29/12/2024

Plan d'Assurance Qualité

Réaliser par : -MAKOUAR Abderrahim
-MARHANI Ayoub
- LACHGAR Hamza
- GHANAMM Ali
- ABOUMADA Abdelghani

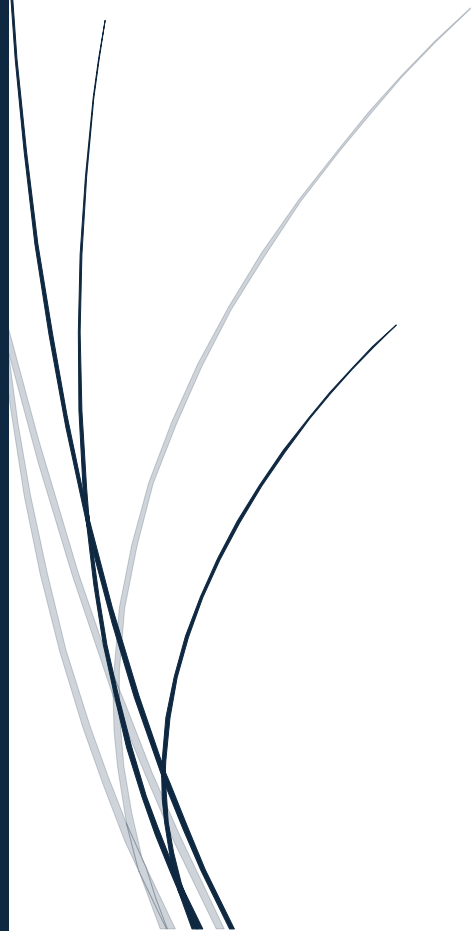

Table des matières

Objet de ce document

Introduction

Le Plan d'Assurance Qualité Projet est un document décrivant comment mettre en œuvre les moyens permettant d'obtenir la qualité nécessaire à la bonne réalisation du projet. Il précise également les dispositions relatives à la conception et à la maîtrise de la qualité pour l'ensemble du cycle de vie d'un système.

Cadre du PAQP

Le but de ce document est de définir la politique qualité de l'entreprise au niveau projet. Le présent document décrit donc les méthodes à employer dans les différents cas de figure pouvant se présenter lors d'un projet.

Terminologie et abréviations

CdP → Chef de Projet

MCT → Modèle Conceptuel de Traitement MOA → Maîtrise d'ouvrage

MO → Maîtrise d'œuvre

PAQP Plan d'Assurance Qualité Projet

PAQL : Plan d'Assurance Qualité Logiciel RQ

 Responsable Qualité

Organisation des ressources humaines

Rôle des différents intervenants

- **Chef de projet**
- **Rôle principal :** Responsable de la gestion globale du projet.
- **Tâches :**
    - Constituer l'équipe projet en identifiant les forces et compétences de chacun.

- Définir et distribuer les rôles et responsabilités au sein de l'équipe.
- Élaborer un planning prévisionnel pour planifier les tâches et échéances.
- Coordonner les activités de l'équipe pour assurer une collaboration efficace.
- Surveiller et vérifier l'avancement des tâches et le respect des délais.
- Valider les résultats et préparer leur présentation aux enseignants ou clients.

Responsable communication/documentation

- **Rôle principal :** Gérer les flux d'information et la documentation.
- **Tâches :**
  - Collecter et rassembler toutes les informations produites par les membres de l'équipe.
  - Assurer que les livrables intermédiaires soient réalisés dans les délais fixés.
  - Communiquer régulièrement avec le client (enseignants ou autres parties prenantes) pour s'assurer que leurs attentes sont bien comprises.
  - Diffuser les documents importants auprès de l'équipe projet pour garantir leur accessibilité.
  - Favoriser la communication interne en organisant des échanges au sein du groupe.

Responsable qualité

- **Rôle principal :** Garantir la conformité et la qualité des livrables.
- **Tâches :**
  - Mettre en place et suivre une démarche qualité adaptée au projet.
  - Contrôler la qualité des livrables intermédiaires pour s'assurer qu'ils répondent aux attentes.
  - Veiller à ce que le produit final respecte les standards de qualité établis et soit conforme aux attentes du client.

Responsable technique

- **Rôle principal :** Assurer les choix techniques et leur mise en œuvre.
- **Tâches :**
  - Identifier et sélectionner les outils ou technologies les plus adaptés aux besoins du projet.
  - Vérifier que les outils sont utilisés correctement par l'équipe.

- Évaluer la faisabilité des choix techniques pour garantir leur pertinence.
- Fournir un soutien technique à l'équipe en cas de difficultés.

Qualité au niveau processus de développement

Modèle retenu dans le cadre du processus de développement et méthodologie utilisée : Scrum

A) Spécification

Objectifs :

- Définir les besoins et prioriser les fonctionnalités dans le **Product Backlog**.
- Décrire les fonctions essentielles du logiciel sous forme d'**User Stories** (exemples : "En tant qu'utilisateur, je veux... pour...").
- Préciser les critères d'acceptation pour chaque User Story.
- Décrire les interactions entre le logiciel et son environnement.

B) *Livrables* :

- **Product Backlog** : Liste priorisée des User Stories.

Conception préliminaire

Objectifs :

- Identifier les composants nécessaires au développement (modules, services, interfaces).
- Planifier et organiser les tâches du **Sprint Backlog** pour le prochain Sprint.

- Réaliser des maquettes ou prototypes pour clarifier certaines fonctionnalités (si nécessaire).
- Assurer la faisabilité technique des fonctionnalités.

Livrables :

- **Sprint Backlog** : Liste des tâches spécifiques à réaliser dans le Sprint en cours.
- **Maquettes ou prototypes** (si requis pour des fonctionnalités spécifiques).
- **Diagrammes d'architecture simplifiés** (si besoin pour guider l'équipe).

Documentation

Règles de gestion et de structuration des documents

Identification des documents

- Chaque document doit être clairement identifié pour faciliter sa gestion.
- Les fichiers électroniques doivent suivre un format de nommage tel que : Rédacteur_Nature du document_Version

Exemple : JM_RapportFinal_V1.0.docx

Présentation des documents

- **Police de caractère** : Times new Roman, taille 12.
- **Structure commune à tous les documents (**:
- **Page de garde** incluant :
  - Titre du document.
  - Date de dernière mise à jour.
  - Numéro de version.
  - État du document (travail, terminé, vérifié, validé).
  - Nom de l'auteur ou des auteurs.
- **Page de sommaire** : Table des matières structurée.
- **Table des mises à jour** retraçant les révisions majeures avec :
  - Numéro de version.
  - Date de mise à jour.

- Objet de la mise à jour.

Etat d'un document

- **Travail** : Document en cours d'élaboration.
- **Terminé** : Document prêt à être diffusé.
- **Vérifié** : Document approuvé par le responsable qualité.
- **ValidéCP** : Document approuvé par le chef de projet.
- **ValidéCL** : Document approuvé par le client (enseignant ou partie prenante)

Gestion des versions

Tous les livrables doivent avoir un numéro de version sous la forme : V X.x

- **X** : Incrémenté pour les modifications majeures.
- **x** : Incrémenté pour les modifications mineures.

Liste des documents principaux

Cahier des charges :

a. *Objectif* **:** Décrire les besoins, contraintes et exigences du projet.

b. *Contenu* **:**

i. Description du contexte.
ii. Objectifs du projet.
iii. Fonctionnalités principales attendues.
iv. Contraintes techniques ou organisationnelles.

c. *État attendu* : Validé par le chef de projet ou l'enseignant.

Rapport de projet :

d. *Objectif* **:** Synthétiser le travail effectué tout au long du projet.

e. *Contenu* **:**

i. Résumé exécutif.
ii. Méthodologie utilisée (ex. : Scrum).
iii. Analyse des résultats.
iv. Conclusion et recommandations.

f. *État attendu* : Vérifié et validé.

Plan d'Assurance Qualité (PAQ) :

g. *Objectif* **:** Garantir la qualité des livrables produits pendant le projet.

h. *Contenu* **:**

i. Méthodologie utilisée pour assurer la qualité.

ii. Normes ou référentiels suivis.

iii. Critères d'acceptation des livrables.

iv. Gestion des revues, tests et validations.

i. *État attendu* : Vérifié par le responsable qualité et validé.

Gestion des modifications

Origine des modifications

Les modifications peuvent être initiées pour plusieurs raisons, notamment :

1. **Correction d'erreur** :
    a. Une erreur a été identifiée dans le document ou le livrable, nécessitant une rectification.
2. **Mise à jour ou complément** :
    a. Une nouvelle information ou un ajustement est nécessaire pour compléter ou mettre à jour le contenu existant.

Procédure de mise en œuvre des modifications

La procédure dépend de la nature des modifications apportées. Les étapes incluent :

1. **Identification de la modification** :
    a. Définir clairement l'objet de la modification (correction ou ajout).
    b. Répertorier la modification dans une **table des mises à jour** pour le suivi.
2. **Analyse de l'impact** :
    a. Évaluer l'effet de la modification sur le document ou le projet (influence sur d'autres sections, livrables ou tâches).
3. **Mise en œuvre de la modification** :
    a. **Pour une correction d'erreur** :

i. Effectuer les ajustements directement dans le document ou livrable concerné.
ii. Augmenter le numéro de version mineure (par exemple, passer de V1.0 à V1.1).

b. **Pour un ajout ou une mise à jour majeure** :
i. Ajouter les informations ou éléments requis.
ii. Réviser le document et augmenter le numéro de version majeure (par exemple, passer de V1.0 à V2.0).

4. **Validation de la modification** :
a. Vérifier que la modification répond aux besoins identifiés (validation par le responsable qualité, chef de projet ou autre partie prenante concernée).
5. **Diffusion** :
a. Distribuer la version mise à jour à l'équipe ou aux parties prenantes concernées.
b. Mettre à jour les références ou liens associés, si nécessaire.

Méthodes, Outils et Règles

Méthodes et méthodologies retenues

Utilisation de UML (Unified Modeling Language) :

- UML est un langage standardisé de modélisation utilisé pour représenter visuellement les systèmes logiciels.
- Il remplace efficacement les formalismes spécifiques à Merise (MCT et MCD) dans de nombreux contextes modernes.
- **Avantages** :
  - Facilite la communication entre les membres de l'équipe grâce à des diagrammes normalisés.
  - Permet de couvrir tous les aspects d'un projet, de l'analyse des besoins à la conception.

Types de diagrammes UML pertinents :

- **Diagrammes de cas d'utilisation (Use Case Diagram)** :

- Représentent les interactions entre les acteurs et les fonctionnalités principales du système.
- **Diagrammes de classes (Class Diagram)** :
  Décrivent la structure statique du système, incluant les classes, attributs, méthodes et relations.
- **Diagrammes de séquence (Sequence Diagram)** :
  Illustrent les interactions dynamiques entre les objets au fil du temps.
- **Diagrammes d'activité (Activity Diagram)** :
  Modélisent les flux de travail ou processus métier.
- **Diagrammes de composants (Component Diagram)** :
  Représentent les différentes parties du système et leurs interactions.

Outils (Outils logiciels, autres outils)

Logiciels pour UML :

- **Draw.io** : Pour créer les diagrammes UML de manière intuitive.
- **StarUML** : Outil spécialisés pour la modélisation UML avancée.

Outils pour la rédaction des livrables :

- **Microsoft Word** : Rédaction des documents.

Règles et normes devant être appliquées

Règles pour UML :

- Utiliser les conventions standardisées de UML (symbole, relation, notation).
- Les diagrammes doivent être suffisamment détaillés pour être exploitables, mais pas surchargés pour rester compréhensibles.
- Respecter une cohérence entre les différents diagrammes (par exemple, entre le diagramme de cas d'utilisation et le diagramme de classes).

Règles générales :

- Respecter les normes enseignées en programmation (L3 et M1).

- Appliquer les bonnes pratiques de modélisation enseignées (séparation des responsabilités, réutilisabilité, modularité).
- Suivre les consignes décrites dans le dossier de gestion de la documentation pour la présentation et la structuration des livrables.

NORMES DE CODE

Initialisation et gestion des objets :

- Utiliser **Autowired** ou **@Bean** pour l'injection de dépendances.
- Les instances doivent être gérées via le conteneur Spring (ne pas utiliser new directement sauf pour des objets locaux simples).

Normes de programmation :

- **Annotations Spring** : Prioriser les annotations Spring Boot (@Controller, @Service, @Repository, etc.) pour la gestion des composants.
- Utiliser @Value pour injecter les configurations depuis les fichiers .properties.

Structure des packages :

Suivre une structure claire et modulaire:

```
com.projectname
  ├── controller
  ├── service
  ├── repository
  ├── model
  ├── config
```

- Chaque package doit contenir les fichiers relatifs à son rôle spécifique.

Documentation

- Ajouter des commentaires **JavaDoc** pour chaque classe, méthode, et variable importante.
- Utiliser des noms descriptifs pour les classes, méthodes, et variables (en anglais).

Noms des entités

- Les noms des classes doivent être en **PascalCase** (exemple : `UserService`).
- Les noms des méthodes et variables doivent être en **camelCase** (exemple : `findUserById`).

Utilisation de bases de données

- Prioriser l'utilisation de **JPA** ou **Spring Data** pour l'accès aux données.
- Les requêtes personnalisées doivent être écrites dans des **interfaces de repository** en utilisant `@Query` si nécessaire.

Configuration

- Utiliser des fichiers de configuration (`application.properties`) pour gérer les paramètres du projet.
- Toutes les configurations sensibles (mots de passe, clés API) doivent être externalisées et stockées dans des environnements sécurisés.

Tests

- Écrire des tests unitaires avec **JUnit** et **Mockito**.
- Utiliser **Spring Boot Test** pour les tests d'intégration avec annotations comme `@SpringBootTest`.

Sécurité

- Implémenter **Spring Security** pour protéger les endpoints sensibles.
- Toutes les données sensibles doivent être encryptées avant stockage.

Gestion des dépendances

- Les dépendances doivent être gérées via **Maven** , avec une version définie pour chaque librairie.
- Ne pas inclure de dépendances inutiles pour réduire la taille de l'application.

Contrôle de version

- Utiliser **Git** pour le suivi des modifications.

Suivi de l’application du plan qualité

Principes

L'application rigoureuse du **Plan d’Assurance Qualité Projet (PAQP)** et des documents associés (tels que les PAQL) est essentielle pour garantir un produit final conforme aux attentes. Il est impératif de vérifier, tout au long du projet, que les règles et les processus définis sont correctement suivis.

Interventions du RQ

Le Responsable Qualité (RQ) joue un rôle central dans le suivi et le respect des normes de qualité. Ses missions incluent :

- **Validation des livrables** :
  - S'assurer que chaque composant produit respecte les critères définis avant validation.
- **Supervision des revues** :
  - Veiller à la bonne tenue des revues qualité et vérifier leur conformité aux procédures.
- **Contrôle de l’application des documents qualité** :
  - Vérifier que les membres de l'équipe utilisent correctement les documents et outils liés à la qualité.

Moments clés des interventions :

Sur demande :

Lorsque le chef de projet ou un membre de l’équipe technique sollicite des conseils ou des validations.

Validation des composants :

Lorsqu’un composant ou livrable est finalisé et doit passer en revue avant acceptation.

Contrôles aléatoires :

Inspection ponctuelle de la production pour vérifier que les standards de qualité sont respectés.

Procédure à suivre en cas de non-respect des règles de qualité

1. Identification de la non-conformité

- Détecter précisément l'écart par rapport aux règles ou standards de qualité établis.
- Documenter la non-conformité en précisant les éléments concernés (livrable, processus, etc.).

2. Sensibilisation

- **Objectif** : Changer la perception négative de la qualité en démontrant ses avantages.
- **Actions** :
    - Expliquer à la personne concernée l'importance des pratiques qualité pour garantir :
        - Un produit final fiable et conforme.
        - Une réduction des erreurs et du travail à refaire.
        - Une satisfaction accrue des clients ou enseignants.
    - Montrer comment l'application des normes de qualité peut faciliter leur travail à long terme (par exemple, en évitant des corrections coûteuses ou répétées).

3. Proposition de solution

- Accompagner la personne dans la mise en conformité du livrable ou du processus.
- Fournir des ressources ou des outils pour simplifier l'application des règles.

4. Correction

- Mettre à jour le livrable ou le processus pour répondre aux critères de qualité spécifiés.
- Faire valider les corrections par le Responsable Qualité (RQ) ou le chef de projet.

5. Suivi

- Vérifier que la non-conformité ne se reproduit pas grâce à un contrôle ponctuel ou un rappel des bonnes pratiques.

Conclusion

Ce **PAQP** vise à établir les dispositions essentielles que la **Maîtrise d'Œuvre (MOE)** doit respecter pour garantir un niveau de qualité élevé tout au long du projet. L'objectif est d'assurer des livrables conformes aux attentes sans imposer de contraintes excessives à l'équipe. En adoptant des règles claires, simples et adaptées aux spécificités du projet, ce PAQP permet à la MOE de concilier efficacité et flexibilité. Ainsi, la qualité devient un élément intégré au processus de développement, favorisant la réussite du projet tout en évitant les lourdeurs administratives.