CS 575
Software Testing and Analysis

Özyeğin University
Graduate School of Engineering

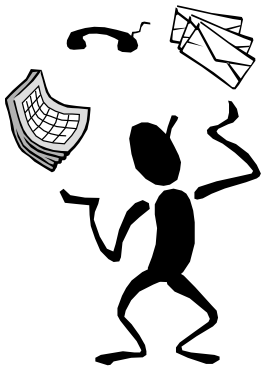
Hasan Sözer
hasan.sozer@ozyegin.edu.tr

Course Objectives

- Explain basic **principles** of software testing
- Summarize basic testing **techniques** and strategies
- Provide a **rationale** for selecting and combining them within a software development process
- Understand **limitations** and **possibilities**

Our Focus Today

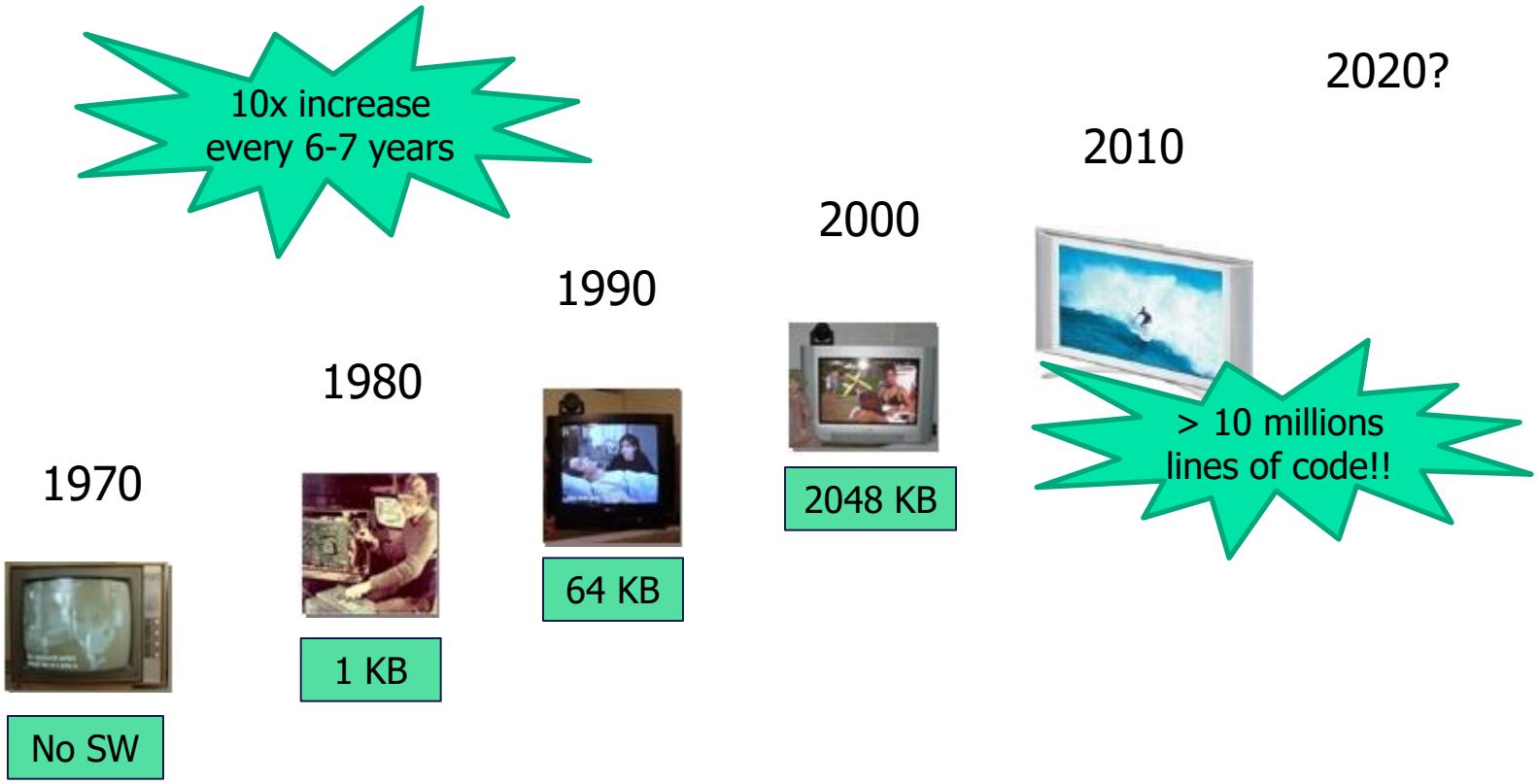
- Introduction & Context
  - Software Dependability, Reliability
- Course Organization
  - Requirements, goals, expectations
  - Study material
  - Schedule
- First part of the course
  - Basic Terminology
  - Software Testing Principles



Increasing Software and Complexity in Systems

- Software systems are getting bigger & more complex
  - it becomes harder to ensure their quality
- Yet, we **depend on** software even more than before
  - airplanes, trains, TV sets, ovens, cell phones, ...
- The quality of the software largely defines the behavior and quality of systems we use

Example: Software Size in TVs



Software failure caused $1.7 trillion in financial losses in 2017

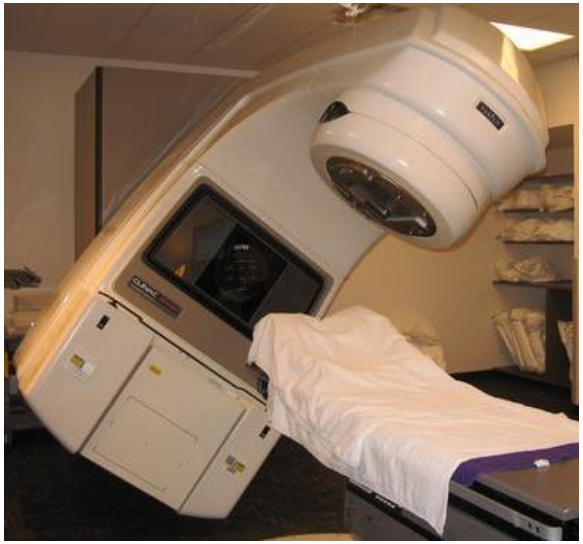
"Software testing company Tricentis found that retail and consumer technology were the areas most affected, while software failures in public service and healthcare were down from the previous year"

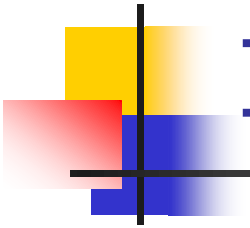
https://www.techrepublic.com/article/report-software-failure-caused-1-7-trillion-in-financial-losses-in-2017/

Software Failures as a Threat for Safety

- **Toyota brakes**: dozens dead, thousands of crashes
- **THERAC-25** radiation machine: poor testing resulted in 3 dead

"On February 8th, Toyota announced recalls of tens of thousands of 2010 Prius and Lexus hybrids to address braking problems, this one caused by a software error." [www.theatlantic.com]

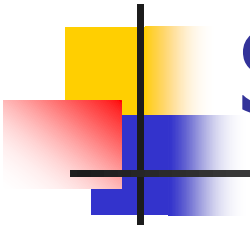




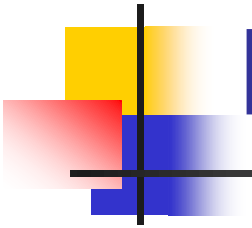
Impact of new technologies

- Advanced development technologies
  - can reduce the frequency of some classes of errors
  - but do **not** eliminate errors
- New development approaches can introduce new kinds of faults, e.g.,
  - deadlock or race conditions for distributed software
  - new problems due to the use of polymorphism, dynamic binding and private state in object-oriented software

Introducing the Context & Scope of the Course

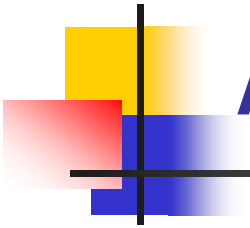


- Dependability: Ability to deliver service that can justifyably be trusted
  - An integrating concept that encompass 5 quality attributes: Availability, **Reliability**, Safety, Integrity and Maintainability
- Security is another composite attribute
  - combines Confidentiality, Integrity and Availability



Dependability attributes

- Availability: readeness for correct service
- Reliability: continuity of correct service
- Safety: absence of catastrophic consequences on the user(s) and the environment
- Integrity: absence of improper system alterations
- Maintainability: ability to undergo modifications and repairs



Attribute inter-relationships

- Safe system operation might depend on the system being available and operating reliably
- A system might be stopped for safety reasons while hindering its availability
- A frequently crashing, unreliable system will also be unavailable
- A system might be very reliable but its availability can be very low due to long-lasting recovery and repair
- Denial of service attacks on a system are intended to make it unavailable.

Reliability
as a software quality attribute

- **Reliability**: The probability that a system will continue to function without **failure** for a specified period in a specified environment

- **Failure**: deviation of the delivered service from compliance with the **specification**

Chain of reliability threats

failure

deviation of the delivered service from compliance with the specification

Chain of reliability threats

propagates to

system error → **failure**

system state that is liable to cause a failure

deviation of the delivered service from compliance with the specification

Chain of reliability threats

activates *propagates to*

fault → **system error** → **failure**

the cause of a system error

system state that is liable to cause a failure

deviation of the delivered service from compliance with the specification

Chain of reliability threats

human error —*leads to*→ **fault** —*activates*→ **system error** —*propagates to*→ **failure**

mistake

the cause of a system error

system state that is liable to cause a failure

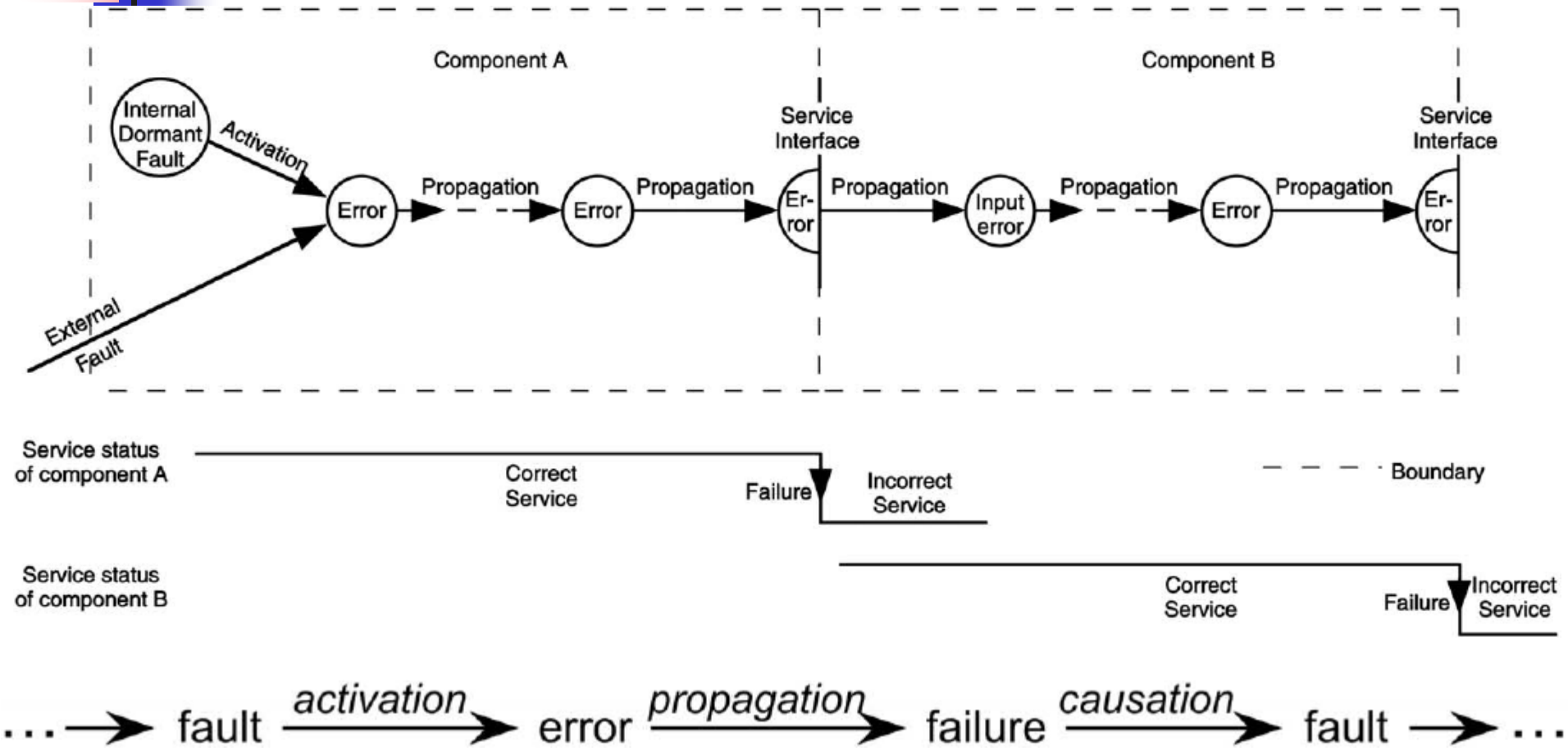
deviation of the delivered service from compliance with the specification

Chain of reliability threats

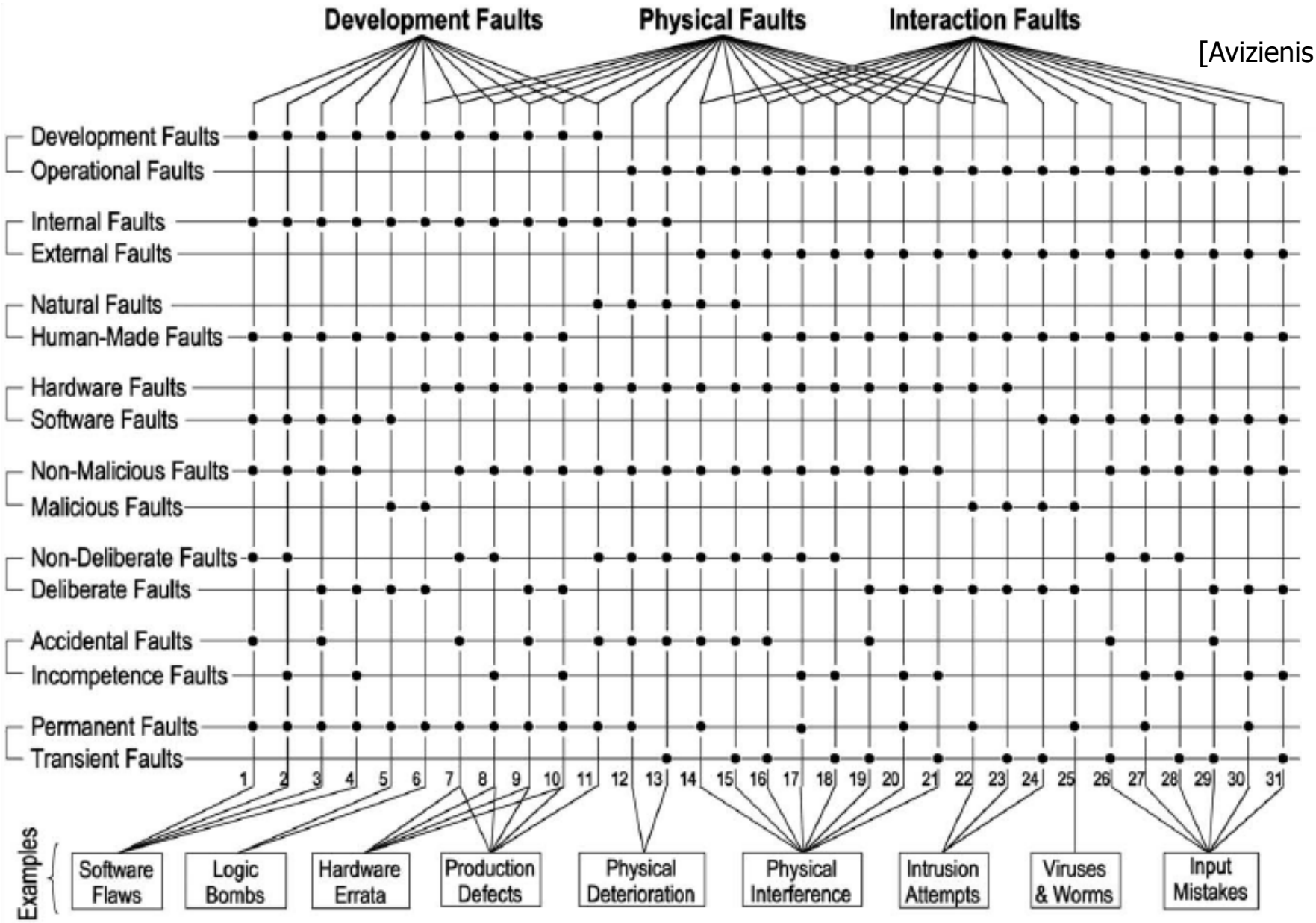
Examples:	**Bug: array index uninitialized**	**Index out-of-bounds**	**Segmentaion fault, system crash**

Chain of reliability threats

[Avizienis 2004]



Fault Classification



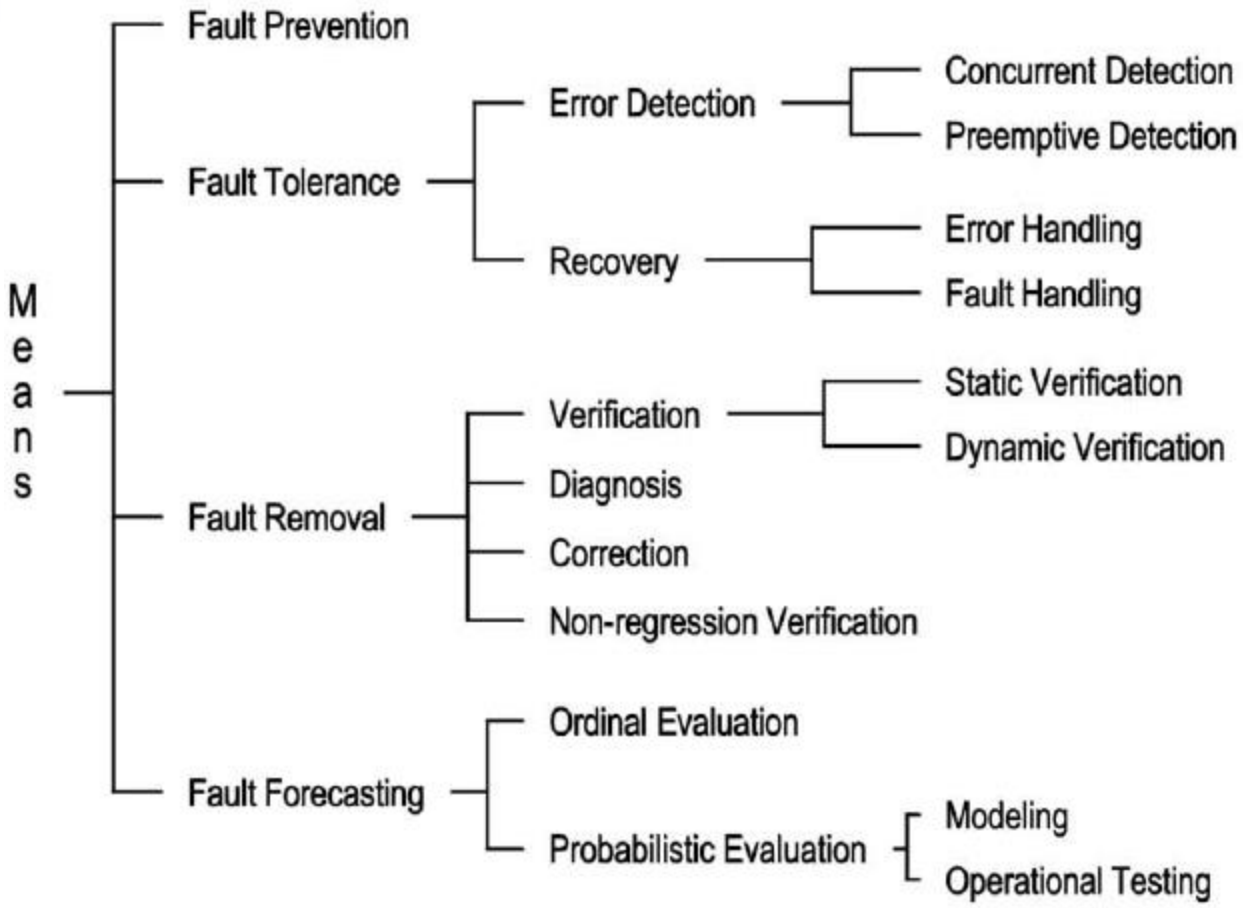
[Avizienis 2004]

How to prevent failures?

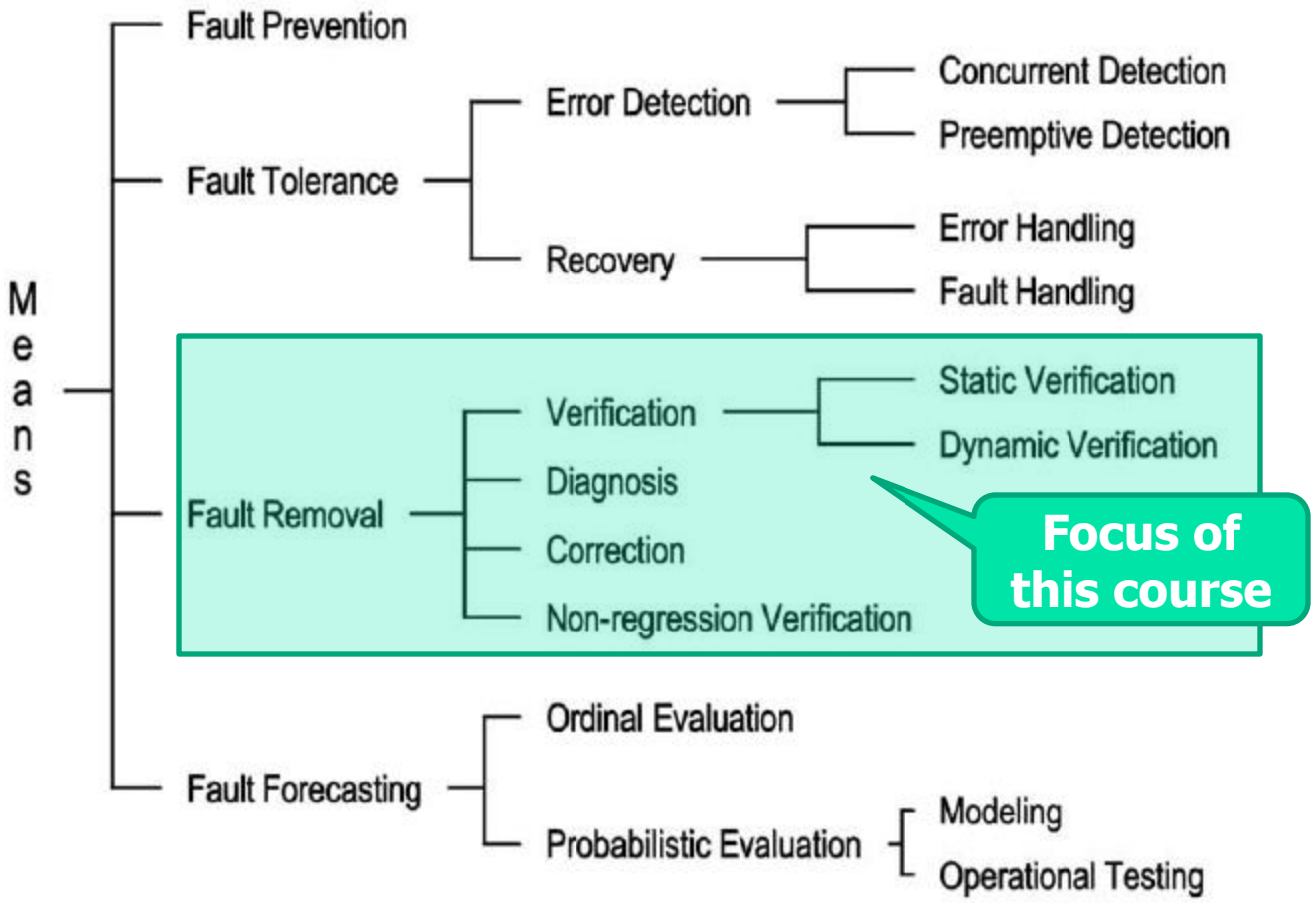
- The chain of threats must be broken
  - Fault prevention (avoidance)
  - Fault removal
  - Fault tolerance

Dependability means

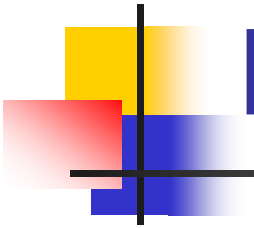
[Avizienis 2004]



Dependability means

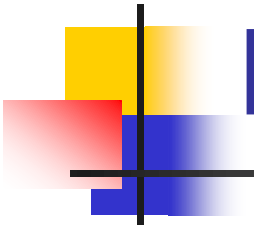


Avizienis 2004



Fault Prevention

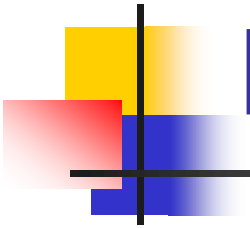
- Development techniques that either minimize the possibility of mistakes or trap mistakes before they result in the introduction of faults
- e.g., process improvement approaches to follow a (more) rigirous software development



Fault Removal

- Verification and validation techniques that increase the probability of detecting and correcting faults before the system becomes operational

- i.e., **Software Testing and Analysis**



Fault Tolerance

- It is usually not feasible to prevent and/or remove all the faults.

"There are two ways to write error-free programs;

only the third one works."

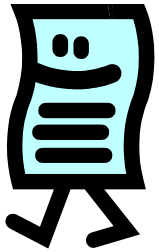
Alan Perlis

- Fault tolerance is the property that enables a system to continue operating properly in the event of the failure of some of its components.

Fault Tolerance (cont'd)

- Fault-tolerant design
  - refers to a method for designing a system so that it will continue to operate,
  - possibly with a reduced functionality/quality,
  - rather than failing completely, when some part of the system fails.

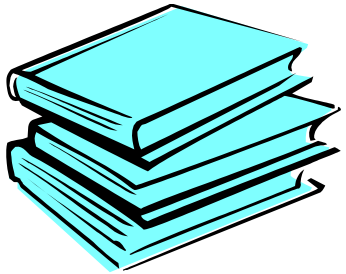
Further reading material..



- A. Avizienis, J.-C. Laprie, B. Randell, C. Landwehr, **Basic Concepts and Taxonomy of Dependable and Secure Computing**, IEEE Transactions on Dependable and Secure Computing, Vol. 1(1), 2004.

Course Organization

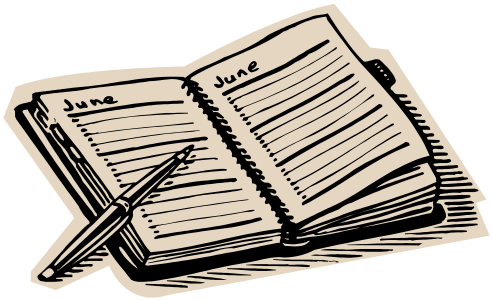
Study Material



- Course Book:
  - P. Amman and J. Offut: Introduction to Software Testing, Cambridge University Press, 2008.
- Recommended supplementary book:
  - M. Pezze and M. Young: Software Testing and Analysis: Process, Principles, and Techniques, Wiley, 2008.
- Course slides, assignments on LMS
  - http://lms.ozyegin.edu.tr

OZU Learning Management System

The List of Topics



Week	Topic
1.	Basic Terminology and Testing Principles
2.	Testing Process and Test-driven Development
3.	Testing Techniques
4.	Control Flow Analysis
5.	Test Coverage Measures
6.	Graph Testing
7.	Software Analysis Techniques and Tools
8.	Fault Localization and Cost Estimation
9.	Concolic Testing
10.	Test Effectiveness Analysis and Test Case Prioritization
11.	Testing Tools and Automation
12.	Testing Web Applications
13.	Testing Graphical User Interfaces
14.	Qualitative Analysis and Diagnosis Techniques

Grading (CS 575)

- 30% Project
  - 5% Proposal
  - 5% Progress Presentation
  - 10% Final Project Report
  - 10% Presentation
- 10% Assignment
- 60% Midterm Exam (x2)

Overall Schedule

February				March				April					May	
5	12	19	26	5	12	19	26	2	9	16	23	30	7	14
		proposal		m1			progress					m2	pres.	assignment
													report	

project / assignment deliverable
exam
no classes

Questions so far..