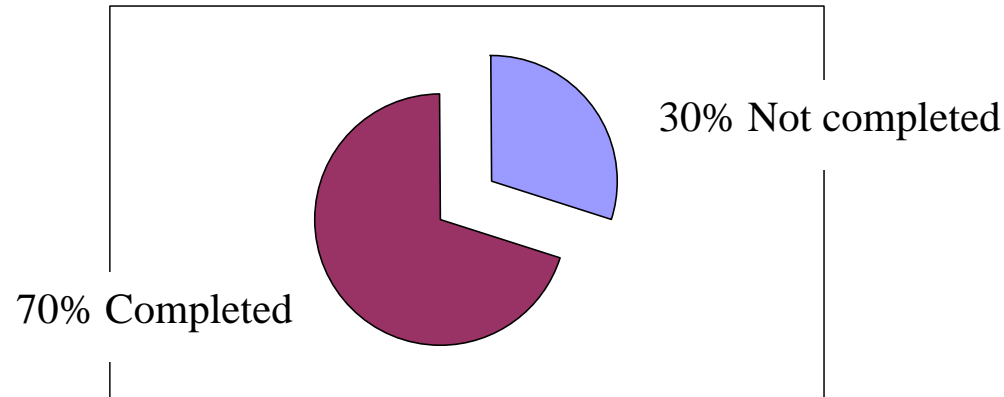

Software Engineering

Introduction

Objectives

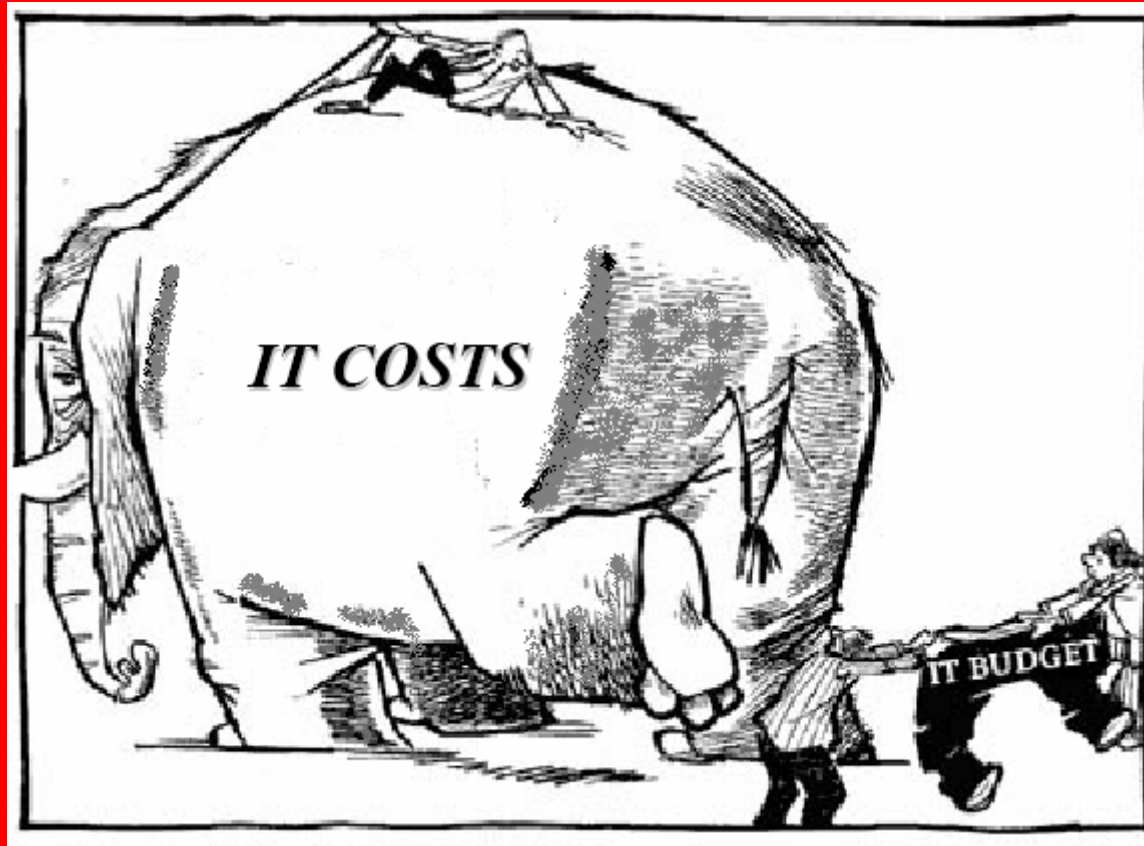
- To introduce software engineering
- To introduce Professional Organizations for Software Engineering standards
- To discuss key questions about software engineering
- To introduce the Systems Development Life Cycle “SDLC”

Software is a Risky Business



- All surveyed projects used waterfall lifecycle.
- 53% of projects cost almost 200% of original estimate.
- Estimated \$81 billion spent on failed U.S. projects in 1995.

IT: Budget & Cost



What is Software Engineering?

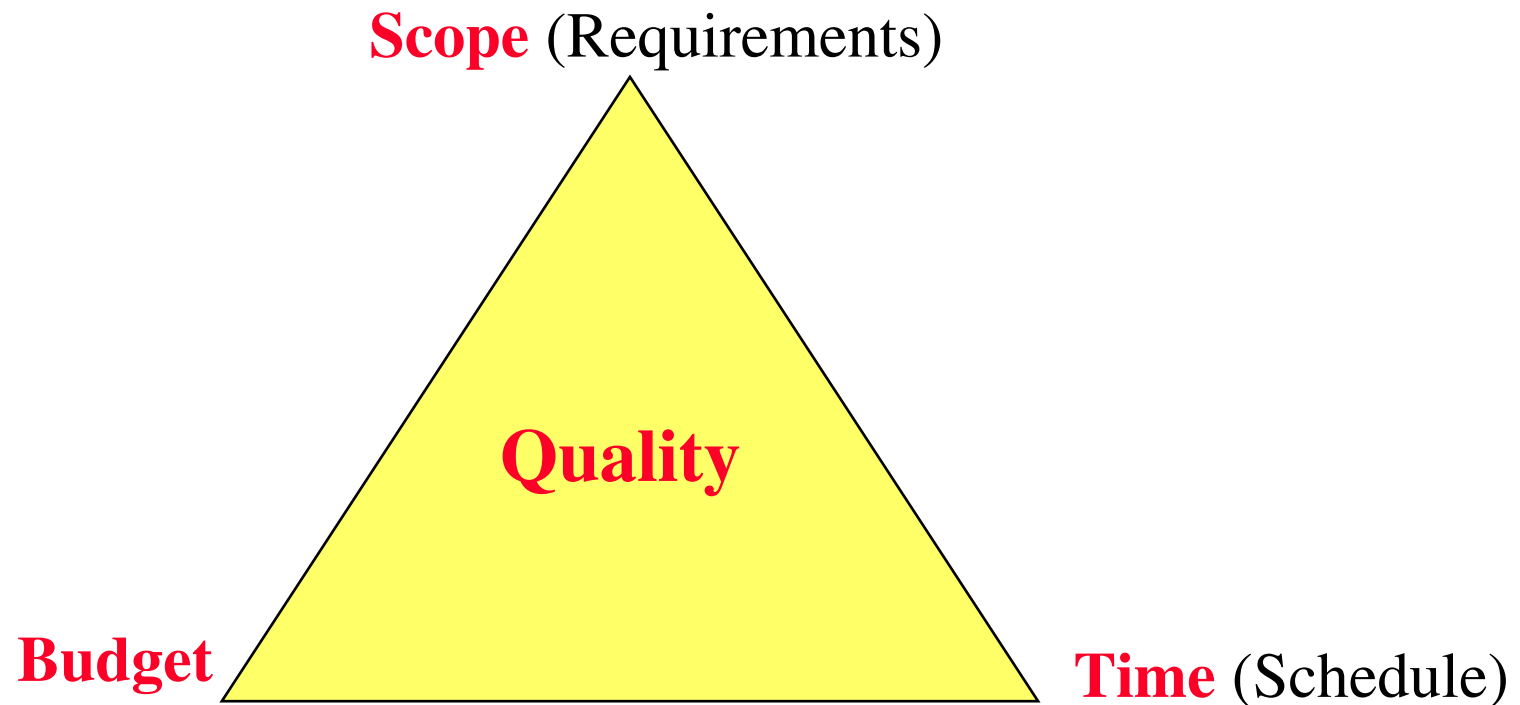
Developing software having:

- High **quality**
- Within **budget**
- On **schedule** (time)
- Satisfying client's **requirements**

Failure Statistics of SW Projects

- Success
 - **On-time,**
 - **On-budget,**
 - **and scope-coverage** (with Most of the Features & Functions)
 - **High quality**
- Failed
 - **Over-budget,**
 - **Over-time,**
 - and/or with **less scope (Fewer Features & Functions)**
- Impaired
 - Cancelled & Unused

Quality & The Triple constraint



Moving Target Problem

- Change is *inevitable*
- planning/preparation
 - Growing companies are always going to change
 - Markets evolve and needs of people change
 - Technology changes
- There is no solution to the moving target problem, so we need to learn to live with it

What is software?

- Software is:
 - Computer programs + associated documentation
- Software products may be:
 - **Generic Software** - developed to be sold to a range of different customers
 - **Bespoke (custom) Software** - developed for a single customer according to given specification

Product Engineering

- Producing a product (e.g. TV, Car, Computer,..)
- Product Engineering:
 - How to produce a product **Optimally**:
 - **Highest** quality
 - **Minimum** resources
 - Under specified constraints

Software costs

- Software costs often dominate system costs.
- System costs includes:
 - Hardware cost
 - Software cost
 - People cost
- Maintenance costs: For systems with a long life, maintenance costs may be several times development costs.

Other related definitions of Software Engineering?

- Software engineering is a **modelling** activity:
Deals with complex systems through modelling
- Software engineering is a **problem-solving** activity: models are used to search for an acceptable solution

What is software engineering? (cont.)

- Software engineering is not a mathematical science: it relies on **empirical** methods (related to experiments on development of previous software)
- Software engineering is a knowledge acquisition activity

What is the difference between software engineering and product engineering?

- Product engineering
 - A prototype is realised & tested
 - Thousands of items are manufactured

- Software engineering
 - **Every software is a prototype**
 - No manufacturing

Software engineers

- Software engineers should:
 - adopt a systematic and organised approach to their work
 - use appropriate **tools** and techniques depending on
 - the problem to be solved,
 - the development constraints and
 - the resources available

What is the difference between software engineering and computer science?

- Computer science is concerned with **theory and fundamentals**
- Software engineering is concerned with the **practicalities of developing useful software**

What is the difference between software engineering and system engineering?

- **System engineering** is concerned with all aspects of **computer-based systems** development including:
 - hardware
 - software
 - and process engineering.
- Software engineering is **part** of system engineering
- System engineers are involved in system specification, architectural design, integration and deployment

Professional Organizations for Software Engineering standards

- Software Engineering Institute (SEI)
- Institute of Electrical and Electronics Engineers (IEEE) as an IEEE standard
- IEEE: Software Engineering Group
- ACM: American Computing Machinery

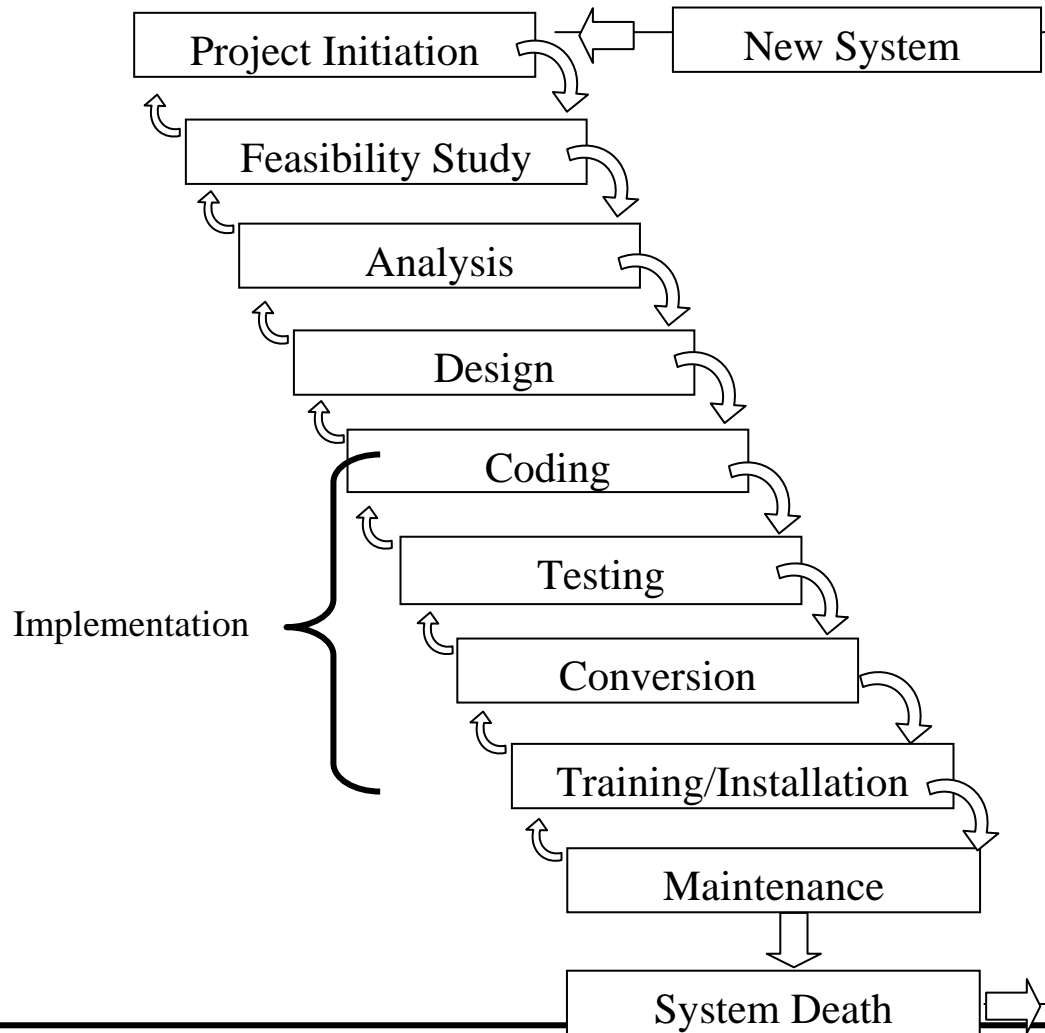
System Development Life Cycle

“SDLC”

Includes the following phases:

- Project Initiation
- Feasibility Study
- Analysis
- Design
- Implementation
 - Coding + Documentation
 - + Testing
 - + Conversion
 - + Training + Installation
- Maintenance

System Development Life Cycle “SDLC”



System Development Life Cycle

“SDLC” (cont.)

Project Initiation

- Client faces a problem
- Needs for an improvement
- Sources of potential projects:
 - Top management
 - Steering committee
 - Users
 - Opportunities
 - Competition

:

System Development Life Cycle

“SDLC” (cont.)

Feasibility Study

- Think of more than one solution: Alternative solutions
- **Four** feasibility Studies:
 1. Economic feasibility (cost benefit analysis)
 2. Technical feasibility (technology available)
 3. Schedule feasibility (delivery date)
 4. Human Resources feasibility (New staff, Training)

System Development Life Cycle

“SDLC” (cont.)

Analysis: Determine and structure system requirements

- Facts Finding / Requirement Elicitation / Domain Info / Requirements capturing
- Requirements Structuring into Diagrams & Text
 - Context Diagram / Data Flow Diagrams “DFDs”
 - ER Diagrams
 - Use Case Diagrams UML (Unified Modeling Language) , Use Case description
 - etc

System Development Life Cycle

“SDLC” (cont.)

Design: Create new System designs

- System Architecture design
- **Interface Design for:**
 - **Subsystems (internal other modules of the system: Inventory, Sales, Purchasing subsystems)**
 - **Other external systems, if any (bank sys, GOSI sys,...)**
- Database Design “Normalised relations”
- Input GUI design “Graphical User Interface”
 - Forms
 - Menus
 - Icons
 - Dialogue boxes, etc
- Output design
 - Reports
 - Queries
- Algorithm design

System Development Life Cycle

“SDLC” (cont.)

Implementation: Translate designs into a working system

- Coding
- Testing
- Documentation
- Data conversion (from old to new system)
- Training
- Installation

System Development Life Cycle

“SDLC” (cont.)

Maintenance: Evolving system

- Requirements **WILL CHANGE** to reflect dynamic environment of business
- Continuous process
- Maintenance types:
 - Corrective: correct existing defects
 - Perfective: improve
 - Adaptive: to new environment / requirements

What is a software process?

- A set of activities whose goal is the development or evolution of software
- Generic activities in all **software processes** are:
 - **Specification** - what the system should do and its development constraints
 - **Development** - production of the software system
 - **Validation** - checking that the software is what the customer wants
 - Evolution “**Maintenance**” - changing the software in response to changing demands

What is a **model**?

- An **abstract representation** of a system that enables us to answer questions about the system
- Used with too large, too small, too complex, or too expensive systems

What is a software process model?

- An abstract representation of a software process, presented from a specific perspective
- Examples of process perspectives are:
 - Workflow perspective - sequence of activities
 - Data-flow perspective - information flow
 - Role/action perspective - who does what

Software Generic Process Models

Four generic process models:

1. Waterfall model
2. Evolutionary development model
3. Formal transformation model
4. Integration from reusable components model

Costs of software engineering

- Roughly:
 - **60%** of costs are development costs,
 - **40%** of costs are **testing** costs.

- For custom software:
 - maintenance costs often **exceed** development costs

Costs of software engineering (cont.)

- Costs vary depending on the type of system being developed and the requirements of system attributes such as performance and system reliability
- Distribution of costs depends on the development model that is used

What are software engineering methods?

- Structured approaches to software development which include:
 - Model descriptions
 - Descriptions of graphical models which should be produced
 - Rules
 - Constraints applied to system models
 - Recommendations
 - Advice on good design practice
 - Process guidance
 - What activities to follow

What is CASE tool (Computer-Aided Software Engineering)

- Software system (tool)
- Provides automated support for software process activities.

- Upper-CASE
 - Tools to support the **early process activities** of requirements and design
- Lower-CASE
 - Tools to support **late process activities** such as programming, debugging and testing

Attributes of good software

- The software should deliver the required functionality and performance to the user
- The software and should be maintainable, dependable and usable

Attributes of good software (cont.)

- **Maintainability**
 - Software should be designed keeping in mind that it will **evolve** to meet changing needs (changes in Business Environment)
- **Reliability**
 - Software must be reliable
- **Efficiency**
 - Software should not make wasteful use of system resources
- **Usability**
 - Software must be usable by the users for which it was designed

Key challenges facing software engineering

Challenges in coping with:

- Legacy systems
 - Old, valuable systems that must be maintained and updated
- Heterogeneity
 - Systems are distributed and include a mix of:
 - Hardware, and
 - Software written using different languages, and
 - Different operating systems
- Delivery
 - Pressure for faster delivery of software

Key challenges facing software engineering (cont.)

- **Cost:**
 - Minimize software development cost
- **Quality:**
 - Search for high quality software
- **Flexibility:**
 - System should have maximum flexibility to reduce maintenance costs

Professional and ethical responsibility

- Software engineering involves wider responsibilities than simply the application of technical skills
- Software engineers must behave in an **honest and ethically responsible way** if they are to be respected as professionals