



Object-Oriented Software Engineering

Practical Software Development using UML and Java
T. Lethbridge and R. Laganière

Chapter 1: Software and Software Engineering

www.lloseng.com

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

- Software Development is hard to automate

www.lloseng.com

© Lethbridge/Laganière 2001

Chapter 1: Software and Software Engineering

2

The Nature of Software ...

Untrained people can hack something together

- Quality problems are hard to quickly notice

Software is easy to modify

- People make changes **without** fully understanding it

Software does not **typically** 'wear out'

- Memory leaks can kill systems!
- S/w deteriorates by having its design changed:
 - erroneously, or
 - in ways that were not anticipated, thus making it complex



www.lloseng.com

© Lethbridge/Laganière 2001

Chapter 1: Software and Software Engineering

3

The Nature of Software

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



www.lloseng.com

© Lethbridge/Laganière 2001

Chapter 1: Software and Software Engineering

4

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
- Software engineers must communicate effectively to identify and understand the problem



www.lloseng.com

© Lethbridge/Laganière 2001

Chapter 1: Software and Software Engineering

5

What is Software Engineering?...

Systematic development and evolution

- An engineering process involves applying well understood techniques in a organized and disciplined way
- Many well-accepted practices have been formally standardized
 - e.g. by the IEEE or ISO
 - Practice can be distant from theory ☺
- 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



www.lloseng.com

© Lethbridge/Laganière 2001

Chapter 1: Software and Software Engineering

6

What is Software Engineering?

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



www.lloseng.com

© Lethbridge/Laganière 2001

Chapter 1: Software and Software Engineering

7

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



www.lloseng.com

© Lethbridge/Laganière 2001

Chapter 1: Software and Software Engineering

8

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

www.lloseng.com

© Lethbridge/Laganière 2001

Chapter 1: Software and Software Engineering

9

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

www.lloseng.com

© Lethbridge/Laganière 2001

Chapter 1: Software and Software Engineering

10

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



www.lloseng.com

© Lethbridge/Laganière 2001

Chapter 1: Software and Software Engineering

11

1.7 Activities Common to Software Projects...

Requirements and specification (Analysis?)

- 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



www.lloseng.com

© Lethbridge/Laganière 2001

Chapter 1: Software and Software Engineering

12

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

www.lloseng.com

© Lethbridge/Laganière 2001

Chapter 1: Software and Software Engineering

13

Activities Common to Software Projects

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

www.lloseng.com

© Lethbridge/Laganière 2001

Chapter 1: Software and Software Engineering

14

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**



www.lloseng.com

© Lethbridge/Laganière 2001

Chapter 1: Software and Software Engineering

15

Other Material from Chapter 1: Software and Software Engineering



www.lloseng.com

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

www.lloseng.com

© Lethbridge/Laganière 2001

Chapter 1: Software and Software Engineering

17

Types of Software

Differences among custom, generic and embedded software

	Custom	Generic	Embedded
Number of <i>copies</i> in use	low	medium	high
Total <i>processing power</i> devoted to running this type of software	low	high	medium
Worldwide annual <i>development effort</i>	high	medium	low

www.lloseng.com

© Lethbridge/Laganière 2001

Chapter 1: Software and Software Engineering

18

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



Software Quality...

Customer:

solves problems at an acceptable cost in terms of money paid and resources used

User:

easy to learn; efficient to use; helps get work done

QUALITY SOFTWARE

Developer:

easy to design; easy to maintain; easy to reuse its parts

Development manager:

sells more and pleases customers while costing less to develop and maintain



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

