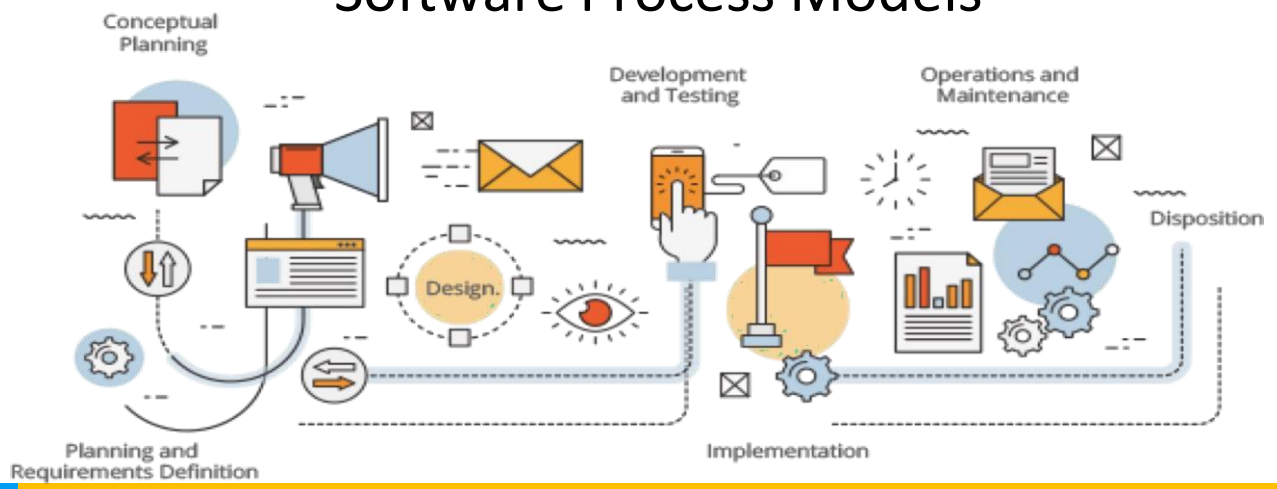
Software Engineering

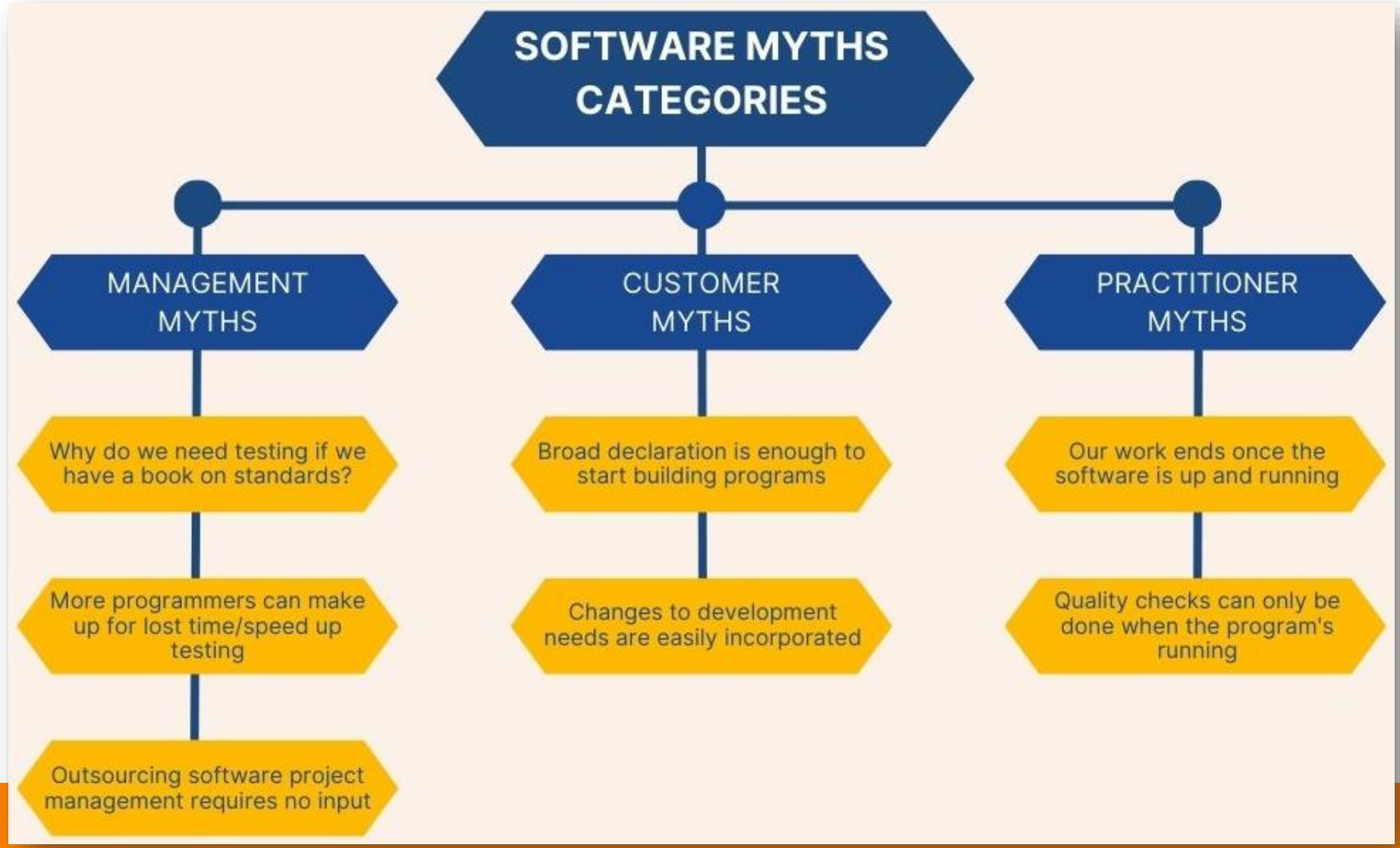
Software Process Models



Revision Lecture 01

1. What was the agenda of NATO conference?
2. What does software crises mean?
3. What is software engineering?
4. How to incorporate change in existing software system?
5. What is the difference between Software Engineering and Computer Science?
6. Why software engineering?

In software engineering, software myths are deceptive beliefs that can cause severe issues for both managers and programmers. Myths about software spread disinformation and misunderstanding.



Software Myths – Management Myth

Myth-1: Our people have a book that's full of standards, best practices and procedures for building software, won't that provide them with everything they need to know?

Reality: The book of standards may exist, but it is rarely consulted. The majority of software developers are unaware of its existence. It also isn't complete and doesn't represent contemporary software engineering standards.

Management Myth

Myth-2: We can recruit more programmers to meet the delivery time if we fall behind schedule (sometimes called the Mongolian horde concept).

Reality: Unlike manufacturing, software development is not a mechanical or mechanistic process.

If newcomers are added to a project, then experienced developers who were already working on the project expend considerable time and energy in getting them up-to-date with the project requirements, current stage, and past issues. This limits the time that engineers and developers--new and old--can spend on actual product development activity. Thus, people can only be added, in a well-coordinated and systematic fashion. If done otherwise, it may just be a waste of time.

Myth: If I opt to outsource the software project management to a third party, all I have to do is sit back and wait for them to finish it.

Reality: If a company does not know how to manage and control software projects internally, it will suffer when outsourcing software projects.

Customer Myths

- **Myth**: A general statement of objective is sufficient to begin writing program
  - we can fill in the details later.
    - Ambiguous requirements – recipe of disaster
    - Effective and continues communication

- **Reality**: The lack of a detailed up-front definition of what the software should do is a major cause of software failures. A formal and detailed description of the information domain and success criteria is absolutely and irrevocably essential.

Myth: Although project requirements continually change these changes can be easily implemented because software is "flexible".

Reality: Software requirements do indeed change, however, the impact of these changes increases drastically as time goes by. **The later the change was introduced the more expensive the change becomes**. If the right amount of effort was put into generating a detailed analysis and design of the system to be built, the customer (you), can review the requirements and recommend the modification you see fit with a low impact on cost. When resources have been committed and design has been established. Change can cause upheaval that requires additional resources and major design modifications.

Practitioner’s Myth

Myth: Once we write the program and get it to work, our job is done.
Reality: practitioner's work is far from over once the software starts running. It still needs to be handed over to the client, and changes, if any, need to be made based on client feedback. Industry statistics actually state that anywhere between 60-80% of software effort goes to waste after it's given to the client for the first time.

Myth: I won't be able to judge the program's quality until it's "running.“
Reality: This false belief is quite common in testing. However, formal technical review is one of the most successful software quality assurance procedures, and it may be used from the start of a project.

Practitioner's Myth

- **Myth:** Software engineering will make us voluminous and unnecessary documentation and it will slow us down.

- **Reality**: Documentation is not just about creating documents. It is about creating quality. Better quality leads to reduced rework, reduced rework results in faster delivery times.

“A successful project is only possible when such myths are dispelled in a timely and effective manner. If these myths are not dispelled, it can lead to a waste of time and effort for everyone involved.”

Software industry is in Crisis!

This is the
SORRY
of Soft
Engine
Toda

- Data on 28
  comple

Managers and Technical Persons are asked:

- ✓ Why d
- ✓ Why a
- ✓ Why c
- ✓ Why d
  develc

- Larger problems,
- Lack of adequate training in software engineering,
- Increasing skill shortage,
- Low productivity improvements.

Some Software failures

Ariane 5

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.

The rocket was destroyed after 39 seconds of its launch, at an altitude of two and a half miles along with its payload of four expensive and uninsured scientific satellites.

Some Software failures

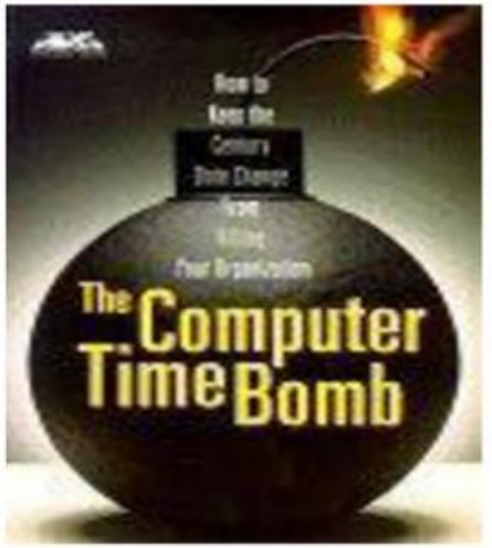
When the guidance system's own computer tried to convert one piece of data the sideways velocity of the rocket from a 64 bit format to a 16 bit format; the number was too big, and an overflow error resulted after 36.7 seconds. When the guidance system shutdown, it passed control to an identical, redundant unit, which was there to provide backup in case of just such a failure. Unfortunately, the second unit , which had failed in the identical manner a few milliseconds before.

Some Software failures

Y2K problem:

It was simply the ignorance about the adequacy or otherwise of using only last two digits of the year.

The 4 digit date format, like 1964, was shortened to 2 digit format, like 64.



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".

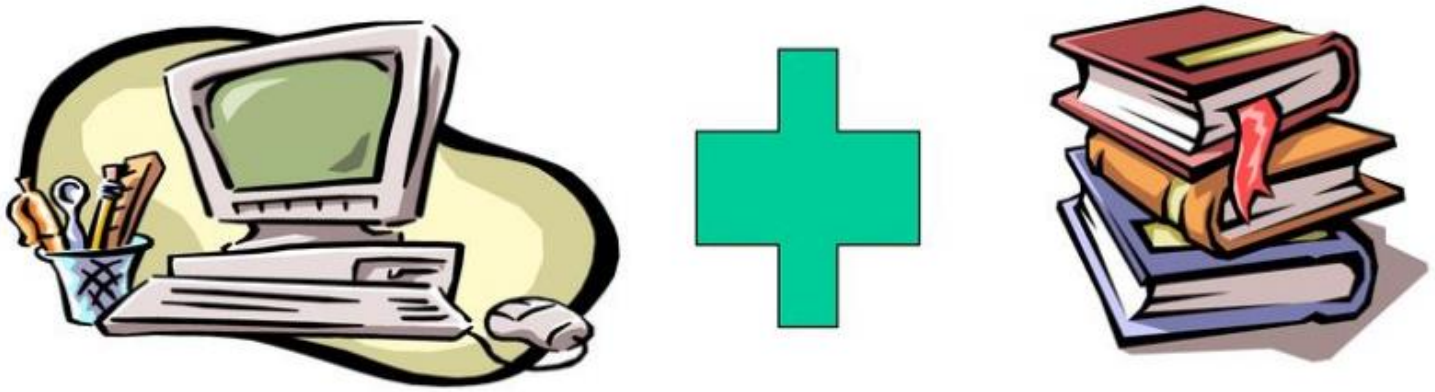
National Health Service

- In the year 2016, it was discovered that the clinical computer system 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.

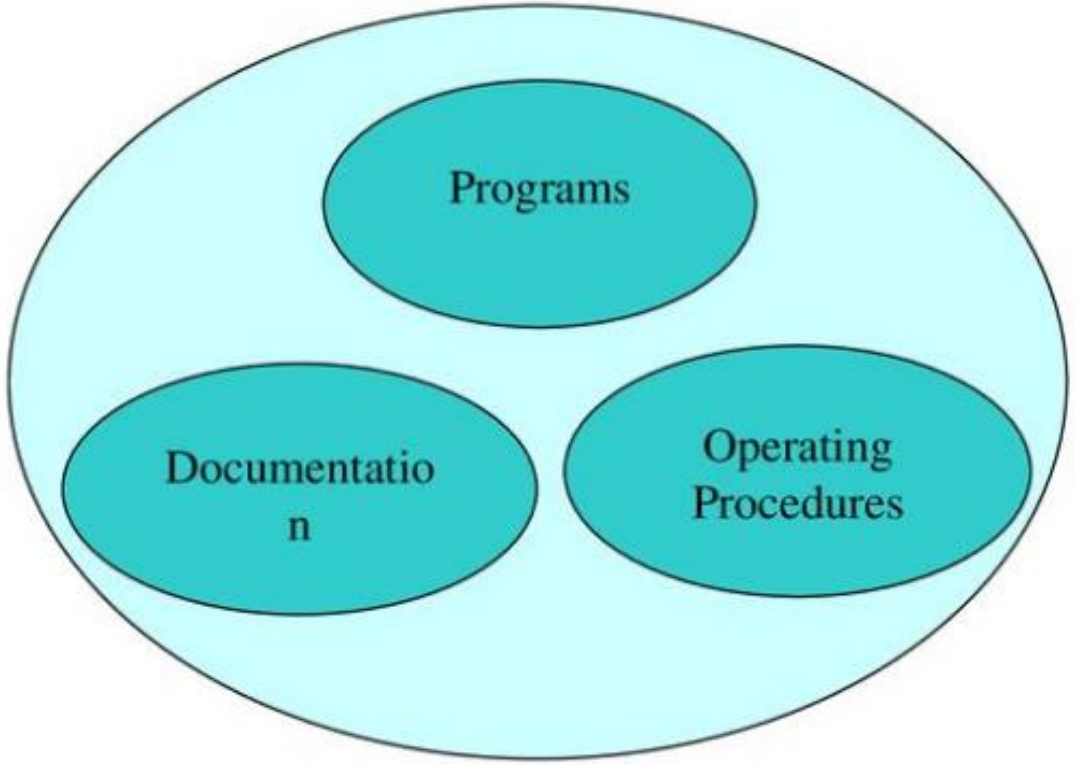
- Software Engineering ≠ Programming
- Programming is actually a pretty small part.
- Other aspects
  - Determining what to build
    - Requirements (what tasks should the Software accomplish?)
    - Specifications (exact operating behaviour)
  - Determining how to build it (Design)
  - Testing (functionality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

What is software?

- **Computer programs** and **associated documentation**



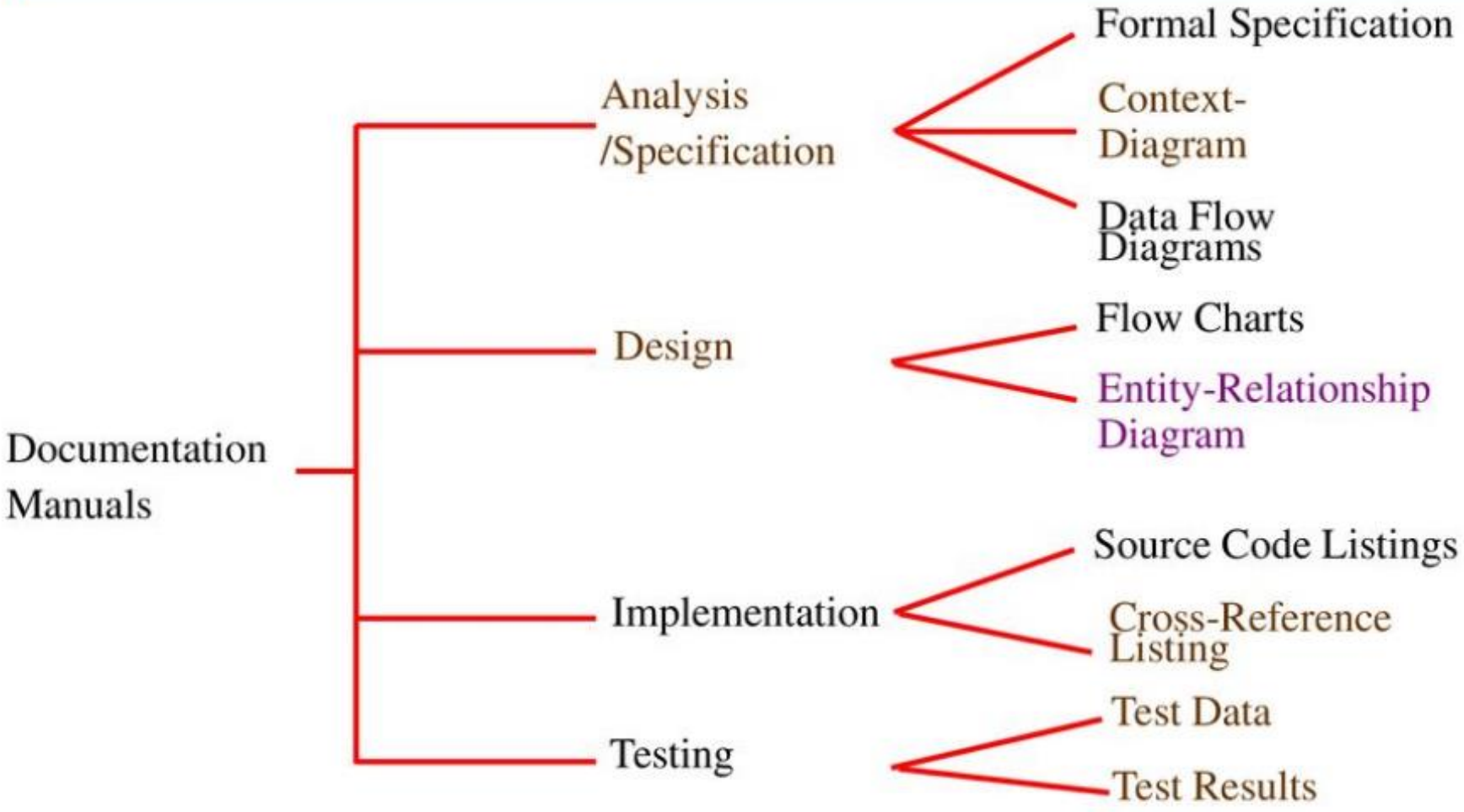
What is software?



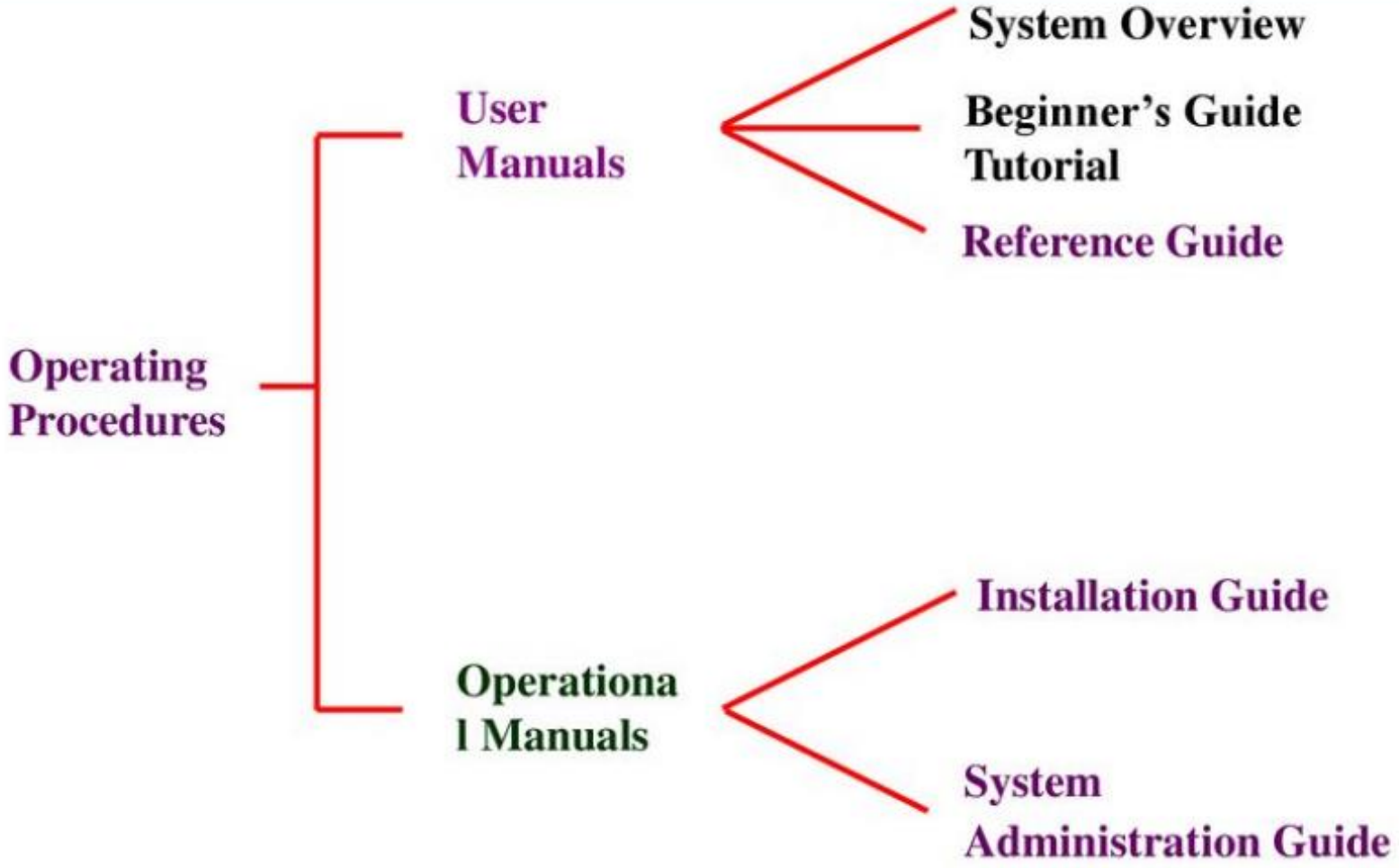
Software=Program+Documentation+Operating Procedures

Components of software

Documentation consists of different types of manuals are



Documentation consists of different types of manuals are



Software Product

- **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
  - **Bespoke** (custom) - developed for a single customer according to their specification

Software is Complex

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

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.