**Key Topics:**

1. **Testing Requirements**
2. **Verifiable Requirements**
3. **Non-functional Requirements**
4. **Hard-to-test Requirements**
5. **Domain Modelling**
6. **Prototyping**

**1. Testing Requirements**

**Why do requirements need tests?**

* Essential to show (validate) that the finished product satisfies the requirement.
* Non-validated requirements are just "wishes."
* Plan verification methods for each requirement, typically using acceptance tests based on user interactions.

**Requirement Test**

* Demonstrates whether a system satisfies a requirement.
* Write tests simultaneously with requirements to form a contract.
* Utilize the “Fit Criterion” to determine if a requirement is met.

**Agile vs. Waterfall**

* Agile integrates testers early, enhancing requirement testability.
* Waterfall delays tester involvement, making changes harder later.

**2****.** **Verifiable Requirements**

**Definition**

* A requirement is verifiable if there is a finite, cost-effective method to check if the software meets the requirement.
* Verification can be achieved using test cases, analysis, or inspection.

**Example:**

* **Performance Speed Requirement:**
  + *Requirement*: System responds to user requests in < 1 second for up to 100 concurrent users.
  + *Test*: Create a script to simulate 100 concurrent user requests and measure response times. Ensure all requests are processed within 100 seconds (1 second per request).

**3.** **Non-functional Requirements**

**Objective** **Metrics**

* **Performance Speed:** Transaction processing per second, response time, screen refresh time.
  + *Example*: System responds in < 1 second for 100 users.
* **Size:** Memory usage in kilobytes or number of RAM chips.
  + *Example*: App uses less than 200 MB on Android.
* **Reliability:** Mean time to failure, probability of unavailability, failure rate.
  + *Example*: System available for 1400 minutes per 24 hours.
* **Robustness:** Restart time after failure, percentage of failure-causing events, data corruption probability.
  + *Example*: No more than 1 minute of data loss if the system crashes.
* **Ease of Use:** Training time, error rate, help frames.
  + *Example*: Users complete the Plan Trip use case within 2 minutes post-tutorial.
* **Portability:** Target-dependent statements, number of target systems.
  + *Example*: App runs on all Android models since 2015 for OS versions 7+.

**4.** **Hard-to-test Requirements**

* **System Requirements:** Affect the entire system, often hard to validate due to dependencies on other functional requirements.
* **Exclusive Requirements:** Specify behaviours to avoid, e.g., system failures must not corrupt databases. Backup and Recovery Testing, Mocking Database Failures
* **Non-functional Requirements:** Require large test sets and extensive validation, e.g., reliability tests.

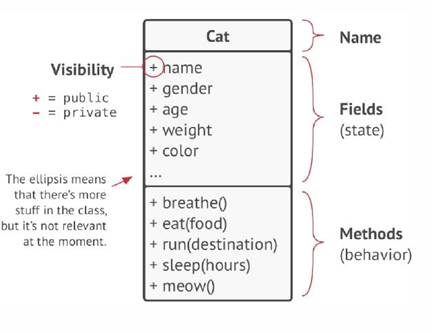
**5.** **Domain Modelling**

A domain is a representation of real-word conceptual classes, not of software components. It is not a set of a set of diagrams describing software classes, or software objects with responsibilities.

**Conceptual Classes**

* Represent real-world entities, not software components.
* Captured using UML class diagrams.

**Classes & Objects**



* **Class:** Template for objects, defines attributes and behaviours.
* **Object:** Instance of a class, with state (data) and behaviour (methods).

**Relationships**

* **Associations:** Interactions between objects.
* **Dependency临时关系:** Temporary links between objects.
* **Aggregation包含关系:** Objects containing references to others.
* **Composition被包含:** "Whole-part" relationships between objects.

**6.** **Prototyping**

**Why Prototype?**

* Validates the software engineer's interpretation of requirements.
* Elicits new requirements and corrects assumptions.
* Prototypes clarify dynamic behaviour and user interfaces better than textual descriptions.

**Types of Prototypes**

* **Software Prototypes:** Executable, created using rapid prototyping tools.
* **Mock-ups:** Screen designs using drawing software or on paper/whiteboards.

**Benefits**

* Prototypes help in understanding and validating requirements.
* They are essential for stakeholder feedback and discovering user needs.

**Detailed Examples for Equations and Metrics**

**Example 1: Performance Speed Test**

* *Requirement*: The system shall respond to user requests in < 1 second under normal load (<100 users).
* *Test Method*: Write a script to simulate 100 user requests and run it multiple times to measure response times.
* *Detailed Steps*:
  1. Create a script simulating a common user action (e.g., login).
  2. Run this script 100 times concurrently.
  3. Measure and record the response time for each request.
  4. Verify that the total time for 100 requests is less than 100 seconds.

**Example 2: Memory Usage Test**

* *Requirement*: The app must use less than 200 MB of memory on Android.
* *Test Method*: Generate the app’s executable and measure its memory usage on various Android versions.
* *Detailed Steps*:
  1. Build the app for different Android versions.
  2. Install the app on devices/emulators for each version.
  3. Use a memory profiling tool to measure the app’s memory usage.
  4. Ensure that memory usage is below 200 MB across all tested versions.