Chapter 1 – Introduction to Software and Software Engineering

Overview

- Software is designed and built by software engineers.
- Software is used by virtually everyone in society.
- Software is pervasive in our commerce, our culture, and our everyday lives.
- Software engineers have a moral obligation to build reliable software that does no harm to other people.
- Software engineers view computer software, as being made up of the programs, documents, and data required to design and build the system.
- Software users are only concerned with whether or not software products meet their expectations and make their tasks easier to complete.

Important Questions for Software Engineers

- Why does it take so long to get software finished?
- Why are development costs so high?
- Why can’t we find all errors before we give the software to our customers?
- Why do we spend so much time and effort maintaining existing programs?
- Why do we continue to have difficulty in measuring progress as software is being developed?

Software

- Software is both a product and a vehicle for delivering a product (information).
- Software is engineered not manufactured.
- Software does not wear out, but it does deteriorate.
- Industry is moving toward construction component-based software, but most software is still custom-built.

Software Application Domains

- System software
- Application software
- Engineering or Scientific Software
- Embedded software
- Product-line software (includes entertainment software)
- Web-Applications
- Artificial intelligence software

New Software Challenges

- Open-world computing
  - Creating software to allow machines of all sizes to communicate with each other across vast networks
- Netsourcing
  - Architecting simple and sophisticated applications that benefit targeted end-user markets worldwide
- Open Source
- Distributing source code for computing applications so customers can make local modifications easily and reliably

Reasons for Legacy System Evolution

- Software must be adapted to meet needs of new computing environments or technology
- Software must be enhanced to implement new business requirements
- Software must be extended to make it interoperable with more modern system components
- Software must be re-architected to make it viable within a network environment

Unique Nature of Web Apps

- Network intensive
- Concurrency
- Unpredictable load
- Availability (24/7/365)
- Data driven
- Content sensitive
- Continuous evolution
- Immediacy (short time to market)
- Security
- Aesthetics

Software Engineering Realities

- Problem should be understood before software solution is developed
- Design is a pivotal activity
- Software should exhibit high quality
- Software should be maintainable

Software Engineering

- Software engineering is the establishment of sound engineering principles in order to obtain reliable and efficient software in an economical manner.
- Software engineering is the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software.
- Software engineering encompasses a process, management techniques, technical methods, and the use of tools

Generic Software Process Framework

- Communication (customer collaboration and requirement gathering)
- Planning (establishes engineering work plan, describes technical risks, lists resources required, work products produced, and defines work schedule)
- Modeling (creation of models to help developers and customers understand the requires and software design)
- Construction (code generation and testing)
- Deployment (software delivered for customer evaluation and feedback)

Software Engineering Umbrella Activities

- Software project tracking and control (allows team to assess progress and take corrective action to maintain schedule)
- Risk management (assess risks that may affect project outcomes or quality)
- Software quality assurance (activities required to maintain software quality)
- Technical reviews (assess engineering work products to uncover and remove errors before they propagate to next activity)
- Measurement (define and collect process, project, and product measures to assist team in delivering software meeting customer needs)
- Software configuration management (manage effects of change)
- Reusability management (defines criteria for work product reuse and establish mechanisms to achieve component reuse)
- Work product preparation and production (activities to create models, documents, logs, forms, lists, etc.)

Essence of Practice

- Understand the problem (communication and analysis)
- Plan a solution (software design)
- Carry out the plan (code generation)
- Examine the result for accuracy (testing and quality assurance)

Understand the Problem

- Who are the stakeholders?
- What functions and features are required to solve the problem?
- Is it possible to create smaller problems that are easier to understand?
- Can a graphic analysis model be created?

Plan the Solution

- Have you seen similar problems before?
- Has a similar problem been solved?
- Can readily solvable subproblems be defined?
- Can a design model be created?

Carry Out the Plan

- Does solution conform to the plan?
- Is each solution component provably correct?

Examine the Result

- Is it possible to test each component part of the solution?
- Does the solution produce results that conform to the data, functions, and features required?

Software Practice Core Principles

- Software exists to provide value to its users
- Keep it simple stupid (KISS)
- Clear vision is essential to the success of any software project
- Always specify, design, and implement knowing that someone else will have to understand what you have done to carry out his or her tasks
- Be open to future changes, don’t code yourself into a corner
- Planning ahead for reuse reduces the cost and increases the value of both the reusable components and the systems that require them
- Placing clear complete thought before any action almost always produces better results