

Lecture Zero

Software Engineering

CSE320

Course details



- **LTP – 3 0 0 [Three lectures/week]**

- **Text Book**

FUNDAMENTALS OF SOFTWARE ENGINEERING by RAJIB MALL, PHI (PRETICE HALL INDIA),

Course Assessment Model

• Marks break up*	
• Attendance	5
• CA (3 online Assignments)	25
• MTT	20
• ETE	50
• Total	<hr/> 100

Detail of Academic Tasks

- AT1: Online Assignment1
- AT2: Online Assignment2
- AT3: Online Assignment3

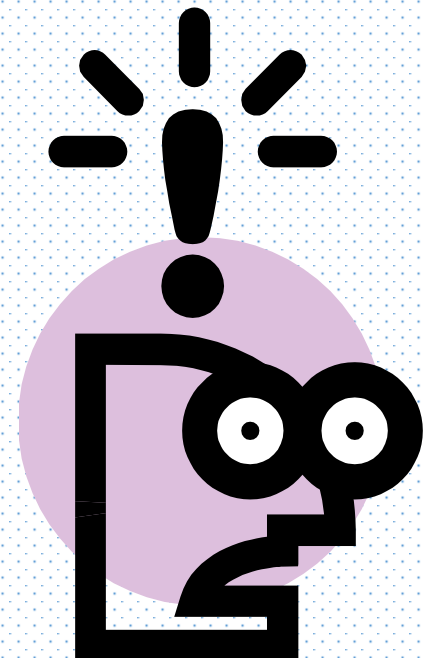
****** 2 best out of three******

MTT and ETT

- MTT:-- All MCQs
- ETT:-- MCQs

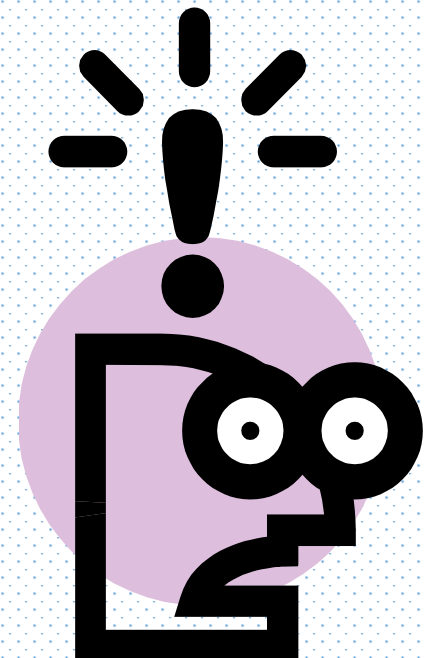
Course Outcomes

- Plan and deliver an effective software engineering process, based on knowledge of widely used development life cycle models.
- Work effectively in a team to analyze the requirements of a complex software system and solve problems by creating appropriate designs that satisfies these requirements.
- Translate a requirements specification into an implementable design, following a structured and organized process.



Course Outcomes

- Formulate a testing strategy for a software system, employing test case design techniques such as functional and structural testing.
- Manage a project including planning, scheduling, estimation and configuration management.
- Recognize current trends in the area of software engineering.



The course contents

Before MTE

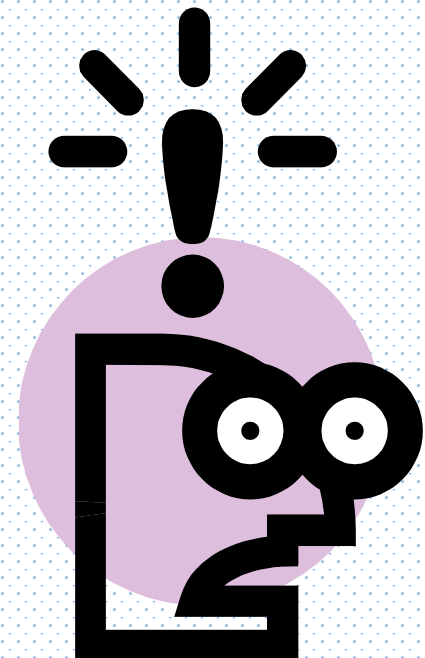
- **Introduction to software engineering :**

Evolution and impact of software engineering, *Software life cycle models*, *Feasibility study*, *Functional and non-functional requirements*, Requirement gathering, Requirement analysis and specification

- **Issues in software design : cohesion, coupling, DFDs**

- **Object modelling :**

Object modelling using UML, Object oriented software development, User interface design, Coding standards and code review techniques



The course contents

- **Testing :**

Fundamentals of testing, White box and black box testing, Test coverage analysis and test case design techniques, Mutation testing, Static and dynamic analysis, Software reliability metrics, Reliability growth modelling.

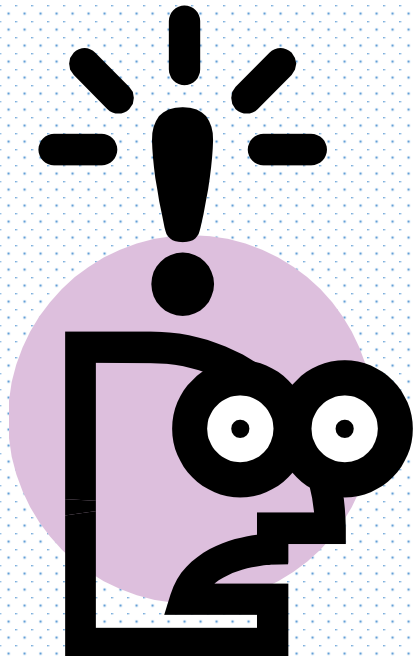
After MTE

- **Software project management :**

Project management, Project planning and control, Cost estimation, Project scheduling using PERT and GANTT charts, Software Configuration Management

- **Quality management :**

Quality management, ISO and SEI CMMI, PSP and Six sigma, Software Maintenance, reuse, CBSD, CASE, Advance topics of Software Engineering.



The hitch...

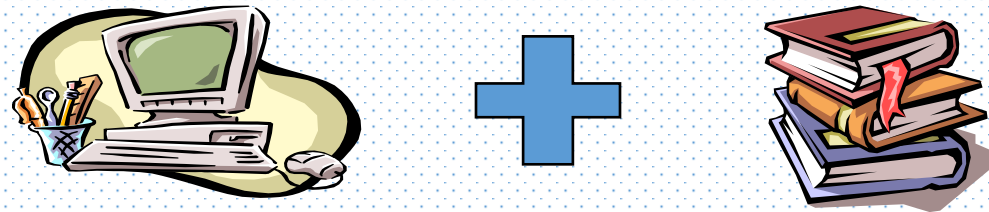
The three BURNING questions in mind...

- What is software? Is it different from Program?
- What is Software Engineering?
- **Why Software Engineering?**
- What are learning outcomes?



What is software?

- ❑ Computer programs and associated documentation



- ❑ Software products may be developed for a particular customer or may be developed for a general market.

- ❑ Software products may be

1. **Generic** - developed to be sold to a range of different customers



2. **Bespoke** - developed for a single customer according to their specification



What is software engineering?

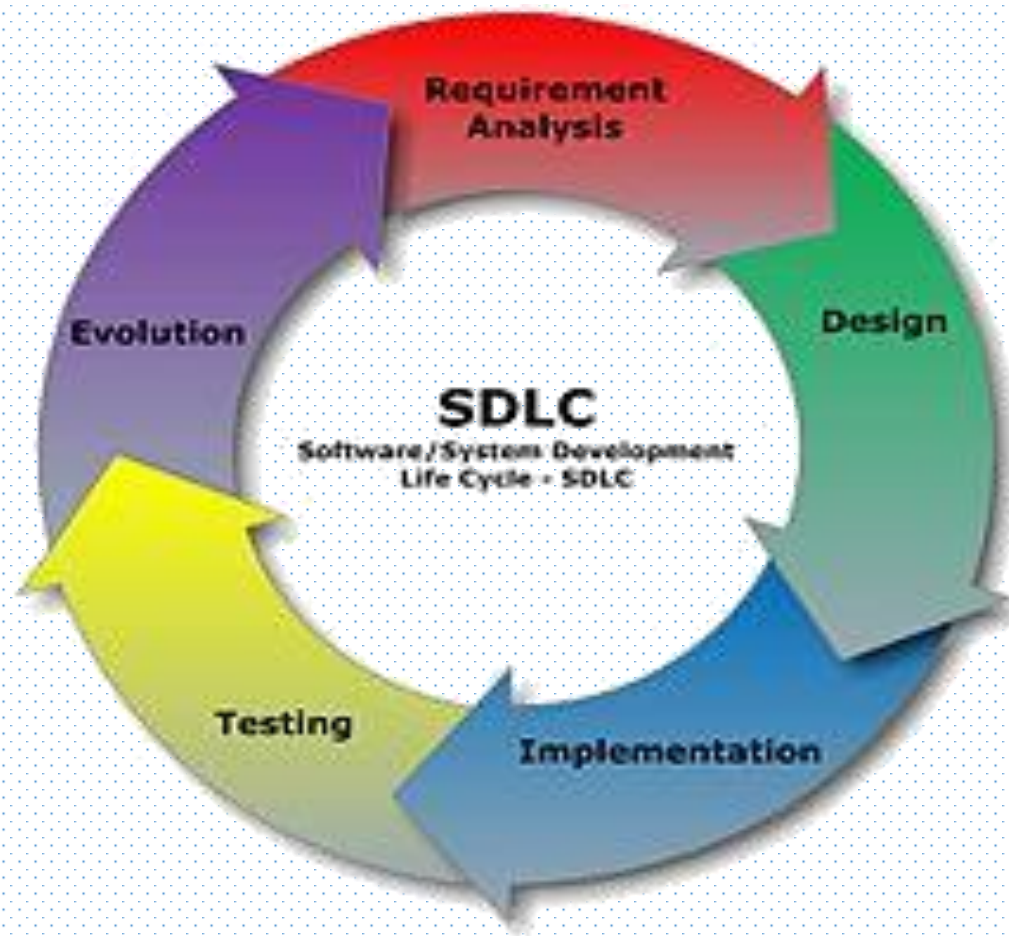
Software engineering is an engineering discipline which is concerned with all aspects of software production

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



Phases of Development



The Role of Software Engineering-1

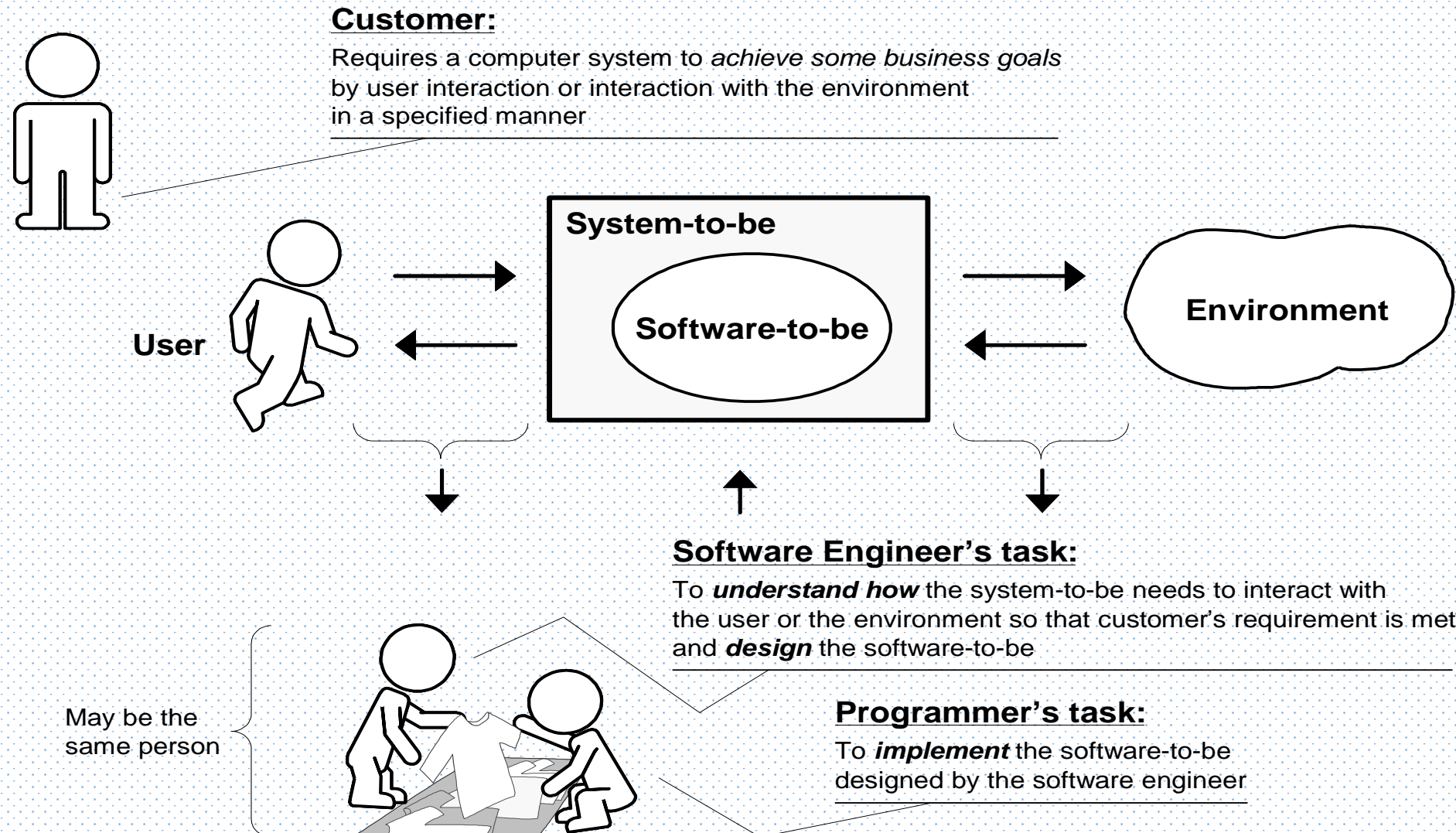
A bridge from customer needs to programming implementation



First law of software engineering

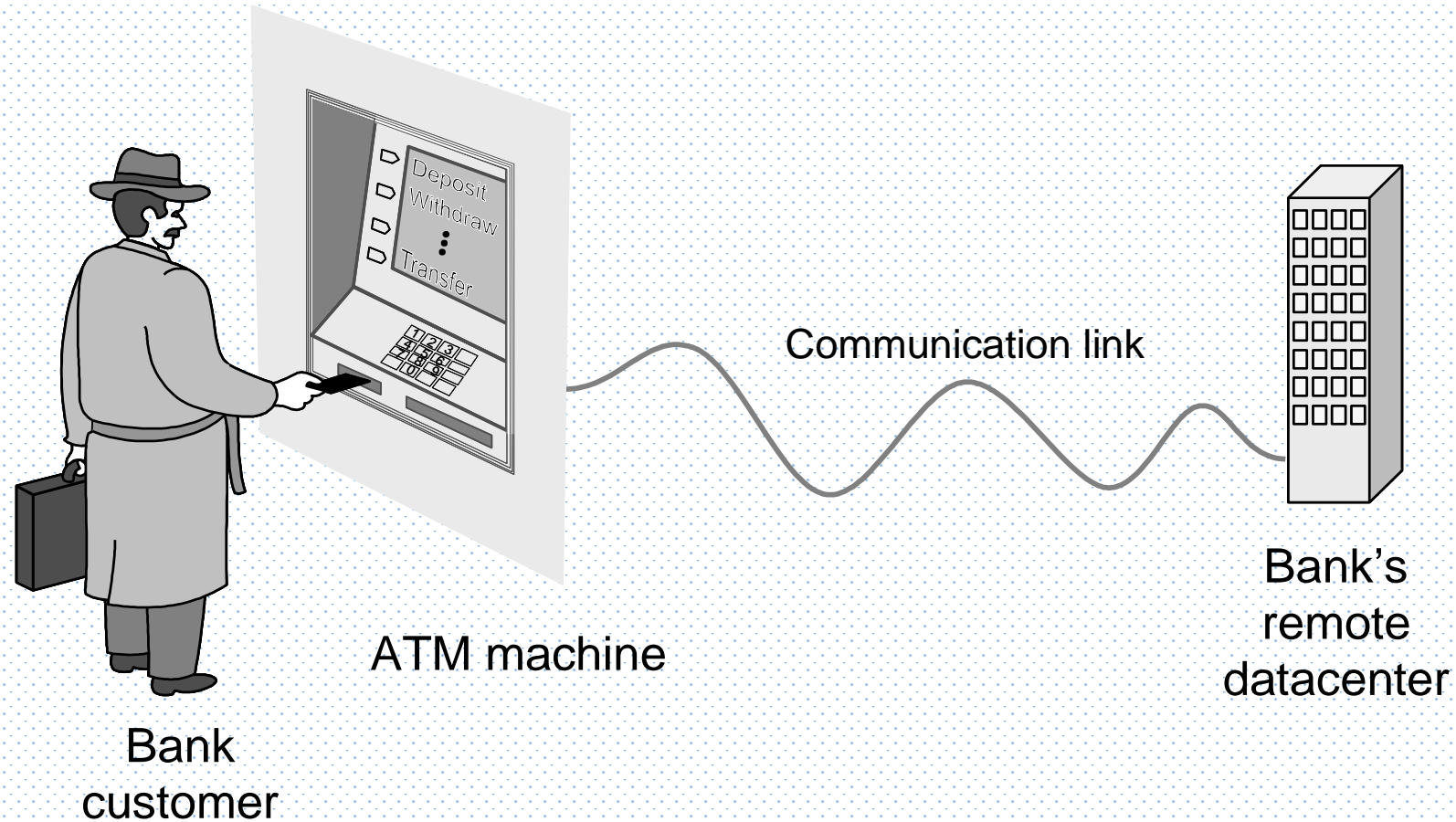
Software engineer must learn the problem domain (problem cannot be solved without understanding it first)

The Role of Software Engineering-2



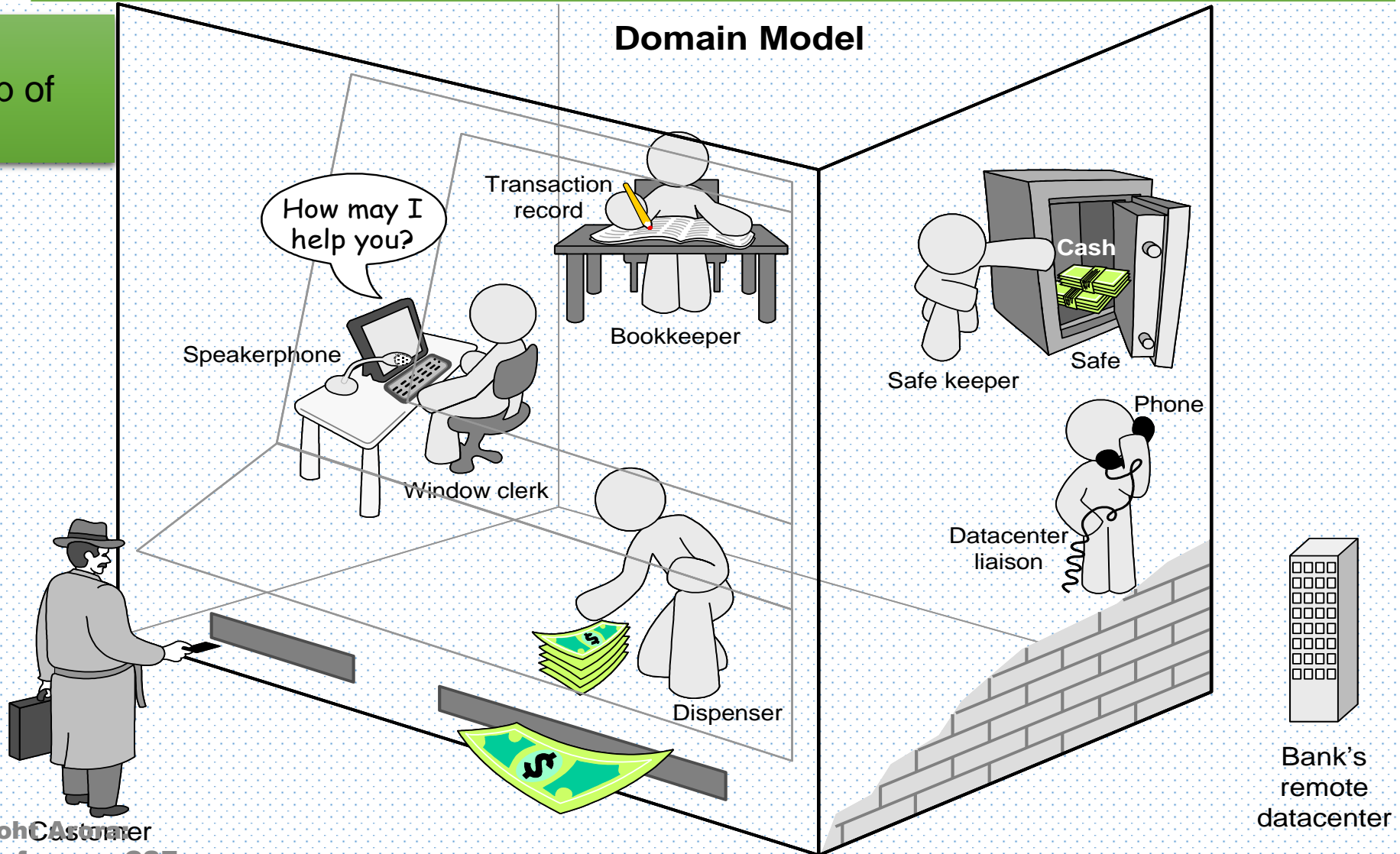
Example: ATM Machine

Understanding the money-machine problem:

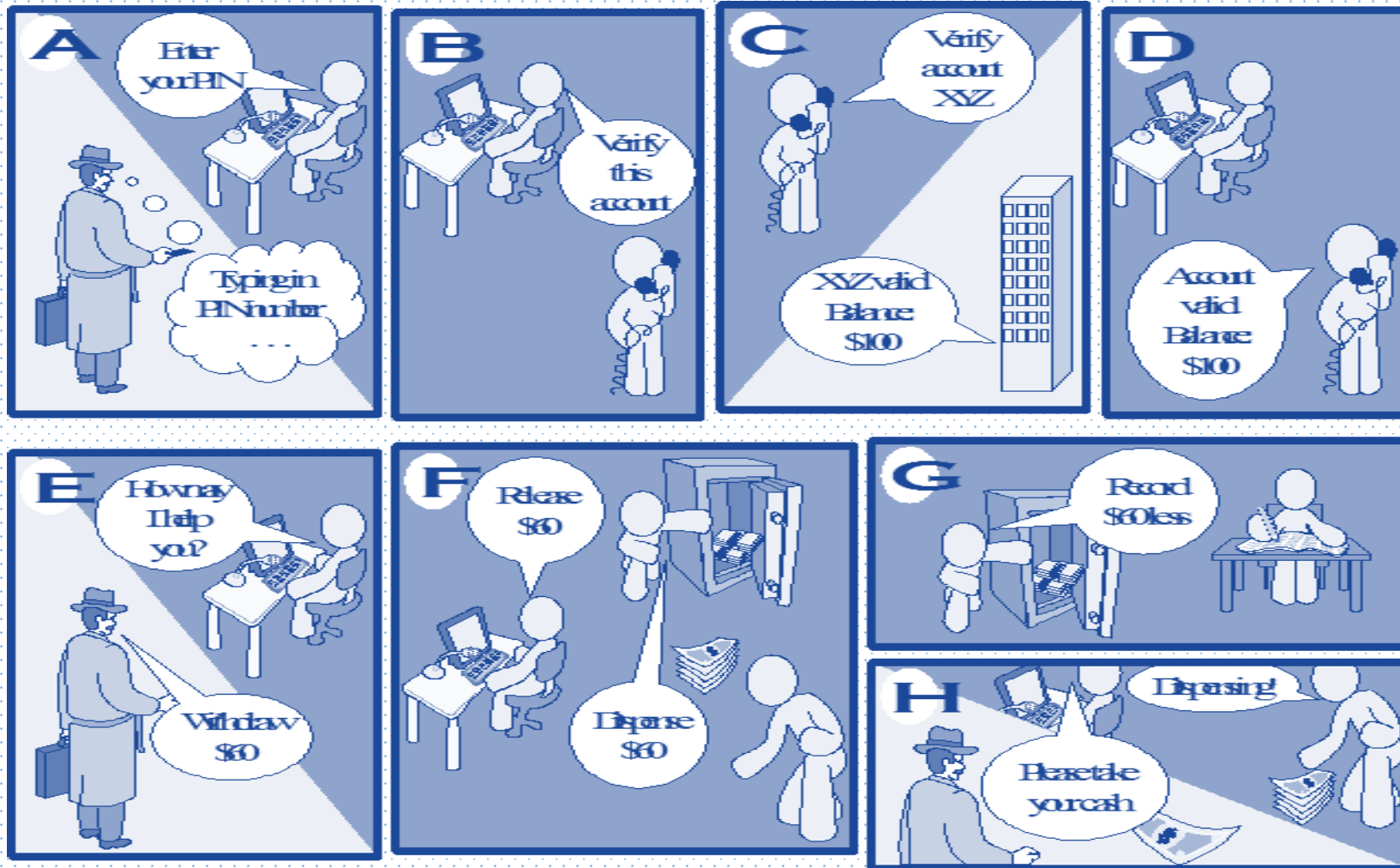


How ATM Machine Might Work

Domain model
created with help of
domain expert



Cartoon Strip: How ATM Machine Works?



Software Engineering Blueprints

- Specifying software problems and solutions is like cartoon strip writing
- Unfortunately, most of us are not artists, so we will use something less exciting:
Designing symbols
- However ...

Second Law of Software Engineering



- **Software should be written for people first**
 - (Computers run software, but hardware quickly becomes outdated)
 - Useful + good software lives long
 - To nurture software, people must be able to understand it

Software Development Methods

➤ Method = work strategy

- The Feynman Problem-Solving Algorithm:
(i) Write down the problem (ii) think very hard, and (iii) write down the answer.

➤ Waterfall

- Unidirectional, finish this step before moving to the next

➤ Iterative + Incremental

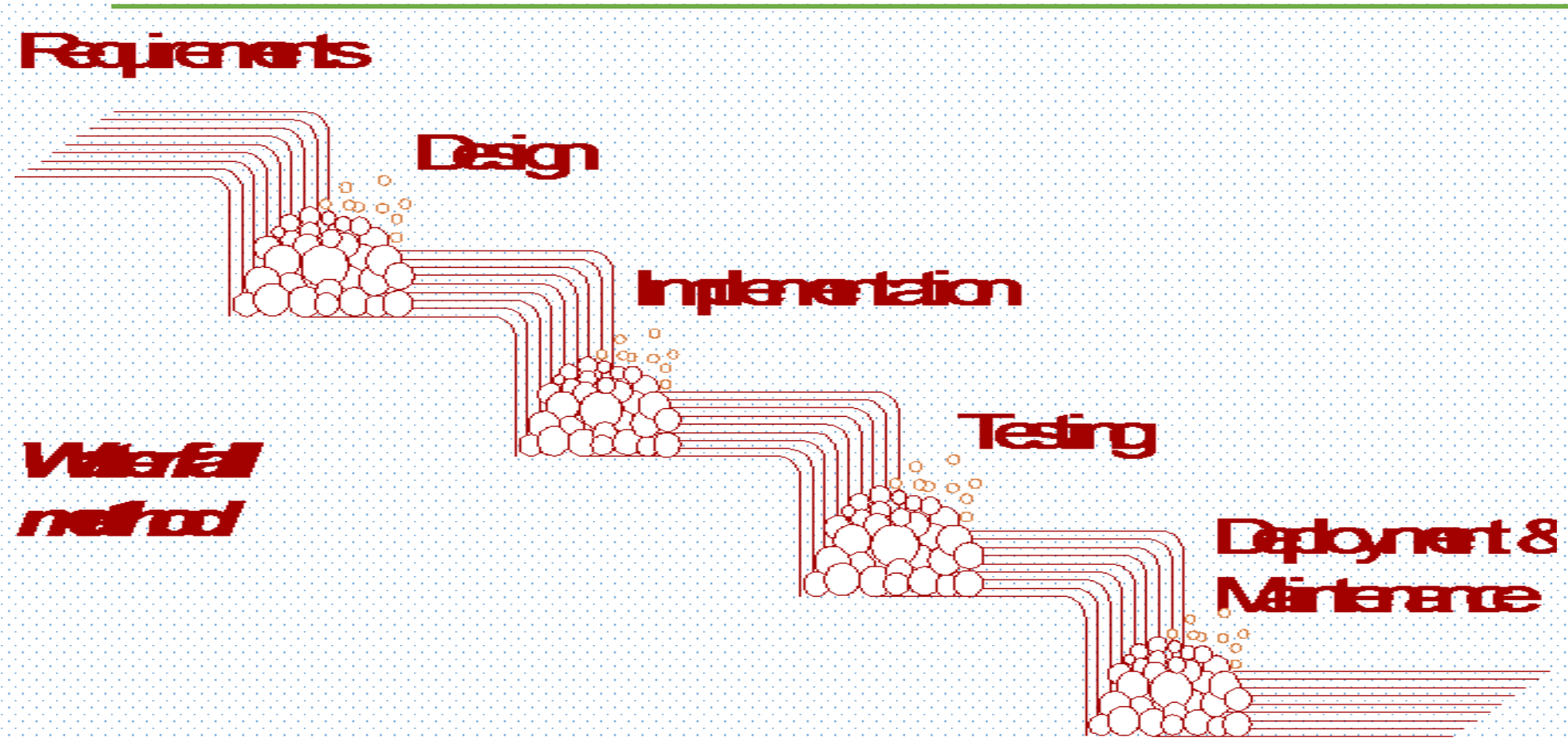
- Develop increment of functionality, repeat in a feedback loop

➤ Agile

- User feedback essential; feedback loops on several levels of granularity

Software Development Methodologies

Waterfall Method



Unidirectional, no way back finish this step before moving to the next

Software myths

1. “If we get behind schedule, we can just add more people”

- ❑ Fact: Adding people to a late project makes it even later.
- ❑ Someone has to teach the new people.

2. “A general statement of objectives is enough to start programming”.

- ❑ Fact: Incomplete requirements are a major cause for project failures.

3. “Changes in requirements are easy to deal with because software is flexible”.

- ❑ Fact: Changes are hard and expensive.
- ❑ Especially during coding and after software deployment.

Software myths

4. “Once we get the program running, we are done”

- ☐ Fact: Most effort comes after the software is delivered for the first time.
- ☐ Bug fixes, feature enhancements, etc

5. “The only product is the running program”

- ☐ Fact: Need the entire configuration
- ☐ Documentation of system requirements, design, programming, and usage

Software crises

- The various software crises are:
 1. Over-budget.
 2. Not delivering product on time.
 3. Product is of poor quality.
 4. Software product is not meeting the customer requirements.

What are the attributes of good software?

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

- **Maintainability**
 - Software must evolve to meet changing needs
- **Dependability**
 - Software must be trustworthy
- **Efficiency**
 - Software should not make wasteful use of system resources
- **Usability**
 - Software must be usable by the users for which it was designed



Next Class: Software Life Cycle Models

www.lpu.in