# CS 487 – SOFTWARE ENGINEERING

## Week 4 Engagement – Proactive and Reactive Quality Assurance

**Suhas Palani** **A20548277** spalani3@hawk.iit.edu

### Proactive and Reactive Quality Assurance

- **Proactive QA**: Focuses on preventing flaws by establishing standards and carrying out inspections to ensure that the product is built with quality from the beginning.

- **Reactive QA**: Involves discovering and repairing bugs after they arise, usually by testing to guarantee the functioning and efficiency of the product.

### The Role of Testing

Testing is essential in software development for:
- Demonstrating correctness and completeness: Ensuring each requirement is met.
- Discovering defects: Identifying issues before release.
- Verification and Validation (V&V): Making sure the right product is constructed properly.

### Testing Levels

- Unit Testing: Tests individual components or units.
- Component Testing: Verifies the operation of interacting entities.
- System Testing: Verifies the system in different scenarios.

### Test-Driven Development (TDD)

- TDD: A method of developing software where tests are composed before code. It ensures clean code and facilitates automated testing, providing documentation and understanding of the system.

### Inspections vs. Testing

- Inspections: Static reviews of code and design documents to ensure adherence to standards.
- Testing: Dynamic execution to find defects, requiring a certain level of completeness.

### Cleanroom Software Development

- Cleanroom Approach: Aims for zero-defect software by using formal methods and rigorous inspections. It emphasizes defect prevention over defect removal, using incremental development and statistical testing to ensure reliability

## Quality Management and Control

- Quality Assurance (QA): Through methodical actions, prevents errors and raises quality.
- Quality Control (QC): Mostly through testing and inspections, QC finds and removes flaws.

## Configuration Management

Supports software evolution by managing changes, versions, and releases. Key activities include:
- Change Management: Formalizes change requests and assesses their impact.
- Version Management: Handles multiple versions of the software.
- System Building: Compiles and links components to create a system version.

## Change, Release, and Version Management

- **Change Management:** Manages planned deployment of changes to control IT infrastructure, minimizing incidents and ensuring efficient handling of changes

- **Release Management:** Coordinates the delivery of new and enhanced IT services, ensuring integration and effective deployment

- **Version Management**: Manages software versions, ensuring proper tracking and integration, and supports parallel development through branching and merging

These topics highlight the importance of integrating both proactive and reactive QA strategies, along with structured management processes, to ensure high-quality software delivery.