1.1 The Nature of Software...

Software is intangible
  Hard to understand development effort

Software is easy to reproduce
  Cost is in its development
    —in other engineering products, manufacturing is the costly stage

The industry is labor-intensive
  Hard to automate

The Nature of Software ...

Untrained people can hack something together
  Quality problems are hard to notice

Software is easy to modify
  People make changes without fully understanding it

Software does not ‘wear out’
  It deteriorates by having its design changed:
    —erroneously, or
    —in ways that were not anticipated, thus making it complex

Conclusions
  Much software has poor design and is getting worse
  Demand for software is high and rising
  We are in a perpetual ‘software crisis’
  We have to learn to ‘engineer’ software
Types of Software...

Custom
For a specific customer

Generic
Sold on open market
Often called
—COTS (Commercial Off The Shelf)
—Shrink-wrapped

Embedded
Built into hardware
Hard to change

Types of Software

Differences among custom, generic and embedded software

<table>
<thead>
<tr>
<th></th>
<th>Custom</th>
<th>Generic</th>
<th>Embedded</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of copies in use</td>
<td>low</td>
<td>medium</td>
<td>high</td>
</tr>
<tr>
<td>Total processing power devoted to running this type of software</td>
<td>low</td>
<td>high</td>
<td>medium</td>
</tr>
<tr>
<td>Worldwide annual development effort</td>
<td>high</td>
<td>medium</td>
<td>low</td>
</tr>
</tbody>
</table>

Types of Software

Real time software
E.g. control and monitoring systems
Must react immediately
Safety often a concern

Data processing software
Used to run businesses
Accuracy and security of data are key

Some software has both aspects

1.2 What is Software Engineering?...

The process of solving customers’ problems by the systematic development and evolution of large, high-quality software systems within cost, time and other constraints

Solving customers’ problems
This is the goal of software engineering
Sometimes the solution is to buy, not build
Adding unnecessary features does not help solve the problem
Software engineers must communicate effectively to identify and understand the problem
What is Software Engineering?…

Systematic development and evolution
An engineering process involves applying well understood techniques in an organized and disciplined way
Many well-accepted practices have been formally standardized
—e.g. by the IEEE or ISO
Most development work is evolution

Large, high quality software systems
Software engineering techniques are needed because large systems cannot be completely understood by one person
Teamwork and co-ordination are required
Key challenge: Dividing up the work and ensuring that the parts of the system work properly together
The end-product that is produced must be of sufficient quality

Cost, time and other constraints
Finite resources
The benefit must outweigh the cost
Others are competing to do the job cheaper and faster
Inaccurate estimates of cost and time have caused many project failures

1.3 Software Engineering and the Engineering Profession

The term Software Engineering was coined in 1968
People began to realize that the principles of engineering should be applied to software development

Engineering is a licensed profession
In order to protect the public
Engineers design artifacts following well accepted practices which involve the application of science, mathematics and economics
Ethical practice is also a key tenet of the profession

1.4 Stakeholders in Software Engineering

1. Users
Those who use the software
2. Customers
Those who pay for the software
3. Software developers
4. Development Managers

All four roles can be fulfilled by the same person
1.5 Software Quality...

**Usability**
Users can learn it and fast and get their job done easily

**Efficiency**
It doesn’t waste resources such as CPU time and memory

**Reliability**
It does what it is required to do without failing

**Maintainability**
It can be easily changed

**Reusability**
Its parts can be used in other projects, so reprogramming is not needed

Software Quality...

The different qualities can conflict
Increasing efficiency can reduce maintainability or reusability
Increasing usability can reduce efficiency

Setting objectives for quality is a key engineering activity
You then design to meet the objectives
Avoids ‘over-engineering’ which wastes money

Optimizing is also sometimes necessary
E.g. obtain the highest possible reliability using a fixed budget

Internal Quality Criteria

These:
Characterize aspects of the design of the software
Have an effect on the external quality attributes
E.g.
— The amount of commenting of the code
— The complexity of the code
Short Term Vs. Long Term Quality

Short term:
- Does the software meet the customer’s immediate needs?
- Is it sufficiently efficient for the volume of data we have today?

Long term:
- Maintainability
- Customer’s future needs

1.6 Software Engineering Projects

Most projects are evolutionary or maintenance projects, involving work on legacy systems
- Corrective projects: fixing defects
- Adaptive projects: changing the system in response to changes in
  - Operating system
  - Database
  - Rules and regulations
- Enhancement projects: adding new features for users
- Reengineering or perfective projects: changing the system internally so it is more maintainable

Software Engineering Projects

‘Green field’ projects
- New development
- The minority of projects

Software Engineering Projects

Projects that involve building on a framework or a set of existing components.
- The framework is an application that is missing some important details.
  - E.g. Specific rules of this organization.
- Such projects:
  - Involve plugging together components that are:
    - Already developed.
    - Provide significant functionality.
  - Benefit from reusing reliable software.
  - Provide much of the same freedom to innovate found in green field development.
1.7 Activities Common to Software Projects...

Requirements and specification

Includes
— Domain analysis
— Defining the problem
— Requirements gathering
- Obtaining input from as many sources as possible
— Requirements analysis
- Organizing the information
— Requirements specification
- Writing detailed instructions about how the software should behave

Activities Common to Software Projects

Design
Deciding how the requirements should be implemented, using the available technology
Includes:
— Systems engineering: Deciding what should be in hardware and what in software
— Software architecture: Dividing the system into subsystems and deciding how the subsystems will interact
— Detailed design of the internals of a subsystem
— User interface design
— Design of databases

Modeling
Creating representations of the domain or the software
— Use case modeling
— Structural modeling
— Dynamic and behavioural modeling

Programming

Quality assurance
Reviews and inspections
Testing

Deployment

Managing the process

1.8 The Eight Themes of the Book

1. Understanding the customer and the user
2. Basing development on solid principles and reusable technology
3. Object orientation
4. Visual modeling using UML
5. Evaluation of alternatives
6. Iterative development
7. Communicating effectively using documentation
8. Risk management in all SE activities
1.9 Difficulties and Risks in Software Engineering

- Complexity and large numbers of details
- Uncertainty about technology
- Uncertainty about requirements
- Uncertainty about software engineering skills
- Constant change
- Deterioration of software design
- Political risks