CS 487 – SOFTWARE ENGINEERING

Week 6 Engagement – Dependability and Reliability

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

1. **System Reliability:**
   - **Reliability**: The system's ability to work without interruptions.
   - **Non-functional requirement**: Like performance and scalability, it's about how well the system works.
   - **User expectations**: Users want systems to always work, but 100% reliability is tough.
   - **Reliability vs. Availability**: If a system isn't available, you can't trust it to be reliable.
   - **Unreliability**: This can show up as errors, wrong answers, missing information, or crashes.
   - **Impact**: When a system is unreliable, it makes it tough to get things done, especially when you need critical information.

2. **Dependability Considerations:**

- **Repairability**: Systems should have diagnostic tools and be easily reparable, with quick, "surgical" repairs.
- **Maintainability**: Systems should be easily and economically adaptable to new functional requirements.
- **Error Tolerance**: The system should be able to automatically detect and fix a small subset of possible errors.
- **Survivability**: The system should be robust and able to survive even when subjected to attack or failure.

3. **Risk Management:**
   - **Resilience**: The system should withstand any danger, recognizing, resisting, and recovering from all attacks.
   - **Risk Management**: Involves several steps:
     - **Identify failures**: Determine what could go wrong.
     - **Compute likelihood**: Assess how likely each failure is.
     - **Examine threats**: Look at possible threats and vulnerabilities.
     - **Specify remedies**: Decide on actions to prevent or fix issues.
   - **Reliability Assessment**: Evaluated through risk assessments that:
     - **Redundantly check**: Repeatedly examine the system.
     - **Cost-benefit analysis**: Weigh the costs of mitigating risks against the benefits of reliability.

4. **Failure Categories:**
   - **Hardware failure:** Failure in the design, manufacturing, or operation of the major hardware components.

- **Software failure:** errors in coding, requirement errors, or any combination of the two. **Operational failure:** The user made an advised action.

5. **Safety-Critical Systems:**

We may classify a system as being primary safety-critical, if failure directly causes harm, or secondary safety-critical, if a failure could result in harm or a dangerous situation could arise.

6. **Security in Systems:**

- **Open Systems**: Designed for data sharing, making them attractive targets for attackers seeking unauthorized access or causing denial of service attacks.
- **Security Engineering**: Involves policies, standards, and measures to minimize threats.
- **Security Management**: Includes:
  - **User access control**: Managing who can access the system.
  - **Deployment strategy**: How the system is set up and maintained.
  - **Monitoring and recovery**: Ability to detect attacks and recover from them.

7. **Designing for Security and Deployment:**

- Security practices such as limiting access, minimizing harm/recoverability through redundancy, and allowing for recovery.
- Deployment will include supporting views of the configuration, minimizing default privileges, and making fixes easier.

8. **Dependable Programming Techniques:**

- Practices that depend or don't depend on the system such as visibility management, input checking, exception handling, recovery, and checking array bounds.
- Practices to avoid (i.e., risky programming) such as dynamic memory allocation, unbounded arrays, recursion.