What is Software Engineering

UVic SEng 265

Daniel M. German
Department of Computer Science
University of Victoria

October 9, 2002 Version: 1.00

What is Software Engineering?

- Software engineering is about the creation of large pieces of software.
- **Definition**: Software engineering is an engineering discipline concerned with all aspects of software production.
- **Engineering discipline**: Engineers apply theories, methods and tools to create solutions to problems within organizational and financial constrains.
- **All aspects of software production**: Engineers are concerned with all the activities related to the creation of software.

Design vs. Manufacturing

- The creation of software is human-intensive
- In other engineering disciplines, the majority of the costs associated with a product are located in manufacturing
- In SEng, software is more design intensive
- Manufacturing cost is insignificant
- Software maintenance can also be costly

Qualities of Software

- Correctness
- Reliability
- Robustness
- Performance
- User friendliness
- Maintainability
- Portability
- Reusability...

Correctness

- A program behaves according to its functional requirements specification
- Correctness verifies that a program satisfies its specifications: testing and formal verification (program proofs)

Reliability

- Reliability is the probability that a system or a capability of a system functions without failure for a specified period in a specified environment.
- Can a user depend on software?
- A program might be correct, but not reliable
  - The program might not want to *cooperate* in the mornings, for example.
- Users are accustom to unreliable software (blue screen of death)
- Almost no other engineering discipline can get away with unreliable products (imagine a TV that stops in the middle of your favorite show)

Robustness

- How well does the system behave when in situations not specified by its requirements?
- For example, faulty equipment, incorrect input, power loss.
- Usually deals with those situations not specified as part of its correctness

Performance

- How well does a program uses the computer resources?
- A program should not waste unnecessary resources
  - Memory
  - CPU time
  - disk...
- How scalable is the program?

User Friendliness

- How well designed is the user interface of the program?
- Involves Human-Computer interaction principles
- A program with a better user interface is more likely to be used
- User interfaces add complexity to a program
- The percentage of code dedicated to the user interface (as compared to the core of the application) has grown dramatically since the GUI appeared.

Maintainability

- A set of attributes that bear on the effort needed to make specified modifications
  - Corrections, improvements, or adaptations of software to changes in environment, requirements and/or specifications.

Portability

- The ability of software to be transferred from one environment to another
- Can be a transfer from one architecture/operating system to another
- Can be a transfer from one programming language to another

Reusability

- How much can we reuse from a software system?
  - Application level reuse: reusing the entire system in a different setting
  - Component reuse: reuse of parts of the system (subsystems or objects)
  - Function reuse: A single function (such as a mathematical one) may be reused

Phases of the Software Life Cycle

- Requirements analysis and definition
- Design:
  - System (architectural) design
  - Program (detailed) design
- Writing the programs (program implementation)
- Testing
  - Unit testing
  - Integration testing
  - System testing
- Evolution (Maintenance)

Requirements Analysis and Definition

- In this stage the requirements of the ”to be developed software” are established.
- These are usually the services it will provide, its constraints and the goals of the software.
- Once these are established they have to be defined in such a way that they are usable in the next stage.
- The user requirements might be contradictory to the developer goals (who want to finish the job).
- Products: requirements documents

Software design

- Software design identifies how each system function will be accomplished
- Composed of architectural (system) and detailed (program) design
  - **Architectural Design** Overall description of the system and its components; it also describes its data structure
  - **Detailed Design** Detailed description of each part of the system.
- Product: design documents

Implementation

- Implementation – involves realising the design through writing code.
- It is the stage we usually most love.
- Each program that composes a system is called a “unit”
- Products: executable code

Testing

- Unit testing: is the verification that every unit (program) meets its specification.
- System testing. All the units are combined and now the whole is tested. When the combined programs are successfully tested the software product is finished.
- Acceptance testing. Getting the user to accept the system, including installation and training courses.
- Product: tested software, audit trail.

Evolution (Maintenance)

- Software is flexible: we can change it any time
- Maintenance: the process of changing a system once it has gone into use
- Software engineering is an evolutive process where software is continually changed over its lifetime in response to changing requirements and customer needs.