Intro to SE

WEEK 3

Ethics and social responsibilities

Carlo ghezzi page 532 and 533

The growing dependence of society on software also places a tremendous social responsibility on the shoulders of software engineers and their managers. When software is being used to monitor the health of patients, control nuclear power plants, apply the brakes in an automobile, transfer billions of dollars in an instant, launch missiles, or navigate an airplane, it is not simply good engineering to build reliable software; it is also the engineer's ethical responsibility to do so.

There are many reports of software defects that have cost lives or tremendous business losses

Defects are unavoidable

By relying on principles and sound methods, as well as the experiences of the past, we can try to prevent and avoid defects and failures

Ethics and social responsibilities

New opportunities for criminal fraud and sabotage

New ways to steal information or shut down any business without physically present

Software must be strong enough to resist such illegal efforts

Programs defects are not just only bugs but those can lead to severe damages

Building reliable software is a technical objective of the software engineer, but it also has ethical and social implications that must guide the actions of the serious professional. In this light, "hacking"—that is, inserting "playful" bugs into programs, creating viruses, writing quick and dirty code just to meet a schedule or a market window, shipping defective software, and even shipping software that works, but does not meet the agreed-upon specifications is unethical.

Software engineering code of ethics

ACM and IEEE developed Software engineering code of ethics and practices. Public interest is focus

The preamble of the code states that

Computers have a central and growing role in commerce, industry, government, medicine, education, entertainment, and society at large. Software engineers are those who contribute by direct participation or by teaching, to the analysis, specification, design, development, certification, maintenance, and testing of software systems. Because of their roles in developing software systems, software engineers have significant opportunities to do good or cause harm. To ensure, as much as possible, that their efforts will be used for good, software engineers must commit themselves to making software engineering a beneficial and respected profession. In accordance with that commitment, software engineers shall adhere to the following Code of Ethics and Professional Practice.

Software engineering code of ethics

1. **Public**: Software engineers shall act consistently with the public interest.
2. **Client and Employer**: Software engineers shall act in a manner that is in the best interests of their client and employer, consistent with the public interest.
3. **Product**: Software engineers shall ensure that their products and related modifications meet the highest professional standards possible.
4. **Judgment**: Software engineers shall maintain integrity and independence in their professional judgment.
5. **Management**: Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance.

Software engineering code of ethics

6. **Profession**: Software engineers shall advance the integrity and reputation of the profession, consistent with the public interest.
7. **Colleagues**: Software engineers shall be fair to, and supportive of, their colleagues.
8. **Self**: Software engineers shall participate in lifelong learning regarding the practice of their profession and shall promote an ethical approach to the practice of the profession.

Relationship of SE and CS

CARLO GHEZZI CHAPTER 1 PAGE 8-13

Programming languages

- Central tool for software development
- SE influence programming languages
  - Influence can be seen by modularity feature
  - Exception handling for reliability
  - Design of OOP
- Programming languages influence SE
  - Requirements and designs should be described according to PL

Operating Systems

- Operating systems were the first really large software systems built
- influence of software engineering techniques on the structures of operating systems can be seen in portable operating systems and operating systems that are structured to contain a small protected kernel that provided a minimum of functionality for interfacing with the hardware and a non-protected part that provided the majority of the functionality previously associated with operating systems.
- Command line interpreter is now a utility
- Abstraction
- Separate what from how
- Separate Policy from mechanism

Database

- New design techniques due to database
- Data independence
- Reading and writing of data without knowing underlying representation
- Attachment of database with software systems
- traditional database technology was incapable of dealing with the problems posed by software engineering processes.
- For example
  - storing large structured objects such as sources programs or use manuals;
  - storing large unstructured objects such as object code and executable code;
  - maintaining different versions of the same object; and strong objects, such as a product, with many large structured and unstructured fields, such as source code, object code, and a user manual.

Artificial Intelligence

- SE techniques used in Expert systems
- Modularization between facts known and rules for processing facts
- AI tools for improving SE tasks like programming assistants

Theoretical models

- Finite state machines
- Pushdown Automata

Management Science

Much of software engineering is involved with management issues. As in any kind of large, multiperson endeavor, we need to do project estimation, project scheduling, human resource planning, task decomposition and assignment, and project tracking. Additional personnel issues involve hiring personnel, motivating people, and assigning the right people to the right tasks.

Management science studies exactly these issues. Many models have been developed that can be applied in software engineering. By looking to management science, we can exploit the results of many decades of study.

In the opposite direction, software engineering has provided management science with a new domain in which to test management theories and models. The traditional management approaches to assembly-line production clearly do not apply to human-centered activities such as software engineering, giving rise to a search for more applicable approaches.

Systems Engineering

The field of systems engineering is concerned with studying complex systems. The underlying hypothesis is that certain laws govern the behavior of any complex system composed of many components with complex relationships. Systems engineering is useful when one is interested in concentrating on the system, as opposed to its individual components. Systems engineering tries to discover common themes that apply to diverse systems—for example, chemical plants, buildings, and bridges.

Software is often a component of a much larger system. For example, the software in a factory monitoring system or the flight software on an airplane are just components of more complicated systems. Systems engineering techniques can be applied to the study of such systems. We can also consider a software system consisting of thousands of modules as a candidate complex system subject to systems engineering laws.

On the other hand, system engineering has been enriched by expanding its set of analysis models—which were traditionally based on classical mathematics—to include discrete models that have been in use in software engineering.