Maîtrise des Grands Projets Informatiques

Plan du cours

- **Introduction aux Concepts de Base de l'Organisation des Entreprises**
- Cycle de vie et cycle de développement logiciel
- Activités de développement logiciel
- Avant-projet
- Collecte des besoins : cas d'utilisation, user's story
- Analyse Conception
- Codage
- Intégration
- Déploiement et production : plateforme de déploiement continu, surveillance des applications, conteneurisation en production (docker)
- Activités de gestion de projets

Introduction aux Concepts de Base de l'Organisation des Entreprises

Définition d’organisation

Une organisation est une entité structurée composée de personnes qui travaillent ensemble pour atteindre des objectifs communs. Elle peut être formelle ou informelle, et ses membres partagent des ressources et des responsabilités pour accomplir des tâches spécifiques.

Exemple d’organisation

Entreprises : Entités commerciales qui produisent des biens ou des services pour générer des profits. Exemples : Apple, Google, Tesla.

ONG (Organisations Non Gouvernementales) : Entités à but non lucratif qui travaillent pour des causes sociales, environnementales ou humanitaires. Exemples : Médecins Sans Frontières, Greenpeace.

Gouvernements : Entités publiques qui gèrent les affaires de l'État et fournissent des services publics. Exemples : Gouvernement français, Gouvernement américain.

Éducatives : Institutions qui fournissent des services éducatifs. Exemples : Universités, écoles.

Structure Organisationnelle des Entreprises

La structure organisationnelle d'une entreprise définit la manière dont les tâches sont réparties, les responsabilités sont assignées, et les informations circulent au sein de l'organisation. Il existe plusieurs types de structures organisationnelles, chacune ayant ses avantages et inconvénients.

Structure Hiérarchique (ou Pyramidale)

Description: Cette structure est caractérisée par une chaîne de commandement claire et verticale. Chaque employé a un supérieur direct.

Exemple: Une grande entreprise de fabrication où les décisions sont prises au sommet et transmises vers le bas.

Avantages: Clarté des rôles et responsabilités, facilité de contrôle.

Inconvénients: Lenteur dans la prise de décision, manque de flexibilité.

Structure Fonctionnelle

Description: Les employés sont regroupés par fonctions spécifiques (marketing, finance, RH, etc.).

Exemple: Une entreprise de services financiers où chaque département est spécialisé dans une fonction spécifique.

Avantages: Expertise spécialisée, efficacité opérationnelle.

Inconvénients: Silos organisationnels, manque de coordination entre départements.

Structure Divisionnelle

Description: L'entreprise est divisée en unités autonomes, souvent basées sur des produits, des marchés ou des zones géographiques.

Exemple: Unilever. Unilever est une multinationale avec des divisions pour chaque région géographique et chaque catégorie de produits (aliments, soins personnels, etc.).

Avantages: Flexibilité, réactivité aux besoins spécifiques des marchés.

Inconvénients: Duplication des ressources, coûts élevés.

Structure Matricielle

Description: Combine les structures fonctionnelle et divisionnelle. Les employés ont deux supérieurs hiérarchiques : un chef de projet et un chef de fonction.

Exemple: IBM. IBM utilise une structure matricielle pour ses projets de développement de logiciels, où les développeurs travaillent sur plusieurs projets simultanément.

Avantages: Flexibilité, utilisation optimale des ressources.

Inconvénients: Complexité, conflits potentiels entre les chefs de projet et les chefs de fonction.

Rôles et Responsabilités dans les Projets Logiciels

Le modèle RACI est un outil de gestion de projet utilisé pour clarifier les rôles et les responsabilités des membres de l'équipe. RACI est un acronyme qui signifie :

- **R**esponsible (Réalisateur)
- **A**ccountable (Approbateur)
- **C**onsulted (Consulté)
- **I**nformed (Informé)

Informed (Informé)

Les personnes qui doivent être informées après la réalisation de la tâche. Elles n'ont pas besoin d'être consultées avant la réalisation de la tâche, mais elles doivent être tenues au courant des résultats.

Exemple : Le responsable de la documentation peut être informé une fois que la fonctionnalité est développée pour qu'il puisse mettre à jour la documentation.

Consulted (Consulté)

Définition : Les personnes dont les avis sont recherchés avant de prendre une décision ou de réaliser la tâche. Elles fournissent des informations ou des conseils.

Exemple : L'architecte logiciel peut être consulté pour s'assurer que la nouvelle fonctionnalité respecte les standards architecturaux.

Responsible (Réalisateur)

Définition : La personne ou les personnes qui effectuent le travail pour accomplir la tâche. Elles sont responsables de l'exécution de la tâche.

Exemple : Dans une tâche de développement de fonctionnalité, le développeur est responsable de coder la fonctionnalité.

Accountable (Approbateur)

Définition : La personne qui est ultimement responsable de la tâche et qui doit rendre des comptes sur son achèvement. Il ne peut y avoir qu'une seule personne comptable pour une tâche donnée.

Exemple : Le chef de projet peut être approbateur pour s'assurer que la fonctionnalité est livrée dans les délais et selon les spécifications.

Avantages du Modèle RACI

Clarification des Rôles : Évite les malentendus et les conflits en clarifiant qui fait quoi.

Responsabilité : Assure que chaque tâche a une personne responsable et comptable.

Communication : Améliore la communication en identifiant qui doit être consulté et informé.

Efficacité : Augmente l'efficacité en évitant les redondances et en assurant que toutes les parties prenantes sont impliquées de manière appropriée.

Utilisation du RACI

Tâche	Chef de Projet	Développeur	Architecte	Testeur	Documentateur
Développer Fonctionnalité	A	R	C	I	I
Tester Fonctionnalité	A	I	I	R	I
Documenter Fonctionnalité	A	I	I	I	R
Déployer fonctionnalité	A	I	C	I	I

Culture d'Entreprise

La culture d'entreprise fait référence aux valeurs, croyances, comportements, normes et pratiques partagées au sein d'une organisation. Elle influence la manière dont les employés interagissent entre eux et avec les parties prenantes externes, ainsi que la manière dont les décisions sont prises et les objectifs sont atteints.

Culture d'Entreprise: Composants

- Valeurs : Les principes fondamentaux qui guident les actions et les décisions de l'entreprise.
- Croyances : Les convictions partagées sur ce qui est important et ce qui est attendu.
- Comportements : Les manières d'agir et de se comporter au sein de l'organisation.
- Normes : Les règles non écrites qui régissent les interactions et les pratiques.
- Pratiques : Les méthodes et les processus utilisés pour accomplir le travail.

Gestion du changement

La gestion du changement est le processus de planification, de mise en œuvre et de gestion des transitions organisationnelles pour minimiser les résistances et maximiser l'adoption des nouvelles pratiques, technologies ou structures. Elle vise à assurer que les changements sont acceptés et intégrés de manière efficace par les employés et les parties prenantes.

Gestion du changement: Composants

- Planification : Identifier les objectifs du changement, les parties prenantes concernées et les impacts potentiels.
- Communication : Informer et engager les parties prenantes tout au long du processus de changement.
- Formation : Fournir les compétences et les connaissances nécessaires pour adopter les nouvelles pratiques.
- Support : Offrir un soutien continu pour aider les employés à s'adapter aux changements.
- Évaluation : Mesurer l'efficacité du changement et apporter des ajustements si nécessaire.

Conclusion

La culture d'entreprise joue un rôle crucial dans la gestion du changement. Une culture forte et positive peut faciliter l'adoption des changements en alignant les valeurs et les comportements des employés avec les objectifs du changement. En revanche, une culture rigide ou résistante au changement peut rendre la gestion du changement plus difficile.

Plan du cours

- Introduction aux Concepts de Base de l'Organisation des Entreprises
- **Cycle de vie et cycle de développement logiciel**
- Activités de développement logiciel
- Avant-projet
- Collecte des besoins : cas d'utilisation, user's story
- Analyse Conception
- Codage
- Intégration
- Déploiement et production : plateforme de déploiement continu, surveillance des applications, conteneurisation en production (docker)
- Activités de gestion de projets

Cycle de vie et cycle de développement logiciel

Estimation et chiffrage

L’estimation ou le chiffrage est le processus de prévision des coûts, des ressources et du temps nécessaire pour développer un logiciel ou une fonctionnalité.

Cette étape est primordiale puisqu’elle permet de définir les attentes et gérer les risques.

Estimation: Estimation par analogie

On compare un projet avec d’autres projets similaire pour pouvoir estimer les coûts et les délais. C’est la méthode la plus simple mais peut manquer de précision si les projets ne sont pas similaires.

Estimation: Estimation par décomposition

On découpe le projet en tâches plus petites et on estime chaque tâche avant d’agréger les estimations pour obtenir le résultat final plus précis. Cette méthode peut être longue et complexe.

Estimation: Estimation Paramétrique

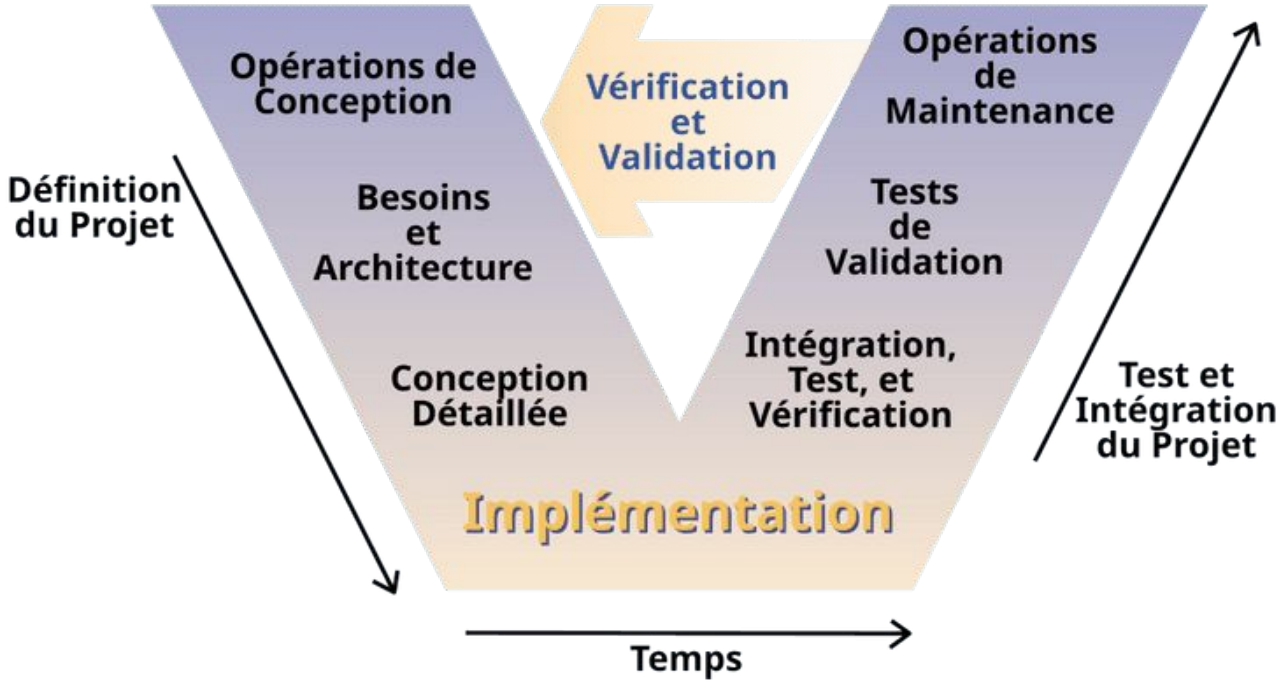
On utilise des algorithmes mathématiques pour évaluer le coût des et les délais à partir de paramètres du projet i.e nombre de fonctionnalités, nombre de fichiers… L’avantage de cette méthode est qu’elle est précise, mais elle nécessite des donnée précises.

Estimation: Estimation par points

Évaluer la taille des fonctionnalités et attribuer des points en fonction de la complexité. Cette méthode est flexible et permet de s’adapter aux changements fréquents, et est subjective.

Cycle en V

Le cycle en V (ou waterfall) est un modèle de développement logiciel qui combine les phases de développement et de test. Il est appelé ainsi parce que les phases de développement et de test forment une structure en forme de V lorsqu'elles sont représentées graphiquement.


Opérations de Conception
Vérification et Validation
Opérations de Maintenance
Définition du Projet
Besoins et Architecture
Tests de Validation
Conception Détaillée
Intégration, Test, et Vérification
Test et Intégration du Projet
Implémentation
Temps

Cycle en V: Phases 1 / 3

Exigences : les exigences font l'objet d'une expression des besoins. Le cas échéant, une étude de faisabilité peut être conduite avant d'engager les travaux;

Analyse : il s'agit, à partir de l'expression de besoin, d'établir le cahier des charges fonctionnel ou les spécifications fonctionnelles;

Conception générale, aussi appelé conception architecturale ou conception préliminaire: il s'agit de concevoir le système qui doit répondre aux exigences et de définir son architecture, et en particulier les différents composants nécessaires;

Cycle en V: Phases 2 / 3

Conception détaillée: il s'agit de concevoir chaque composant, et la manière dont ils contribuent à la réponse aux besoins;

Mise en œuvre: il s'agit de réaliser chaque composant nécessaire. Pour les composants et systèmes logiciels, l'activité est essentiellement le codage;

Test unitaire: il s'agit de vérifier le bon fonctionnement et la conformité de chaque composant à sa conception détaillée;

Cycle en V: Phases 3 / 3

Intégration et test d'intégration: il s'agit d'assembler le système à partir de tous ses composants, et de vérifier que le système dans son ensemble fonctionne conformément à sa conception générale;

Test système (anciennement « tests fonctionnels ») : vérification que le système est conforme aux exigences;

Test d'acceptation (également appelés « recette » dans le contexte de la sous-traitance) : validation du système par rapport à sa conformité aux besoins exprimés.

Cycle en V: Avantages

- Structure claire et bien définie.
- Facilite la traçabilité des exigences.
- Les tests sont planifiés dès le début.

Cycle en V: Inconvénients

- Peu flexible face aux changements.
- Les tests sont souvent réalisés tardivement dans le cycle.
- Peut être coûteux et long.

Processus Agiles

Les processus agiles sont des méthodologies de développement logiciel qui mettent l'accent sur la flexibilité, la collaboration et la livraison rapide de valeur.

Méthodes

Dans les années 1990 plusieurs nouvelles méthodes de développement voient le jour comme par exemple, Scrum, l’extreme programming (XP) ou le Crystal.

Agile Manifesto

En 2001, 17 développeurs de différents horizons se regroupent pour discuter des méthodes de développement. Suite à ces discussions, mènent à la création d'un manifeste agile regroupant un ensemble de pratiques qui permettent de développer des applications dans les meilleures conditions.

12 principes ont également été déposés pour guider la mise en oeuvre de l'agile.

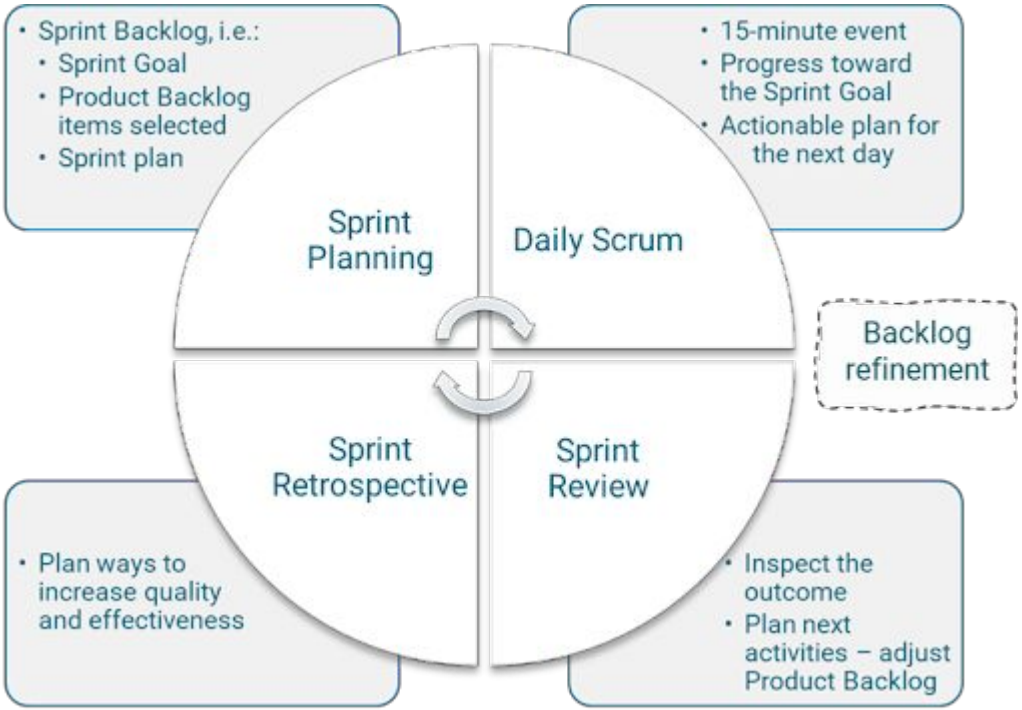
Processus Agiles: Scrum

L’objectif de Scrum est de livrer des incréments de produit fonctionnels à intervalles réguliers. Chacun de ces interval est appelé un Sprint.

Cette méthode de gestion de projet s’accompagne de plusieurs cérémonies qui ont lieu à plusieurs phases du projet.

Des rôles particuliers sont nécessaires, le product owner ainsi que le scrum master ainsi que l’équipe de développement devront effectuer des actions avant, pendant et après les différentes cérémonies.

Scrum


• Sprint Backlog, i.e.:
• Sprint Goal
• Product Backlog items selected
• Sprint plan
Sprint Planning
Daily Scrum
• 15-minute event
• Progress toward the Sprint Goal
• Actionable plan for the next day
Backlog refinement
Sprint Retrospective
Sprint Review
• Plan ways to increase quality and effectiveness
• Inspect the outcome
• Plan next activities – adjust Product Backlog

Scrum: Artefact

En scrum il y a plusieurs artefacts:

- Le Product Backlog: Il s’agit de la liste des fonctionnalités à développer.
- Le Sprint Backlog: C’est la liste des tâches qui seront effectuées pendant un Sprint
- L’incrément produit: C’est la version fonctionnelle du produit à la fin de chaque sprint.

Scrum: Cérémonies

- Sprint Planning
- Daily Standup
- Sprint Review
- Sprint Retrospective

Scrum: Backlog Refinement

Le backlog refinement aussi appellé backlog grooming consiste à s’assurer que chaque élément du backlog est bien compris par toute l’équipe.

On découpera également les plus grosses tâches en plus petites tâches en fonction de leur complexité.

Chaque tâche sera estimée par l’équipe.

La priorisation de ces tâches est ensuite effectuée par le product owner.

Enfin, toutes les dépendances de ses tâches sont identifiées afin de s’assurer qu’elles soient prêtes à être sélectionnées pour le développement.

Scrum: Sprint Planning

L’objectif du sprint planning sont les suivants: définir les objectifs du sprint.

Ceci a s’effectue en choisissant les éléments du backlog qui seront traités pendant le sprint.

Enfin, la planification des tâches est effectuée, les tâches sont redécoupées en plus petites tâches si le besoin s’en fait ressentir.

Scrum: Sprint Retrospective

La Sprint Retrospective est une cérémonie clé dans la méthodologie Scrum, qui se déroule à la fin de chaque sprint.

Elle vise à améliorer continuellement le processus de développement en permettant à l'équipe de réfléchir sur le sprint écoulé, d'identifier les points forts et les points à améliorer, et de définir des actions concrètes pour le prochain sprint.

Scrum: Sprint Retrospective

Il existe plusieurs templates possible pour les retrospectives.

Celles-ci se concentrent souvent sur ces points:

- Évaluer ce qu'il s'est bien passé
- Évaluer ce qui aurait pu mieux se passer / être évité
- Anticiper les risques du prochain sprint

Scrum: Daily Scrum

Le Daily Scrum, également connu sous le nom de Daily Stand-up, est une réunion quotidienne courte et structurée dans la méthodologie Scrum. Elle vise à synchroniser l'équipe de développement, à suivre l'avancement du travail et à identifier les obstacles qui pourraient entraver la progression du sprint

Scrum: Daily Scrum Format

La réunion ne doit pas dépasser 15 minutes.

Chaque participant répond à ces questions:

1. Qu'ai-je accompli depuis la dernière réunion ?
2. Qu'est-ce que je prévois de faire aujourd'hui ?
3. Quels obstacles rencontre-je ?

Réunit les équipes de développement, le product owner et le scrum master.

Scrum: Sprint Review

La Sprint Review, également connue sous le nom de Sprint Demo. Elle se déroule à la fin de chaque sprint et vise à présenter les fonctionnalités développées pendant le sprint aux parties prenantes, à recueillir leurs retours et à ajuster le backlog du produit en conséquence.

Scrum: Sprint Review déroulé

- Présentation : L'équipe de développement présente les fonctionnalités développées pendant le sprint.
- Démonstration : Une démonstration en direct des fonctionnalités est souvent réalisée pour montrer leur fonctionnement.
- Feedback : Les parties prenantes fournissent des retours sur les fonctionnalités présentées.
- Discussion : Une discussion ouverte sur les retours et les ajustements nécessaires.

Scrum: Product Owner

Le PO est responsable de la gestion du produit et de la maximisation de sa valeur pour l'entreprise et les utilisateurs finaux.

Il a plusieurs responsabilités:

- Définir la vision du produit
- Gérer le backlog du produit
- Collaborer avec les parties prenantes
- Participer aux cérémonies Scrum
- Prendre des décisions

Scrum: Scrum Master

Le Scrum Master est responsable de faciliter le processus Scrum, d'éliminer les obstacles pour l'équipe de développement et de promouvoir l'auto-organisation et l'amélioration continue.

Il a les responsabilités suivantes:

- Facilitation des Cérémonies Scrum
- Élimination des Obstacles
- Promotion de l'Auto-Organisation
- Amélioration Continue
- Formation et Coaching
- Communication et Collaboration

Processus Agiles: Kanban

Kanban est une méthodologie de gestion de projet et de flux de travail qui met l'accent sur la visualisation, la limitation du travail en cours (WIP) et l'amélioration continue.

Originaire du système de production Toyota, Kanban a été adapté pour la gestion de projets logiciels et d'autres types de travaux.

Kanban: Tableau

Exemple de tableau Kanban:

To Do	In progress	To Test	Done
User Auth			
	Update User		
		Delete User	
			Add User

Kanban: Décomposition

- Colonnes: Représentent les différentes étapes du processus.
- Cartes: Chaque carte représente une tâche ou un élément de travail.
- Limite du nombre de tâches parallèles.

TDD

Cette méthode de développement a été redécouverte par Kent Beck:

La description originale du TDD se trouvait dans un ancien livre sur la programmation: Il disait que vous prenez la bande d'entrée, tapez manuellement la bande de sortie que vous attendez, puis programmez jusqu'à ce que la sortie réelle corresponde à la sortie attendue.

Après avoir écrit le premier framework xUnit en Smalltalk, je me suis souvenu avoir lu cela et j'ai essayé. C'était l'origine du TDD pour moi. Quand je décris le TDD à des programmeurs plus âgés, j'entends souvent : "Bien sûr. Comment pourrait-on programmer autrement ?" C'est pourquoi je parle de mon rôle comme étant celui de "redécouvrir" le TDD.

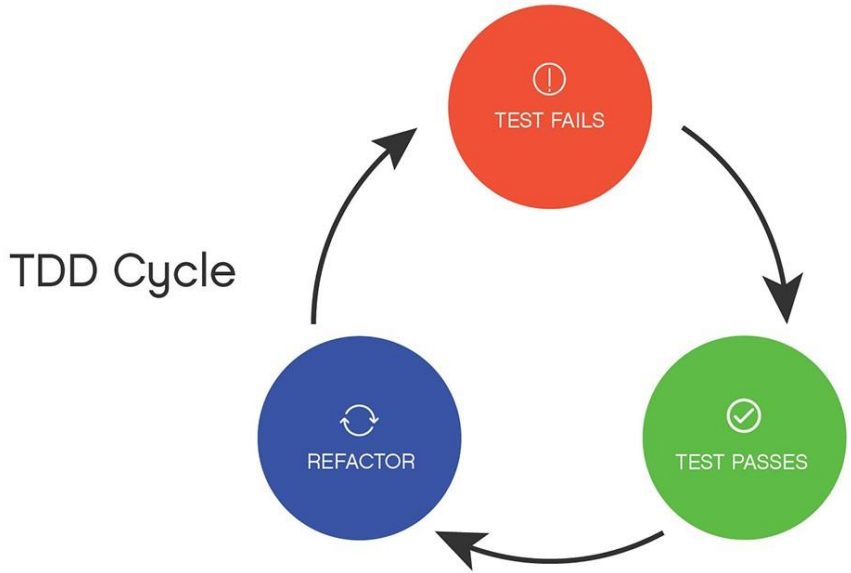
TDD: Les 3 lois

- Loi n°1: Écrivez un test qui échoue avant d’écrire le code de production correspondant.
- Loi n°2: Écrivez une seule assertion à la fois, qui fait échouer le test ou qui échoue à la compilation.
- Loi n°3: Écrivez le minimum de code de production pour que l'assertion du test actuellement en échec soit satisfaite.

TDD: Les étapes

1. Lister les scénarios pour la nouvelle fonctionnalité
2. Écrire un test pour un élément de la liste
3. Exécuter tous les tests. Le nouveau test devrait échouer – pour des raisons attendues
4. Écrire le code le plus simple qui passe le nouveau test
5. Tous les tests devraient maintenant passer
6. Refactoriser si nécessaire tout en s'assurant que tous les tests continuent de passer

TDD: Red Green Refactor



Software Craftmanship

Le Software Craftmanship s’appuie sur différentes méthodes de développement et s’appuie sur les principes de son manifesto.