Computers and Risk

EECE 692

Presented by Chukwunweike Ugbome

Howard University

Computers and Risk

- “We seem not to trust one another as much as would be desirable. In lieu of trusting each other, are we putting too much trust in our technology?… Perhaps we are not educating our children sufficiently well to understand the reasonable uses and limits of technology”

  -T.B. Sheridan

  Trustworthiness of command and Control System

Computers

- Defined as the electronic machine..
- It's invention 50 years ago has drastically altered our society
- The uniqueness and power of the digital computer over other machines stems from the fact that, for the first time, we have a general purpose machine.
- We no longer need to build a mechanical or analog autopilot from scratch

Computers

- Diagram
- Software + General-Purpose Computer

  =Special-Purpose

These steps are then loaded into the computer, which while executing the instructions, in effect becomes the special-purpose machine(autopilot)

If changes are needed, the instructions can be changed instead of building a different physical machine from scratch

It's advantages have led to an explosive increase in their use, including their introduction into potentially dangerous systems.

2.1 The Role of Computers in Accidents

- What Computers do;
- Few systems today are built without computers to provide control functions, to support design, and sometimes to do both
- Computers now control most safety-critical devices
- Computers often replace traditional hardware safety interlocks and protection systems-sometimes with devastating results
- Even if the hardware protection devises are kept, software is often used to control them

The Role of Computers in Accidents

- What Computers cause;
- A relatively new breed of hazards and associated problems have appeared
- They appear primarily in flight control systems, armament control systems, navigation systems and cockpit displays
- They add new dimensions to the human-error problem
- Some of the hazards result from the crew's multitude of choices in aircraft management system, often during prioritization of tasks
- Conversely, computer –based systems are supposed to relieve pilot workload, but perhaps too much in some instances with resultant complacency and/or lack of situation awareness

The Role of Computers in Accidents

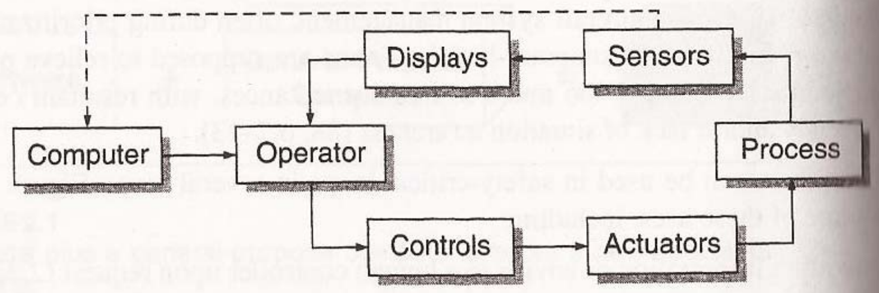
- **Ways computers are used in safety-critical loops**

1. Providing information or advice to a human controller upon request (2.2a)
2. interpreting data and displaying it to the controller, who makes the control decisions (2.2b)
3. Issuing commands directly, but with a human monitor of the computer's actions providing varying levels of input(2.2c)
4. Eliminating the human from the control loop completely(2.2d)

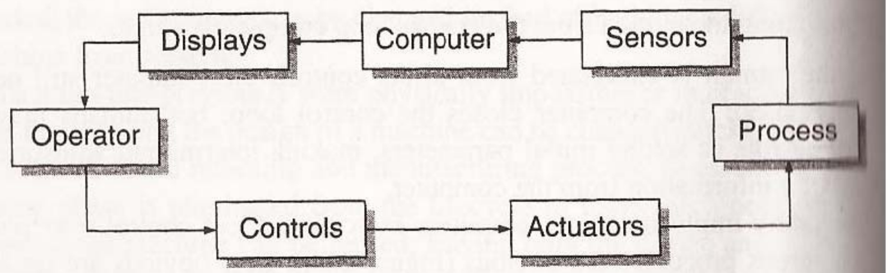
The Role of Computers in Accidents

- Providing information or advice to a human controller upon request (2.2a)

- interpreting data and displaying it to the controller, who makes the control decisions (2.2b)



(a) Computer provides information and advice to controller, perhaps by reading sensors directly.

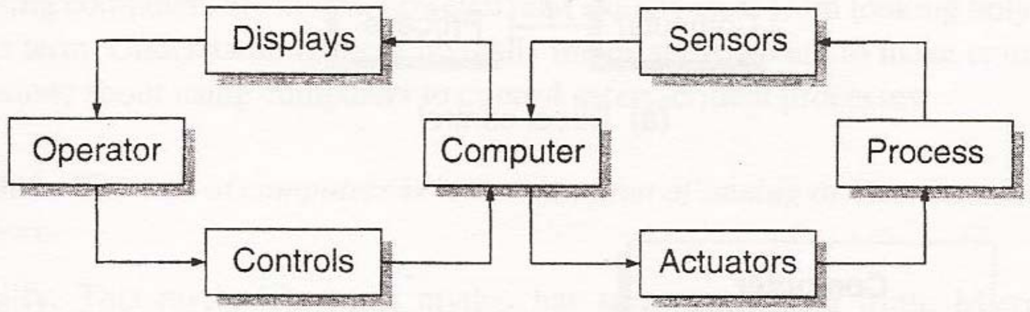


(b) Computer reads and interprets sensor data for operator.

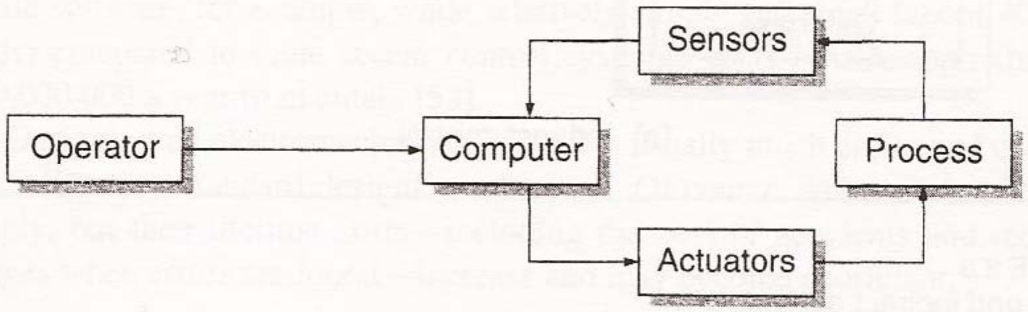
The Role of Computers in Accidents

- Issuing commands directly, but with a human monitor of the computer's actions providing varying levels of input(2.2c)

- Eliminating the human from the control loop completely(2.2d)



(c) Computer interprets and displays data for operator and issues commands; operator makes varying levels of decisions.



(d) Computer assumes complete control of process with operator providing advice or high-level direction.

The Role of Computers in Accidents

- Even if human is eliminated from direct control, the computer still needs to be supervised: the computer closes the control loop but humans may be assigned the role of setting initial parameters, making intermittent adjustments and receiving information from the computer

- SO WHAT?

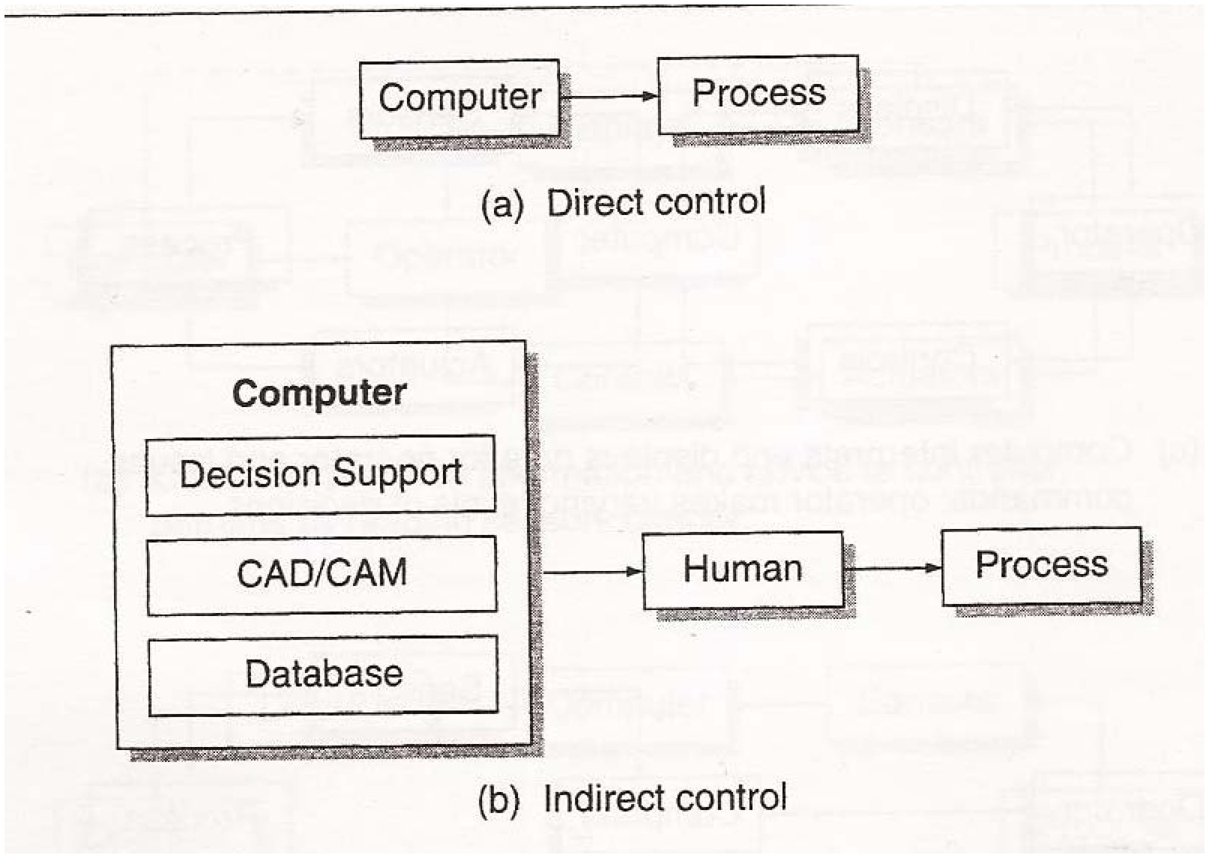
The Role of Computers in Accidents

- The safety implications of computers exercising direct control over potentially dangerous processes can be beyond our imagination
- Figure 2.3a depicts the obvious danger and the safety laps
- Less obvious are the dangers when (as depicted in fig2.3b)

The Role of Computers in Accidents

- Figure (a) depicts the absolute control of a system by computer

- Figure (b) describes an indirect control



The Role of Computers in Accidents

- **Safety implications:**

1. software-generated data is used to make safety-critical decisions (such as air traffic control and medical blood analyzers)
2. software is used in design analysis (such as CAD/CAM)
3. safety-critical data (such as blood bank data) is stored in computer data bases

The FDA has received reports of software errors in medical instruments that led to mixing up patients names and data, as well as reports of incorrect output from laboratory and diagnostic instruments (such as patient monitors, electro-cardiogram analyzers and imaging devices

The Role of Computers in Accidents

- **Safety implications-direct control:**

❑ In 1979, the discovery of an error in the software used in the design of nuclear reactors and their supporting cooling systems resulted in the Nuclear Regulatory Commission’s temporary shutdown of five nuclear power plants that did not satisfy earthquake standards.

❑ There is a serious danger in overreliance on the accuracy of computer outputs and data bases.

The Role of Computers in Accidents

- **Safety implications-indirect control**

- In some cases, companies and government agencies have agued that software that generate data but does not make decisions such as air traffic control software is not safety critical or is less than direct-control software because the human controller makes the ultimate decision, not the computer.

- If diagnostic devices produce incorrect results, the errors may be readily noticed or may be inconsistent with other clinical signs.

- The risk to the patient is less than in the case of software –driven devices that directly affect patients.

The Role of Computers in Accidents

- **Safety implications-indirect control**

❑ Although risk may be reduced by the use of a human intermediary , this reduction is by no means assured

❖ If system safety truly is to be increased, then all the components whose operation can directly or indirectly affect safety must be considered, and the related hazards must be eliminated or reduced

The Role of Computers in Accidents

- **Cost implication and complexity:**

❑ Computers add difficulty and cost to accident investigations

➢ For example, in the case of the Therac-25 medical accelerator, overdoses were first denied and not investigated or were attributed to transient hardware failures.

➢ Even if the possibility of software error is investigated, subtle errors that cause accidents in well-tested and sometimes well-used systems are not easy to find (or to prove that they may or may not exist).

➢ One software error cost millions of dollars to investigate-- it caused the loss of an F-14 military aircraft.

❑ The widespread use of computers in safety-critical systems is creating new problems for software and system engineers.

Methods to ensure the safety of computer-controlled systems have lagged behind the development of these systems

The Role of Computers in Accidents

- Proven system safety engineering techniques do not include software, and because of the unique characteristics of this new technology, are not easily adapted to software.
- Recent introduction of computers to control potentially dangerous systems and the relatively safe nature of computer itself(in terms of explosion, fire, or other direct hazards), few software engineering techniques have been developed to cope with safety problems
- For the most part, standard software engineering techniques and processes are being used to develop safety-critical software without any consideration of the special factors and unique requirements for enhancing safety.

Risk?

- Operating and design staff have been made to complain that programmers resent being watched or checked and that they produce programs that are not resistant to mistakes, cannot tolerate plant errors, and are difficult to understand.
- No doubt, programmers make similar remarks about operators and designers.

2.2 Software Myths

- If there are problems, why are computers being used so widely?
- ➢ The basic reason is that computers provide a level of power, speed and control not otherwise possible;
- ➢ -they are relatively light and small
- ➢ -other supposed advantages of using computers are myths
- To make competent decisions about using computers to control safety-critical processes, it's important to understand these myths

Software Myths

- Myth classification: based on initial decision to employ computers to control safety-critical processes
- Myth 1
- Myth 2
- Myth 3
- Myth 4
- Myth 5
- Myth 6
- Myth 7

Myth 1

- The cost of computers is lower than that of analog or electromechanical devices
- Reality : this myth like most myths have some superficial truth
- Microcomputer hardware is cheap relative to other electromechanical devices, however
- The cost of writing and certifying highly reliable and safe software to make that microprocessor useful together with the cost of maintaining the software without compromising reliability and safety, can be enormous

Myth 1 cont.

- The on-board space shuttle software, for example while relatively simple and small (about 400,000 words) compared to more recent control systems costs NASA approximately $100.000,000 annually to maintain
- Designing an electromechanical system is usually much easier and cheaper, especially when standard designs can be used.
- Software can be built cheaply, but then life costs—including the costs of accidents and required changes when errors are found—increase and may become exorbitant

Myth 2

- Software is easy to change
- Again, this myth is superficially true:
  - Unfortunately making changes without introducing error os extremely difficult
  - like the hardware, the software must be completely reverified and recertified every time a change is made, at what may be an enormous cost
  - software quickly become more “brittle” as changes are made—the difficulty of making a change without introducing errors may increase over the lifetime of the software

Myth 3

- Computers provide greater reliability than the devices they replace

❑ Reality : although true in theory

- -software does not “fail” in the sense this term usually implies in engineering
- -there is little evidence to show that erroneous behavior of software is not a significant problem in practice

Myth 3 cont.

- A study by the British royal signals and radar establishment used commercially available tools to examine the number of errors in software written for some highly safety critical systems
- ❑ -up to 10% of the program modules or individual function were shown to deviate from the original specification in one or more modes of operation
- ❑ -discrepancies were found even in software that had undergone extensive checking using sophisticated test platforms
- ❑ -many of the detected anomalies were too minor to have any perceptible effects

For example, a discrepancy of 1 part in 32,000 in a computation using 16-bit arithmetic

Myth 4

- Increasing software reliability will increase safety
- Reality: Software reliability can be increased by removing software errors that are unrelated to system safety thus increasing reliability while not increasing safety at all
- Software reliability is defined as compliance with requirements specification while most safety critical software errors can be traced to errors in the requirements
- Safety and reliability while partially overlapping are not the same thing: Increased computer or software reliability does not necessarily result in increased system safety

Myth 5

- Testing software or "proving"
- ❑ Reality: The limitations of software testing are well known
- ❑ Basically the large number of states of most realistic software makes exhaustive testing impossible only a relatively small part of the state space can be covered
- The use of mathematical techniques to verify the consistency between the software instructions and the specifications is another way to gain assurance
- ❑ However such verification will not solve all of our problems

Myth 6

- Reusing software increases safety
- Reality: Although reuse of proven software components can increase reliability, reuse has little or no effect on safety
- Reuse can actually decrease safety because of the complacency it engenders and because the specific hazards of the new system were not considered when the software was originally designed and constructed
- Example of safety problems arising from reuse of software include the following
- The therac-20 parts of which were reused for the therac-25 contained the same error responsible for at least two deaths in the therac-25

Myth 6 …

- Software used successfully for air traffic control for many years in the united states was reused in great britan with less success
- Aviation software written for use in the northern hemisphere often creates problems when used in the southern hemisphere
- Safety is not a property of the software itself but rather a combination of the software design and the environment in which the software is used

Myth 7

- Computers reduce risk over mechanical systems

❑ Reality: Computers have the potential to decrease risk but not all uses of computers achieve this potential

- Computers can automate tedious and potentially hazardous jobs such as spray painting and electric art welding thus reducing the risk to workers in this particular jobs

1. Arguments:

- Computers allow finer control in which they can check parameters more often, perform complicated computations in real time, and take action quickly

Counter argument:

- Computers do provide finer control computations in real time and they can take action quickly but finer control allows the process to be operated closer to its optimum and the safety margins can be cut.

Myth 7 …

2. Argument

- Automated systems allow operators to work further away from hazardous areas

Counter argument

- Because of lack of familiarity with the hazards, more accidents may occur when operators do have to enter hazardous areas

3. Argument

- By eliminating operators, human errors are eliminated

Counter argument

- Operator errors are replaced by human design and maintenance errors: Humans are not removed from the system, they are merely shifted to different jobs

Myth 7cont.

- **4. argument :**
- Computers have the potential to provide better information to operators and thus to improve decision making
- Counter argument:
- While theoretically true, in reality this potential is very difficult to achieve
- 5. argument:
- Software does not fail
- Counter argument:
- This common belief is true only for a very narrow definition of "failure"

Software myth summary

- Computers have the potential to increase safety, and surely this potential will be realized in the future. But we can not assume that we know enough to accomplish this goal.

- Any increased potential may not be realized if those building the system use it to justify taking more risks.

2.3 Why Software Engineering is Difficult

- Why do we have so much trouble engineering software?

1. analog versus Discrete state systems
2. the “Curse of flexibility”
3. Complexity and Invisible Interfaces
4. Lack of historical usage information

Why Software Engineering is Difficult

- **Analog versus Discrete (software) state system**

- ❑ in control system, the computer is usually simulating the behavior of an analog controller
- ❑ the translation of the function from analog to digital form may introduce inaccuracies and complications
- ❑ The same type of mathematical analysis used to predict the behavior of physical systems do not apply to discrete (software) systems
- ❑ factors such as time, finite-precision arithmetic and concurrency are difficult to handle (there is progress but we are far from being able to handle even small software)
- ❑ mathematical specification or proves of software properties may be the same size as the program, more difficult to construct and often harder to understand than the program
- ❑ mathematical specification of discrete systems are as prone to error as the code itself

Why Software Engineering is Difficult

- **The "Curse of Flexibility"**
- A computer's behavior can be easily changed by changing its software
- ❑ --in reality, the apparent low cost and ease of changing software is deceptive
- ❑ --it encourages major and frequent change, which often increases complexity rapidly and introduces errors
- ❑ --flexibility encourages redefinition of task late in the development process in order to overcome deficiencies found in other parts of the system
- ❑ --major design modifications are much more difficult to make than minor ones as the properties of the physical material in which the design is embedded provide natural constraints on modification

Why Software Engineering is Difficult

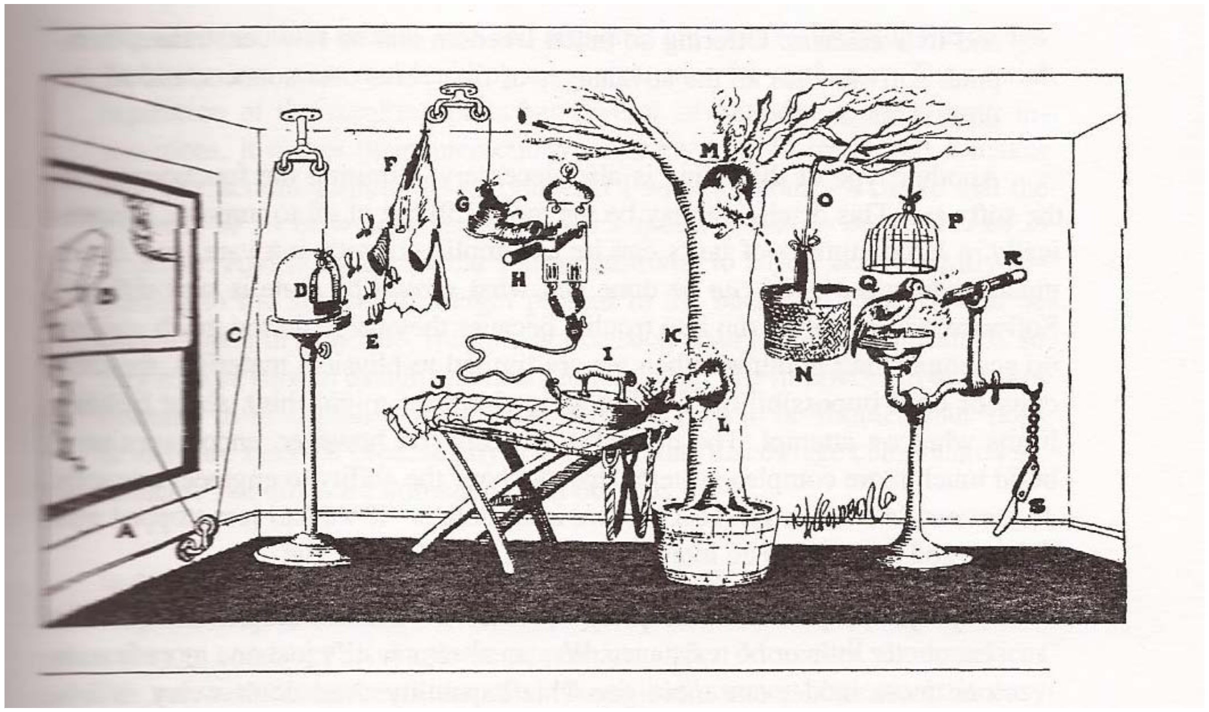
- The "Curse of Flexibility" cont.

❑ While natural constraints enforce discipline on the design, construction and modification of a physical machine, these constraints do not exist for software

❑ Nature imposes discipline on the design process which helps to control complexity, in contrast software has no corresponding physical limitations or natural laws which makes it too easy to build enormously complex designs (figure 2.4)

Why Software Engineering is Difficult

- Software has no physical limitations or natural laws which makes it too easy to build enormously complex designs



Why Software Engineering is difficult

- The myth of software flexibility also encourages premature construction before we fully understand what we need to do
- Another trap of software flexibility is the ease with which partial success is attained often at the expense of unmanaged complexity
- Software works correctly most of the time, but not all of the time
- Once a programs complexity has become unmanageable, each change is likely to hurt as to help
- Like airplane complexity, software complexity can be controlled by appropriate designed discipline which people must impose not nature

Why Software Engineering is difficult

- **Complexity and invisible interfaces**
- one way to deal with complexity is to break the complex object in to pieces or models
- Errors occur because the human mind is unable to fully comprehend the many conditions that can arise though the interactions of components
- An interface between two programs is comprised of all the assumptions that the programs make about each other
- Humans can only cope with very little complexity

Why Software Engineering is difficult

- **Lack of historical usage information**
- A final difficulty with software not found in hardware systems is that no historical usage information is available to allow measurements, evaluations, and improvements on standard designs

2.4 The reality we face

- When systems were composed of only electromechanical and human components, engineers knew that random wear out failures and human errors could be reduced and mitigated but never completely eliminated
- They accepted ways to build systems that were robust and safe despite random failures
- Design errors could also be handled fairly well through testing and reused of proven design
- In reality the time to create perfect software is never there, and never can be

Questions!!!!