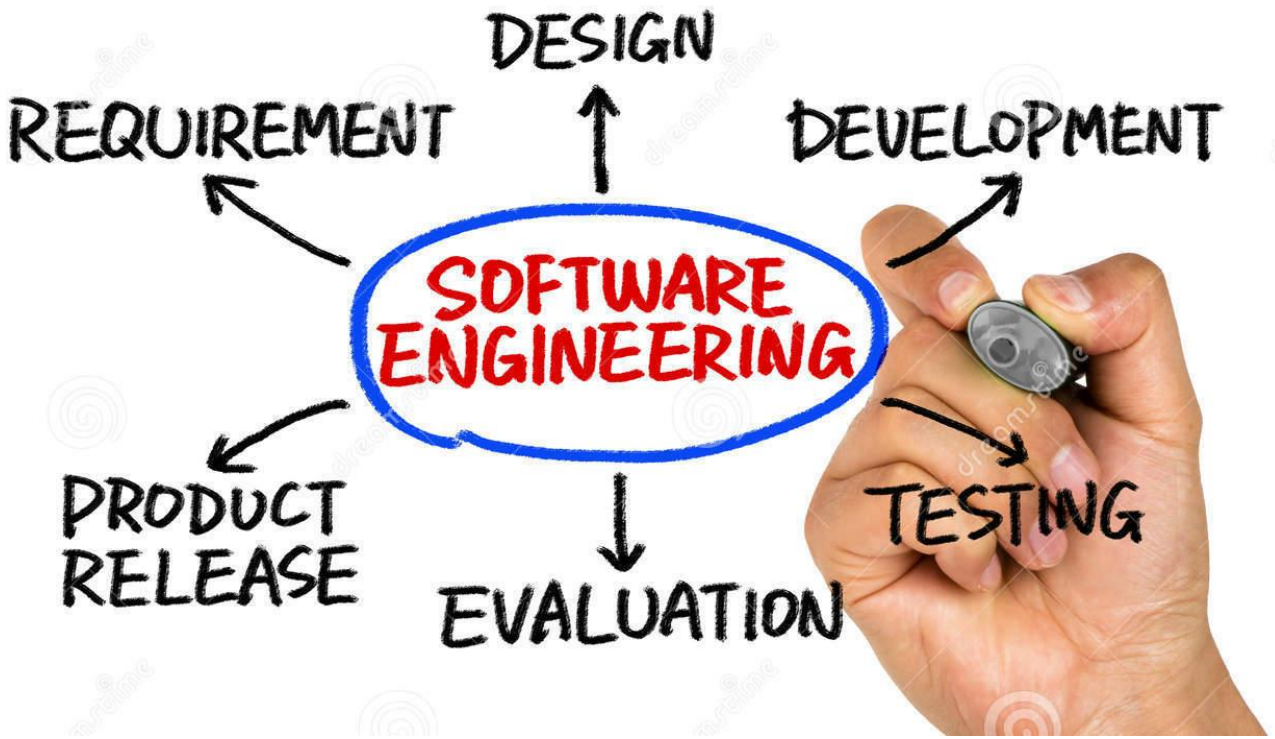

# Fundamentals of Software Engineering

LECTURE 2: SE MOTIVATION, APPLICATION TYPES AND PROCESSES




# Software Engineering

# What is Software Engineering?

- As defined in IEEE Standard 610.12:
  - *The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software.*
- Your opinion?
- This definition is descriptive, not prescriptive
  - It does not say how to do anything
  - It just say what qualities S.E. should have
  - As a result, many people understand SE differently

# Why Software Engineering

- It is the best way or programs to get the following points:

- To provide the best output of software system.
- To make easy to use the software systems and develop them.
- To improve the rate of production.
- To maintain the budget for development of Software system.
- Job satisfaction of software engineering.

# Why Software Engineering

- It is necessary to produce a high-quality software PRODUCT to fulfill the below given points.
  - Consistency
  - Improved quality
  - Minimum cost
  - Within time
  - Reliability &
  - Fulfill the need of user

# Software Engineering Myths: Practitioner

"Let's write the code, so we'll be done

"The sooner you begin writing code, the longer it'll take to

60 80% of effort is expended after first delivery

"Until I finish it, I cannot assess its

Software and design reviews are more effective than testing (find 5 times more bugs)

"There is no time for software engineering

But is there time to redo the software?

# Software Engineering Myths: Customer

"We can refine the requirements

- A recipe for disaster.

"The good thing about software is that we can change it later

- As time passes, cost of changes grows rapidly

# Software Engineering Myths: Management

We have books with rules. Isn't that everything my people need?"

Which book do you think is perfect for you?

"If we fall behind, we add more

"Adding people to a late software project, makes it later" Fred Brooks (The Mythical Man Month)

"We can outsource

–If you do not know how to manage and control it internally, you will struggle to do this with outsiders

# Software Engineering

- Software Engineering ≠ Programming
- Programming is actually a pretty small part.
- Other aspects
  - Determining what to build
    - Requirements (what tasks should the Software accomplish?)
    - Specifications (exact operating behaviour)
  - Determining how to build it (Design)
  - Testing (functionality and experience)
  - Debugging (functional and performance)
  - Maintaining the program (new APIs, platforms, minor features)
  - More and more, these are not linear steps.

# Software Crisis

- Software Engineering is a response to "Software Crisis"
- What is a "Software Crisis"?
  - it refers to the difficulty of writing correct, understandable, and verifiable computer programs. The roots of the software crisis are complexity, expectations, and change.
- Does any of the software you use have problems?
- Does any of the software you use not have problems?

# Software Development Problems

- Always (usually) Late
- Exceeds cost
- Doesn't satisfy specifications
- Bad or weak design
- Not understandable or modifiable
- Unreliable
- Inefficient (time, memory, disk)
- Not portable
- Not secure

**MODERN RESOLUTION FOR ALL PROJECTS**

| | 2011 | 2012 | 2013 | 2014 | 2015 |
|---|---|---|---|---|---|
| SUCCESSFUL | 29% | 27% | 31% | 28% | 29% |
| CHALLENGED | 49% | 56% | 50% | 55% | 52% |
| FAILED | 22% | 17% | 19% | 17% | 19% |

*The Modern Resolution (OnTime, OnBudget, with a satisfactory result) of all software projects from FY2011-2015 within the new CHAOS database. Please note that for the rest of this report CHAOS Resolution will refer to the Modern Resolution definition not the Traditional Resolution definition.*

# Success, Challanged and Failed Project Ratio Based on Scale

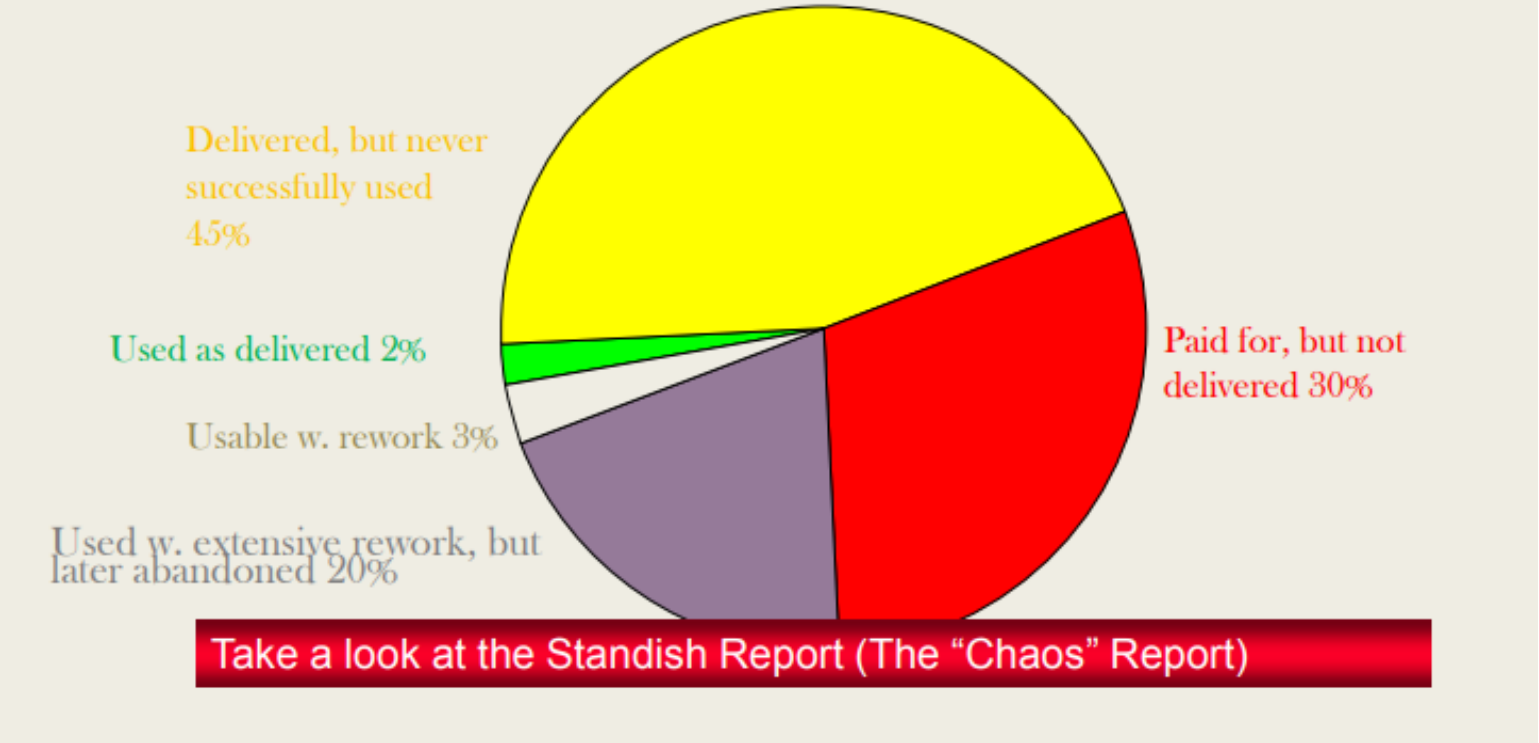
**CHAOS RESOLUTION BY PROJECT SIZE**

| | SUCCESSFUL | CHALLENGED | FAILED |
|---|---|---|---|
| Grand | 2% | 7% | 17% |
| Large | 6% | 17% | 24% |
| Medium | 9% | 26% | 31% |
| Moderate | 21% | 32% | 17% |
| Small | 62% | 16% | 11% |
| TOTAL | 100% | 100% | 100% |

*The resolution of all software projects by size from FY2011–2015 within the new CHAOS database.*

# Why Software Engineering?

## 9 SOFTWARE PROJECTS TOTALING $96.7 MILLION: WHERE THE MONEY WENT



# Software Production has a Poor Track Record Example: Space Shuttle Software



- Cost: $10 Billion, millions of dollars more than planned
- Time: 3 years late
- Quality: First launch of Columbia was cancelled because of a synchronization problem with the Shuttle's 5 onboard computers.
  - – Error was traced back to a change made 2 years earlier when a programmer changed a delay factor in an interrupt handler from 50 to 80 milliseconds.
  - – The likelihood of the error was small enough, that the error caused no harm during thousands of hours of testing.

Substantial errors still exist. – Astronauts are supplied with a book of known software problems "Program Notes and Waivers".

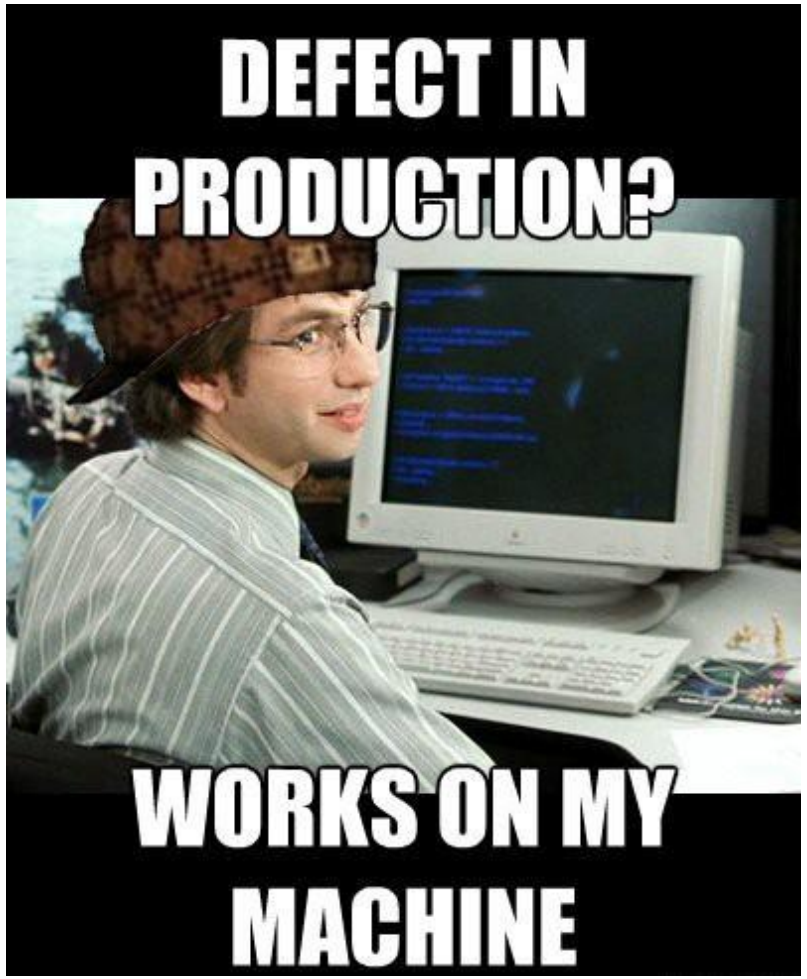
# Ariane 5 Flight 501 – Reusability Failure

- Europe's newest un-manned satellite-launching rocket reused working software from its predecessor, the Ariane 4. Unfortunately, the Ariane 5's faster engines exploited a bug that was not found in previous models. Thirty-six seconds into its maiden launch the rocket's engineers hit the self destruct button following multiple computer failures. In essence, the software had tried to cram a **64-bit number** into a **16-bit space**. The resulting overflow conditions crashed both the primary and backup computers (which were both running the exact same software).

- The Ariane 5 had cost nearly **$8 billion to develop** and was carrying a **$500 million satellite payload** when it exploded.

# National Health Service

- In the year 2016, it was discovered that the clinical computer system SystmOne had an error that since 2009 had been miscalculating patient's medicines who were at risk of heart attack. As a result, many patients suffered heart attacks or strokes since they were told they were at low-risk, while other suffered from the side-effects of taking unnecessary medication




# End User's Conclusion

# What is a Product?

- A product is something that is produced and sold in large quantities, often as a result of a manufacturing process.
- Computer software (Apps) is a work product that software professionals/ practitioners build/ developed and then support over many years. These work products includes programs that executes within computers of any size and architecture
- Software Engineering encompasses a process, a collection of methods and an array of tools that allow professionals to build high quality computer software product.
- Computer Software product can be used in ATM, Mobile and other tools.

# Who is an engineer?

- An engineer is a person who uses scientific knowledge to design, construct, and maintain engines and machines or structures such as roads, railways, and bridges.

# What is a 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:
  - Generic - developed to be sold to a range of different customers (developed as a product)
  - Bespoke (custom) - developed for a single customer according to their specification (developed as a project)

# 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 organized approach to their work and use appropriate tools and techniques depending on the problem to be solved, the development constraints and the resources available.

# What is the work product?

- From the Software engineer's pint of view, the work product is the set of programs, content (data/ docs) and other work products that support computer software Apps.
- From the user's point of view, the work product is a tool or product that some how makes the user's world better.

# Difference between software engineering and computer science?

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

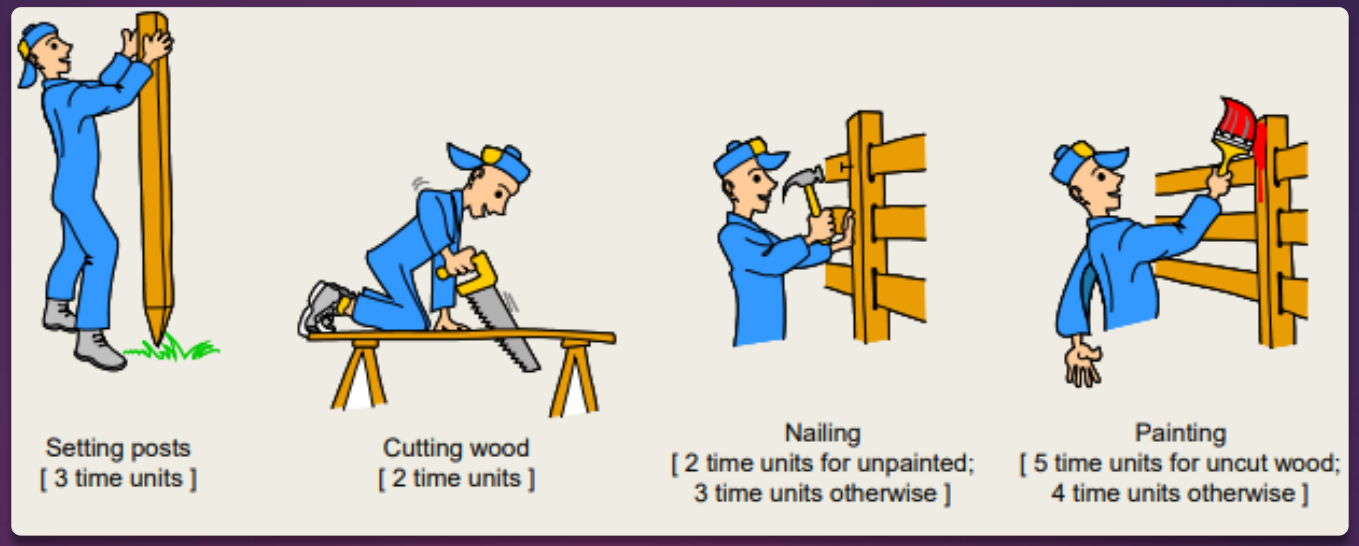
- Computer science theories are currently insufficient to act as a complete underpinning for software engineering

# Software Engineers and their Companions?

- Software Engineers
- Software Developers
- Software Designers
- Database Experts
- Database Administrators
- Quality Assurance Experts
- IT System Network Experts
- IT Supporting Staff
- End Users (Clients)

# Software is Complex

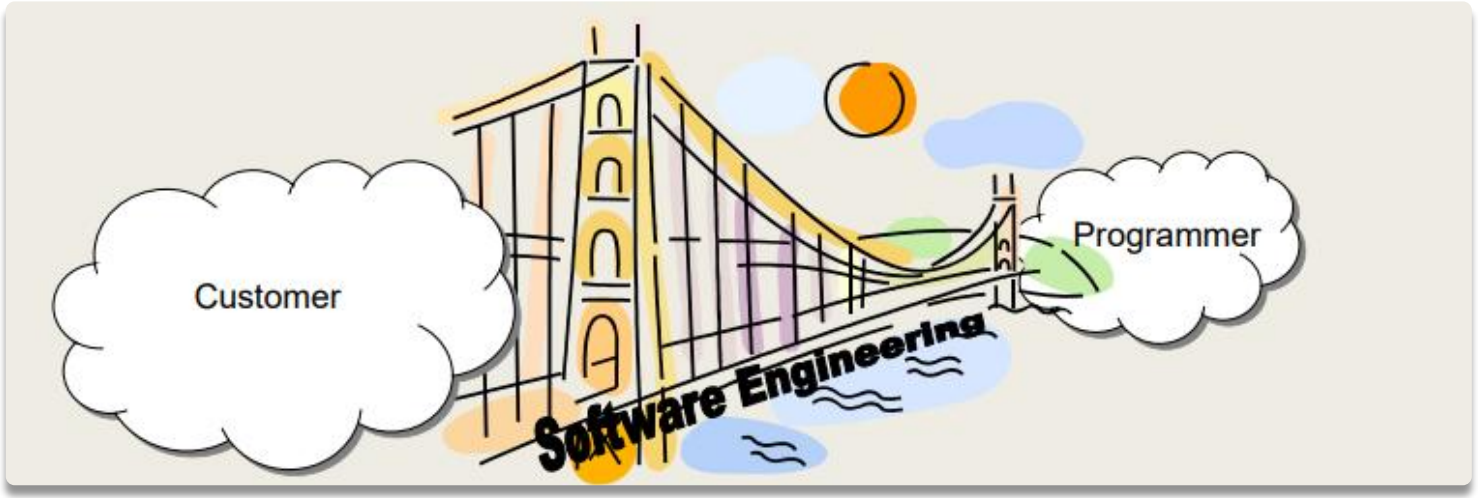
- Complex ≠ complicated
- Complex = composed of many simple parts related to one another
- Complicated = not well understood, or explained




Complexity Example: Scheduling Fence Construction Tasks
Setting posts < Nailing, Painting
Cutting < Nailing …
shortest possible completion time = ?
"simple" problem, but hard to solve without a pen and paper

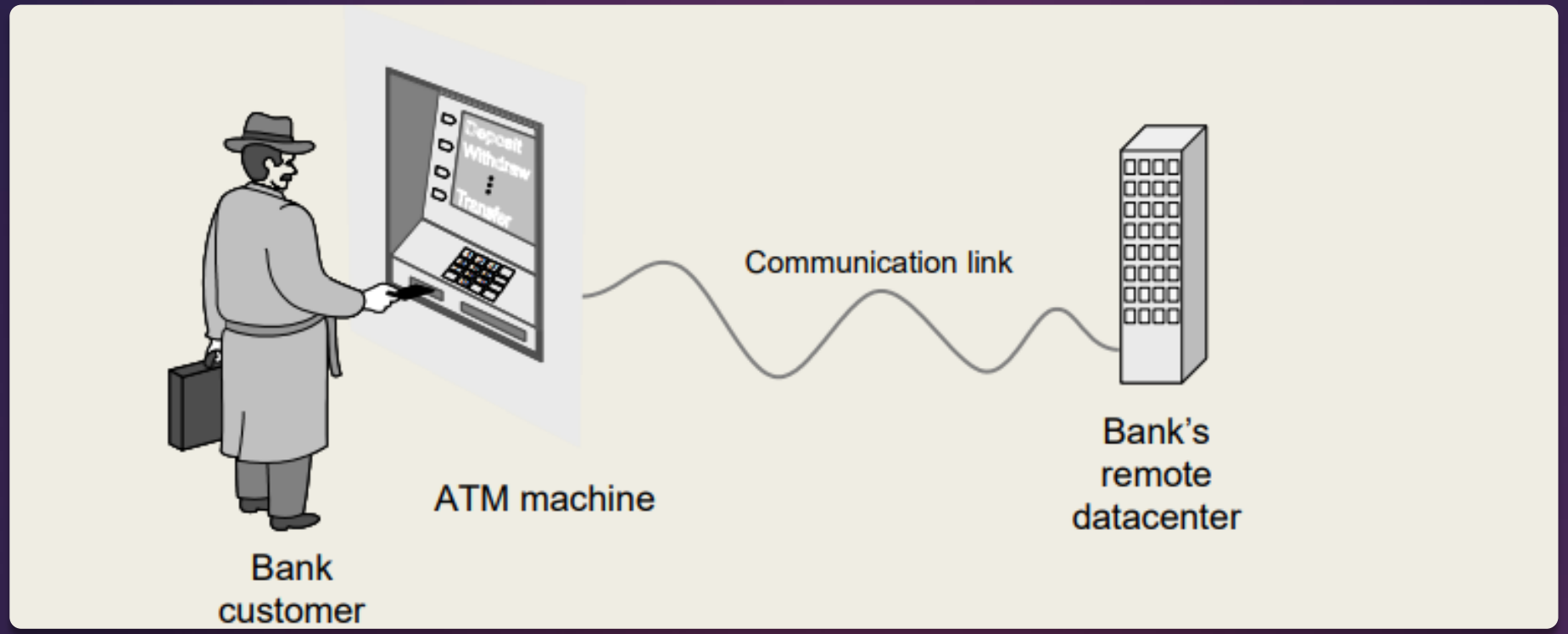
# The Role of Software Engg.

- **First law of software engineering**
- Software engineer is willing to learn the problem domain (problem cannot be solved without understanding it first)



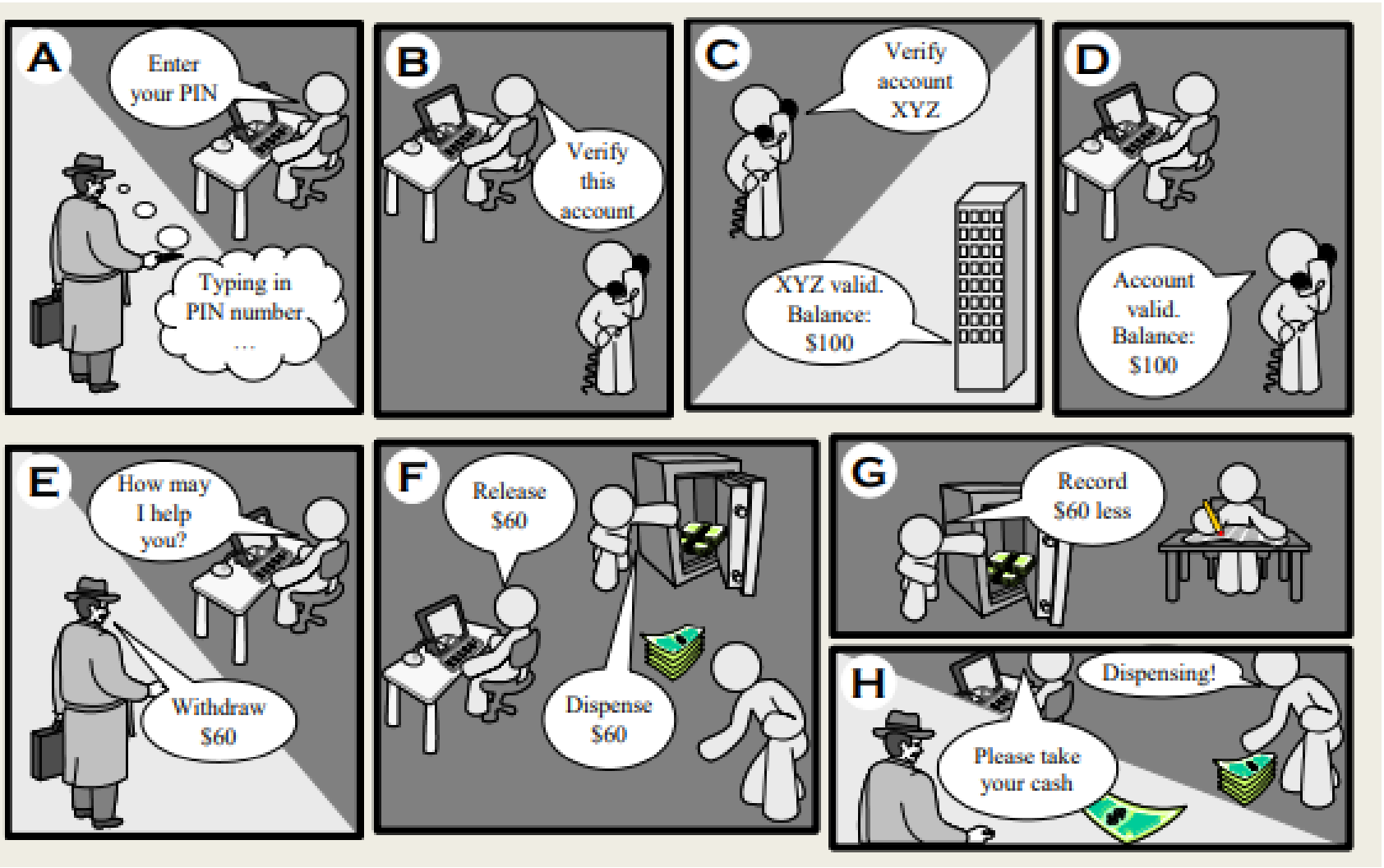
# Second Law of Software Engineering

- 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.




# Example – ATM Machine (How it works)

# Cartoon Strip: How ATM Machine Works




Thank
You