



SHAPIRO
SHER
GUINOT &
SANDLER



Legal Ramifications of Faulty Software

William A. McComas, Esquire

June 2006

Historically Defective Software Had Little Impact

- Used by experts in large corporations (IBM, UNISYS)
- Design was simple (single functions)
- Leading vendors marketed products under tight contracts
- Few lawyers with the expertise to handle such cases
- As software moved from this controlled environment to an open architecture
 - Software's usefulness entered in our everyday lives which meant people tolerated it even when the quality was poor
 - Software has been the fastest growing industry so regulators have stayed away



Now, Faulty Software Is More Than An Inconvenience



- Software errors cost the US \$59.5 billion annually (.6% of gdp) - National Inst. of Stds
- Half the costs are borne by end users rather than developers
- 10 defects per 1,000 lines of code (the typical program size exceeds 1 MM lines of code)
- An insurance company recently spent \$3MM to defend software claims in litigation



Examples of Costly Software Defects

- Year 2000 remediation, litigation and support
- Ground Based Altitude system caused the crash of a plane that killed 228 people (97)
- Recall of 39,000 trucks, tractors, & buses for software glitch in anti-locking brakes (99)
- Mars Polar Lander crashed when software shut the engines off at 100 feet \$165MM
- *State of Ohio v. PeopleSoft* (\$510 Million) – System caused Cleveland State to lose \$5MM in revenue because system did not accurately track and collect bills
- Voting Machines – is the software secure?



Why The Paradigm is Changing

- Software development has become more complex over the years (from single function routines, large dedicated programs, multi-programmed systems to multi-processing systems)
- Commercial pressure to bring products to the market quickly leads to mistakes
- The software industry's lack of liability while manufacturers have been held liable for widgets
- Poor Work Methodology (including bonuses for lines of code) has led to large disasters
- NASA demonstrated that you can build software without bugs – The Space Shuttle programmers developed 3 lines of code a day
- The Software Industry is maturing
- Insurance carriers – Chubb, St. Paul and others are paying out large amounts in claims



The Standard For Code Is Being Raised by others Due to Liability Concerns



- Software is so pervasive in industrial equipment that companies need assurances that their equipment will work
- Plaintiffs lawyers are looking for the next asbestos litigation - membership in ABA computer law group has grown exponentially since 1980
- Data Integrity Summit Jan. 2006 – Alston Bird attorney shocked the audience of CIOs & CFOs by stating that software should be 100% secure
 - However in a survey – 64% of developers said they could not write secure applications
- Who should be held liable? (End User, developer entity or developers personally)
 - Traditionally Corporations (*i.e.*, End Users or the developer of the software)
 - Now the move is towards making developers personally liable
 - “Software developers should be held liable for the security of the code they write” (Howard Schmidt former White House Cyber Security Advisor June 16, 2006)
- As ID theft grows, the risk allocation for software flaws will be the battle ground and may settle the allocation for all software
 - Companies that permit unsecured data transmissions and storage are being held responsible (PetCo, ChoicePoint, Worm penetrating Yahoo mail, VA)



The Software Industry is Recognizing the Demand For Higher Software Standards



- Microsoft's Trustworthy Computing Initiative held up coding for 10 weeks to teach employees to spend more time on quality
- Sustainable Computing Consortium – measures reliability of code



Sources of Legal Liability

- **Strict Liability** – damages caused by or threatened by unreasonable dangerous practices
- **Tort** (negligence) – Developer did not take reasonable steps to . . .
- **Product Warranty** – assurances (statutory or contract) that products purchased will perform as stated (UCC – UCITA)
- **Statutory** – defects in title (*i.e.*, IP infringement)
- **Contract** – I promised to do something and I did not (performance measures)
- **Other sources** – common law, contributory negligence and vicarious liability (responsible for third party acts).



Mitigating Liability Exposure is a three prong approach (Business, technical and legal)



- **Technical –**
 - Developers should invest in tools and methodologies to improve product quality
 - Apply quality control measures from the start to finish
- **Business –**
 - Focus on the development cycle - software defect repair is as much as 100 times more expensive than defect prevention
 - Don't limit ***searching*** for software defects to the development cycle include post production
- **Legal –**
 - Clear third party software (both title and performance) while developing and making updates.
 - Make sure you have contracts in place – Limit Liability, disclaim warranties, and limit indemnities.



