Professional Issues

Part VI:

Liability for Defective Software

Dr. Amanda Sharkey
a.sharkey@shef.ac.uk

Department of Computer Science
University of Sheffield
• Assignment

• News:
  • High Court judges say that Parliament must be allowed to vote on Brexit (triggering Article 50 to leave the EU)
  • Tesco bank: 20,000+ accounts affected by fraudulent activity
1. Introduction

How dependent is society on computer systems and computer decision making?

Recent advances in computer hardware have led to fault-tolerant systems.

*Software has now become the dominant factor in the reliability of computer systems.*

At the same time, the computer industry is facing a problem which has been called the ‘software crisis’.
The Standish Group (www.standishgroup.com) have collected information about IT project failures since 1994.

In a recent report (2009), they noted a marked decrease in project success rates:
32% of all projects succeeding (delivered on time, on budget, with required features and functions),
44% challenged (late, over budget, and/or with less than the required features and functions), and
24% failed (cancelled prior to completion, or delivered and never used).

The Standish Group’s 2011 CHAOS Study shows that 66% of projects are either “challenged” or downright failures, leaving just 34% of projects to be considered successful.

We need ways of enhancing software reliability and tackling the problems of the software crisis
• **1.1. Computer aided mistakes**
  • **1.1.1. Ariane 5**
  • **1.1.2. Cancer Radiotherapy Treatment**
  • **1.1.3. The Airbus Crashes**
Some Computer-Aided Mistakes

The Ariane 5 Launch (right)

Cancer Therapy Treatment (below)

The Airbus Crashes
The Ariane 5 Rocket

- Developed from the Ariane 4 rocket
- Its first flight failed:
  - 40 seconds after leaving the ground,
  - at an altitude of 3700m,
  - the rocket veered sharply to one side,
  - and was automatically destroyed.
- Its payload included equipment built here!
- Video of the launch can be seen at: <http://www.youtube.com/watch?v=kYUrqdUyEpI>.
• Ariane 5 used some software designed for Ariane 4

• Module which ran after launch
  – Did calculations related to velocity
  – Ariane 5 travels faster than Ariane 4
  – Calculations produced numbers bigger than program designed to handle – overflow
  – Caused system to halt.
• The vehicle performed a normal flight until approximately $H_1 + 37$ seconds.
• Shortly after that time, it suddenly veered off its flight path, broke up, and exploded.
• A preliminary investigation of flight data showed:
  – normal behaviour of the launcher up to $H_1 + 36$ seconds;
  – failure of the back-up Inertial Reference System, followed immediately by
  – failure of the active Inertial Reference System;
  – swivelling into the extreme position of the nozzles of the two solid boosters and, slightly later, of the Vulcain engine, causing the launcher to veer abruptly;
  – self-destruction of the launcher correctly triggered by rupture of the links between the solid boosters and the core stage.
• The origin of the failure was thus rapidly narrowed down to the flight control system and more particularly to the Inertial Reference Systems,
  – which obviously ceased to function almost simultaneously at around $H_1 + 36.7$ seconds.
• The launcher started to disintegrate at about $H_1 + 39$ seconds
  – because of high aerodynamic loads,
  – due to an angle of attack of more than 20 degrees,
  – that led to separation of the boosters from the main stage,
  – in turn triggering the self destruct system of the launcher.
• This angle of attack was caused by full nozzle deflections of the solid boosters and the Vulcain main engine.
• These nozzle deflections were commanded by the On-Board Computer (OBC) software on the basis of data transmitted by the active Inertial Reference System (SRI 2).
• Part of these data at that time did not contain proper flight data,
  – but showed a diagnostic bit pattern of the computer of the SRI 2,
  – which was interpreted as flight data.
• The reason why the active SRI 2 did not send correct attitude data was that the unit had declared a failure due to a software exception.
The OBC could not switch to the back-up SRI 1,  
  – because that unit had already ceased to function during the previous data cycle (72 milliseconds period),  
  – for the same reason as SRI 2.

The internal SRI software exception was caused during execution of a data conversion from 64-bit floating point to 16-bit signed integer value.  
  – The floating point number which was converted had a value greater than what could be represented by a 16-bit signed integer - an Operand Error.  
  – The data conversion instructions (in Ada code) were not protected from causing an Operand Error,  
  – although other conversions of comparable variables in the same place in the code were protected.

These data conversions were not protected because it had been assumed for Ariane 4 that the values could not go out of range.

This assumption did not apply to Ariane 5, and so caused the accident.

The full report is at <http://sunnyday.mit.edu/accidents/Ariane5accidentreport.html>.
Radiotherapy Treatment

- Linear accelerators are used to produce a beam of ionising radiation (X-rays) to treat tumours.
- The tumour is treated every day over a number of weeks.
- The regime is calculated to maximise killing of tumour cells and spare surrounding healthy tissues.
- There have been a number of cases in which the delivered dose was higher than the planned dose (e.g. 20% higher).
- These were due to software errors.
- A particular case was the Therac-25 machine.


• Therac-25 incidents 1985-1987
• Massive overdoses of radiation to 6 patients
• Sometimes overdose repeated because machine’s display said no dose given.
• Frequent malfunctions – usually underdoses. Operators used to seeing error messages.
• Little information given about errors.
• Why so many overdoses?
  – Not realised it was the cause of first overdoses
  – Manufacturers insisted errors could not have been caused by machine.
  – First patient told the operator she had been “burned” but told this was impossible
  – Overconfidence in software
  – Later – hardware safety interlocks installed (need to ensure proper protective device in place)
• Automation bias, automation complacency
  • The Glass Cage, Nicholas Carr (2015)
  • 1995 1500 passenger ocean liner Royal Majesty
  • state of the art automated navigation system
  • GPS antenna came loose, navigation system lost its bearings
  • Captain and crew did not notice
  • mate on watch failed to spot navigational buoy they should have passed
    • assumed he had just missed it
    • 20 miles off course, the ship ran aground
Airbus A320

• First fly-by-wire airliner.
• No conventional instruments.
• Several crashes in which:
  – instrumentation indicated wrong altitude,
  – low altitude and lowered landing gear were interpreted as ‘landed’,
  – and would not allow pilot to alter throttle,
  – lack of compression of landing gear was interpreted as ‘not landed’,
  – so braking was delayed (9 s), and
  – the plane ran off the runway.
• Control systems for aircraft
  • mechanical
  • hydraulic
  • fly by wire

• Airbus: as systems fail, control reverts to the computer not the pilot
• Boeing: Fly by wire, but pilots can override the computer
• A320 crash at Mulhouse-Habsheim Airport in 1988
• A low-speed fly by as demonstration flight of Airbus plane
• Should have taken place at 100ft, instead descended to 30 feet.
• 3 passengers died.
• Controversial disputes over causes – Captain blamed, but real cause unclear
• Pilots blamed fly-by-wire system
1.2. Causes of the software crisis

Little data has been collected on the software development process, so there little historical data to guide the estimation of cost and time requirements.

Specifications are often vague. Communication between the developer and the customer is often poor.

Software is difficult to maintain, since maintainability has not been stressed as a criterion for acceptance.

Middle and upper level managers with no background in software are given responsibility for software development.
In his book “The mythical man-month”, Frederick Brooks (1975) notes that software development is not a mechanistic process like manufacturing.

When a software project is behind schedule, you cannot simply add more programmers to catch up.

**Brooks law**: ‘adding man power to a late software engineering project makes it later’
Some factors in Computer System Errors and Failures

- [from Baase, 2003, A Gift of Fire]
- Interaction with physical devices which do not work as expected
- Incompatibility of software and hardware
- Management problems and pressure to get system out quickly
- Inadequate attention to safety risks
- Not planning or designing for unexpected inputs or circumstances
- Insufficient testing
- Reuse of software without adequate testing
- Overconfidence in software
- Carelessness
- Problems with management of the use of a system
  - Data entry errors
  - Inadequate training of users
  - Errors in interpreting results or output
  - Insufficient planning for failures

- Lack of market or legal incentives to do a better job
‘Success actually requires avoiding many separate possible causes of failure’ (Diamond, in Baase, 2003)

1.3. Solutions

There are no complete solutions, but various partial ones.

• Increasing formalisation of software production.

• Procedures for automated mathematical proofs of program correctness,
  – but these are currently limited to relatively small programs
  – E.g. up to a few thousand lines of code.

• Sophisticated products to assist in the creation and testing of computer programs.

• However, software engineering must also encompass new methods for management control and review of software development:
  – managing people is as important as managing the products.
1.4. A quality standard for software

The BSI (British Standards Institution) is the national standards body. The BSI Kitemark indicates that the product has been tested and conforms to standards.

There are currently no standards which refer directly to software production;
   – but there are standards for relevant activities:
     – BS 7925 Part 1: 1998 for software testing;

Quality standards for software should be valuable for two reasons:
   They provide an independent yardstick for both the developer and the customer to resolve quality issues for the project.
   They can be used as a guide for setting up a developer’s own quality control system.

Note that any manufacturer may *claim* to have met a British Standard. However, only products bearing the Kitemark have actually been *tested by the BSI* for conformity to the standard.
1.4.1. The ISO 9000 Quality Standard

The international standard for quality systems is ISO 9000.

This standard covers the production of anything from toys to computer hardware and software.

ISO 9000 requires the manufacturer to have a documented quality system which covers the aspects of the standard.

The monitoring process for a software house involves checking that documents such as requirements specification, design plans, test plans and so on are being produced at an early stage of development, and that they are being adhered to.

Q. Any problems with this scheme?
• Criticisms include emphasis on documentation and certification rather than quality.
2. Liability for Computer Aided Mistakes
WHOSE FAULT IS IT?
Who is responsible?

- Accountability
  - the person (or collective group) who is the appropriate agent
    - to respond
    - to give a report
    - to go to jail or pay compensation
    - to bear guilt and remorse
• Responsibility
  • **Role responsibility**: what’s expected from social role (Computer professional?)
  • **Causal responsibility**
    • John failed to stop at the stop sign and caused a car accident
    • Elena did not put out the camp fire and caused the forest fire
      • (but other factors, e.g. time of year, where the camp fire was)
  • **Blameworthy?**
    • John?
      • his brakes failed?
• Liability
  • how the situation is treated legally
  • who is liable to pay damages or compensate when certain events occur?
  • Legal liability often related to one of the senses of responsibility
• Example: from Deborah Johnson, (2001) Computer Ethics
• Software vendor sells software to customer X, but lies about why the software can do. X installs the package, but it crashes and disrupts X’s business for several weeks, causing loss of revenue
• Software vendor may be liable
• X would have to show that vendor had been dishonest, and failed to fulfil role responsibility
• i.e. that vendor was blameworthy
• Also X would have to show that vendor was causally responsible for the loss of revenue
2. Liability for Computer Aided Mistakes

The determination of liability for computer-aided mistakes depends on a number of factors, including

- whether there is a contract
- the nature of the software
- whether the mistake resulted from an error in a computer program
- whether the person using the software was adequately trained

If a person suffers loss as a result of a defect in some computer product (hardware or software), one or more of the following areas of law may provide a remedy:

- contract law
- law of negligence
- negligent misstatement
- product liability (Consumer Protection Act 1987)
2.1. **Contract law**

- Contractual liability is often the simplest route to a remedy. However, only the parties to a contract can sue upon it.

  *Privity of contract, under common law of contract*

- If software is written for a client, this constitutes a contract for the supply of a service between the software developer and the client.

- Such a contract is covered by the Supply of Goods and Services Act 1982 (SGSA).

- The supply of expert systems or other software which provides advice may also fall within the jurisdiction of the SGSA, since such software could be deemed to be supplying a service.

- However, the SGSA does not apply to other off-the-shelf software, since software is not ‘goods’.
2.1.1. The SGSA and contractual liability

- The SGSA implies a term into contracts for the supply of services that the supplier must carry out the service with reasonable care and skill.

If a supplier does not take reasonable care, he will be in breach of contract and will be liable for damages.

- The definition of ‘reasonable’ is subjective
2.1.1. The SGSA and contractual liability (continued)

Contracts may include exemption clauses, which exclude or restrict the liability of a party who is in breach of contract.

- However, the scope of exemptions is limited by the Unfair Contract Terms Act 1977:
  - Liability for death or personal injury resulting from negligence cannot be excluded or limited.
  - In the case of other damage, liability can only be excluded or restricted where the term purporting to do this satisfied a test of ‘reasonableness’.
  - Liability for defective products under the Consumer Protection Act 1987 cannot be excluded or limited.
2.2. Negligence

Negligence imposes liability on a person who has acted carelessly. To be able to sue in negligence, three essential ingredients must be present:

1. a duty of care owed to the injured party
2. a breach of that duty of care
3. consequential loss, i.e. loss which is a direct result of the breach of duty of care.

The *consequential loss* may be to persons or to property. Errors in software may cause loss indirectly. Errors in hardware may cause loss directly (e.g. electric shock)
Landmark negligence case
• **Landmark negligence case:**
  • Donoghue v Stevenson (1932)
  • Claimant was bought a bottle of ginger beer by a friend in a cafe.
  • Bottle was opaque, and contents could not be seen.
  • The cafe owner poured half of it into a glass, and the claimant drank it. Then the rest of the bottle poured into the glass – included decomposed snail.
  • Claimant suffered shock and gastroenteritis
  • Claimant could not sue in contract – her friend had bought the beer.
  • But House of Lords held that the manufacturer was under legal duty to take reasonable care that the article, which could not be visually inspected, was free from defect likely to cause injury to health.
  • Manufacturer had “duty of care” to those likely to be injured.
• No snails in software or computers, but computer ‘bugs’ could cause injury.
• E.g. Air traffic control
• E.g. Software for calculating tax liability for self-assessment.
2.2.1. Implications of the law of negligence

The fact that an action in negligence does not depend on a contract has important repercussions:

• If a program is licensed by a publisher, the program author may be liable in negligence even though s/he is not a party to the licence agreement.

• A person suffering loss through the negligence of a hardware manufacturer will have a claim in negligence against the manufacturer even though the contract of sale exists with a dealer.
There are limitations to the law of negligence:

• **A duty of care must be owed to the injured party.** A software developer is only liable to those who it is reasonable to believe could be adversely affected by any negligent act or omission.

• The claimant must show that the defendant was negligent

• Also, damages may be reduced if the claimant has contributed to the negligence.

  e.g. a badly made computer that is an electrical hazard - but if a person tampered with the machine, the damages could be reduced in proportion to their contribution to the accident
2.3. Negligent misstatement

- Liability for negligence is significant with regard to software that provides advice. Tortious liability for negligent advice is called *negligent misstatement*.

- If the advice generated by an expert system (decision support system) is incorrect, the developers of the system may be liable in negligent misstatement to the recipient of the advice.

These persons include:

- Experts who provide the knowledge
- Knowledge engineers who formalise the knowledge
- Programmers and analysts

However, two factors may negate or reduce liability:

- The people who develop the system have little control over the way the system is used or interpreted.

- Liability may be avoided by including a disclaimer which excludes legal responsibility for the advice.
• *tortious, from tort*:
• **Tort law**: A body of rights, obligations, and remedies that is applied by courts in civil proceedings to provide relief for persons who have suffered harm from the wrongful acts of others. The person who sustains injury or suffers pecuniary damage as the result of tortious conduct is known as the plaintiff, and the person who is responsible for inflicting the injury and incurs liability for the damage is known as the defendant.
2.3.1. Liability for defective advice from decision support or expert systems

The diagram assumes that experts and knowledge engineers are consultants to the software company and not its employees (this is often the case).
2.3.2. Factors affecting liability for decision support or expert system errors

- A person using a system to advise a client is potentially liable under contract and negligence law.
- As with normal software, liability will not be avoided simply because the program has a fault.
- To determine whether there has been negligence, it will be important to consider whether it is reasonable for the person using the system for the purpose of advising others to rely on the system’s output.
- This will depend on whether the error would have been made by a competent professional having the standard and type of skill that the defendant is held to have.
2.4. Consumer Protection Act 1987

- Product liability imposes a liability on the producer of a defective product. In the UK, product liability is provided for by the **Consumer Protection Act 1987**.
- The CPA protects consumers from faulty or defective products which cause damage to persons or property.
- Manufacturers and suppliers are obliged to ensure that goods conform to the contract description, are fit for the intended purpose and are of reasonable quality.
- Initially, some confusion regarding the applicability of the CPA to software; now clear that software does not meet the definition of ‘product’ used in the Act.
- The CPA *does apply to computer hardware*.
- Manufacturers would be liable under the CPA if links between VDU use and illness are ever proven.
- Product liability does not apply to software
- But it does apply to a defective product that incorporates software
- E.g. Defect in software that controls a microwave oven will result in the microwave being defective.
• But Consumer Protection Act does allow ‘state of the art defence’

• Manufacturers can claim that they cannot be held liable for designs or products that reflect ‘state of the art’

• i.e. the state of scientific and technical knowledge at the relevant time was not such that the producer of the product might be expected to have discovered the defect
• See also **Health and Safety at Work Act 1974**
• General responsibility for health and safety precautions in the workplace.
• But most responsibility falls to employers
• - infringement of act is a **criminal offence**.

• Act places a duty on designers and manufacturers to ensure equipment is safe.
• Implications for safety-critical software: if worker or member of the public is injured as the result of an error in software of computer controlled machine, software developers could be liable for prosecution if they cannot demonstrate sufficient care in its design and development.
2.5 Vicarious Liability

• Form of strict secondary liability
• Employers might be liable for employees’ actions
• Parents might be liable for childrens’ actions
• Owners might be liable for animals’ actions

• Artificial Intelligence systems (self-driving cars) might be seen as employees and vicarious liability might apply
3. **Exemption Clauses**

Already mentioned in contract law notes. Two types:

- **Exclusion clause** – total exemption from a breach of contract;
- **Limitation clause** – limits liability to a specified amount.
- Exemption of liability is limited by the **Unfair Contract Terms Act 1977 (UCTA)**.
- UCTA is unclear regarding software, since it states that:

  > Sections 2 to 4 of this Act do not extend to ... Any contract so far as it relates to the creation or transfer of a right or interest in any patent, trademark, copyright, registered design, technical or commercial information or other intellectual property...

- Sections 2 to 4 of the UCTA cover liability for breach, negligence and unreasonable indemnity clauses.
3.1. Does the UCTA apply to software contracts?

- The essence of most software contracts is the granting of a licence to use the software – the creation of a right under copyright law.

- A number of software companies considered that they could largely ignore the UCTA and exclude or restrict their liability for defects.

- The courts have taken a more restrictive approach, as in the following case

3.1.1. The Salvage Association v Cap Financial Services Ltd (1992)

- See Bainbridge pages 222-223 (254 edition 6), Ayers page 106.
The Salvage Association v CAP Financial Services Ltd (1995)

- The Salvage Association invited tenders for the computerisation of its accounting system.
- CAP Financial Services submitted a bid (£30,000) for a feasibility study and was awarded the contract.
- CAP was awarded a second contract (£291,654) to develop and implement the system by July 18th 1988, using ORACLE technology.
- The system was ready for user training in July, but:
  - the software was unusable, and
  - contained a large number of errors requiring a lot of work to resolve.
- New completion dates were agreed, but the claimant terminated the contract on July 13th 1989.
- Salvage Association claimed damages of £291,388 paid under the contracts + £564,162 for wasted expenditure.
- Cap Financial Services claimed that the contract limited their liability to £25,000.
• Both contracts said that the defendant would assign appropriately qualified staff to the work.
• The judge (in the High Court) ruled that there was a breach of these terms (and other issues as well).
• The judge ruled that the Unfair Contract Terms Act 1977 applied to aspects of the contract which did not deal with the creation or transfer of IPR.
• The judge held that the terms limiting liability to £25,000 were unfair, and awarded total damages of £662,926:
  – £291,388 paid by the claimant under the contract,
  – £231,866 for wasted expenditure,
  – £139,672 for wasted management time.
**4. Example**

- A consulting engineer has obtained computer software to help them to design buildings.
- A defect in a design produced using the software could be caused by a fault in the software itself, or by the software being used incorrectly.
- The diagram shows the potential liability of the engineer and the software company towards a client of the engineer, should the client suffer damage because of a fault in the structural design caused by the software or the manner in which the software was used.
4.1. Contractual Liability

- The client can sue the engineer on the basis of the contract between them.

Q. Can the client sue the software company on the basis of contract law?

- The engineer’s contractual liability, under terms implied by the Supply of Goods and Services Act 1982, is to exercise reasonable care and skill in carrying out the service.
- The standard of ‘reasonable care’ is a subjective one, based on the technical competence of the engineer.
- If the software is shown to be defective, the engineer may still be liable to the client.
- The question is whether the engineer has exercised reasonable care and skill in choosing the software, testing it, checking its suitability for the task and in using it.
4.2. Negligence

- A client suffering loss will usually sue on the basis of the contract, although there may also be a legal remedy under the tort of negligence.
- Third parties injured or damaged because of a mistake by the engineer must rely on the law of negligence.
- For example, if a wall designed by the engineer using the software falls down, an injured passer-by could sue the engineer in negligence.
- For a claim of negligence, there must be a duty of care owed to the injured person and a breach of that duty.
- It is likely that the engineer will have a duty of care to the employees using the building he has designed, since it is likely that an employee would be injured should the wall collapse.
- Breach of duty of care will arise even if the design fault is the direct result of an error in the software, if the engineer was negligent in the choice of software, the way it was used or because the output was not checked.
5. Liability for Year 2000 problems

• Were software developers liable for loss suffered because of Year 2000 (Y2K) problems?
• Depends when software was purchased, and if a contract is involved.
• Actions for breach of contract must usually be brought within 6 years, so there was no chance of claiming for problems with legacy systems.
• For recent software, Y2K problem was well known; supplier would be expected to correct errors under warranty. The ‘reasonable care’ principle applied here.
• If ability to process dates beyond 1999 was mentioned (or implied) in contract, supplier was liable.
• Inevitable ‘grey areas’ with systems that were not written so long ago that Y2K compliance wasn’t necessary, nor so recently that Y2K compliance was obviously needed.
A group of **civil engineers** are involved in a major construction project, and have duties that include liaison with **architects** and **builders**, to whom they recommend appropriate structural materials for different parts of the building. The major tool they use to provide this advice is a **program that calculates stress analysis using plans from a CAD package**, and a small **expert system** which contains expertise on the physical properties of building materials such as different kinds of steel, ducting, concrete and insulation.

The stress analysis system has a bug that produces arithmetic errors in some calculations. The engineers do not notice the odd values produced by the system and these values are used as a basis for the new building. In any case, the calculations cannot be checked by other means because they are too complex, too numerous and the project deadlines are pressing. In selling the package to engineers and architects, the developers of the system have promoted it as being the safest and most reliable system in the world.

The expert system used in the package also contains incorrect information. Specifically, the steel used in parts of the building is an alloy that has very good corrosion properties at the cost of a slightly diminished strength. The expert from whom the expert system was constructed did not fully understand the difference in strength with this particular alloy, and hence this factor is not taken into account when calculating the stresses to which the building will be subjected.

Half way through the construction process, the building is unable to support the loads being placed upon it and a crane on the uppermost floor crashes through the lower floors killing two workmen. An analysis of the disaster shows that the arithmetic bug and the misunderstanding of the strength of the steel essentially interacted to bring about the failure. That is, had more conventional materials been used then the errors in the calculations would not have been of any consequence, but when combined with the weaker structural properties of the steel used, the calculations were totally inappropriate.
• **Question 1**
  Who has the greatest moral and ethical responsibility for the accident? The engineers who failed to recognise stress values that were incorrect? The developers and commercial backers who demanded such a tight schedule that the stress calculations could not be checked? The software developers who supplied a faulty product? The ‘expert’ whose knowledge of the properties of building materials was inadequate?

• **Question 2**
  Two parties could consider suing for damages here; the property developer whose building has collapsed, and the families of the dead workmen.
  Under what conditions could these two parties sue, and who would be liable? What is the potential liability under (i) contract law (ii) negligence (iii) negligent misstatement and (iv) product liability.
6. Summary

- The reliability of computer systems is now determined by software.
- The current situation is something of a 'software crisis'. Software is delivered late, doesn't meet requirements and is often of suspect quality.
- Possible solutions to the software crisis are
  (i) formalised software design
  (ii) quality standards.
- ISO 9000 checks the standard of software development. It does not ensure quality of the software product.
- A remedy for computer aided mistakes can be provided by
  (i) contract law
  (ii) the law of negligence
  (iii) negligent misstatement
  (iv) product liability.
- Product liability imposes a liability on the producer of a defective product. Only applies to hardware.
- Software contracts are not exempt from the Unfair Contract Terms Act 1977.