Why a Discussion on Ethical Issues in Software Engineering is Overdue

Klaus BRUNNSTEIN
IFIP TC-9 Chairman
University of Hamburg, Germany

Engineering disciplines have often experienced learning curves in the responsibility of engineers, as legal measures and professional responsibility were often developed significantly later than the technical discipline's methods and tools. Understanding ethical issues related to their work developed in many disciplines often as a consequence of experienced incidents.

Famous incidents such as the sinking of the instably built Swedish warship VASA as well as of TITANIC said to be 'the safest ship in all times', the collapse of the Tay Bridge or the burning of the Zeppelin in Lakehurst are deeply stored in the memory of mankind. Sometimes, incidents ended the career of a discipline (as in the Lakehurst incident). In other cases, it helped, under the incident's impact, to fastly upgrade its methods and knowledge and to equally develop a discipline-related professional behaviour, including discussions of professional consciousness, of partly informal or formal Codes of Professional Behaviour, and to educate responsible engineers.

There were interesting exemptions from such patterns. In ship-building, there are legions of incidents including many tens of thousands of drowned sailors until the 'arts of shipbuilding' (though organized at least since medieval times in guilds with enforced professional standards) developed into a (rather) safety-conscious and ethically mature discipline. On the other side, few disciplines such as bio-engineering and genetics even started with ethical discussions and formalized codes, to prevent serious incidents (but it must be admitted, that these discipline may be too young to judge whether their benevolent intentions will survive contemporary fights of parties interested preferably in markets and growth).

There are also examples such as in nuclear energy production, where early discussions about professional behaviour were deliberately overruled by interest groups in politics and industry, closely assisted by engineering disciplines, with inevitable consequences (e.g. Three-Mile Island and Tchernobyl incidents). In many cases, technicians differentiate between engineers and users responsibility; engineers must build safe technics, and users are responsible to use them safely. From a holistic view, the responsibility of the engineer would also include the product can be safely used: Nuclear energy is a good example how experts sometimes reduce their scope to oversee risks of usage. Public and many experts are today even unaware that significantly safer technical options are available, where the 'nuclear fire'

J. Berleur et al. (eds.), *Ethics of Computing*

would inherently extinguish in case of malfunctions (using a heap of Uranium and C balls, which together form the critical mass by its topology, such that any loss of cooling will result in reducing the heap below its critical mass).

As incidents seems to play a major role in helping a discipline to become mature, what can be said about the status of maturity of Information and Communication Technologies (ICT)? Is it possible to apply the patterns of experiences of other disciplines also to ICT? Compared to other disciplines, ICT exists for a relatively short time (roughly 50 years, about 10 of which widely distributed). Moreover, ICT mixes engineering aspects with many application disciplines and non-engineering sciences. Finally, ICT related disciplines (e.g. Computer Science, Informatics and Cognitive Sciences) may even constitute new types of constructive sciences. ICT applications e.g. in organizations immediately change previous habits often without prior analysis; control of the human brain is often replaced by hard or soft artifacts which users and customers rarely understand and can hardly control. In contrast to traditional disciplines which develop methods and collect experience on a controlled set of reproducible experiments, methods of software engineering produce new realities which immediately replace the old ones, without caring for experience. There are even techniques (e.g. in 'learning systems') where previous test results are reproducible.

It seems that such immediate impact of new concepts, ideas, systems, products or methods in ICT not only produces specific incidents with less human ability to interfere but also shapes the consciousness of too many techno-minded people in the ICT community NOT to care for implication and effects of their work. As examples, some experiences of Software Engineering shall be described:

- In September 1993, a Lufthansa Airbus A-320 crashed at Warsaw airport when pilots were unable to apply computer controlled brakes for 9 seconds; 2 people were killed, and about 60 were injured, many severely. Analysis of the incident revealed that the Automatic Brake System 'performed to specification', according to which safe landing implies that BOTH pairs of main landing gear wheels MUST have ground contact AND all started rolling. When the wind turned, during final approach, fast from front to rear, the plane was significantly (30 mph) too fast, with too much lift on the wings. With only one gear on the runway, pilots did not understand why brakes did not apply; clearly, the designed model and the pilots model did not match. When the other gear finally contacted the wet runway, the remaining 1,000 metres were not sufficient to stop the machine. Evidently, the experienced pilots (one of whom was Lufthansa's check captain) did not fully understand the implemented algorithm; in the critical moment, the model in the pilot's brain differed critically from the implemented model.

  Also on the A-320 airplane (which was appraised by its manufacturer as 'most technologically advanced airliner in the world'), two previous accidents were at least influenced by the pilot's display which did not sufficiently distinguish between two seriously different modes of approach. Only many months after the incidents, an improved display was available from the manufacturer. In critical application areas concerned with life and health, badly designed man-machine interfaces may contribute significantly to serious incidents.

- Autopilots are a significant improvement in ease-of-handling of small to large airplanes. It seems unbelievable that more than 30 cases have been experienced in the last 30 years where B-747 (Jumbo) autopilots decided without any instruction and without apparent reason to make the plane roll and descend, by about 10,000 feet. Recently, the same was

reported from B-757 and B-767. It is hard to believe that after over 30 years of such incidents, neither the origin nor any cure against such system behaviour is visible; evidently, responsible authorities display an attitude such as 'hey, nothing has happened so far, so why worry?'

- In the case of THERAC-25, a computer-controlled medical radiation machine, persons suffering from cancer could be radiated with low-energy electron beams (up to 2 Million Electron Volts, MeV) or high-energy X-ray beams (up to 25 MeV), produced from 25 MeV 'hot' electrons in collisions with a Tungsten target, thus converting hot electrons to hot X-ray. Evidently, such a system must reliably exclude a state where 'hot' electrons fall immediately on a body. Due to bad design, implementation faults as well as bad managerial habits, 2 people were killed and several seriously burnt before partial re-design and software upgrades seem to have cured the problems. This is an example of irresponsible behaviour with respect to not investigating the incidents immediately when they started. Moreover, hardware locks on a smaller machine (THERAC-20) had been removed when upgrading to the larger one; it was irresponsible not to put back these interlocks immediately as government agencies and others recommended.
- Bad design of the man-machine interface was a major source of a military 'expert' system (AEGIS) shooting down a civilian Airbus during the Iran-Iraq war (1st Gulf war), killing over 170 persons. Some essential information on which the 'destroy-the-target' decision was based was not visible on the primary display. Instead of being made accountable for killing of innocent passengers, the officer responsible for shooting the civilian Airbus even received military merits, as he had followed the design philosophy of the system engineers implied in the name (in the Latin sense: Nomen est omen!): AEGIS, shield of Greek Goddess Pallas Athene, was understood (both by old Greeks as well as military system designers) to protect 'the Good Guys' who are inside the shield's scope from 'the Bad Guys' who are outside.
- Contemporary 'security policies' follow the same idea as the AEGIS designers, in modeling ('Bell-LaPadula model') computer systems like military camps, to protect the good guys inside from the bad ones outside. To become a good guy, you must be 'certified', and on a high level of certification, such systems (e.g. administrators) are even regarded as instantiation of trustworthiness ('trusted persons'). Unfortunately, even such concepts have not protected major sensitive systems and data from being invaded (e.g. with 'Trojan Horses') or somehow compromised. Systems and networks on which scientists, governments and enterprises rely on more than any other technology before can easily be 'infected' with self-reproducing software ('viruses', 'worms', 'bacteria') which may spread without control and perform whatever was intended by the creators of such creatures, with more unforeseen effects. What was regarded previously as a nightmare for administrators, recent implementations of 'agents' can now wander freely through systems and networks and introduce such self-reproducing software, without any means of control over such agents. This again is hardly a sign of professional maturity of those who propagate such ideas!
- Since the summer 1994, the scientific community (and later the public) became aware of the fact that a divide (DVI) instruction of INTEL's Pentium processor may not work properly under certain conditions. Only then did several scientists detect that months of calculations had produced unreliable results; some even complained about this finding although responsible scientific behaviour requires accuracy tests at any stage. Moreover,

even better educated users do not understand that microprocessors are products of special forms of software (better: firmware) engineering. As in software, bugs are not rare in chips; indeed, previous INTEL processors (eso. 80386 and 80486) had bugs in (floating point) arithmetic.

Such incident analysis reveals some basic paradigms which unavoidably produce risk-prone systems. Based on the heritage of von Neumann's assumption about basic concepts of computers, deficiencies of the ICT community in understanding such problems can be observed in contemporary discussion on RISC versus CISC. With performance as the dominant goal (the Olympic words: 'citius, altius, fortius' deliberately avoid modern concepts like 'control' and 'safety'), Complex Instruction Sets of Computers are Reduced to those sets efficiently to be executed very often. Unfortunately, instructions used to safeguard the status of a process or a program's memory against access by other processes do NOT belong to the often executed instructions. Therefore, RISC enlarges the risk as it diminishes protection levels. The use of RISC architectures in economic applications may be compared to a person who goes into rocky mountains with dancing shoes; what one needed in such environments, were spikes to guarantee good contact with the hard rocks. Similarly, better protection needs advanced CISC, while using RISC is risky. There are many more examples where safety looses over performance, thus making incidents less avoidable.

Such selected examples demonstrate how urgently a discussion of Professional Conduct and Codes of Ethics is needed. Let's get started.