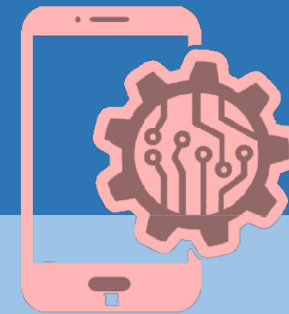


Lecture 1.1

# Introduction to Software Engineering



# Y2K bug (millennium bug)

- The Y2K bug was a computer Flaw or bug, that may have caused problems when **dealing with dates** beyond December 31, 1999.
- The flaw, faced by computer programmers and users all over the world on January 1, 2000, is also known as the **"millennium bug"**.
- Computer engineers used a two-digit code for the year. Instead of a date reading 1970, it read 70.



As the year 2000 approached, computer programmers realized that computers might not interpret 00 as 2000, but as 1900



# Ariane 5 - One bug, one crash

- ❖ It took the European Space Agency 10 years and \$7 billion to produce Ariane 5, a giant rocket capable of hurling a pair of three-ton satellites into orbit with each launch and intended to give Europe overwhelming supremacy in the commercial space business.
- ❖ Issue was a small computer program trying to stuff a 64-bit number into a 16-bit space.



A **16 bits** variable can have a value of  $-32.768$  to  $32.767$ .

On the other hand, a **64 bits** variable can have a value of  $-9.223.372.036.854.775.808$  to  $9.223.372.036.854.775.807$  (that's almost an infinity of options).

# “No Silver Bullet”

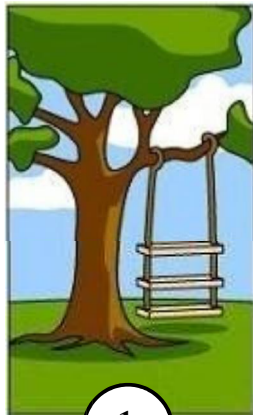
---

The blame for software bugs belongs to:

- Software companies
- Software developers
- Legal system
- Universities

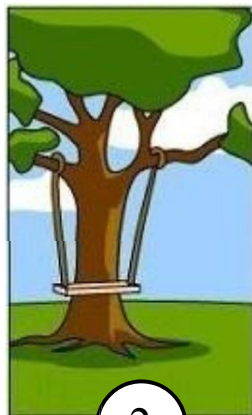
# Why to Study Software Engineering?

Software Development Life Cycle **without** Software Engineering



1

How the  
Customer  
Explains  
Requirement



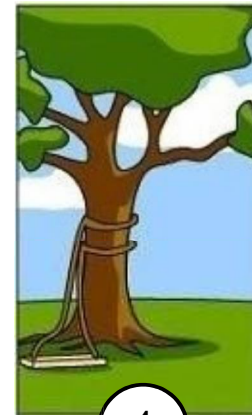
2

How the  
Project  
Leader  
understand it



3

How the  
System  
Analyst  
design it



4

How the  
Programmer  
Works  
on it

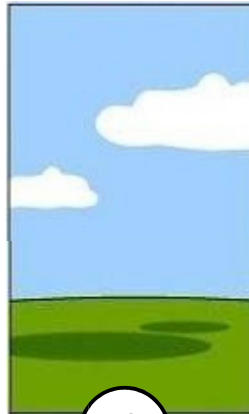
# Why to Study Software Engineering?

Software Development LifeCycle **without** Software Engineering



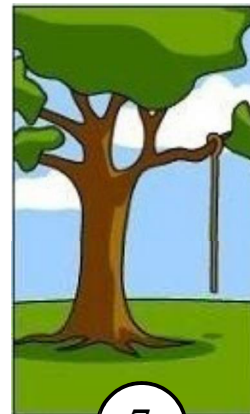
5

How the  
Business  
Consultant  
describe it



6

How the  
Project was  
documented



7

What  
Operations  
Installed

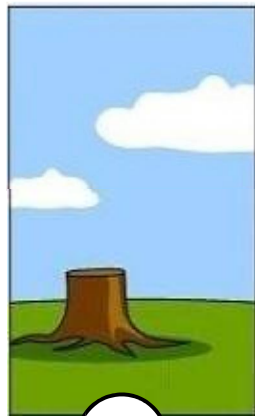


8

How the  
Customer  
was  
billed

# Why to Study Software Engineering?

Software Development Life Cycle **without** Software Engineering



9

How it was  
supported





10

What the  
customer  
really needed

Software development  
**Process** needs to be  
**engineered** to avoid  
the **communication gape**  
& to **meet the actual**  
**requirement** of customer  
within stipulated budget  
& time



# SDLC **without** Software Engineering

Customer Requirement	Solution
<ul style="list-style-type: none"> <li>• Have one trunk</li> <li>• Have four legs</li> <li>• Should carry load both passenger &amp; cargo</li> <li>• Black in color</li> <li>• Should be herbivorous</li> </ul>	<ul style="list-style-type: none"> <li>• Have one trunk</li> <li>• Have four legs</li> <li>• Should carry load both passenger &amp; cargo</li> <li>• Black in color</li> <li>• Should be herbivorous</li> </ul>
	 <p>Our value added Also gives milk</p>



# Why Software Engineering?

---

- \_ Change in nature & complexity of software
- \_ Concept of one “guru” is over
- \_ We all want improvement

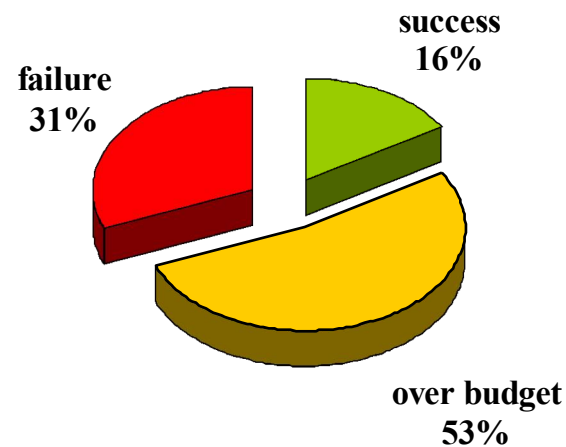


Ready for change

# The Evolving Role of Software

---

Software industry is in Crisis!

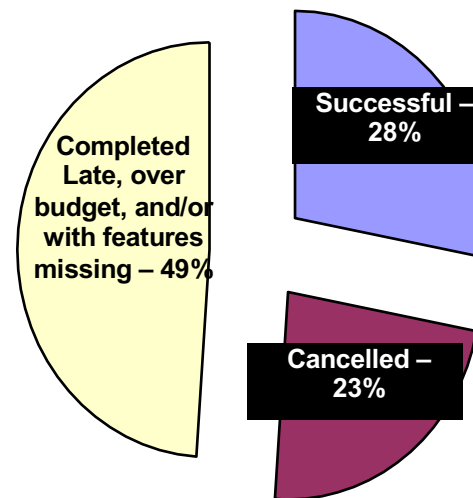


Source: The Standish Group International, Inc (CHAOS research)

# The Evolving Role of Software

---

This is the  
**SORRY** state  
of Software  
Engineering  
Today!






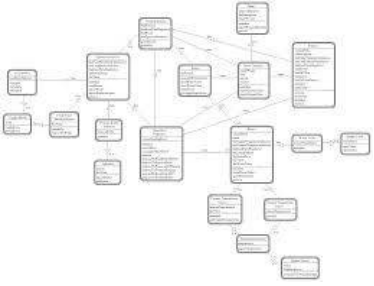


- Fewer than 20% of innovation focused projects are coming to life, according to a new report from Oracle, primarily caused by lack of focus and leadership, poor processes, and ongoing resistance to change.

# The Evolving Role of Software

---

As per the IBM report, “31% of the project get cancelled before they are completed, 53% over-run their cost estimates by an average of 189% and for every 100 projects, there are 94 restarts”

# Software Engineering

Engineering			
Software Engineering			
	Design	Build	Product

*Software is developed or engineered, it is not manufactured in the classical sense.*

# What is Software?

□ Software is

- ✓ instructions (computer programs) that when executed provide desired features, function, and performance;
- ✓ data structures that enable the programs to adequately manipulate information and
- ✓ documentation that describes the operation and use of the programs.



Computer  
Program

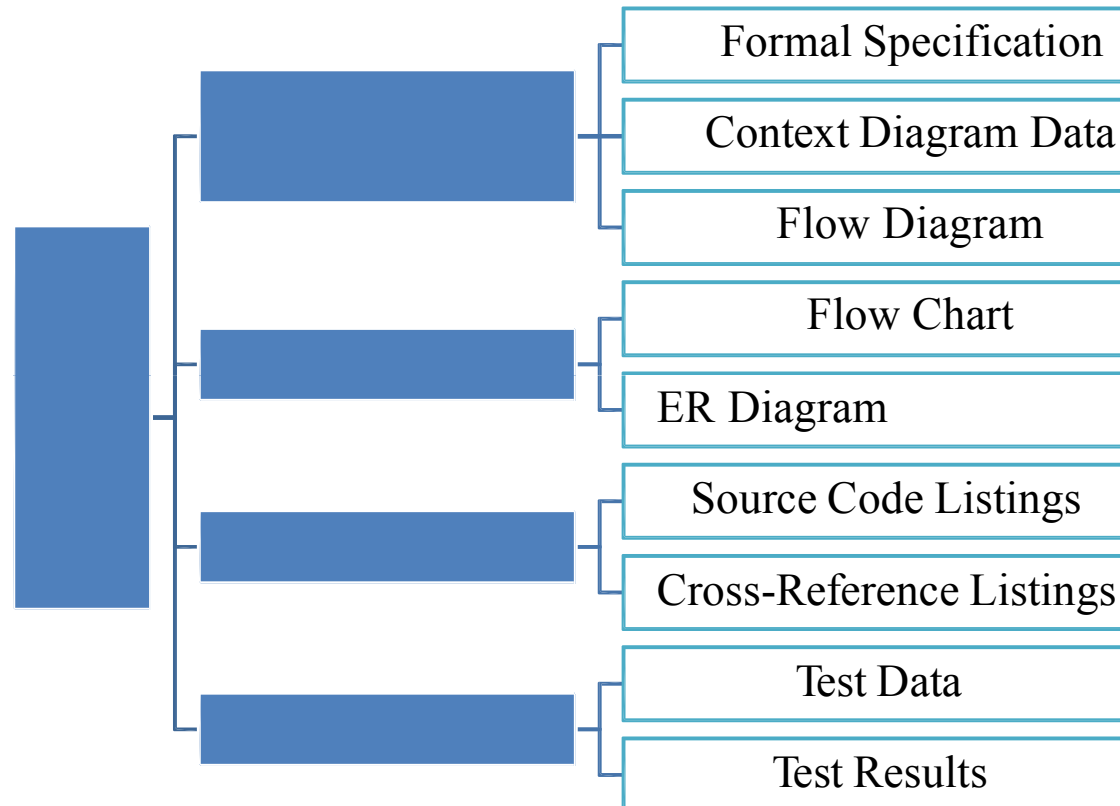


Data  
Structure



Documents  
Soft & Hard

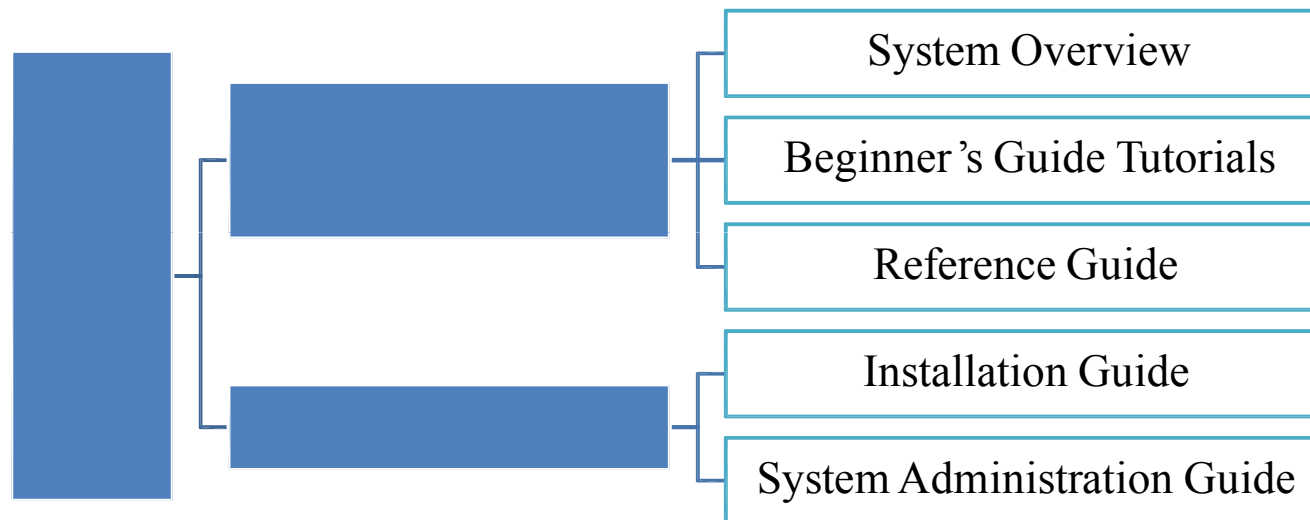
# List of Documentation Manuals





# List of Documentation Manuals

---

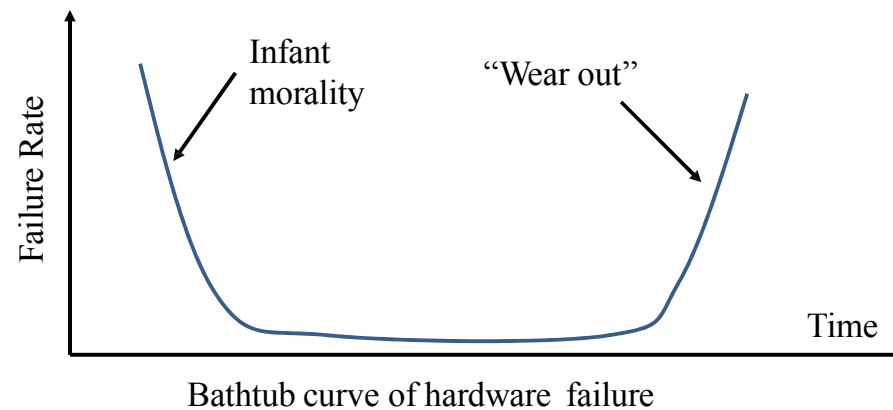


# Characteristics of Software

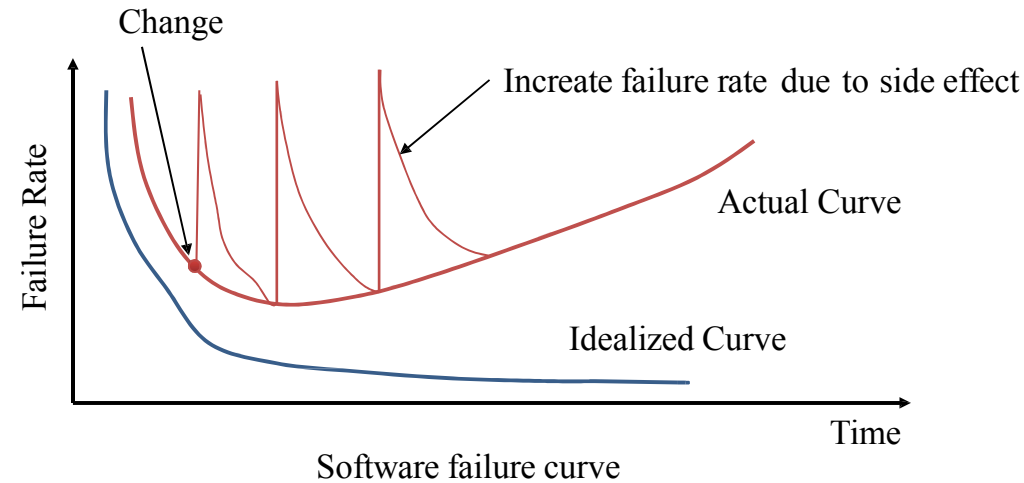
## □ Software is developed or engineered

- It is not manufactured like hardware
  - Manufacturing phase can introduce quality problem that are nonexistent (or easily corrected) for software
  - Both requires construction of “product” but approaches are different

## □ Software doesn't “wear-out”



# Characteristics of Software cont.



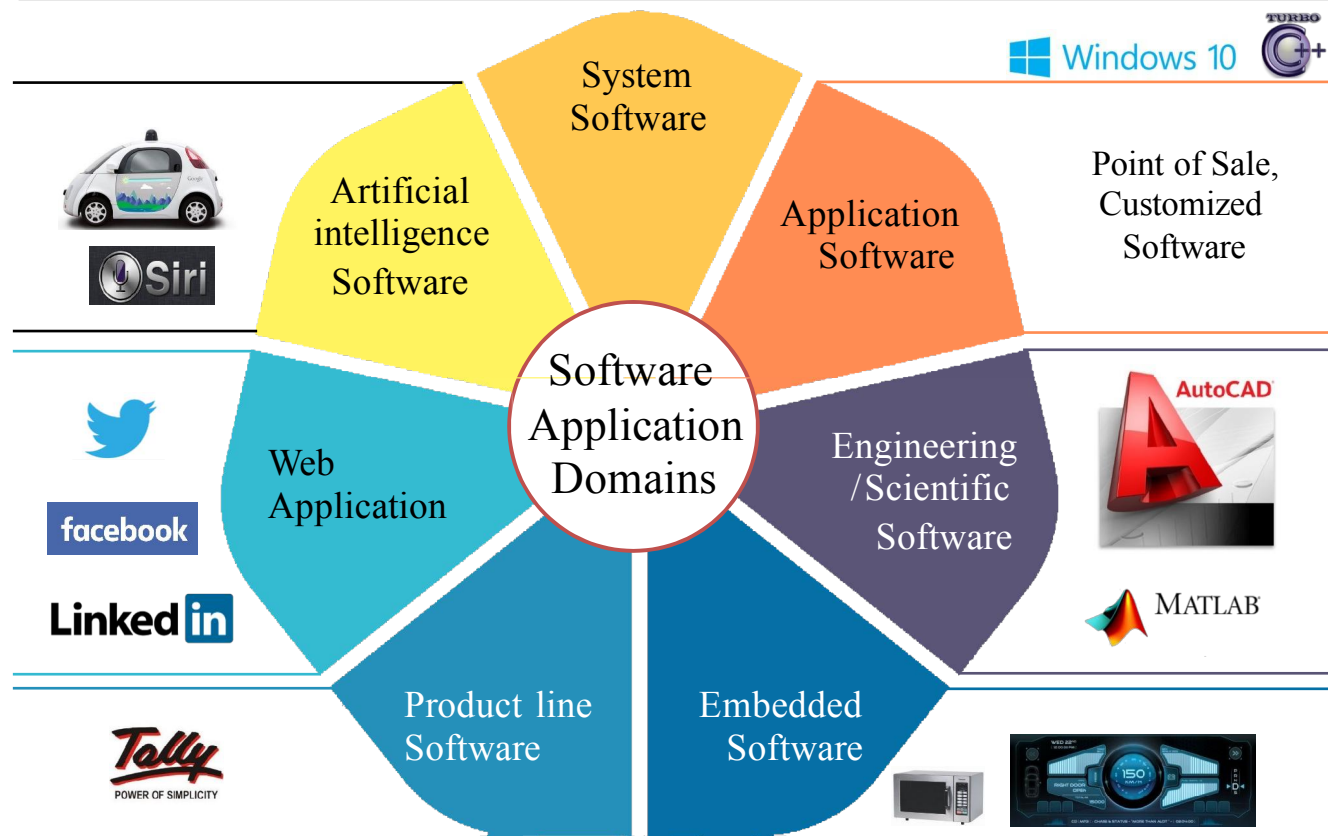
- Although the industry is moving toward component based construction, most software continues to be custom built

# Software Characteristics: An Example

## Comparison of constructing a bridge vis-à-vis writing a program

Sr No	Constructing a bridge	Writing a program
1	The problem is well understood	Only some parts of the problem are understood, others are not
2	There are many existing bridges	Every program is different and designed for special applications
3	The requirement for a bridge typically do not change much during construction	Requirements typically change during all phases of development
4	The strength and stability of a bridge can be calculated with reasonable precision	Not possible to calculate correctness of a program with existing methods
5	When a bridge collapses, there is a detailed investigation and report	When a program fails, the reasons are often unavailable or even deliberately concealed
6	Engineers have been constructing bridges for thousands of years	Developers have been writing programs for 50 years or so
7	Materials (wood, stone, iron, steel) and techniques (making joints in wood, carving stone, casting iron) change slowly	Hardware and software changes rapidly

# Software Application Domains



# Software Engineering

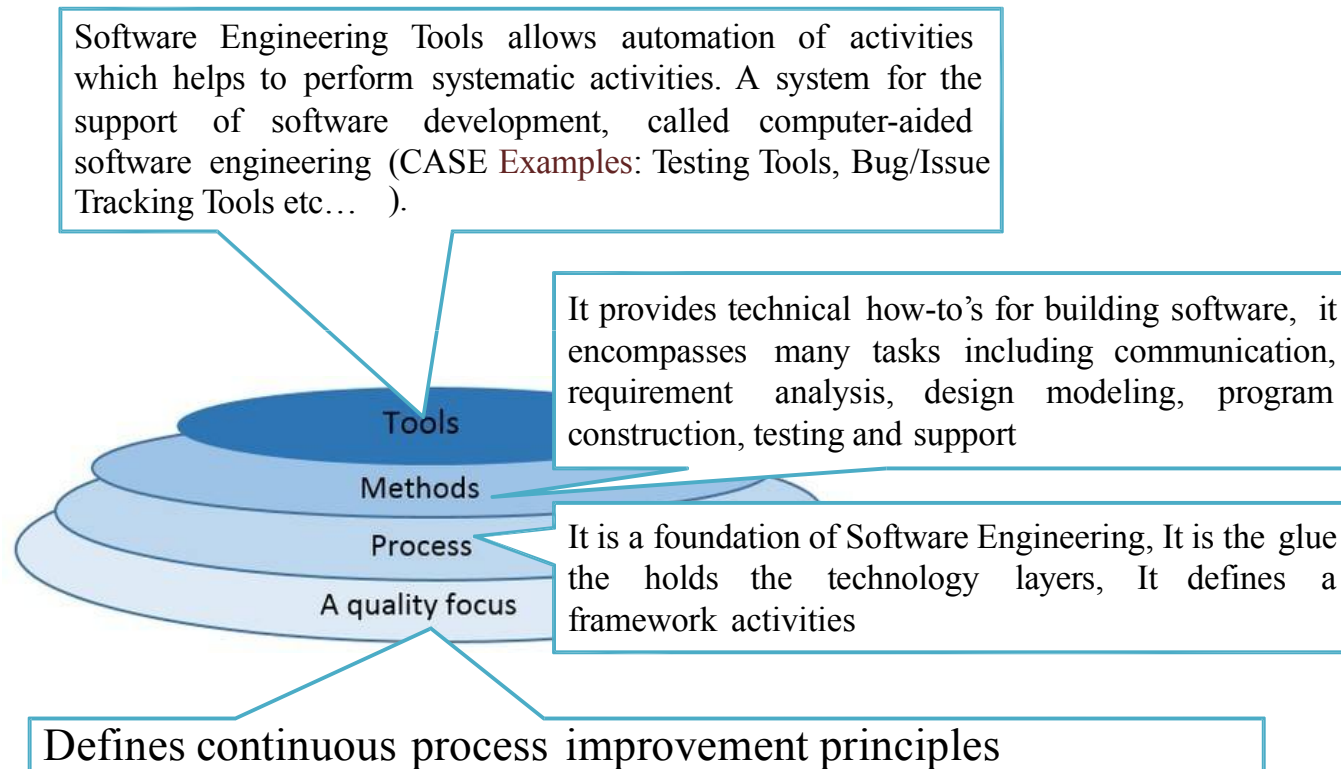
Software engineering is the establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently in real machines.

The IEEE definition: The application of a **systematic, disciplined, quantifiable** approach to the **development, operation, and maintenance of software**, that is the application of engineering to software.

Software Engineering is the science and art of building (designing and writing programs) a software systems that are:

- 1) on time
- 2) on budget
- 3) with acceptable Quality With
- 4) full scope

# Software Engineering Layered Approach





# Software Engineering Cont.

---

- Software Engineering is a layered technology
- **Quality**
  - Main principle of Software Engineering is Quality Focus.
  - An **engineering approach** must have a focus on quality.
  - Total Quality Management (TQM), Six Sigma, ISO 9001, ISO 9000-CAPABILITY MATURITY MODEL(CMM) CMM & similar approaches encourages a continuous process improvement culture
- **Process layer**
  - It is a foundation of Software Engineering
  - It is the glue the holds the technology layers
  - It defines a framework with activities for effective delivery of software engineering technology

# Software Engineering Cont.

---

## □ Method

- It provides **technical how-to's** for building software
- It encompasses many tasks including communication, requirement analysis, design modeling, program construction, testing and support

## □ Tools

- Computer-aided software engineering (**CASE**) the scientific application of a **set of tools** and **methods** to a software system which is meant to result in high-quality, defect-free, and maintainable software products.
- CASE tools automate many of the activities involved in various cycle phases. life

## Software Myths (Management Perspectives)

---

Management may be confident about good standards and clear procedures of the company

But the taste of any food item  
is in the eating;  
not in the Recipe !



## Software Myths (Management Perspectives)

---

Company has latest computers and state-of-the-art software tools, so we shouldn't worry about the quality of the product

The infrastructure is only one of the several factors that determine the quality of the product!



## Software Myths (Management Perspectives)

---

Addition of more software specialists, those with higher skills and longer experience may bring the schedule back on the track!

Unfortunately,  
that may further delay the schedule

!



## Software Myths (Management Perspectives)

---

Software is easy to change

The reality is totally different



## Software Myths (Developer Perspectives)

---

Once the software is demonstrated, the job is done

Usually, the problems just begin!





## Software Myths (Developer Perspectives)

---

Software quality can not be assessed before testing

However, quality assessment techniques should be used through out the software development life cycle

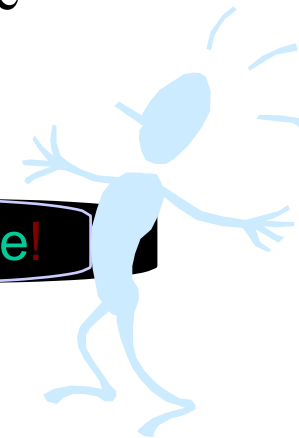


## Software Myths (Developer Perspectives)

---

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

Tested code is only one of the deliverable!



# End of Class

# Let's Talk

