Ex1: les qualités

- ✓ Le logiciel ne fournit pas toutes les fonctionnalités attendues. la qualité manquante c'est la fiabilité
- ✓ L'utilisateur trouve que l'apprentissage pour utiliser le logiciel est difficile. la qualité manquante c'est **utilisabilité**
- ✓ Les résultats donnés par le logiciel sont parfois erronés. la qualité manquante c'est **fiabilité**
- ✓ Le logiciel consomme beaucoup de CPU pour des requêtes qui semblent simples, la qualité manquante c'est la **performance**

exo2

1. Indiquer la ou les phases où est produit chacun des documents

Documents dans le cycle de vie d'un logiciel :	
Manuel utilisateur final	Phase d'implémentation
Conception architecturale	Phase de conception
Plan d'assurance qualité	Phase de planification du projet
Spécifications des modules	Phase de conception
Code source	Phase d'implémentation
Cahier des charges	Phase de faisabilité
Plan de test	Phase de spécification
Manuel utilisateur préliminaire	Phase de spécification
Conception détaillée	Phase de conception
Estimation des coûts	Phase de planification
Calendrier du projet	Phase de planification
Rapport des tests	Phase de test
Documentation	Phase d'implémentation

2. Le modèle spirale peut être vu comme une succession de modèle « en cascade » ou « en v »

Ex3:

Cahier des charges précis = risques maîtrisés mais non nuls, donc se laisser la possibilité de revenir sur une étape précédente donc le cycle de vie le plus adéquat c'est le modèle en cascade

Il permet de valider la qualité et les travaux de l'étape qui vient de s'achever. Et aussi permettre de corriger certaines erreurs ou apporter des modifications en décidant de revenir à l'étape précédente si cela est nécessaire pour le bien du projet, Simplicité de mise en œuvre, mais il n'implique pas de participation importante du client sur toutes les phases. (Plus de détails sur les avantages et les inconvénients voir le cours)

Exo 4 :

Le modèle de cycle de vie utilisé par cette entrepris c'est le modèle «en cascade"

Exo 5

Identifier les risques pour chaque approche

Solution 1 : développement par l'université

•le cout de SGBD qui peut être trop élevé
•Temps de développement trop important
•Manque de compétences en matière de développement
•Apparition d'autres coûts supplémentaires comme l'achat d'équipements et les recrutements.
•risques de bugs et de maintenance

Solution 2 : Achat d'un logiciel et adaptation

•Le logiciel peut ne pas couvrir correctement/totalement les besoins
•Difficultés d'adaptation
•Indépendance à un fournisseur
•mises à jour difficiles

Solution 3 : Elaboration d'un même cahier des charges et sous-traitance (développement effectué par une société de développement)
•Coordination complexe, ce qui a une influence sur l'établissement d'un cahier de charges commun
•Lenteur du processus d'analyse des besoins et délais de décision prolongés
•Apparition de conflits concernant le financement du projet
• Non équité des frais à engager par rapports aux besoins des différentes parties
•Problème des mises à jour qui peuvent ne pas arranger toutes les parties

Décision :

1.l'approche la plus collaborative (se joindre à un groupe d'autres universités) pourrait réduire les coûts et les risques de développement, mais nécessite une bonne coordination

2.L'achat d'un système comparable pourrait être plus rapide, mais moins flexible

3.Développer un système en interne peut offrir une personnalisation complète, mais avec des risques élevés

Finalement, l'approche la plus équilibrée serait de se joindre à un groupe d'autres universités pour partager les coûts et les ressources tout en répondant aux besoins communs