Software Failures

Why does a software system fail?

Causes of software failure

Causes of Software Failure

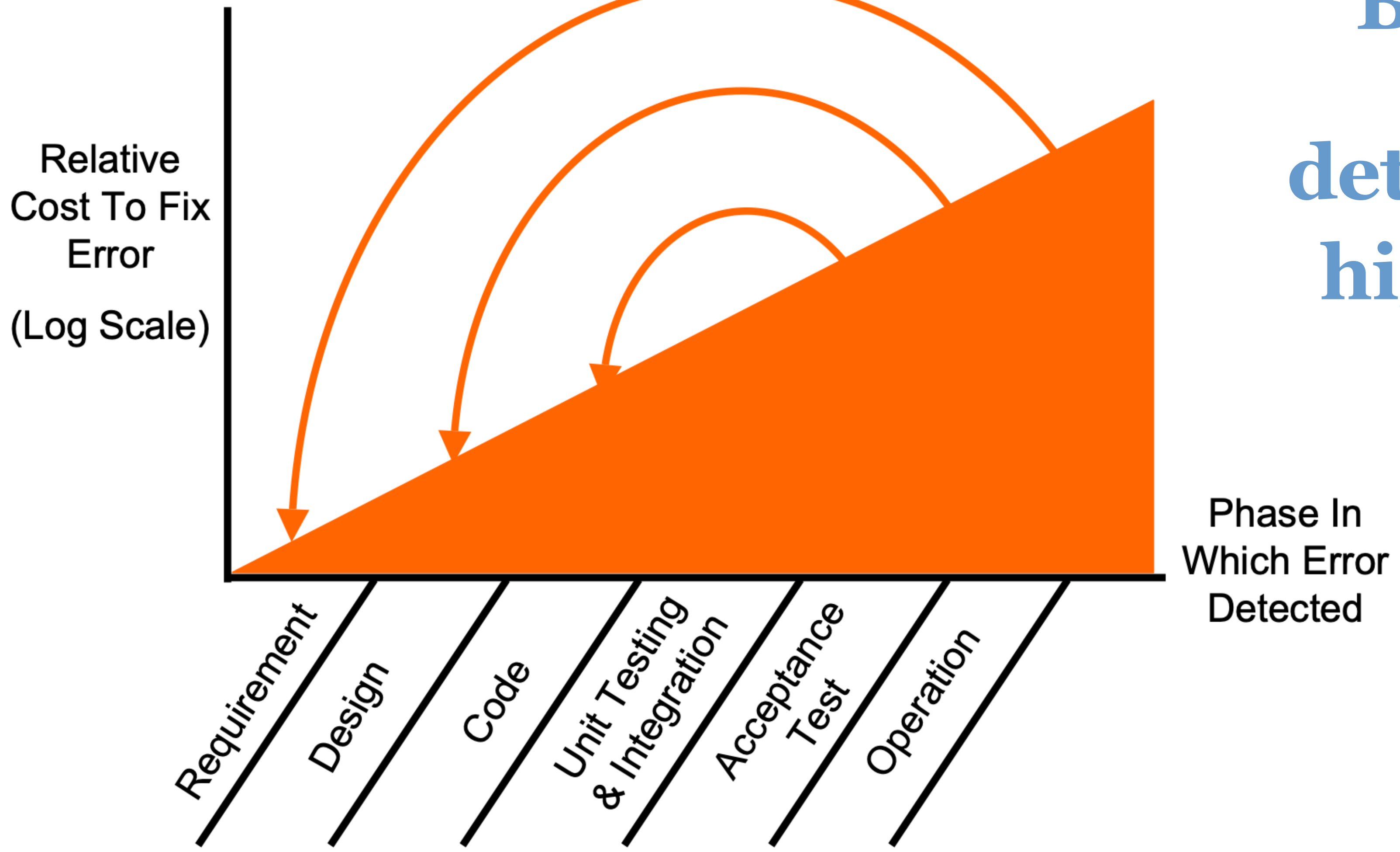
Many factors can cause software failure, however, there are some general causes, including:

-Undetected bugs!
-Co-evolution of software
-Costs factors
-Risk factors

Greater complexity= greater changes = potential errors!

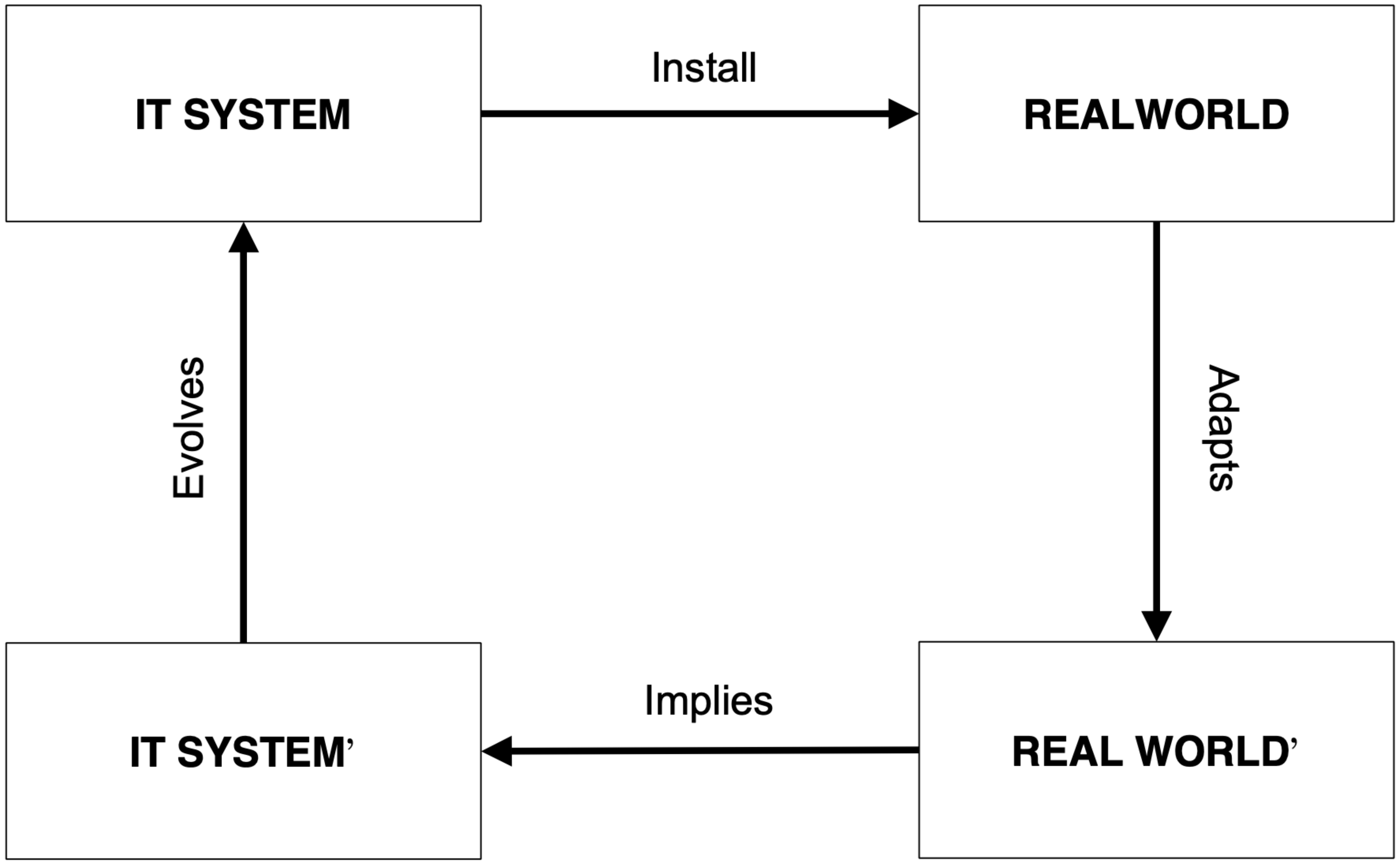
Causes: Bugs

Cost of delayed error detection


Relative Cost To Fix Error
(Log Scale)
Phase In Which Error Detected
Requirement
Design
Code
Unit Testing & Integration
Acceptance Test
Operation

Bugs: the the later detected, the higher cost to fix

Causes: IT System co-evolution- eternal loop



Causes: Costs

***System Development*.**

System Requirements	2
Hardware Requirements	8
Software Requirements	10
Software Design	12
Coding	13
Unit Test	24
Integration Test	13
Documentation	6
System Test	12
TOTAL	**100**

Causes: But total Costs

Pre-Delivery

– System Development	100
– Installation	15

Post-Delivery - Maintenance

– Defect Removal	60
– Environmental Changes	60
– Enhancements	180
TOTAL	**415**

What Do Coders Actually Do?

Reading Code (Code Reviewing)	16%
Job Communications	25%
Personal & Business Calls	9%
Training	6%
Electronic Mail	9%
Surfing The Web	9%
Other	13%
Writing Code	13%

- Initial writing code is 13% of 100/415 of 13% of development.

=>THUS **CODING** IS ONLY 0.004 of TOTAL DEVELOPMENT COST

Risk Factors: DELPHI Study

9.5	Lack of top management commitment to the project.	♣
8	Failure to gain user commitment.	♣
8	Misunderstanding the requirements.	♦
7.5	Lack of adequate user involvement.	♦
7	Failure to manage end user expectation.	♦
6.5	Change of scope of the project.	♦
6.5	Lack of required skills in the development project.	♣
6.5	Lack of frozen requirements.	♦
6	Introduction to new technology.	♠
6	Insufficient staffing.	♣
5	Conflicts between end user departments.	♦

1 = less important

10 = most important

4 organisation factors ♣ 6 requirements ♦ 1 new technology ♠

Software Engineering …

Did software engineering overcome these issues?

Software Engineering: Progress

Important progress in Software Engineering:

- Ability to produce more **complex** software has increased
- New technologies have led to **new SE approaches**
- A better understanding of the **activities** involved in software development
- Effective **methods** to specify, design and implement software have been developed
- New **notations** and **tools** have been produced

What is a software process?

Software Process (SP) is a **set of activities** whose goal is the development or evolution of software

Fundamental activities in all software processes are:

Specification - what the system should do and its development constraints

Development - production of the software system (design and implementation)

Validation - checking that the software is what the customer wants

Evolution - changing the software in response to changing demands

What is a Software Process Model (SPM)?

SPM is a simplified representation of a software process, presented from a specific perspective

- **Examples of process perspectives:**

  Workflow perspective represents inputs, outputs and dependencies
  Data-flow perspective represents data transformation activities
  Role/action perspective represents the roles/activities of the people involved in software process

- **Generic process models**
  - **Waterfall**
  - **Evolutionary development (commonly known as agile)**
  - **Formal transformation**
  - **Reuse-oriented: Integration from reusable components**

What are the costs of software engineering?

Roughly **60% of costs are development costs, 40% are testing costs**. For custom software, evolution costs often exceed development costs

Costs vary depending on the type of system being developed **and the requirements** of system attributes, for example for high system performance and reliability costs can be high.

Distribution of costs depends on the development model that is used

What is CASE ? (Computer-Aided Software Engineering)

Software systems which are intended to provide automated support for software process activities, such as requirements analysis, system modelling, debugging and testing

Upper-CASE

Tools to support the early process a requirements and design

Lower-CASE

Tools to support later activities such as programming, debugging and testing

Software Characteristics

Does software have special characteristics?

Software versus Program

Do “software” or “program” mean the same?

- **Program**: a set of instructions written in a particular programming language for a specific purpose

- **Software**: a combination of program(s), documentation (development documents) and operating procedure documents (provided to customers at the time of release).

Software versus Program

Development Documents, **include**:

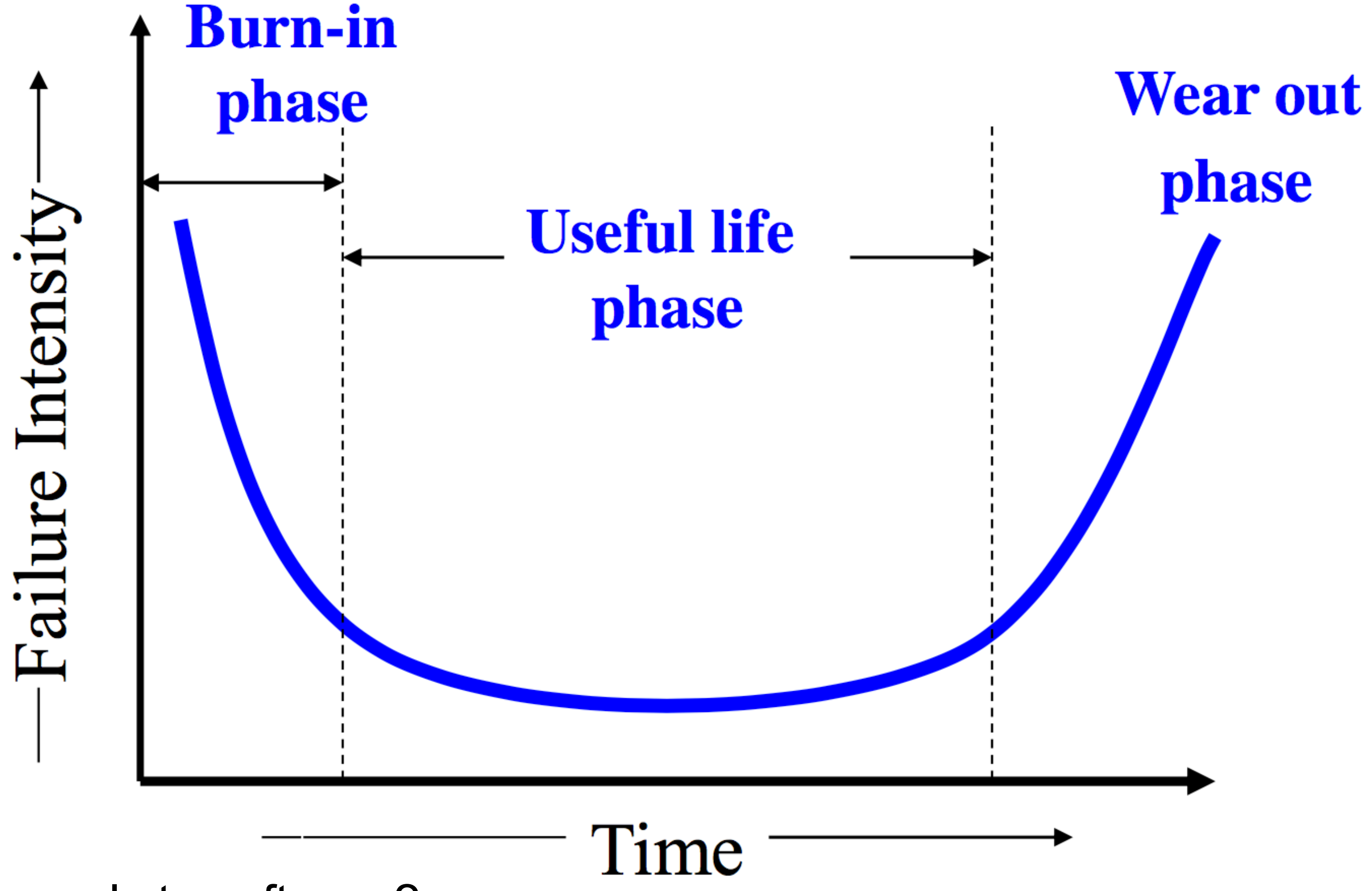
- Software Requirements and Specification document
- Software Design Document
- Test plan document
- Test suite document
- Source code etc.

Operating Procedure Documents, include:

- Installation manual
- System administration manual
- Beginner’s guide tutorial
- System overview
- Reference guide etc.

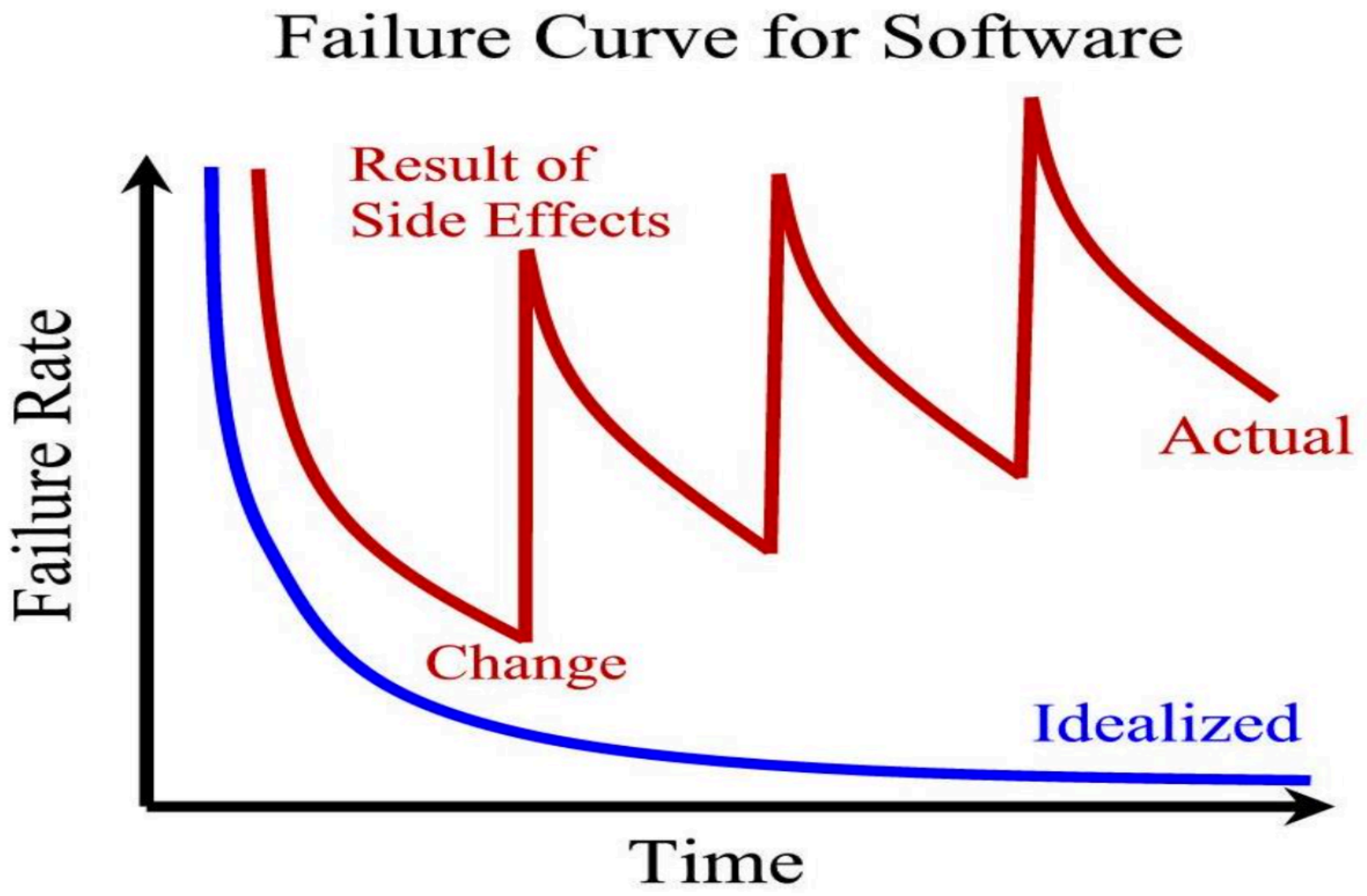
Software is intangible and does not wear out?

Normal tangible Products life-cycle phases.



Do all these phases apply to software?

Software is Reusable!



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 be able evolve to meet changing needs with minimal effort and time
- **Dependability**
  - Software must be trustworthy
- **Efficiency**
  - Software should not make wasteful use of system resources
- **Acceptability and Usability**
  - Software must be acceptable and usable by the users for the purpose it was designed for.

What are the key challenges that are still facing software engineering?

Software engineering in the 22st century still faces three key challenges:

- **Legacy systems**
  - Old, valuable systems must be maintained and updated
  - However can these systems be kept functional? how newly developed systems can work or interoperate with these old systems?
- **Increasing Diversity and Heterogeneity**
  - Systems are distributed and include a mix of different hardware and software
  - How software systems could be developed to work in heterogeneous environments
- **Dependability and Delivery**
  - Having trustworthy software with faster delivery of software product (time-to-market)
  - How to achieve a trustworthy system?