

Module 1 : Introduction aux Tests Logiciels (2h)

1. Pourquoi tester un logiciel ?

Objectifs des tests logiciels

- Déetecter les **bogues** avant la mise en production.
- Améliorer la **fiabilité** et la **qualité** du logiciel.
- Réduire les **coûts de maintenance** en identifiant les erreurs tôt dans le développement.
- Garantir la conformité aux **spécifications** et aux attentes des utilisateurs.

❗ **Exemple :** Un bug non détecté dans une application bancaire peut entraîner des pertes financières importantes.

2. Concepts clés

Vérification vs Validation

<i>Terme</i>	<i>Définition</i>	<i>Exemple</i>
<i>Vérification</i>	Vérifier que le logiciel est construit correctement par rapport aux spécifications.	Vérifier que la fonction <code>calculate_total()</code> retourne bien une somme correcte en interne.
<i>Validation</i>	Vérifier que le logiciel répond aux besoins de l'utilisateur final.	Tester que le montant affiché au client correspond bien à ses attentes.

❗ **Exemple en Python :** Vérification avec un test unitaire.

```
def calculate_total(prices):
    return sum(prices)

assert calculate_total([10, 20, 30]) == 60 # Vérification de la
fonctionnalité
```

Assurance Qualité (QA) vs Test Logiciel

- **Assurance Qualité (QA)** : Processus global qui garantit que le développement respecte des normes de qualité.
- **Test Logiciel** : Processus spécifique visant à **déetecter les défauts** avant le déploiement.

💡 **Exemple** : La QA inclut la mise en place d'une méthodologie Agile, alors que les tests incluent des tests unitaires et d'intégration.

Tests Manuels vs Tests Automatisés

Type de test	Avantages	Inconvénients
Manuels	Utile pour les tests exploratoires, tests UX/UI.	Lents, sujets aux erreurs humaines.
Automatisés	Rapides, répétables, intégrables dans un pipeline CI/CD.	Temps initial d'implémentation, maintenance des tests.

💡 **Exemple en Python** : Test manuel vs test automatisé avec pytest.

```
# Test manuel : L'utilisateur clique sur "Valider" et vérifie visuellement le résultat.

# Test automatisé avec pytest :
def test_calculate_total():
    assert calculate_total([5, 15, 10]) == 30 # Vérification automatique

# Commande pour exécuter le test :
# pytest test_script.py
```

3. Cycle de vie des tests logiciels

Modèles de développement et stratégies de test

- **V-Model** : Tests planifiés dès la phase de conception.
- **Agile** : Tests intégrés en continu avec **Test-Driven Development (TDD)**.
- **DevOps & CI/CD** : Automatisation des tests et exécution continue.

💡 **Exemple** : Workflow GitHub Actions exécutant des tests à chaque commit.

```
name: Run Tests
on: [push, pull_request]
jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
      - name: Install dependencies
        run: pip install pytest
      - name: Run tests
        run: pytest
```

4. Types de Tests Logiciels

Tests Statiques (sans exécution du code)

- **Revues de code** (peer review).
- **Analyse statique** avec flake8, pylint.
- **Détection des vulnérabilités** avec bandit.

💡 Exemple : Analyse statique avec flake8.

```
pip install flake8
flake8 my_script.py
```

Tests Dynamiques (avec exécution du code)

Type de test	Objectif	Exemple
Tests unitaires	Vérifier les fonctions individuellement.	pytest, unittest
Tests d'intégration	Tester l'interaction entre plusieurs modules.	API tests avec requests
Tests fonctionnels	Vérifier le comportement global du logiciel.	Selenium pour UI testing

💡 Exemple en Python : Test d'intégration avec requests (API test).

```
import requests

def test_api_response():
    response = requests.get("https://jsonplaceholder.typicode.com/posts/1")
    assert response.status_code == 200
```

Conclusion du Module 1

- Tester un logiciel réduit les risques et améliore la qualité.
- Différencier vérification vs validation et QA vs tests logiciels.
- Adapter les tests aux modèles de développement (V-Model, Agile, DevOps).
- Maîtriser les tests statiques et dynamiques pour assurer la robustesse du code.