

# Software Engineering

## Lecture 1: Introduction to software engineering



# Course Objectives

- To learn about all the difficulties in developing software so that we can avoid myths in software design.
- To learn about different software processes so that we can choose a suitable one.
- To learn to design high-quality efficient software so that it is usable and maintainable.
- To learn about advanced methods for software engineering.

# Reference book

- <https://iansommerville.com/software-engineering-book/>

## Software Engineering, Tenth Edition

A comprehensive textbook on software engineering

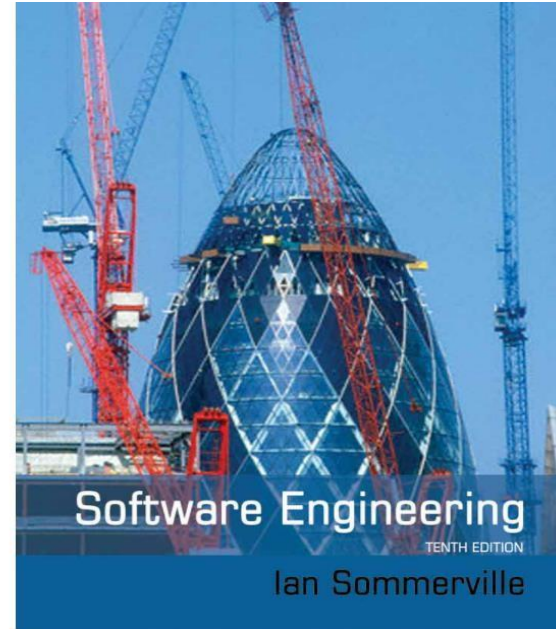
The tenth edition of my Software Engineering textbook was published in April 2015. The book is organized into four parts and focuses on the methods, tools and techniques used in the development of software systems. This edition is oriented towards systems engineering with new chapters on systems engineering, resilience engineering and systems of systems.

'Software Engineering' has been designed to support both introductory and advanced courses in software engineering.

*GDPR compliance: I do not set cookies or store personal information about visitors to this site.*

**Twitter:** @iansommerville

**Email:** [iansommerville at geemail.com](mailto:iansommerville@gmail.com)





# References

- David Farley (2022). Modern Software Engineering, 1st Edition. PEARSON.
- <https://www.lucidchart.com/pages/uml-class-diagram>
- <https://www.geeksforgeeks.org/software-engineering-cocomo-model>
- <https://www.tutorialspoint.com/uml/index.htm>





# LMS

- Course Materials: Lectures and Lab Assignments
- Online Quizzes
- Project Submission
- Lab Assignment Submission





# Grades

- Midterm: 15 Degree
- Activities
  - Project: 20 Degree
  - Quizzes (Practical/LMS): 20 Degree (4 quizzes)
  - Practical Assignments : 20 Degree (LMS + Discussion)
- Final Exam: 25 Degree



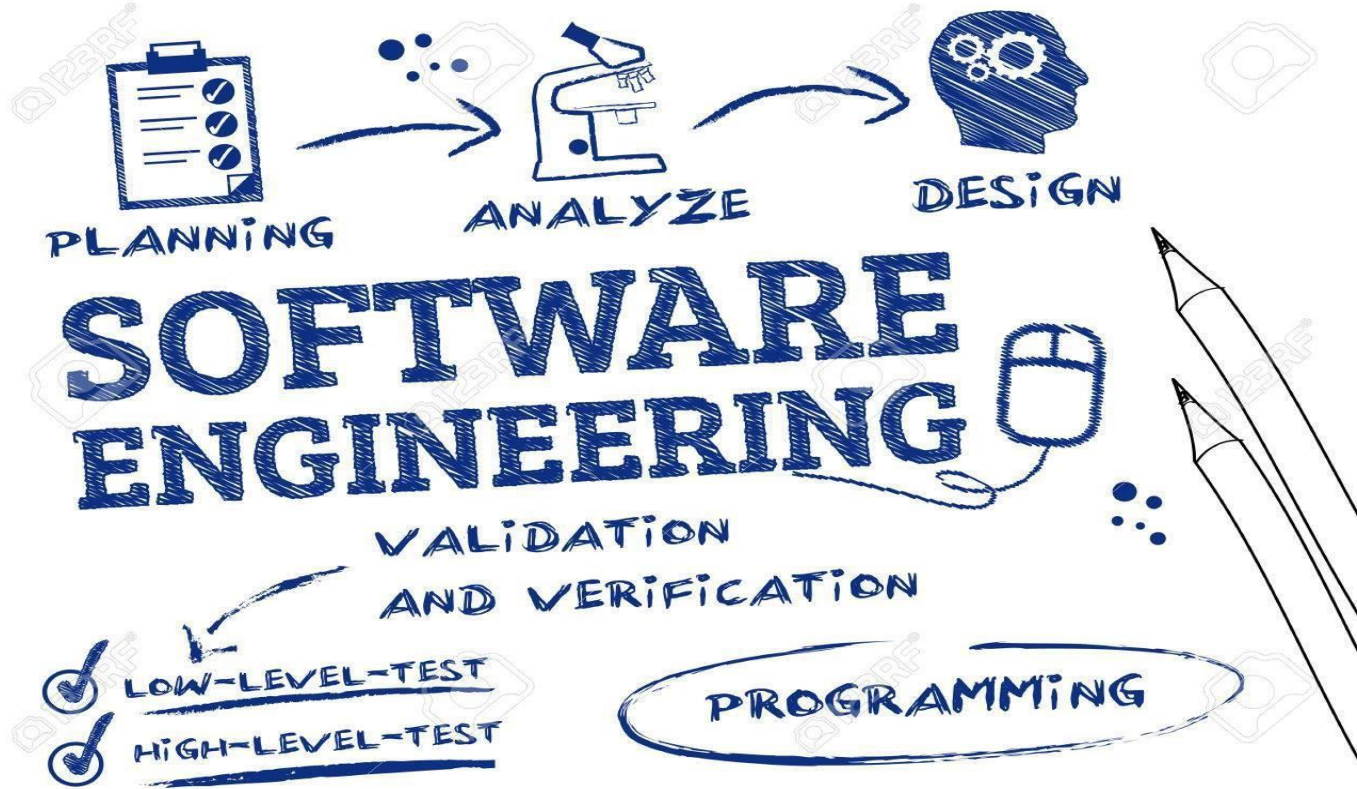
# Course Contents

- Introduction to Software Engineering
- Software Processes
- Requirements Engineering
- Software Design
- Software Testing and Verification
- Software Project Management

# Introduction to Software Engineering



# What is Software Engineering?



# What is Software Engineering?

- The term **software engineering** is the product of two words, **software**, and **engineering**.
- The **software** is a collection of integrated programs.
- Software subsists of carefully-organized instructions and code written by developers on any of various particular computer languages.
- Computer programs and related documentation such as **requirements, design models and user manuals**.
- **Engineering** is the application of **scientific** and **practical** knowledge to **invent, design, build, maintain, and improve frameworks, processes, etc.**

# What is Software Engineering?

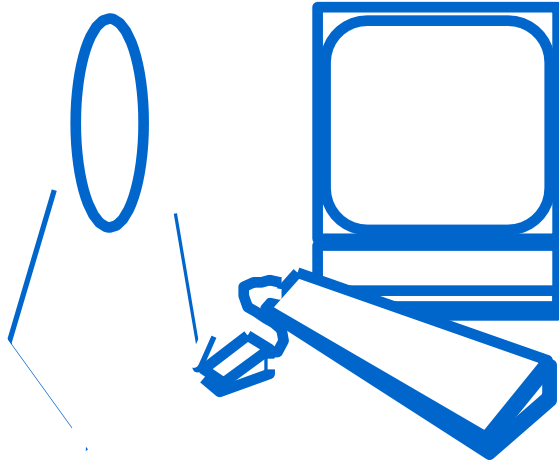
- **Software Engineering** is an engineering branch related to the evolution of software product using systematic well-defined scientific **principles**, **techniques**, and **procedures**.
- The result of software engineering is an effective and reliable software product.
- What is delivered to the customer is called the **product**.
- **Software products** may be developed for a **particular customer** or may be developed for a **general market**.
- It may include **source-code specification documents**, **manuals**, **documentation**, etc.



# What is Software?

Software is a set of items or objects that form a “configuration” that includes

- programs
- documents
- data ...



# What is Software?

Or you may want to say:

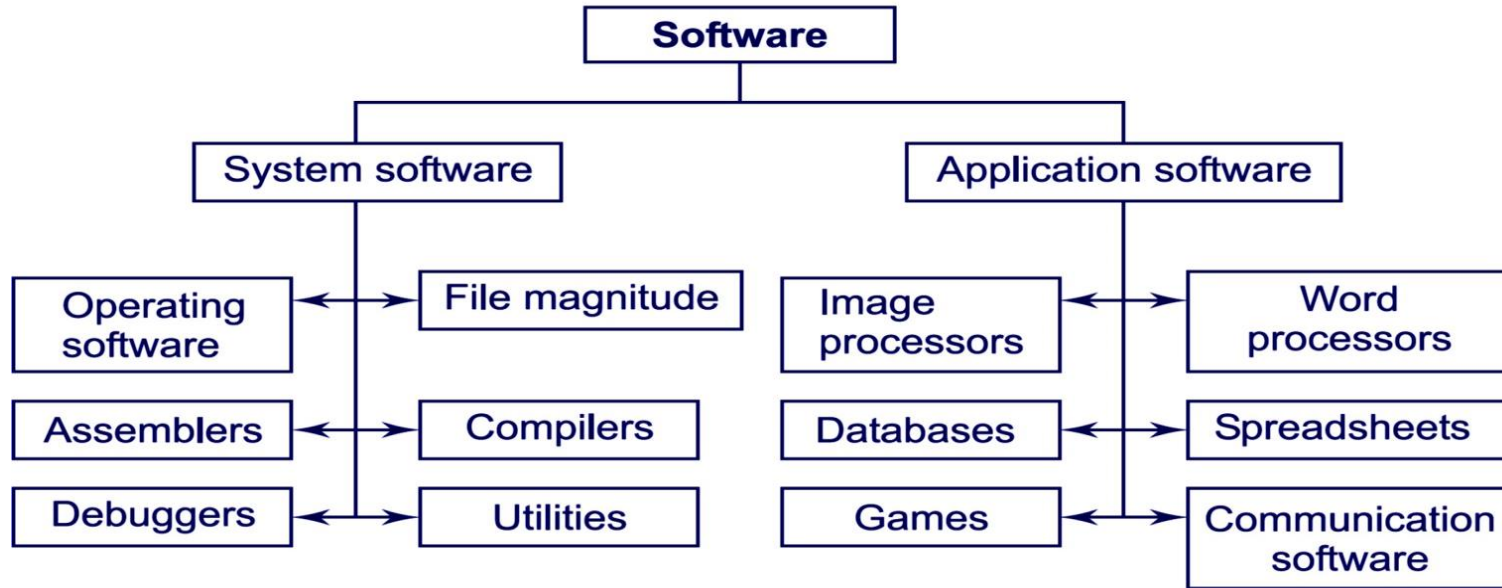
- ❑ Software consists of
  - (1) instructions (computer programs) that when executed provided desired function and performance,
  - (2) data structures that enable the programs to adequately manipulate information, and  
بشكل صحيح
  - (3) documents that describe the operation and use of the programs.

# What is Software?

But these are only the concrete part of software that may be seen, there exists also **invisible part** which is more important:

- ❑ Software is the **dynamic behavior** of programs on real computers and **auxiliary equipment**.  
المعدات المساعدة
- ❑ "... a software product is a **model** of the real world, and the real world is **constantly changing**."
- ❑ Software is a **digital form of knowledge**.

# Categories of Software



- **System Software:** as the Operating Systems, Compilers, Assemblers, Debuggers, and Utilities.
- **Application Software:** Programs that do real work for users, as word processors, spreadsheets, and database management systems.

# Software Examples


- System software** - compilers, editors.
- Real time software**, software that monitor/control real world event - air traffic control, navigation
- Embedded software** - keypad of microwave oven, mp3 players, cell phones
- Personal computer software** – operating system and other application software
- Web based software** – Google calendar, web-based systems for different organizations



# Importance of Software

- It is the **engine** that drives business decision-making
- It serves as the basis for modern scientific investigation and engineering problem-solving
- It is embedded in all kinds of systems, such as transportation, medical, telecommunications, military, industrial processes, entertainment, office products, etc.

# Unique Characteristics of Software

- Software is **malleable** (flexible)
- Software construction is **human-intensive**
- Software is **intangible** and **hard to measure**
- Software problems are usually **complex** غير ملموس
- Software directly depends upon the **hardware**
  - It is at the top of the system engineering “food chain” يتدهور او يتحلل 
- Software **doesn't wear out** but will **deteriorate**
- Software solutions require **unusual rigor** حل مش سهل

# Casting the Term

- The field of software engineering was born in NATO Conferences, 1968 in response to chronic failures of large software projects to meet schedule and budget constraints
- Since then, term became popular because software is getting more and more important to industry and business but the “software crisis” still persists.

# What is Software Engineering?

- Different focuses for this term exist in various textbooks. Some are listed below.
- The application of a systematic, **disciplined**, quantifiable approach to development, operation, and **maintenance** of software; that is, the application of engineering to software. (**IEEE** *Standard Computer Dictionary*, 610.12, ISBN 1-55937-079-3, 1990)

# What is Software Engineering? (ctd)

- Software engineering is concerned with the theories, methods and tools for developing, managing and evolving software products. (I. Sommerville, 6ed.)
- A discipline whose aim is the production of quality software, delivered on time, within budget, and satisfying users' needs. (Stephen R. Schach, Software Engineering, 2ed.)
- Multi-person construction of multi-version software (Parnas, 1987)

# What is Software Engineering? (ctd.)

- The practical application of scientific knowledge in the design and construction of computer programs and the associated documentation required to develop, operate and maintain them (B.W. Boehm)
- The establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines (F.L. Bauer)

# What is Software Engineering? (ctd.)

- The technological and managerial discipline concerned with systematic production and maintenance of software products that are developed and modified on time and within cost constraints (R. Fairley)
- A discipline that deals with the building of software systems which are so large that they are built by a team or teams of engineers (Ghezzi, Jazayeri, Mandrioli)

# Other Definitions of Software Engineering

- “A systematic approach to the analysis, design, implementation and maintenance of software.” (*The Free On-Line Dictionary of Computing*)
- “The systematic application of tools and techniques in the development of computer-based applications.” (Sue Conger in *The New Software Engineering*)
- “Software Engineering is about designing and developing high-quality software.” (Shari Lawrence Pfleeger in *Software Engineering -- The Production of Quality Software*)



# So, Software Engineering is ...

## ■ Scope

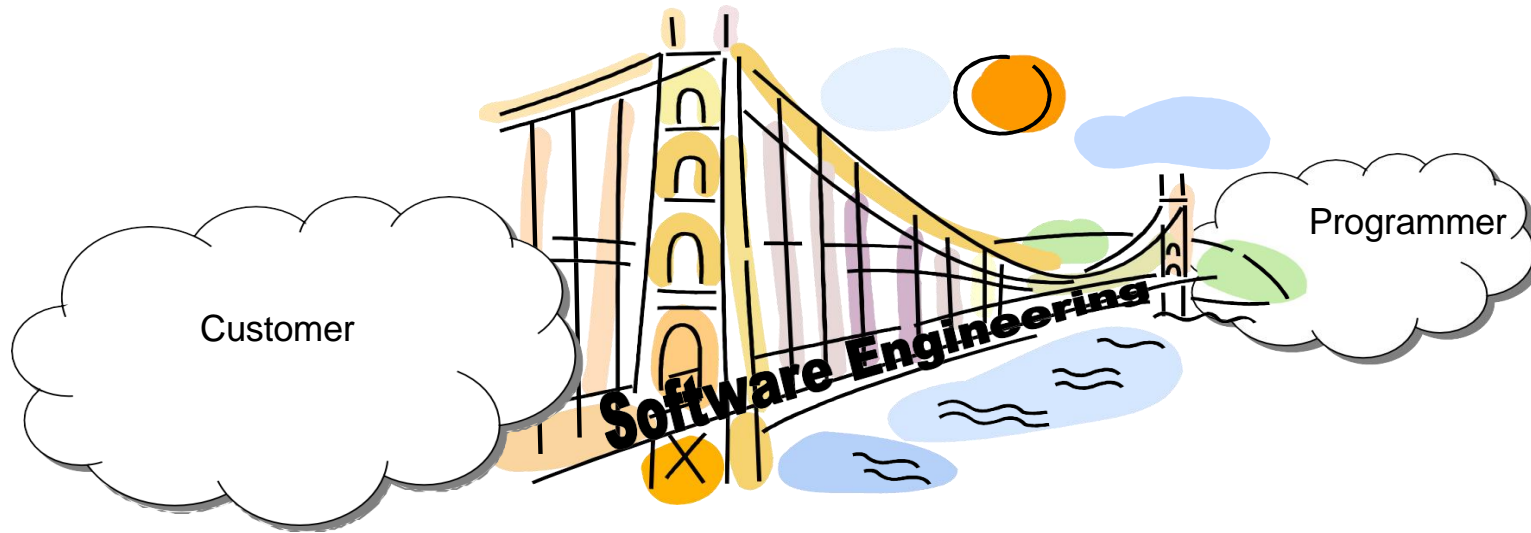
- ❑ study of software process, development principles, techniques, and notations

## ■ Goals

- ❑ production of quality software,
- ❑ delivered on time,
- ❑ within budget,
- ❑ satisfying customers' requirements and users' needs

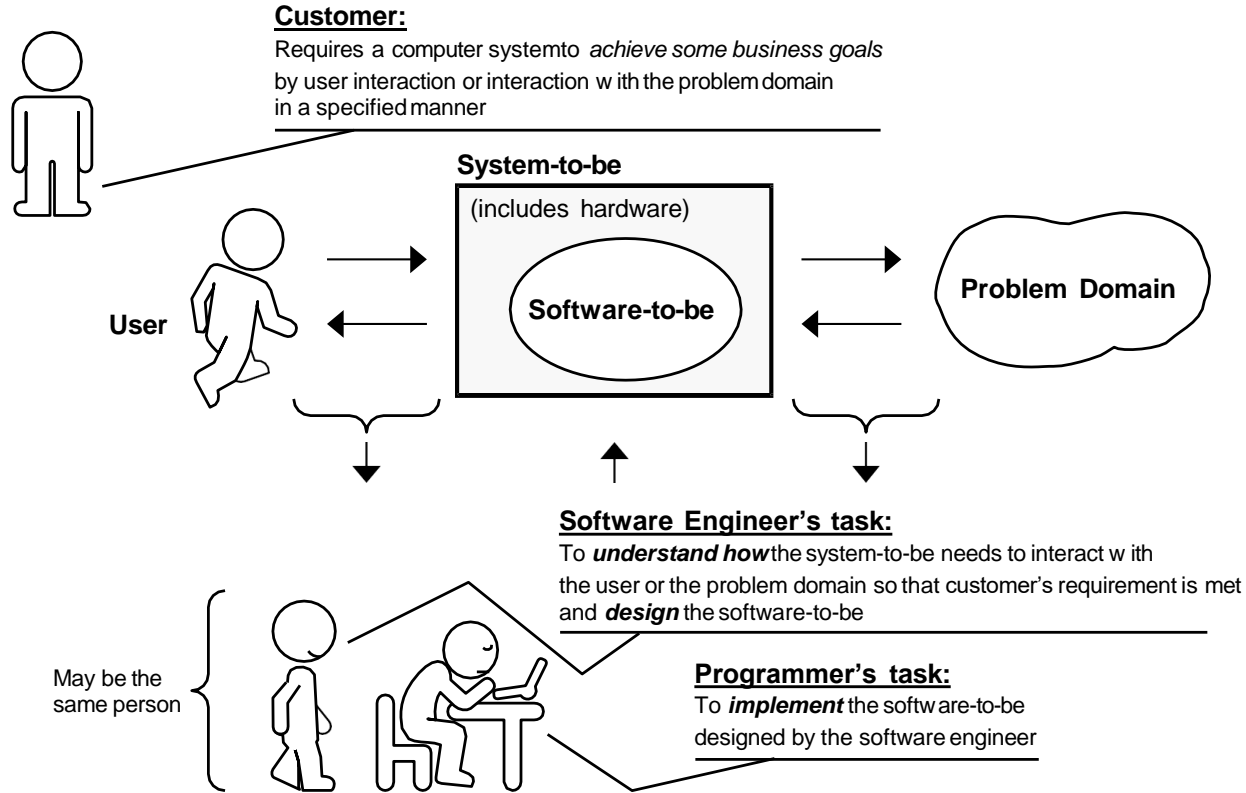
# The Role of Software Engineering

A bridge from customer needs to programming implementation



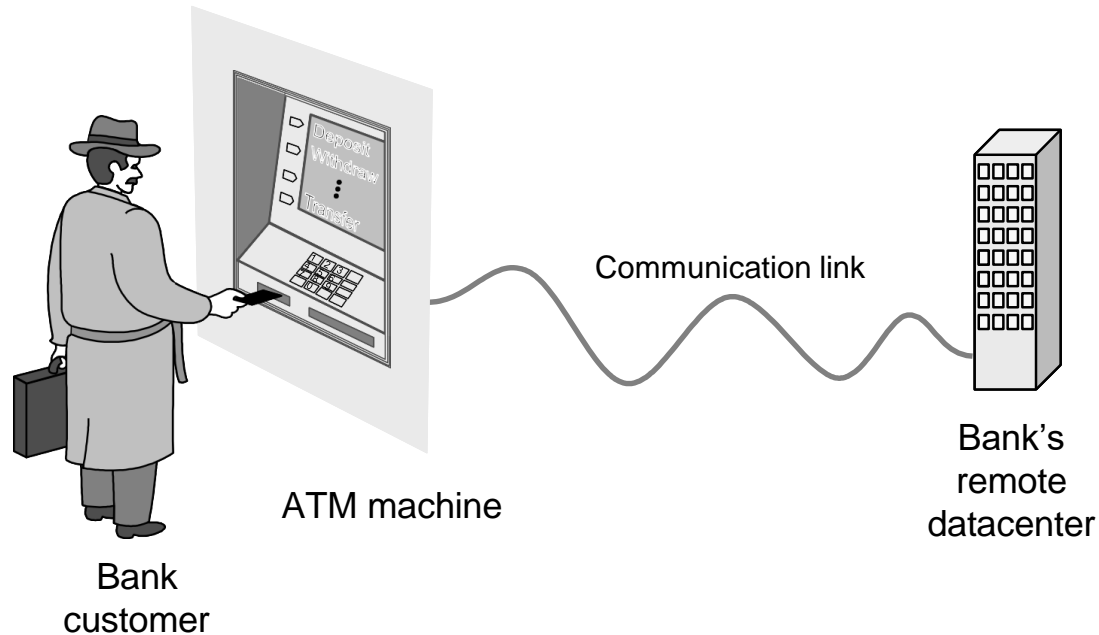
Software engineer is willing to learn the problem domain  
(problem cannot be solved without understanding it first)

# The Role of Software Engineering



# Example: ATM Machine

Understanding the money-machine problem:

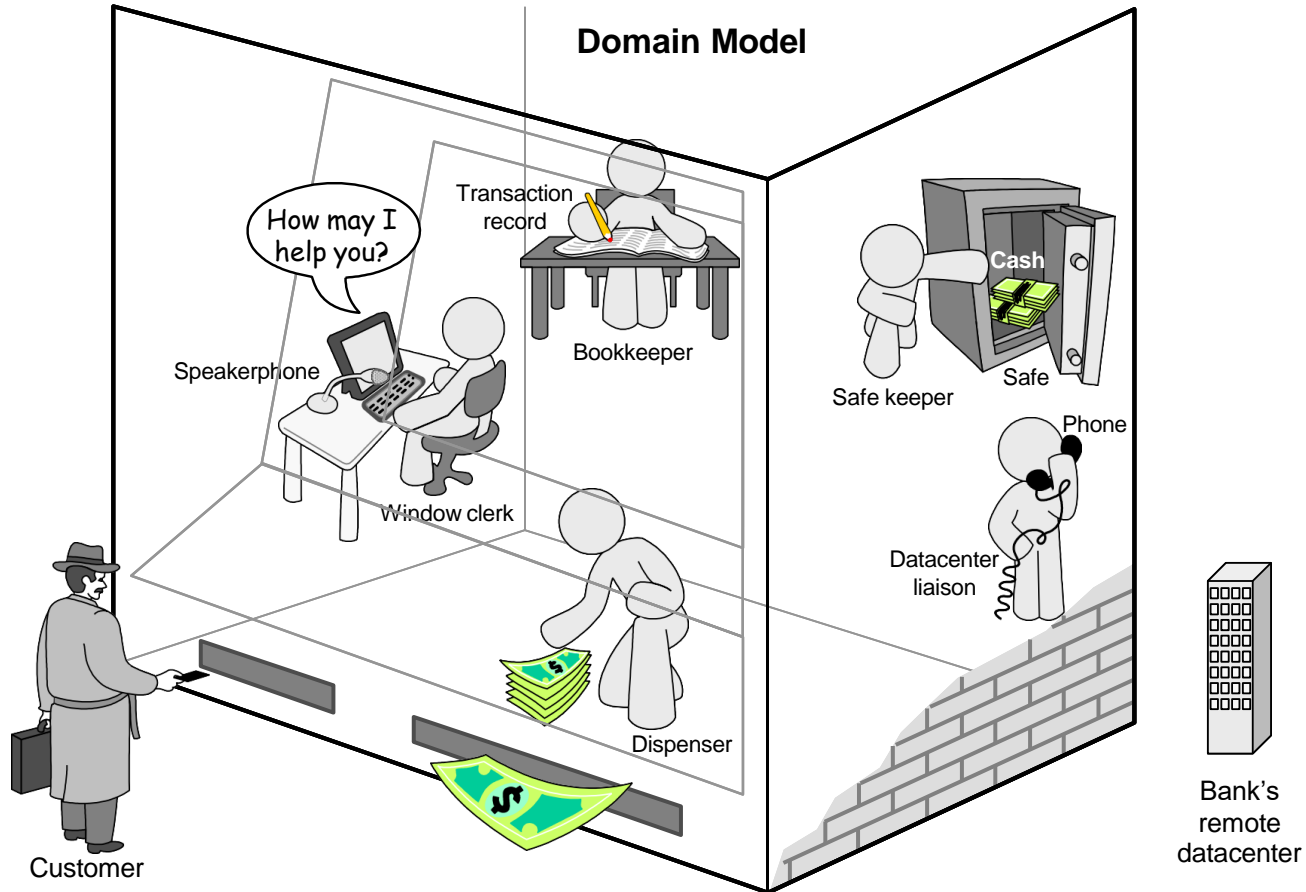


# Problem-solving Strategy

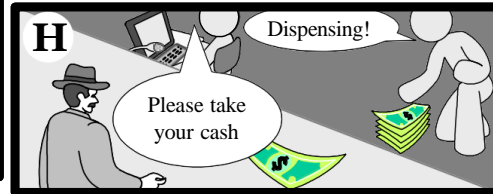
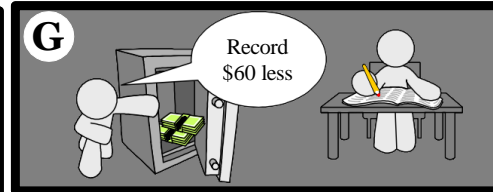
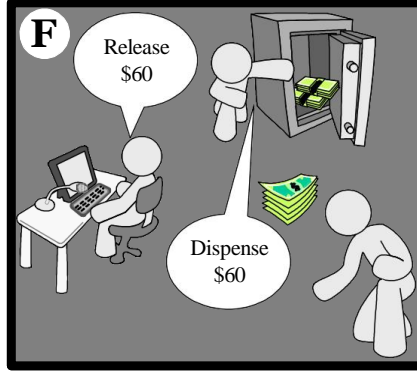
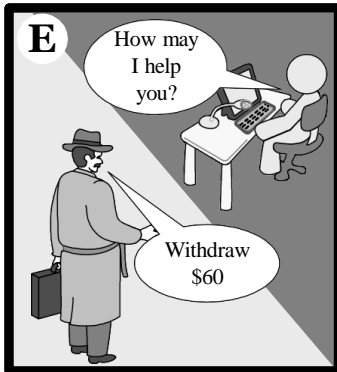
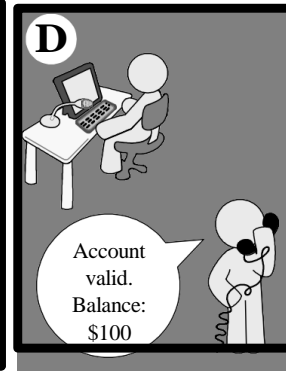
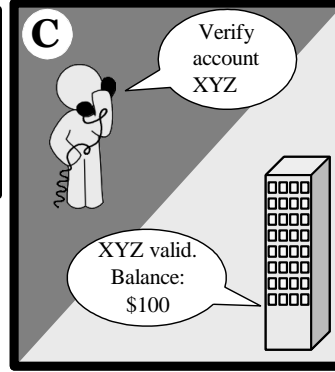
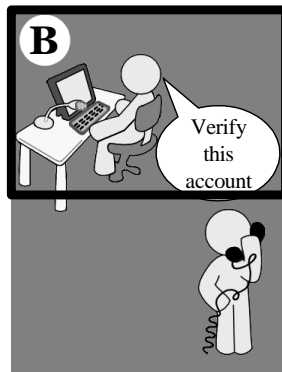
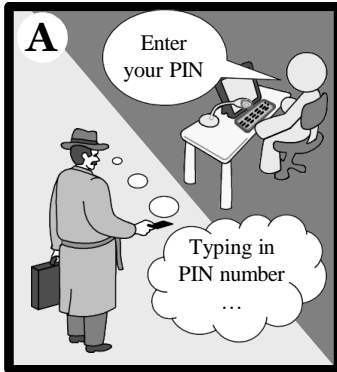
Divide-and-conquer:

- Identify logical parts of the system that each solves a part of the problem
- Easiest done with the help of a domain expert who already knows the steps in the process (“how it is currently done”)
- Result:  
A Model of the Problem Domain (or “domain model”)

# How ATM Machine Might Work



# Cartoon Strip: How ATM Machine Works



# Software Engineering Principles

- Software-engineering principles deal with both the process of software engineering and the final product.
- The right process will help produce the right product, but the desired product will also affect the choice of which process to use.
- Both are important.



# Software process activities

A software process is the related set of activities whose goal is the development or evolution of software.

*Activities:*

- **Software specification**, where customers and engineers define the software that is to be produced and the constraints on its operation.
- **Software development**, where the software is designed and programmed.
- **Software validation**, where the software is checked to ensure that it is what the customer requires.
- **Software evolution**, where the software is modified to reflect changing customer and market requirements.

# Confused with Programs and Products

Programs	Software Products
■ Usually small in size	■ Large
■ Author himself is sole user	■ Large number of users
■ Single developer	■ Team of developers
■ Lacks proper user interface	■ Well-designed interface
■ Lacks proper documentation	■ Well documented & user-manual prepared
■ Ad hoc development.	■ Systematic development

# What are the attributes of good software?

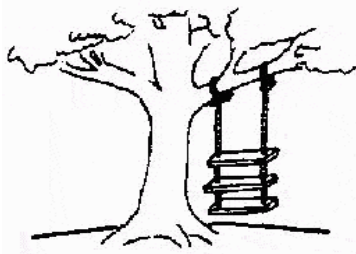
- Good software should deliver the required functionality and performance to the user and should be maintainable, dependable, efficient and usable

Examples - A banking system must be secure  
- Iterative game software –responsive  
- A telecommunication software -reliable

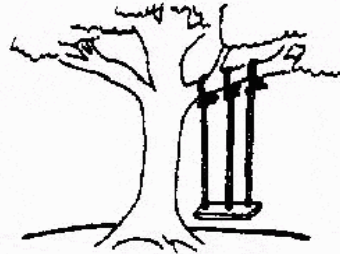
# What are the attributes of good software?

- Maintainability
  - Software must evolve to meet changing needs of customers.
- Dependability - has a range of characteristics reliability, security, and safety. Dependable software should not cause physical or economic damage in the event of system failure.
- Efficiency
  - Software should NOT waste system resources such as memory and processor time.
- Usability
  - Software must be usable and acceptable by the users for which it was designed, it should have an appropriate user interface and enough documentation.

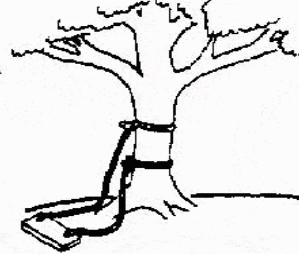
# How is Software usually Constructed ...



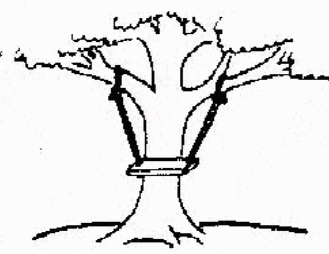
The requirements specification was defined like this



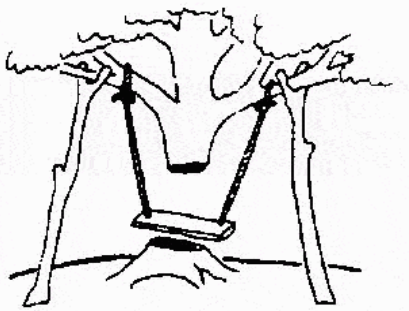
The developers understood it in that way



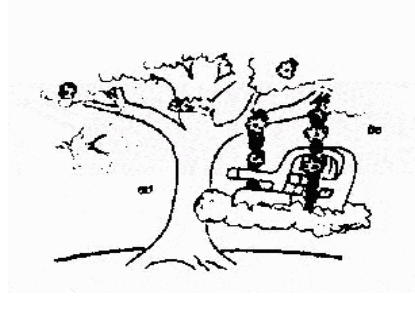
This is how the problem was solved before.



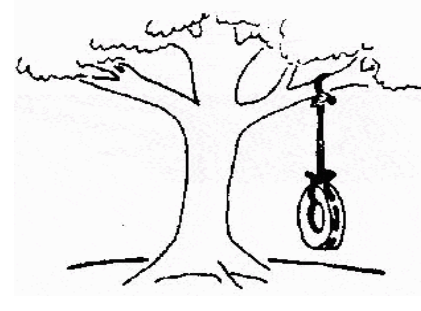
This is how the problem is solved now



That is the program after debugging



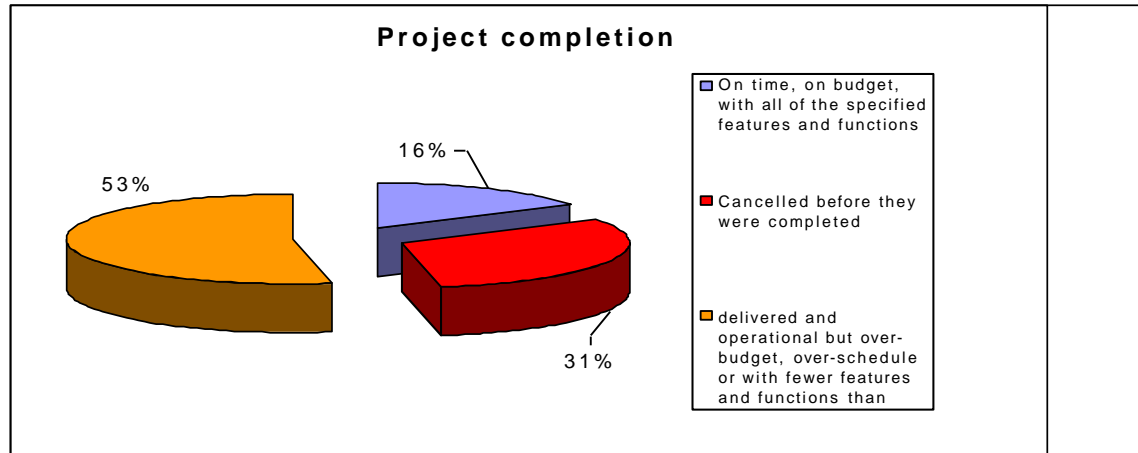
This is how the program is described by marketing dept.



This, in fact, is what the customer wanted ...<sup>35</sup>;-)

# Symptom of Software Crisis

- About **US\$250 billions** spent per year in the US on application development
- Out of this, about **US\$140 billions** wasted due to the projects getting abandoned or reworked; this in turn because of not following best practices and standards



# Symptom of Software Crisis

- 10% of client/server apps are abandoned or restarted from scratch
- 20% of apps are significantly altered to avoid disaster
- 40% of apps are delivered significantly late

**Source: 3 year study of 70 large c/s apps 30 European firms.  
Compuware (12/95)**

# Observed Problems

- Software products:
  - ❑ fail to meet user requirements
  - ❑ crash frequently
  - ❑ expensive
  - ❑ difficult to alter, debug, enhance
  - ❑ often delivered late
  - ❑ use resources non-optimally



# Software Crisis



# Causes of Software Crisis

- The cost of owning and maintaining software was as expensive as developing the software
- Projects was running over-time
- Software was very inefficient
- Software was low quality
- Software often did not meet requirements
- The average software project overshoots its schedule by half
- Software was never delivered

# Solution of Software Crisis

- Reduction in software over-budget
- The quality of software must be high
- Less time needed for software project
- Experience working team member on software project
- Software must be delivered

# Software Myths (Customer Perspectives)

- A general statement of objectives is sufficient to get started with the development of software. Missing/vague requirements can easily be incorporated/detailed out as they get concretized. تجسیدھا
- Application requirements can never be stable; software can be and has to be made flexible enough to allow changes to be incorporated as they happen.

should be stable if i can make it  
stable

# Software Myths (Developer Perspectives)

Once the software is demonstrated, the job is done.

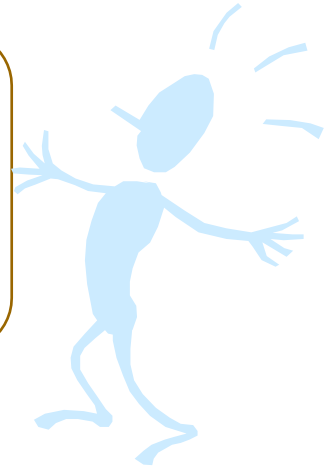
*Usually, the problems just begin!*



# Software Myths (Developer Perspectives)

Until the software is coded and is available for testing, there is no way for assessing its quality.

*Usually, there are too many tiny bugs inserted at every stage that grow in size and complexity as they progress thru further stages!*



# Software Myths (Developer Perspectives)

The only deliverable for a software development project is the tested code.

*The code is only  
the externally visible component  
of the entire software complement!*



# Software Myths (Management Perspectives)

As long as there are good standards and clear procedures in my company, I shouldn't be too concerned.

*But the proof of the pudding  
is in the eating;  
not in the Recipe !*

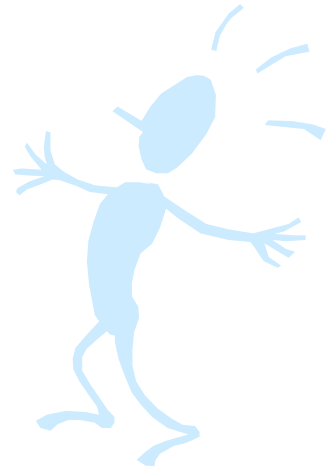




# Software Myths (Management Perspectives)

As long as my software engineers(!) have access to the fastest and the most sophisticated computer environments and state-of-the-art software tools, I shouldn't be too concerned.

*The environment is  
only one of the several factors  
that determine the quality  
of the end software product!*



# Software Myths (Management Perspectives)

When my schedule slips, what I have to do is to start a fire-fighting operation: add more software specialists, those with higher skills and longer experience - they will bring the schedule back on the rails!

*Unfortunately,  
software business does not  
entertain schedule compaction  
beyond a limit!*



# Misplaced Assumptions

- All requirements can be pre-specified
- Users are experts at specification of their needs
- Users and developers are both good at visualization
- The project team is capable of unambiguous communication

# Software Programming ≠ Software

- **Software programming:** the process of translating a problem from its physical environment into a language that a computer can understand and obey. (*Webster's New World Dictionary of Computer Terms*)
  - ❑ Single developer
  - ❑ "Toy" applications
  - ❑ Short lifespan
  - ❑ Single or few stakeholders
    - Architect = Developer = Manager = Tester = Customer = User
  - ❑ One-of-a-kind systems
  - ❑ Built from scratch
  - ❑ Minimal maintenance

# Software Programming ≠ Software Engineering

- Software engineering
  - Teams of developers with multiple roles
  - Complex systems
  - Indefinite lifespan عمر غير محدود
  - Numerous stakeholders
    - Architect ≠ Developer ≠ Manager ≠ Tester ≠ Customer ≠ User
  - System families
  - Reuse to amortize costs
  - Maintenance accounts for over 60% of overall development costs

# Software Engineering vs. Computer Science

- **Computer science** is concerned with theory and fundamentals; This field involves the understanding and application of both abstract and concrete knowledge.
- **Software engineering** is a field largely concerned with the application of engineering processes to the creation, maintenance, and design of software for a variety of different purposes.

# Software Engineering vs. System Engineering?

- **System engineering** is concerned with all aspects of computer-based systems development including hardware, software and process engineering.
- System engineers are involved in system specification, architectural design, integration and deployment.
- Software engineering is part of this process concerned with developing the software infrastructure, control, applications and databases in the system.
- Systems engineering is older than Software Engineering:
  - Complex industrial systems such as trains and chemical plants.
- As the percentage of software systems has increased, software engineering techniques are finding their way into systems engineering.

# Why is Software Engineering required?

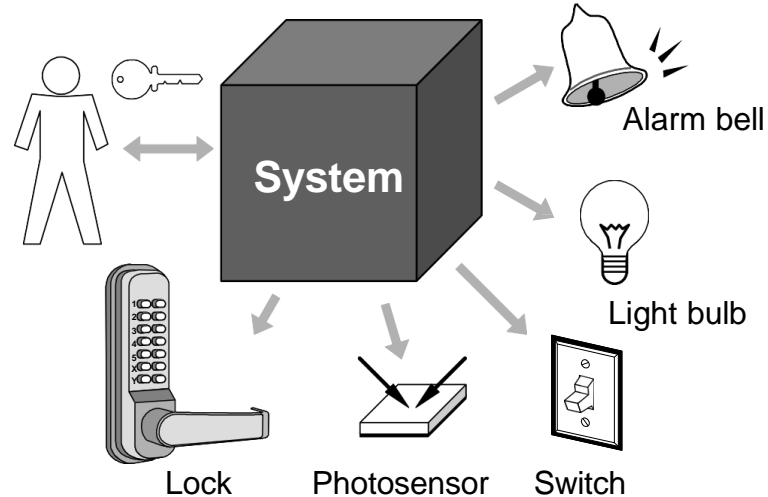
- Software Engineering is required due to the following reasons:
  - To manage Large software
  - For more Scalability
  - Cost Management
  - To manage the dynamic nature of software
  - For better quality Management



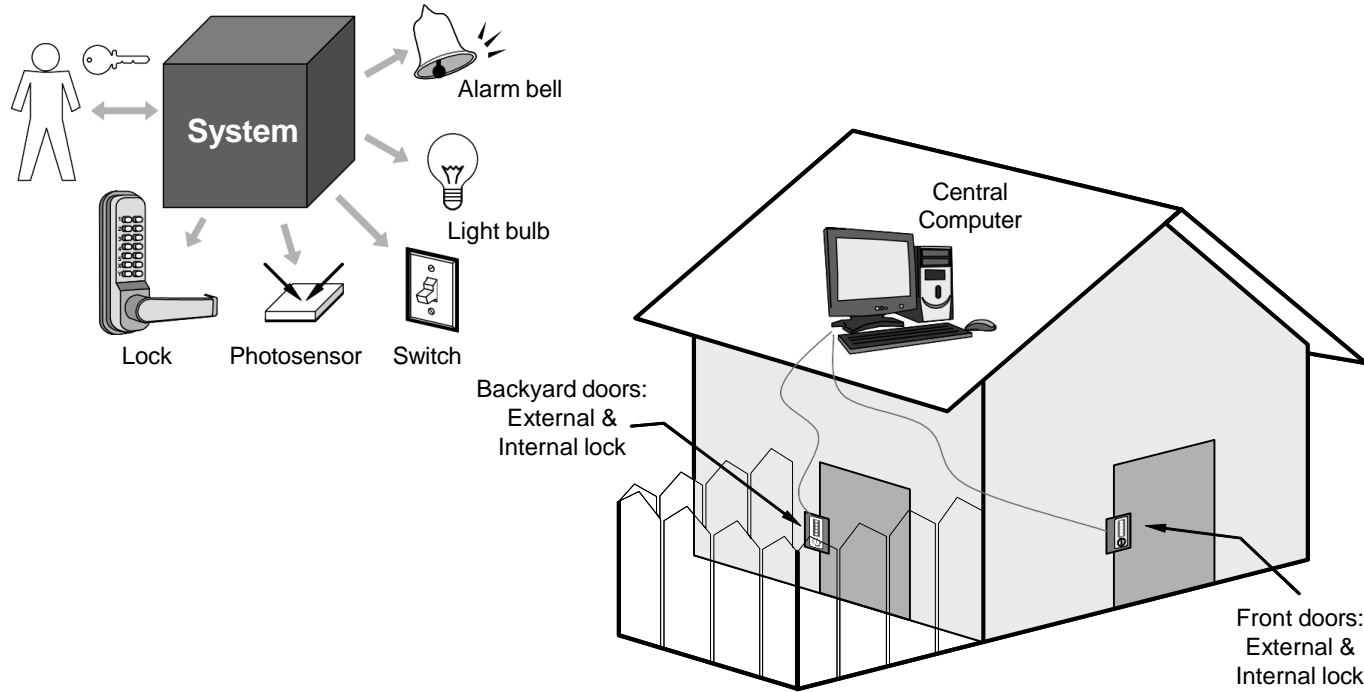
# Case Study: Home Access Control

## ■ Objective: Design an electronic system for:

- Home access control
  - Locks and lighting operation
- Intrusion detection and warning

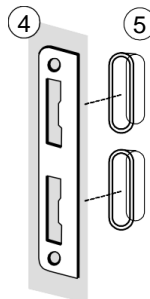
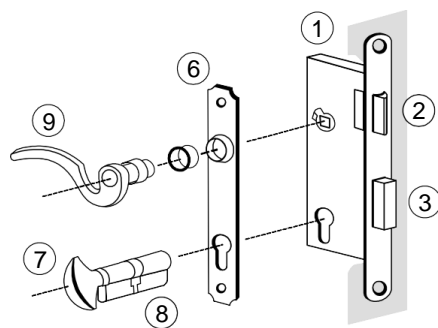


# Case Study – More Details

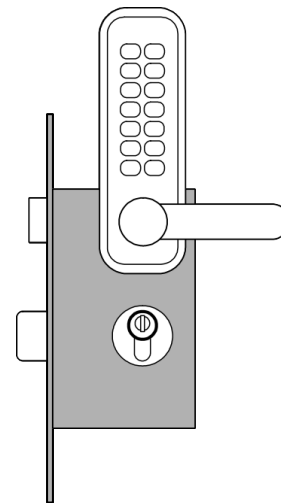
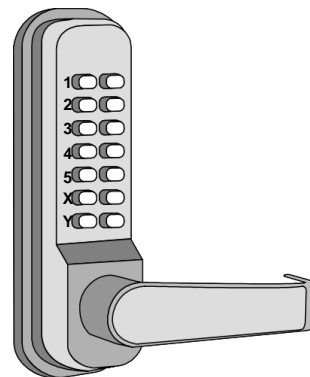


# Know Your Problem

## Mortise Lock Parts



- ① Lock case
- ② Latch bolt
- ③ Dead bolt
- ④ Strike plate
- ⑤ Strike box
- ⑥ Protective plate
- ⑦ Thumb-turn
- ⑧ Lock cylinder
- ⑨ Left hand lever




# Concept Maps

Useful tool for problem domain description.

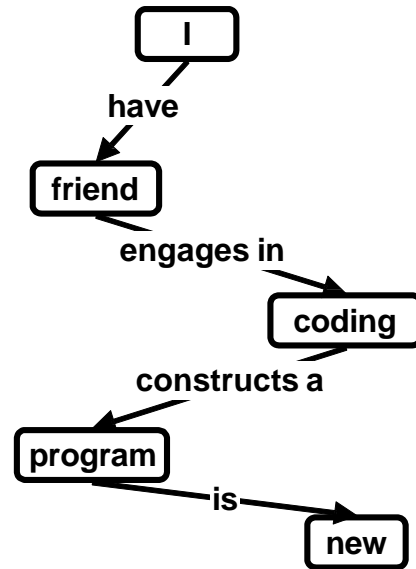
- a diagram that depicts suggested relationships between concepts

SENTENCE: “My friend is coding a new program”

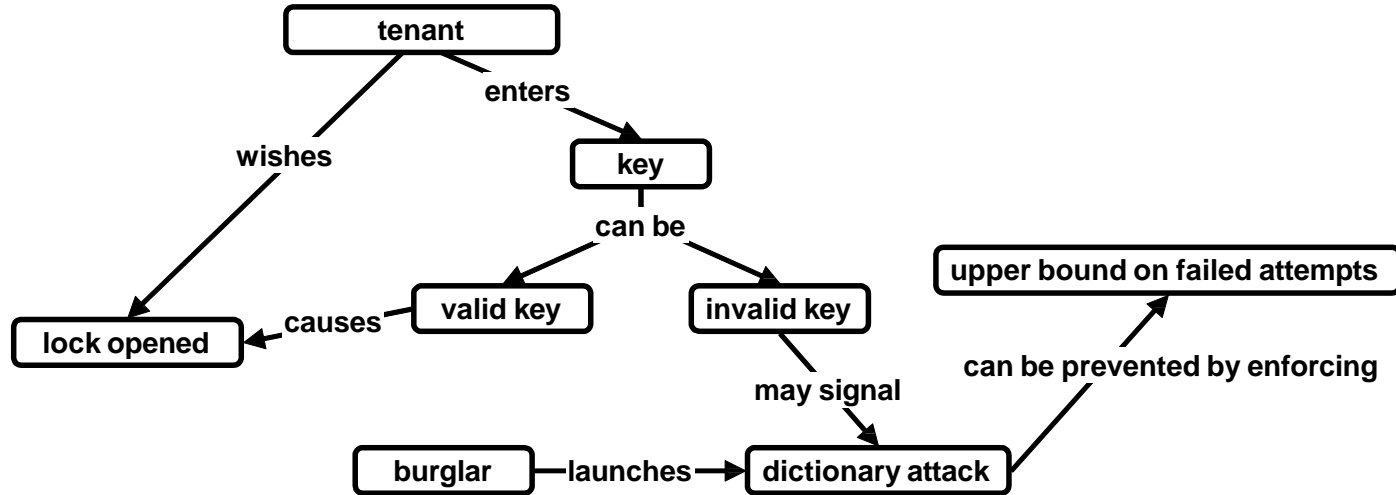
translated into propositions



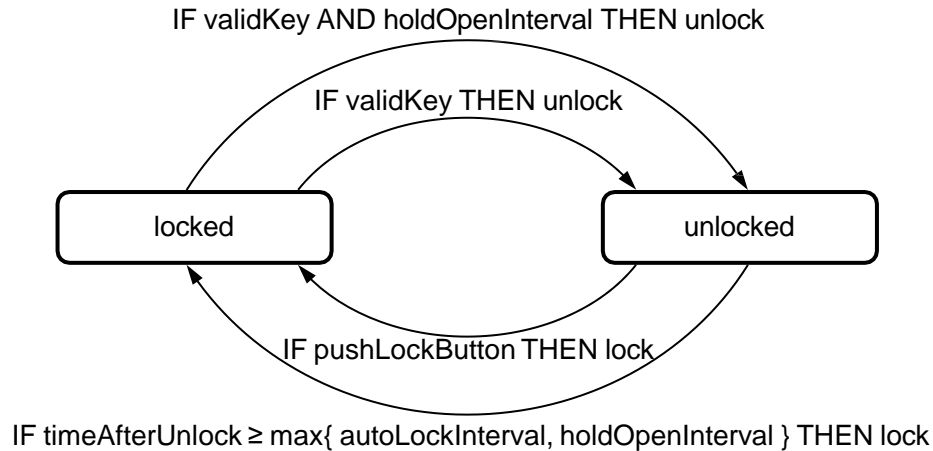
Proposition	Concept	Relation	Concept
1.	I	have	friend
2.	friend	engages in	coding
3.	coding	constructs a	program
4.	program	is	new



# Concept Map for Home Access Control

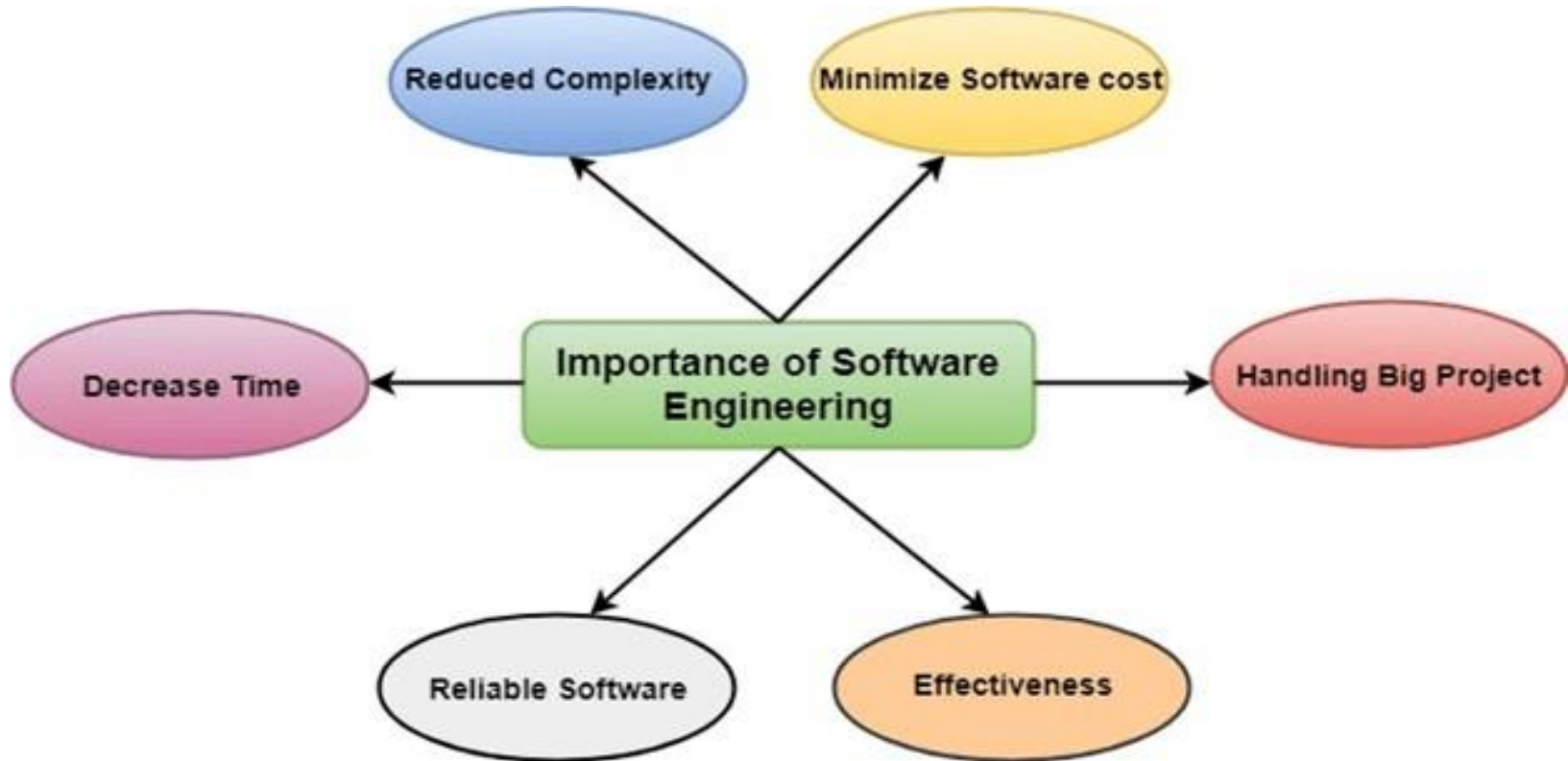


# States and Transition Rules



... what seemed a simple problem, now is becoming complex

# Importance of Software Engineering



# Key points

- **Software engineering** is an engineering discipline which is concerned with all aspects of software production.
- **Software** products consist of developed programs and associated documentation.
- Essential **product attributes** are maintainability, dependability, efficiency and usability.



# Key points

- The **software process** consists of activities which are involved in developing software products.
- Basic **activities** are software specification, development, validation and evolution.