1. Data Conceptualization
2. Python
3. Testing & Debugging
4. Basics of Software Engineering
5. Software is becoming so prevalent in nearly everything we do
6. Failures impact everyone

c. Cost of failure becoming very high

i. Financial

ii. Loss of life

iii. Time

iv. Loss of equipment

v. Inconvenience

d. Describe what Software Engineering is

i. “Technological”

ii. “Managerial”

iii. “Systematic”

iv. Development that is “on time” and “within cost estimates”

It is engineering versus a science because the focus is on production, practicality, quality, maintenance, reuse, standards, teams, management, and so on.

To put it simply, software engineering is about the following:

* Engineering
* Participating in a systematic, careful, disciplined, scientific activity
* Building software systems, particularly larger ones (hacking or debugging until it works won’t work)
* Modifying software systems over time (not just focused on creating new applications)

e. Programming in the large vs Programming in the small

The size of a project matters. You must consider the number of people involved, how long the task is expected to take, what the risks and consequences are, and, of course, economics.

|  |  |  |  |
| --- | --- | --- | --- |
| **Category** | **Programmers** | **Duration** | **Product Size (Lines of Code)** |
| Trivial | 1 | 1–4 weeks | Less than 500 |
| Small | 1–2 | 1–6 months | 500–3,000 |
| Medium | 2–5 | 6 months–2 years | 3,000–20,000 |
| Large | 5–20 | 2–3 years | 20,000–100,000 |
| Very Large | 20–200 | 3–6 years | 100,000–1,000,000 |
| Extremely Large | More than 200 | >More than 6 years | More than 1,000,000 |

f. Software Development Lifecycle and Phases

i. Requirements – Requirements are statements of what the system should do. This is from the customer of client point of view and is not expressed in terms of a solution. Requirements should be: Clear, unambiguous statements of what must be done, consistent, complete, feasible, and testable.

ii. Design – Design is a description of how a solution will be implemented. It is a model or blueprint that includes the specifications for building something that will satisfy or meet the requirements. The design phase is done before the implementation phase.

iv. Integration – This phase involves combining different parts of the system to ensure they work together. This also involves integrating the system with the environment.

iii. Coding/Implementation – Implementation(aka coding) begins once the developer receives the design documents. These designs are translated into source code and all components of the software are implemented.

v. Maintenance – Once the software passes through all the previous phases, it is released to the client and placed in the environment for which it is designed. It then enters the maintenance phase.

vi. (… don’t forget testing throughout! )

g. Relative cost or time per phase

i. Which phase takes up the biggest chunk? Be able to explain why

Requirements – 7%

Design – 6%

Module Coding – 5%

Module Testing – 7%

Integration – 8%

Maintenance – 67%

Note on maintenance: Fixing errors isn’t the only thing that happens in maintenance. This is called “corrective maintenance” and constitutes only about 20%m of maintenance. Other activities include “perfective maintenance” (60%) where the software is enhanced to add new features, and “adaptive” maintenance(20%) where the system is adapted to make it work in changing environments.

h. Requirements

i. Client point-of-view

ii. Objective CS5010 | Pg 3

iii. Validation tests

i. Functional requirements

i. “What” the system or application does (describes a function or behavior)

j. Non-functional requirements

i. “How” the system does things (qualifies a given functional requirement)

ii. A “quality” or property the product has (e.g. efficiency)

k. Constraint – a design requirement directly from the client

l. Cost increases as faults are found later / importance of front-end

1. Software Development methods/Models
2. SCRUM