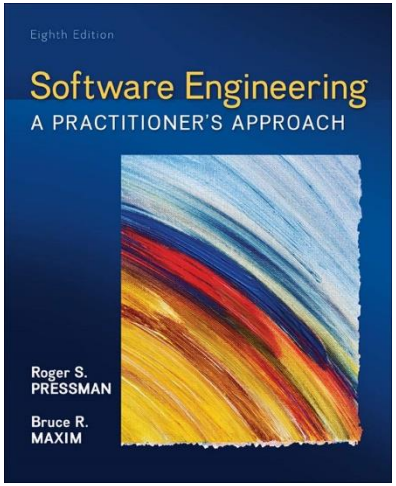
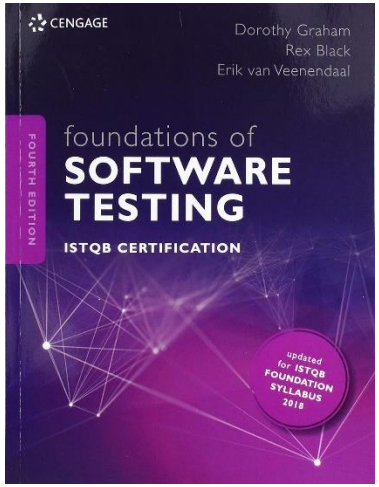
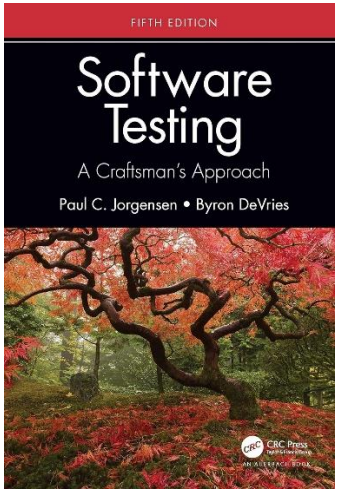
UNITE D'ENSEIGNEMENT	MANUEL DE REFERENCE	LIVRE 2	LIVRE 3	LIVRE 4
SE 333 Software Testing	 **Auteur :** Roger S. Pressman & Bruce R. Maxim **Éditeur :** McGraw-Hill Education **Edition :** 9e édition (2019/2020) **Langue :** Anglais **Pages :** env. 1408 pages (selon la version 9e édition) **ISBN-10 :** 1260548007 **ISBN-13 :** 978-1260548006	 **Auteurs :** Dorothy Graham, Rex Black & Erik van Veenendaal **Éditeur :** Cengage Learning EMEA **Edition :** 4e (ou 5e) édition **Langue :** Anglais **Pages :** env. 288 pages **ISBN-10 :** 1473764793 **ISBN-13 :** 978-1473764798	 **Auteur :** Paul C. Jorgensen (avec Byron DeVries pour certaines éditions) **Éditeur :** CRC Press / Taylor & Francis Group **Edition :** 5e édition (2021) **Langue :** Anglais **Pages :** env. 550 pages **ISBN-10 :** 1000391493 **ISBN-13 :** 978-1000391497	

A. Objectifs et description du cours

1. Description détaillée du cours

Le cours **Software Testing (SE 333)** est une unité d'enseignement avancée consacrée aux **processus, méthodes, techniques et outils permettant de garantir la qualité des logiciels tout au long de leur cycle de vie**.

Ce cours ne se limite pas à la phase de test logiciel en fin de développement ; il adopte une **approche globale de l'assurance qualité logicielle (Software Quality Assurance – SQA)**, intégrée **avant, pendant et après** le processus de développement logiciel.

Il couvre de manière approfondie :

- Les **fondements de la qualité logicielle** et les principes de l'ingénierie de la qualité ;
- Les **processus de test logiciel** aux différents niveaux :
  - Tests unitaires,
  - Tests d'intégration,
  - Tests système,
  - Tests d'acceptation ;
- Les **méthodes de test** :
  - Génération manuelle et automatique de cas de test,
  - Tests fonctionnels,
  - Tests boîte noire et boîte blanche ;
- Les techniques de **vérification statique** :
  - Revues de code,
  - Inspections,
  - walkthroughs ;
- Les techniques de **vérification dynamique** et d'analyse comportementale ;
- Les **métriques logicielles**, indicateurs de qualité et anti-patterns ;
- Les **modèles et cadres de référence en assurance qualité logicielle** ;
- Les pratiques de **gestion de configuration logicielle** et de contrôle des versions ;
- L'usage d'**outils professionnels** (environnements de développement, frameworks de test, outils de gestion du code).

Le cours intègre également des **études de cas réels**, des **travaux pratiques**, ainsi que des **activités de projet**, permettant aux étudiants de développer une compréhension **opérationnelle et professionnelle** des pratiques modernes de test et de qualité logicielle.

Il vise à former des étudiants capables non seulement de **détecter les défauts**, mais surtout de **concevoir des logiciels de qualité**, maintenables, fiables et conformes aux exigences fonctionnelles et non fonctionnelles.

2. Objectifs généraux du cours

À l'issue de ce cours, l'étudiant devra être capable de :

- Comprendre les **principes fondamentaux de la qualité logicielle** et de l'assurance qualité dans les projets logiciels modernes ;
- Expliquer le rôle stratégique du **test logiciel** dans la fiabilité et la maintenabilité des systèmes informatiques ;
- Analyser et appliquer des **processus structurés de test** à différents niveaux d'un système logiciel ;
- Utiliser des **métriques et modèles de qualité** pour évaluer un produit logiciel ;
- Identifier et corriger les **mauvaises pratiques de conception** (code smells, anti-patterns) ;
- Mettre en œuvre des **techniques de test manuel et automatisé** ;
- Appliquer des méthodes de **vérification statique et dynamique** pour améliorer la qualité du code ;
- Exploiter des **outils professionnels de développement, de test et de gestion de configuration** ;
- Intégrer la qualité logicielle comme une **dimension transversale du cycle de développement logiciel**.

Ces objectifs visent à doter l'étudiant de compétences à la fois **théoriques, méthodologiques et pratiques**, alignées sur les exigences du secteur professionnel du génie logiciel.

3. Positionnement stratégique du cours

Le cours **Software Testing and Quality Assurance (SE 433)** occupe une place **hautement stratégique** dans les formations orientées vers :

- Génie logiciel
- Développement d'applications (web, mobile, enterprise)
- Ingénierie des systèmes informatiques
- Assurance qualité logicielle et DevOps
- Cybersécurité et fiabilité des systèmes

Il constitue un **socle essentiel** pour comprendre que la qualité logicielle :

- Ne repose pas uniquement sur le test final,
- Mais sur une **démarche continue**, intégrée aux décisions de conception, de développement et de maintenance.

Ce cours prépare directement les étudiants à des rôles tels que :

- Software Quality Engineer
- Test Engineer / QA Engineer
- Software Engineer
- DevOps Engineer
- Technical Lead

Il contribue également à développer une **culture de la qualité**, indispensable dans les environnements professionnels modernes caractérisés par :

- L'intégration continue,
- Le déploiement continu,
- Les architectures complexes et évolutives,
- Les exigences élevées de fiabilité et de conformité.

B. Prérequis

CONDITION(S) PRÉALABLE(S) : IT 130.

C. Résultats d'apprentissage (Learning Outcomes)

À l'issue de l'unité d'enseignement **Software Testing and Quality Assurance (SE 433)**, l'étudiant devra démontrer qu'il a atteint les résultats d'apprentissage suivants :

Résultat d'apprentissage 1 : Compréhension des fondements de la qualité logicielle

À la fin du cours, l'étudiant sera capable d'expliquer de manière claire, structurée et argumentée les principes fondamentaux de la qualité logicielle et de l'assurance qualité dans le développement logiciel moderne.

Il devra être capable de :

- Définir les notions de qualité logicielle et d'assurance qualité logicielle ;
- Expliquer le rôle stratégique de la qualité dans le cycle de vie d'un logiciel ;
- Distinguer assurance qualité, contrôle qualité et test logiciel ;
- Analyser l'impact de la qualité logicielle sur la fiabilité, la maintenabilité et l'évolutivité des systèmes informatiques.

Résultat d'apprentissage 2 : Maîtrise des processus de test logiciel

À l'issue du cours, l'étudiant sera capable d'analyser et d'appliquer des processus de test logiciel structurés à différents niveaux d'un système logiciel.

Il devra être capable de :

- Expliquer les niveaux de test (unitaire, intégration, système, acceptation) ;
- Décrire les objectifs spécifiques de chaque niveau de test ;

- Analyser la place du test dans le cycle de développement logiciel ;
- Justifier le choix d'une stratégie de test en fonction du contexte d'un projet logiciel.

Résultat d'apprentissage 3 : Application des techniques de test logiciel
À la fin du cours, l'étudiant sera capable de mettre en œuvre et de comparer différentes techniques de test logiciel afin d'évaluer la conformité et la robustesse d'un système.
Il devra être capable de :

- Distinguer les tests fonctionnels et non fonctionnels ;
- Appliquer les principes des tests boîte noire et boîte blanche ;
- Concevoir des cas de test manuels et automatisés ;
- Analyser la couverture et l'efficacité des tests mis en place.

Résultat d'apprentissage 4 : Utilisation des méthodes de vérification statique et dynamique
À l'issue de cette unité d'enseignement, l'étudiant sera capable d'expliquer et d'utiliser des méthodes de vérification statique et dynamique pour améliorer la qualité du logiciel.
Il devra être capable de :

- Expliquer les principes de la vérification statique (revues, inspections, walkthroughs) ;
- Analyser les avantages et limites des techniques de vérification statique ;
- Expliquer le fonctionnement de la vérification dynamique ;
- Identifier les défauts détectables par chaque type de vérification.

Résultat d'apprentissage 5 : Analyse de la conception logicielle et des anti-patterns
À la fin du cours, l'étudiant sera capable d'identifier, analyser et corriger les mauvaises pratiques de conception logicielle susceptibles d'affecter la qualité du logiciel.
Il devra être capable de :

- Expliquer le concept de code smell et d'anti-pattern ;
- Analyser une conception logicielle du point de vue de la qualité ;
- Identifier les défauts de conception courants ;
- Proposer des améliorations basées sur des principes de bonne conception et de refactoring.

Résultat d'apprentissage 6 : Compréhension et usage des métriques de qualité logicielle

À l'issue du cours, l'étudiant sera capable d'utiliser des métriques et indicateurs pour évaluer la qualité d'un produit logiciel.
Il devra être capable de :

- Définir les principales métriques de qualité logicielle ;
- Expliquer le rôle des métriques dans l'assurance qualité ;
- Interpréter des résultats issus de métriques logicielles ;
- Analyser les limites et les risques d'une mauvaise interprétation des indicateurs de qualité.

Résultat d'apprentissage 7 : Utilisation des outils et environnements de test logiciel

À la fin du cours, l'étudiant sera capable d'exploiter des outils professionnels de développement, de test et de gestion de configuration dans un contexte d'assurance qualité logicielle.
Il devra être capable de :

- Utiliser des environnements de développement et de test ;
- Expliquer le rôle des outils de gestion de versions dans la qualité logicielle ;
- Intégrer les outils de test dans un processus de développement ;
- Analyser l'apport des outils dans l'automatisation et la fiabilité des tests.

Résultat d'apprentissage 8 : Intégration de l'assurance qualité dans un projet logiciel

À l'issue de l'unité d'enseignement, l'étudiant sera capable d'intégrer une démarche globale d'assurance qualité dans un projet logiciel.
Il devra être capable de :

- Analyser un projet logiciel sous l'angle de la qualité ;
- Proposer une approche structurée d'assurance qualité ;
- Justifier les choix méthodologiques et techniques liés à la qualité ;
- Évaluer les impacts organisationnels et techniques de l'assurance qualité sur un projet logiciel.

D. Critères d'évaluation (Assessment Criteria)

Critères liés au Résultat 1 : Compréhension des fondements de la qualité logicielle

Pour démontrer l'atteinte de ce résultat, l'étudiant devra être capable de :

- Expliquer de manière précise et cohérente les concepts fondamentaux de qualité logicielle et d'assurance qualité logicielle ;

- Distinguer clairement assurance qualité, contrôle qualité et test logiciel, en mettant en évidence leurs rôles respectifs ;
- Analyser l'impact de la qualité logicielle sur la fiabilité, la maintenabilité et l'évolutivité d'un système logiciel ;
- Démontrer une compréhension structurée de l'importance stratégique de la qualité dans le cycle de vie du logiciel.

Critères liés au Résultat 2 : Maîtrise des processus de test logiciel

L'étudiant devra démontrer qu'il est capable de :

- Décrire avec exactitude les différents niveaux de test logiciel (unitaire, intégration, système, acceptation) ;
- Expliquer les objectifs spécifiques et la portée de chaque niveau de test ;
- Analyser la place et le rôle des processus de test dans un cycle de développement logiciel ;
- Justifier de manière argumentée une stratégie de test adaptée à un contexte de projet donné.

Critères liés au Résultat 3 : Application des techniques de test logiciel

Pour satisfaire ce résultat, l'étudiant devra être capable de :

- Distinguer correctement les tests fonctionnels et non fonctionnels ;
- Expliquer et appliquer les principes des tests boîte noire et boîte blanche ;
- Concevoir des cas de test pertinents et cohérents avec les exigences du logiciel ;
- Analyser la couverture et l'efficacité des tests afin d'évaluer la qualité du produit logiciel.

Critères liés au Résultat 4 : Vérification statique et dynamique

L'étudiant devra démontrer sa capacité à :

- Expliquer clairement les principes de la vérification statique et de la vérification dynamique ;
- Décrire les techniques de revues, inspections et walkthroughs ;
- Analyser les avantages et limites de chaque méthode de vérification ;
- Identifier les types de défauts détectables par les approches statiques et dynamiques.

Critères liés au Résultat 5 : Analyse de la conception logicielle et des anti-patterns

Pour démontrer l'atteinte de ce résultat, l'étudiant devra être capable de :

- Identifier les mauvaises pratiques de conception logicielle (code smells, anti-patterns) ;

- Analyser les conséquences de ces pratiques sur la qualité du logiciel ;
- Expliquer les principes du refactoring comme moyen d'amélioration de la qualité ;
- Proposer des améliorations cohérentes basées sur des principes de bonne conception logicielle.

Critères liés au Résultat 6 : Utilisation des métriques de qualité logicielle

L'étudiant devra démontrer qu'il est capable de :

- Définir les principales métriques utilisées pour évaluer la qualité logicielle ;
- Expliquer l'utilité et les limites des métriques de qualité ;
- Interpréter correctement des indicateurs de qualité logicielle ;
- Analyser les risques liés à une mauvaise interprétation ou à une utilisation inappropriée des métriques.

Critères liés au Résultat 7 : Usage des outils et environnements de test

Pour satisfaire ce résultat, l'étudiant devra être capable de :

- Expliquer le rôle des outils de développement, de test et de gestion de configuration dans l'assurance qualité ;
- Démontrer une compréhension du lien entre automatisation des tests et qualité logicielle ;
- Analyser l'apport des outils professionnels dans la fiabilité et la répétabilité des tests ;
- Justifier l'intégration des outils dans un processus de développement logiciel structuré.

Critères liés au Résultat 8 : Intégration de l'assurance qualité dans un projet logiciel

L'étudiant devra démontrer sa capacité à :

- Analyser un projet logiciel sous l'angle de la qualité ;
- Proposer une démarche globale et cohérente d'assurance qualité logicielle ;
- Justifier les choix méthodologiques et techniques liés à la qualité ;
- Évaluer les impacts organisationnels et techniques de l'assurance qualité sur la réussite d'un projet logiciel.

E. Évaluation de l'apprentissage et pondération

1. Principes généraux de l'évaluation

L'évaluation de l'apprentissage dans le cadre du cours **Software Testing (SE 333)** vise à mesurer de manière rigoureuse et équilibrée la capacité de l'étudiant à :

- Comprendre les fondements de la qualité logicielle et de l'assurance qualité ;
- Analyser et appliquer des processus et techniques de test logiciel ;
- Utiliser des méthodes de vérification statique et dynamique ;
- Exploiter des métriques et outils de qualité logicielle ;
- Intégrer une démarche globale d'assurance qualité dans un projet logiciel.

Le dispositif d'évaluation repose sur une **approche critériée**, orientée vers l'atteinte effective des résultats d'apprentissage, et combine des évaluations théoriques, pratiques et intégratives.

2. Structure de l'évaluation proposée

a) Travaux pratiques et études de cas – 20 %

Les travaux pratiques et études de cas permettent d'évaluer la capacité de l'étudiant à appliquer concrètement les concepts de test et d'assurance qualité logicielle.
Ils portent notamment sur :

- L'analyse de cas réels de projets logiciels ;
- La conception de cas de test à différents niveaux ;
- L'identification de défauts, d'anti-patterns et de problèmes de qualité ;
- L'utilisation d'outils de test et de vérification.

Cette composante évalue principalement les résultats d'apprentissage liés à l'application pratique, à l'analyse critique et à la compréhension opérationnelle de la qualité logicielle.

b) Projet de qualité logicielle et rapport final – 30 %

Le projet constitue une évaluation intégrative majeure du cours.
Il consiste en l'analyse et la mise en œuvre d'une démarche d'assurance qualité appliquée à un système logiciel donné.
L'étudiant devra démontrer sa capacité à :

- Analyser un logiciel du point de vue de la qualité ;
- Définir une stratégie de test et d'assurance qualité cohérente ;
- Appliquer des techniques de test et de vérification ;
- Utiliser des métriques et outils adaptés ;
- Justifier ses choix méthodologiques et techniques.

Cette évaluation mesure la capacité de synthèse, d'analyse globale et d'intégration des résultats d'apprentissage.

c) Examen de mi-parcours (Semaine 6) – 20 %

L'examen de mi-parcours vise à évaluer l'acquisition des connaissances fondamentales et méthodologiques abordées durant la première moitié du cours.
Il porte notamment sur :

- Les principes de la qualité logicielle ;
- Les processus et niveaux de test ;
- Les techniques de test logiciel ;
- Les méthodes de vérification statique et dynamique ;
- Les métriques de base de la qualité logicielle.

Cet examen permet de vérifier la compréhension conceptuelle et la capacité d'analyse individuelle de l'étudiant.

d) Examen final (Semaine 12) – 30 %

L'examen final évalue la maîtrise globale et approfondie de l'ensemble des notions abordées dans le cours.

Il couvre :

- L'intégration de l'assurance qualité dans un projet logiciel ;
- L'analyse de la conception et des anti-patterns ;
- L'utilisation avancée des métriques et outils de qualité ;
- La mise en perspective stratégique de la qualité logicielle dans les projets modernes.

Cet examen vise à mesurer la capacité de l'étudiant à mobiliser l'ensemble des connaissances et compétences acquises.

3. Récapitulatif de la pondération

La pondération globale de l'évaluation de l'apprentissage est répartie comme suit :

- Travaux pratiques et études de cas : **20 %**
- Projet de qualité logicielle et rapport final : **30 %**
- Examen de mi-parcours : **20 %**
- Examen final : **30 %**

Total : 100 %

F. Plan et échéancier du cours sur 12 semaines

Semaine 1 — Introduction à la qualité logicielle et au test logiciel

Contenus abordés :

Introduction générale au cours. Présentation des notions de qualité logicielle, d'assurance qualité logicielle et de test logiciel. Distinction entre qualité perçue, qualité interne et qualité externe. Rôle du test dans le cycle de vie du logiciel.

Objectifs pédagogiques :

- Comprendre les concepts fondamentaux de qualité logicielle.
- Situer le test logiciel dans le cycle de développement.

- Identifier les enjeux liés à la non-qualité logicielle.

Semaine 2 — Métriques logicielles et indicateurs de qualité
Contenus abordés :
Introduction aux métriques de qualité logicielle. Mesure de la qualité du code, de la complexité, de la maintenabilité et de la fiabilité. Présentation des indicateurs et de leurs limites. Analyse des risques liés à une mauvaise interprétation des métriques.
Objectifs pédagogiques :
- Comprendre le rôle des métriques dans l'assurance qualité.
- Interpréter des indicateurs de qualité logicielle.
- Analyser de manière critique l'utilisation des métriques.

Semaine 3 — Mauvaises pratiques de conception et anti-patterns
Contenus abordés :
Étude des code smells et des anti-patterns courants. Identification des défauts de conception logicielle. Analyse de l'impact de ces pratiques sur la qualité, la maintenabilité et l'évolutivité du logiciel.
Objectifs pédagogiques :
- Identifier les mauvaises pratiques de conception.
- Analyser leurs conséquences sur la qualité du logiciel.
- Développer une capacité critique face à la conception logicielle.

Semaine 4 — Refactoring et amélioration de la qualité logicielle
Contenus abordés :
Principes et objectifs du refactoring. Techniques de refactoring courantes. Lien entre refactoring, qualité logicielle et tests. Bonnes pratiques pour améliorer la conception sans modifier le comportement fonctionnel.
Objectifs pédagogiques :
- Comprendre le rôle du refactoring dans la qualité logicielle.
- Appliquer des techniques de refactoring de base.
- Relier refactoring et stratégie de test.

Semaine 5 — Gestion de configuration logicielle et outils de développement
Contenus abordés :
Principes de la gestion de configuration logicielle. Gestion des versions, contrôle des changements et traçabilité. Présentation des outils de développement et de test (environnements, frameworks, outils de versioning).
Objectifs pédagogiques :

- Comprendre l'importance de la gestion de configuration.
- Expliquer le rôle des outils dans l'assurance qualité.
- Analyser l'impact de la gestion des versions sur la qualité logicielle.

Semaine 6 — Examen de mi-parcours
Contenus évalués :
- Fondements de la qualité logicielle
- Métriques et indicateurs de qualité
- Code smells et anti-patterns
- Refactoring
- Gestion de configuration logicielle

Objectif pédagogique :
Évaluer la maîtrise des concepts fondamentaux et des premières techniques d'assurance qualité logicielle.

Semaine 7 — Processus et niveaux de test logiciel
Contenus abordés :
Présentation des niveaux de test : tests unitaires, d'intégration, système et d'acceptation. Objectifs, responsabilités et positionnement de chaque niveau dans le cycle de développement.
Objectifs pédagogiques :
- Comprendre les différents niveaux de test logiciel.
- Expliquer leurs objectifs respectifs.
- Justifier une organisation cohérente des tests.

Semaine 8 — Techniques de test logiciel : boîte noire et boîte blanche
Contenus abordés :
Étude des techniques de test fonctionnel et structurel. Tests boîte noire et boîte blanche. Conception de cas de test. Analyse de la couverture de test.
Objectifs pédagogiques :
- Distinguer les différentes techniques de test.
- Concevoir des cas de test pertinents.
- Analyser l'efficacité des tests.

Semaine 9 — Génération manuelle et automatique de cas de test
Contenus abordés :
Méthodes de génération de cas de test. Tests manuels et automatisés. Introduction aux outils et frameworks de test automatique. Avantages et limites de l'automatisation.

Objectifs pédagogiques :

- Comprendre les principes de l'automatisation des tests.
- Comparer tests manuels et tests automatisés.
- Analyser les contextes favorables à l'automatisation.

Semaine 10 — Vérification statique du logiciel

Contenus abordés :

Méthodes de vérification statique : revues de code, inspections, walkthroughs. Détection précoce des défauts. Rôle de la vérification statique dans l'amélioration de la qualité logicielle.

Objectifs pédagogiques :

- Expliquer les principes de la vérification statique.
- Identifier les défauts détectables sans exécution du code.
- Analyser l'apport des inspections et revues.

Semaine 11 — Vérification dynamique et intégration de l'assurance qualité

Contenus abordés :

Principes de la vérification dynamique. Tests d'exécution, analyse des résultats. Intégration globale de l'assurance qualité dans un projet logiciel. Études de cas et synthèse des approches vues.

Objectifs pédagogiques :

- Comprendre la vérification dynamique.
- Analyser les résultats de tests exécutés.
- Intégrer une démarche complète d'assurance qualité.

Semaine 12 — Examen final

Contenus évalués :

- Processus et techniques de test logiciel
- Vérification statique et dynamique
- Métriques et outils de qualité
- Intégration de l'assurance qualité dans un projet logiciel
- Vision globale et stratégique de la qualité logicielle

Objectif pédagogique :

Évaluer la capacité de l'étudiant à mobiliser l'ensemble des connaissances et compétences acquises durant le cours.

G. Livres recommandés

1. Livre recommandé dans le syllabus de référence

Software Engineering: A Practitioner's Approach
Auteur : **Roger S. Pressman**
Édition : **7e édition** (ou éditions ultérieures disponibles)
Éditeur : **McGraw-Hill Education**
Cet ouvrage constitue une référence classique et structurante en génie logiciel.
Dans le cadre du cours **Software Testing and Quality Assurance**, il est particulièrement pertinent pour :

- La compréhension globale du cycle de vie du logiciel ;
- Les fondements de la qualité logicielle ;
- Les principes de test logiciel et d'assurance qualité ;
- L'intégration des activités de test dans les processus de développement.

Ce livre offre une vision méthodologique solide, indispensable pour replacer le test et la qualité logicielle dans une approche d'ingénierie logicielle complète.

2. Livres complémentaires récents recommandés

Les ouvrages suivants permettent d'approfondir les aspects **techniques, pratiques et modernes** du test logiciel et de l'assurance qualité.

Foundations of Software Testing – ISTQB Certification
Auteurs : **Rex Black, Erik van Veenendaal, Dorothy Graham**
Édition : **4e édition (2018+)**
Éditeur : **Cengage Learning**
Ce livre est aligné sur les standards internationaux de test logiciel (ISTQB).
Il permet de renforcer :

- La compréhension structurée des processus de test ;
- Les niveaux et techniques de test ;
- Le vocabulaire professionnel du test logiciel ;
- La rigueur méthodologique en assurance qualité.

Il est particulièrement adapté à une approche académique et professionnelle du test logiciel.

Software Testing: A Craftsman's Approach
Auteur : **Paul C. Jorgensen**
Édition : **5e édition**
Éditeur : **CRC Press / Taylor & Francis**

Cet ouvrage met l'accent sur l'aspect pratique et rigoureux du test logiciel.
Il est recommandé pour :

- L'étude approfondie des techniques de test boîte noire et boîte blanche ;
- La conception de cas de test efficaces ;
- L'analyse de la couverture de test ;
- La compréhension des enjeux techniques du test.

Effective Software Testing: A Developer's Guide
Auteur : **Mauricio Aniche**
Édition : **2022**
Éditeur : **Manning Publications**
Ce livre récent est particulièrement pertinent pour les environnements modernes de développement logiciel.
Il permet d'approfondir :

- Les tests unitaires et d'intégration ;
- La testabilité du code ;
- Les bonnes pratiques de test orientées développeur ;
- L'intégration des tests dans les pipelines de développement.

Refactoring: Improving the Design of Existing Code
Auteur : **Martin Fowler**
Édition : **2e édition (2019)**
Éditeur : **Addison-Wesley**
Cet ouvrage est une référence incontournable pour la compréhension :

- Des code smells ;
- Des anti-patterns ;
- Des techniques de refactoring ;
- Du lien entre qualité du code et test logiciel.

Il complète parfaitement les parties du cours relatives à la conception logicielle et à l'amélioration continue de la qualité.

3. Synthèse des livres recommandés

Les ouvrages recommandés pour ce cours couvrent de manière cohérente :

- Les fondements du génie logiciel et de la qualité ;
- Les processus et techniques de test logiciel ;
- Les bonnes pratiques professionnelles modernes ;
- L'analyse, l'amélioration et la maintenabilité du code.

Ils offrent à l'étudiant une combinaison équilibrée entre **approche académique**, **références normatives** et **pratiques industrielles actuelles**.

H. Correspondance contenus ↔ chapitres des livres

Abréviations utilisées

Pour faciliter la lecture, les abréviations suivantes sont utilisées :

- **[PRESS]** : *Software Engineering: A Practitioner's Approach* – Roger S. Pressman
- **[ISTQB]** : *Foundations of Software Testing* – Graham, Black, van Veenendaal
- **[JOR]** : *Software Testing: A Craftsman's Approach* – Paul C. Jorgensen
- **[ANICHE]** : *Effective Software Testing* – Mauricio Aniche
- **[FOWLER]** : *Refactoring: Improving the Design of Existing Code* – Martin Fowler

Semaine 1 — Introduction à la qualité logicielle et au test logiciel

Contenu du cours :

Qualité logicielle, assurance qualité, rôle du test dans le cycle de vie du logiciel.

Correspondance livres :

- **[PRESS]** : Chapitre 1 – The Nature of Software
- **[PRESS]** : Chapitre 8 – Software Quality Assurance
- **[ISTQB]** : Chapitre 1 – Fundamentals of Testing

Semaine 2 — Métriques logicielles et indicateurs de qualité

Contenu du cours :

Mesure de la qualité, métriques de code, complexité, maintenabilité, interprétation des indicateurs.

Correspondance livres :

- **[PRESS]** : Chapitre 23 – Software Metrics
- **[JOR]** : Chapitre 2 – Test Metrics and Measurement
- **[ISTQB]** : Chapitre 2 – Testing Throughout the Software Lifecycle

Semaine 3 — Mauvaises pratiques de conception et anti-patterns

Contenu du cours :

Code smells, anti-patterns, impact sur la qualité logicielle.

Correspondance livres :

- **[FOWLER]** : Chapitre 3 – Bad Smells in Code
- **[PRESS]** : Chapitre 12 – Design Concepts
- **[ANICHE]** : Chapitre 1 – Testability and Code Quality

Semaine 4 — Refactoring et amélioration de la qualité logicielle
Contenu du cours :
Principes et techniques de refactoring, lien entre refactoring et tests.
Correspondance livres :

- **[FOWLER]** : Chapitres 1 et 2 – Refactoring Principles
- **[ANICHE]** : Chapitre 6 – Refactoring with Tests
- **[PRESS]** : Chapitre 14 – Design for Quality

Semaine 5 — Gestion de configuration logicielle et outils
Contenu du cours :
Gestion des versions, configuration logicielle, outils de développement et de test.
Correspondance livres :

- **[PRESS]** : Chapitre 22 – Software Configuration Management
- **[ISTQB]** : Chapitre 5 – Test Management
- **[ANICHE]** : Chapitre 9 – Tooling and Test Automation

Semaine 6 — Examen de mi-parcours
Chapitres à réviser :

- **[PRESS]** : Chapitres 1, 8, 12, 14, 22, 23
- **[ISTQB]** : Chapitres 1 à 3
- **[FOWLER]** : Chapitres 1 à 3

Semaine 7 — Processus et niveaux de test logiciel
Contenu du cours :
Tests unitaires, intégration, système et acceptation.
Correspondance livres :

- **[ISTQB]** : Chapitre 2 – Testing Levels
- **[JOR]** : Chapitre 4 – Unit Testing
- **[PRESS]** : Chapitre 17 – Software Testing Strategies

Semaine 8 — Techniques de test : boîte noire et boîte blanche
Contenu du cours :
Tests fonctionnels et structurels, conception de cas de test, couverture.
Correspondance livres :

- **[JOR]** : Chapitres 5 et 6 – Black-Box and White-Box Testing
- **[ISTQB]** : Chapitre 4 – Test Techniques
- **[PRESS]** : Chapitre 18 – Testing Techniques

Semaine 9 — Génération manuelle et automatique de cas de test
Contenu du cours :
Automatisation des tests, frameworks de test, limites et bénéfices.
Correspondance livres :

- **[ANICHE]** : Chapitres 3 et 4 – Automated Testing
- **[JOR]** : Chapitre 7 – Test Automation
- **[ISTQB]** : Chapitre 6 – Tool Support for Testing

Semaine 10 — Vérification statique du logiciel
Contenu du cours :
Revues de code, inspections, walkthroughs.
Correspondance livres :

- **[PRESS]** : Chapitre 15 – Reviews and Inspections
- **[ISTQB]** : Chapitre 3 – Static Testing
- **[JOR]** : Chapitre 9 – Static Analysis

Semaine 11 — Vérification dynamique et intégration de l'assurance qualité
Contenu du cours :
Tests d'exécution, analyse des résultats, intégration globale de l'assurance qualité.
Correspondance livres :

- **[PRESS]** : Chapitres 17 et 18 – Testing and Quality Integration
- **[ANICHE]** : Chapitre 8 – Test Strategy Integration
- **[ISTQB]** : Chapitre 5 – Test Planning and Control

Semaine 12 — Examen final
Chapitres à maîtriser :

- **[PRESS]** : Chapitres 8, 12, 14, 15, 17, 18, 22, 23
- **[ISTQB]** : Ensemble de l'ouvrage
- **[JOR]** : Chapitres 4 à 9
- **[ANICHE]** : Chapitres clés sur tests automatisés et intégration
- **[FOWLER]** : Chapitres sur refactoring et code smells