: *This programming <u>product</u>* offers the level of safety I designed into it
  - Liability if my product *fails to meet my own [internal] design standards* involving safety
    - no likely social benefits to random experimentation, no consent
      - simple to prove a case, not much to it?
      - we defeated our own purpose as engineers?
    - standard of judgement is *internal, technical*

Software Design Defect

- Hypothesis: *This design* offers a reasonable level of safety
  - No liability - hypothesis proved true, consent based on *social need for the info*
    - Lots of information developed during the case
      - though consider settlements that allow sealing of records (private interests ahead of public interests?)
  - Liability - hypothesis false, no consent, no social need for this information
    - Standard of judgment is *external*, *social*

Defects in Instructions and Warnings

- Hypothesis: My product includes adequate information about residual design risks to render it reasonably safe for consumers
  - No liability - hypothesis proved, consent given
  - Liability - hypothesis disproved, no consent
    - Standard of judgment is *external, social*

Bibliography (experiment)

- [Bro95] Brooks, The Mythical Man-Month, Addison-Wesley, 1995
- [LT93] Leveson, Turner, "An Investigation of the Therac-25 Accidents," IEEE Computer, July, 1993
- [MS89] Martin, Schinzinger, Ethics in Engineering, 2d Ed., McGraw-Hill, 1989
- [Par90] Parnas, "Evaluation of safety-Critical Software," CACM June, 1990
- [PER84] Perrow, Normal Accidents: Living with High Risk Technologies, Basic Books, NY, 1984
- [PET85] Petroski, To Engineer is Human: The Role of Failure in Successful Design, Vintage, NY, 1992

Bibliography (liability)

- [BD81] Brannigan, Dayhoff, *Liability for Personal Injuries Caused by Defective Medical Equipment,* 7 Am. J. Law and Medicine 123 (1981)
- [Boe88] Boehm, *A Spiral Model of Software Development and Enhancement*, IEEE Computer, May, 1988
- [Bro95] Brooks, The Mythical Man-Month, Addison-Wesley, 1995
- [Ham92] Hamlet, *Are We Testing for True Reliability?*, IEEE Software, July 1992
- [Lev95] Leveson, Safeware, Addison-Wesley, 1995
- [LT93] Leveson, Turner, *An Investigation of the Therac-25 Accidents*, IEEE Computer, July, 1993
- [Par86] Parnas, *A Rational Design Process: How and Why to Fake It*, IEEE TSE, Feb, 1986
- [Pet92] Petroski, To Engineer is Human: The Role of Failure in Successful Design, Vintage, NY, 1992
- [Res99] Restatement of the Law, 3rd, Products Liability, ALI, 1999
- [SB82] Swartout, Balzer, *On the Inevitable intertwining of Specification and Implementation,* CACM, July, 1982
- [TRK96] Turner, Richardson, King, *Legal Sufficiency of Testing Processes*, SAFECOMP 96, Springer
- [Wol93] Wolpert, *Product Liability and Software Implicated in Personal Injury,* Defense Counsel Journal 519, October, 1993