Compte Rendu n°2 du Génie Logiciel

Nom : DERMOUCHE

Prénom : Mohammed Ryad

Groupe : TD2

1)-Première partie :

Le **SWEBOK** (Software Engineering Body of Knowledge) est un guide de référence exhaustif qui rassemble les connaissances, pratiques, et méthodologies clés en ingénierie logicielle. Publié par l'IEEE, il couvre 15 domaines de connaissance essentiels, dont l'un des premiers est **l'exigence logicielle**. Elle précise ce qu'un logiciel doit accomplir pour répondre aux attentes des utilisateurs ou des parties prenantes.

Les exigences logicielles sont organisées en plusieurs sous-domaines correspondant aux étapes cruciales de leur gestion : les fondamentaux, l'élicitation, l'analyse, la validation, la spécification et la gestion des exigences. Chacun de ces sous-domaines contribue à assurer que les besoins sont correctement identifiés et mis en œuvre dans le produit final.

Les fondamentaux d'exigence permettent de structurer le processus de développement en fonction des attentes du projet :
-**Exigences fonctionnelles :** Elles définissent les fonctionnalités que le logiciel doit offrir.
-**Exigences non-fonctionnelles** : Elles concernent la qualité du logiciel, telles que la performance ou la sécurité.

Partie Individuelle :

Dans le cadre de mon projet précédent, il est important de noter que les notions d'exigences, telles qu'elles sont décrites dans le **SWEBOK** , n'ont pas été respectées. À l'époque, mon équipe et moi n'avions pas conscience de l'importance cruciale des exigences dans la gestion et la réussite d'un projet logiciel. Nous ne savions pas non plus comment formaliser ces exigences de manière à guider le développement de façon efficace.
Toutefois, il manque une véritable discussion approfondie sur les **exigences** du projet en relation avec ce que le SWEBOK recommande.

Exigences fonctionnelles :

-L'application doit permettre de visualiser un graphe et de colorier chaque sommet en respectant la contrainte de coloration.
-L'algorithme doit attribuer une couleur à chaque sommet tout en s'assurant que deux sommets adjacents ne partagent jamais la même couleur.

Exigences non fonctionnelles :

- **Performance** : L'application doit être capable de gérer des graphes complexes avec un temps de réponse inférieur à un seuil donné.
- **Fiabilité** : L'application doit être testée à travers une suite de tests unitaires couvrant au moins 80 % du code.
- **Utilisabilité** : L'interface doit être suffisamment intuitive pour qu'un utilisateur puisse visualiser et interagir avec les graphes sans formation préalable

Exigences de processus :

-Le projet doit respecter un processus de développement itératif, avec des revues hebdomadaires pour ajuster la répartition des tâches et évaluer l'avancement.

-Des points de contrôle doivent être établis toutes les deux semaines pour assurer que le développement progresse conformément au plan.

En analysant notre projet à la lumière des principes du SWEBOK, nous constatons que de nombreuses difficultés auraient pu être évitées avec une meilleure formalisation des exigences, notamment en clarifiant les exigences non fonctionnelles et en prenant en compte les propriétés émergentes plus tôt dans le projet.

2)-Deuxième partie :

Langages de programmations : Python, C, JAVA ?
On a utilisé **Python** comme langage de programmation pour ce projet.

Liste de contrôle pour Python:

1-Syntaxe :

Commentaires : # pour les commentaires sur une seule ligne et ''' ou """ pour les commentaires sur plusieurs lignes.
Variables : Pas besoin de déclarer le type, par exemple x = 5 ou nom = "Alice".
Opérateurs : +, -, *, /, ….

2-Indentation :

Espaces : On 4 espaces pour l'indentation.
Blocs de code : L'indentation définit les blocs de code (pas de {} ou begin...end).
Règles de Typage :

- **Dynamique** : Les types sont déterminés au moment de l'exécution.
- **Fortement Typé** : Les types ne sont pas implicitement convertis, par exemple, 1 + "2" génère une erreur.

3-Structure

- **Fonctions** : avec def, par exemple def ma_fonction():
- **Classes** : class, par exemple class MonObjet
- **Modules** : avec import, par exemple import math
- **Conditions** :if, elif, else.
- **Boucles** :for et while.

4-Gestion des dépendances :

Avec pip pour installer les bibliothèques nécessaires :

- pip install nom_de_la_bibliothèque

5-Documentation :

Avec des docstrings et créer un fichier README.md pour expliquer le projet.

6-Visualisation :

Utilisation des bibliothèques comme matplotlib ou networkx pour visualiser le projet.

7-Tests :

Ecriture des tests pour le code, en utiliseant unittest ou pytest

Amélioration :

1-Installation de Python :

Assurez-vous que Python est installé sur votre système. Vous pouvez télécharger la dernière version de Python depuis le site officiel python.org.

2-Configuration de l'environnement :

Utilisez un environnement virtuel pour isoler vos projets. Vous pouvez créer un environnement virtuel avec venv

3-Structure du projet :

Organisez votre projet avec une structure de dossiers claire :

- **my_project/**
- **├── my_module/**
- **│ ├── __init__.py**
- **│ └── my_script.py**
- **├── tests/**
- **│ └── test_my_script.py**
- **├── requirements.txt**
- **└── README.md**