SE 2AA4 Winter 2007

01 Software Engineering as an Engineering Discipline

William M. Farmer

Department of Computing and Software
McMaster University

4 January 2007

What is Software Engineering?

- An area of engineering that deals with the development of software systems that:
  - Are large or complex.
  - Exist in multiple versions.
  - Exist for large periods of time.
  - Are continuously being modified.
  - Are built by teams.
- Software engineering is the "application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software" (IEEE 1990).
- D. Parnas (1978)—the father of the McMaster software engineering program—said it is "multi-person construction of multi-version software".
- Like other areas of engineering, software engineering relies heavily on mathematical techniques
  - Logic and discrete mathematics are more important than continuous mathematics.

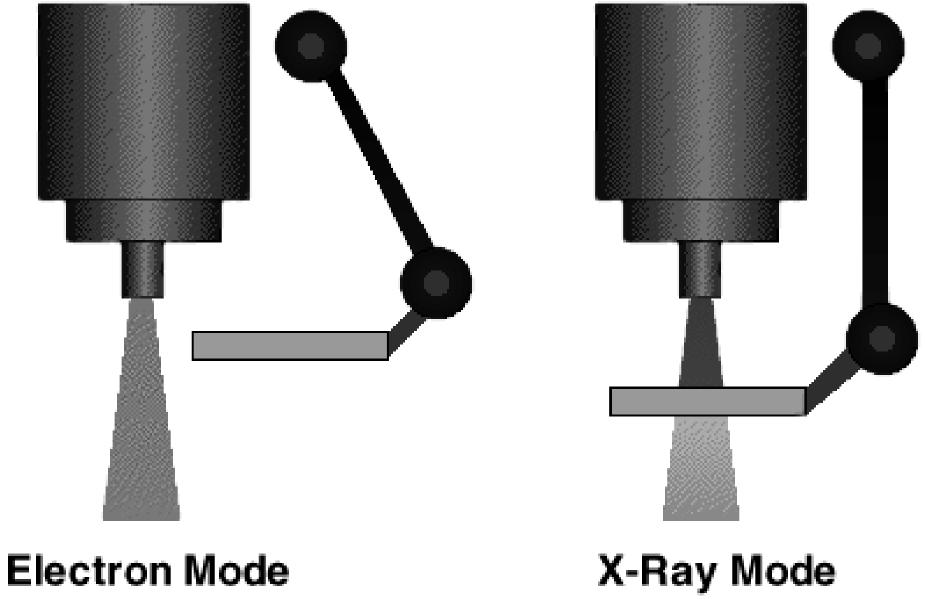
Software Engineering in System Design

- A physical system is often controlled by a software system called an embedded system.
- As a result, software engineering is often a crucial part of system design.
- Examples of embedded systems:
  - Cell phones.
  - Nuclear power plants.
  - Automobiles.
  - Aircraft.
  - Programmable household devices.
  - MP3 players.
  - Radio Frequency Identification (RFID) tags.
- Embedded systems are rapidly appearing everywhere.
- The developers of software for an embedded system need to understand both the software and the physical device.

Example: Therac-25 (1)

- The Therac-25 was a radiation therapy machine for treating cancer.
  - Produced by the Atomic Energy of Canada Limited (AECL).
  - Controlled by software.
- How it worked:
  - Provided both electron beam and X-ray treatment.
  - The machine produced low- to high-energy electron beams.
  - X-rays were produced by rotating a target into the path of a high energy electron beam.
- Used in several clinics across North America.

Example: Therac-25 (2)



Example: Therac-25 (3)

- In six separate incidents in the 1980s, Therac-25 machines delivered overdoses of radiation causing severe physical damage or even death to the patients being treated.
  - The second incident, which took place in Hamilton, resulted in an administration of 13,000–17,000 rads of radiation (200 rads is a regular treatment and 1000 rads can be fatal).
  - Three patients ultimately died from radiation poisoning.
- What went wrong:
  - Software failed to detect that the target was not in place.
  - Software failed to detect that the patient was receiving radiation.
  - Software failed to prevent the patient from receiving an overdose of radiation.

Example: Therac-25 (4)

Causes of the failure:

- Inadequate software design.
- Inadequate software development process.
  - Coding and testing done by only one person.
  - No independent review of the computer code.
  - Inadequate documentation of error codes.
  - Poor testing procedures.
- Software was ignored during reliability modeling.
- No hardware interlocks to prevent the delivery of high-energy electron beams when the target was not in place.

The Great Gulf

- Engineers do not sufficiently understand or care about software.
    - Many of the basic principles of software design and development are largely unknown to engineers.
    - Engineers often do not appreciate the challenges and dangers inherent in software for embedded systems.
- Software developers lack engineering training and professionalism.
    - There is an entrenched culture of producing software without any guarantee whatsoever.
    - There is no system for certifying either software or software developers.
    - Most software developers lack the engineering background needed to produce software for embedded systems.

Challenges and Opportunities for Engineering

- Challenges:
  - Engineers need to design systems that have safe, correct, high-quality software.
  - Software engineers need to produce software they can guarantee.
- Opportunities:
  - Software tools can greatly enhance the capabilities of engineers.
  - Software can greatly increase the effectiveness of the devices engineers design.

Attributes of a Good Software Engineer

- Is a good engineer!
- Can program in the large as well as in-the-small.
- Has a solid understanding of computing and software.
- Is comfortable with working with models at different levels of abstraction.
- Can communicate and work effectively with other team members.

Software Development Process

- A rational development process is needed to produce quality software.
- Any proposed rational process is necessarily an idealization.
    - Humans inevitably make errors.
    - Communication between humans is imperfect.
    - Many things are not understood at the start.
    - Supporting technology always has limitations.
    - Requirements change over time.

Software Presentation

- Every software product should include documentation that presents the product to clients, reviewers, users, and maintainers.
- It is useful to produce documentation that makes it appear as if the software product were developed by a rational process.
  - Mathematicians have long followed this approach in presenting their results.
  - See D. Parnas, "A Rational Design Process: How and Why to Fake It", in: D. Hoffman and D. Weiss, Software Fundamentals, Addison Wesley, 2001.

Software Development Phases

1. Requirements: What is the problem that needs to be solved? What are the product requirements that need to be satisfied?
2. Design: How will the problem be solved? How will the product requirements be satisfied?
3. Implementation: What is a solution to the problem? What is an executable implementation of the design?
4. Verification: What behavior does the product exhibit? Is the behavior correct?
5. Delivery and Maintenance: How will the product be delivered? What needs to be maintained? How will it be maintained?

Software Life Cycle Models

- Waterfall model:
  - Development follows the logical order of the phases given above in a linear fashion.
  - Is an idealization of the software development process that is rarely realized.
- Other life cycle models:
  - Refinement
  - Incremental
  - Spiral
  - Prototyping