Summary of Lecture 10: Test Management

This lecture covers **Test Management**, including **software testing concepts, fault handling, testing levels, test cases, model-based testing, test doubles, and automated testing with Robot Framework**.

Key Terminology in Testing

Term	Definition
Failure	Any deviation from the expected behavior.
Erroneous State (Error)	A system state that may lead to failure.
Fault (Bug)	The root cause of an error (e.g., coding mistakes).
Validation	Checking if the system behaves as expected.

Types of Faults and Errors

✅ **Faults in Interface Specification**

- Mismatch between what a client expects and what the server provides.

✅ **Algorithmic Faults**

- Missing initialization, incorrect branching, missing null checks.

✅ **Mechanical Faults**

- Hardware issues (e.g., overheating, sensor failure).

✅ **Errors**

- Wrong user input, concurrency issues, null references.

📌 **Example:**
A train derails due to a **faulty compass** giving wrong readings.

- **Fault** → Compass error.
- **Error** → Train in wrong direction.
- **Failure** → Train crash.

Software Testing: Breaking the Product to Find Faults

🔹 **Purpose of Testing**

- Find **bugs & weaknesses** in software.
- Show that faults **exist** (not prove the software is perfect).
- Requires **a different mindset** from development.

🔹 **"Don't test to pass, test to fail!"**

- Successful testing means **finding problems before users do**.

Fault Handling Techniques

Approach	Description
Fault Avoidance	Prevent issues using **good design & coding standards**.
Fault Detection	Identify faults through **various testing techniques**.
Fault Tolerance	Make systems **resilient** to faults (e.g., error handling, redundancy).

Types of Testing

1️⃣ Blackbox Testing 🏴

- **Focus:** Only inputs & outputs, without looking at internal code.
- **Goal:** Ensure the system meets **functional requirements**.
- **Example:** Checking if a login form accepts valid credentials.

2️⃣ Whitebox Testing ⚪

- **Focus:** Internal structure & logic of the software.
- **Goal:** Ensure every function, condition, and flow is tested.
- **Example:** Testing whether all branches in an `if-else` statement execute correctly.

Testing Levels (From Small to Large)

Level	Who Performs It?	Goal
Unit Testing	Developers	Test individual components in isolation.
Integration Testing	Developers	Test multiple modules working together.
System Testing	Developers	Verify that the full system meets requirements.
Acceptance Testing	Clients/Users	Ensure the system meets user needs.

📌 **Example:** Testing a shopping cart system

- **Unit Test** → Checking if "Add to Cart" function works.
- **Integration Test** → Ensuring checkout integrates with payment gateway.
- **System Test** → Validating the entire shopping experience.
- **Acceptance Test** → Customer verifies the system before launch.

Test Model & Test Cases

🔹 **System Under Test (SUT)**

- The **specific part of the system** being tested.

🔹 **Test Model**

- A **collection of test cases** that check specific system behaviors.

🔹 **Test Case Components**

1️⃣ **Name** – Descriptive test name.
2️⃣ **Preconditions** – The initial system state.
3️⃣ **Input** – Data entered into the system.
4️⃣ **Expected Output** – The correct system response.

📌 **Example Test Case:**

Test Name	Precondition	Input	Expected Output
Login Test	User exists	Email + Password	"Login Successful"

Model-Based Testing (MBT)

📌 **What is MBT?**

- Testing based on **system models** instead of manually written test cases.

✅ **Advantages**

- **Early testing** → Detect issues before coding starts.
- **Reduces manual effort** → Test cases are **automatically generated**.
- **Better traceability** → Easier to track issues back to requirements.

📌 **Models Used in MBT**

- **State-Machine Diagrams** (for workflows).
- **Decision Tables** (for conditional logic).
- **Flowcharts** (for process visualization).

Test Doubles: Replacing Real Dependencies in Testing

🔹 **What is a Test Double?**

- A **fake object** that replaces real system components **to simplify testing**.

🔹 **Types of Test Doubles:**

Type	Description	Example
Dummy	Placeholder object (not used)	Empty parameter in function calls.
Fake	Works but is not production-ready	In-memory database instead of MySQL.
Stub	Provides fixed responses	Always returns "User found" for any login request.
Mock	Simulates real behavior & checks calls	Ensures "sendEmail()" was called after registration.

📌 **Example:** Testing an online auction system

- Replace **real payment system** with a **mock payment processor** to simulate transactions.

Robot Framework: Automated Testing

📌 **What is Robot Framework?**

- **An open-source automation testing tool** for Acceptance Test-Driven Development (ATDD).
- **Uses human-readable syntax** (keyword-driven, data-driven, Gherkin).

- Supports **Python, Java, Selenium, etc.**

📌 **Example: Running a Calculator Test in Robot Framework**

```
*** Test Cases ***
Addition Test
  [Documentation]  Test if calculator adds numbers correctly
  Push Button  1
  Push Button  +
  Push Button  2
  Push Button  =
  Result Should Be  3
```

✅ **Key Features of Robot Framework:**

- **Supports multiple programming languages.**
- **Easy to read & write test cases.**
- **Generates automated test reports (HTML format).**

Final Takeaways

✅ **Testing is about finding bugs, not proving software is perfect.**
✅ **Different levels of testing exist (Unit, Integration, System, Acceptance).**
✅ **Test cases should be structured with clear preconditions, inputs, and expected outputs.**
✅ **Test doubles (mocks, stubs) help isolate components for better testing.**
✅ **Robot Framework simplifies test automation.**

Keywords

- **Failure, Error, Fault**
- **Validation & Verification**
- **Blackbox vs. Whitebox Testing**
- **Unit Testing, Integration Testing, System Testing, Acceptance Testing**
- **System Under Test (SUT)**
- **Test Model & Test Case**
- **Model-Based Testing (MBT)**
- **Test Doubles (Stub, Mock, Fake, Dummy)**
- **Robot Framework**
- **Automated Testing**