
Fiche TD N° 1

Cours #1 : Généralités sur le génie logiciel

Répondre aux questions suivantes

1. Quelles sont les difficultés technologiques liées au GL ?

- ✚ Engagements irréalistes ;
- ✚ Gestion et conduite de projet inadéquates ;
- ✚ Manque de contrôle ;
- ✚ Technologies inappropriées (méthodes, outils, langages) ;
- ✚ Validation et vérification insuffisantes, etc.

2. Quelles sont les différentes approches qui peuvent être adopter lors de la phase spécification ?

La complexité et les dimensions de la spécification peuvent varier énormément en fonction de l'environnement d'utilisation du logiciel et des objectifs auxquels il répond. Les formes de spécification :

- ✚ Les approches formelles: Utilisent des outils mathématiques et des méthodes de preuve pour construire un logiciel correct.
- ✚ Les approches informelles utilisent :
 - Langage naturel structuré : des modèles de fiches, et des formulations, dont le sens est explicité et moins ambiguë.
 - Description algorithmique : un langage de programmation abstrait est utilisé pour donner une vision opérationnelle du système.
 - Notation graphique: des diagrammes annotés généralement par du texte en langage structuré, représentent le système
- ✚ Les approches semi-formelles
 - Elles visent à introduire un langage normalisé pour décrire le logiciel et sa spécification (UML,etc).

3. Quelle est la différence la plus importante entre le développement des produits logiciels génériques et spécifiques? Qu'est-ce que cela peut signifier dans la pratique pour les utilisateurs des produits logiciels génériques ?

- La spécification pour le développement de produits logiciels générique est bien menée par le développeur du produit quant aux produits logiciels spécifiques, la spécification est détenue et contrôlée par le client (demandeur du produit logiciel). Les utilisateurs de produits génériques n'ont pas aucun contrôle sur la spécification de logiciels et peuvent donc pas contrôler l'évolution du produit. Le développeur peut décider d'inclure/ exclure des fonctionnalités et changer d'interface utilisateur. Cela pourrait avoir des implications pour les processus métiers de l'utilisateur et ajouter des couts de formation supplémentaire lorsque de nouvelles versions du système sont installées.

4. Ecrire un cahier de charges qui permet de développer une application logicielle pour une entreprise économique.

Consignes pour la rédaction d'un cahier des charges :

- Un cahier des charges correct exprime un besoin **nécessaire** (paramètre nécessaire à la conception et la réalisation du projet), **réalisable** (besoin techniquement et financièrement réalisable), **vérifiable** (existence des moyens de vérification pour ce qui est requis).
- Un cahier des charges doit être complet , cohérent
- Les phrases et paragraphes doivent être courts.
- Un cahier des charges doit rester aussi général que possible afin de ne pas brider la conception.
- Un cahier des charges doit être clair, (l'idée ou le besoin qu'il exprime n'est pas susceptible d'être interprété différemment de ce que pense le maître d'ouvrage).
- Le cahier des charges doit servir tout au long du projet

Exemple d'un cahier des charges :

1. Présentation du projet

- Contexte : environnement dans lequel s'inscrit le projet (stratégie, domaine, etc.)
- Objectifs : résultats que le projet doit atteindre.
- Description de l'existant : environnement logiciel et matériel du logiciel, Système existant.
- Critères d'acceptabilité du produit : procédure de validation, Critères d'acceptation.

2. Expression des besoins

- Besoins fonctionnels : fonctions que le logiciel doit réaliser, Les spécifications fonctionnelles peuvent être classées par importance.
- Besoins non fonctionnels: les spécifications non fonctionnelles sont toutes les spécifications qui n'expriment pas une fonction du logiciel (contraintes de performance, système d'exploitation cible...).

3 Contraintes

- Coûts : budget alloué au projet, Moyens matériels et logiciels mis à disposition.
- Délais : date de livraison du produit, Echéances intermédiaires.
- Autres contraintes : autres contraintes à prendre en compte (normes techniques, clauses juridiques, etc.)

4 Déroulement du projet

- Planification : articulation des grandes phases du projet et des principaux jalons.
- Plan d'assurance qualité : procédures adoptées pour contrôler la qualité du logiciel.
- Documentation : description de la documentation devant accompagner le logiciel à sa livraison.
- Responsabilités
 - Maîtrise d'ouvrage : On appelle maître d'ouvrage (notée MOA) l'entité porteuse du besoin, définissant l'objectif du projet, son calendrier et le budget consacré à ce projet. Le résultat attendu du projet est la réalisation d'un produit, appelé ouvrage.

-Maîtrise d'œuvre : Le maître d'œuvre (notée MOE) est l'entité retenue par le maître d'ouvrage pour réaliser l'ouvrage, dans les conditions de délais, de qualité et de coût fixées par ce dernier conformément à un contrat.

5. En donnant des exemples, définir la fiabilité et la performance des logiciels ? donner aussi leurs métriques ?

Fiabilité: le logiciel est conforme à ses spécifications (le logiciel fonctionne raisonnablement en toutes circonstance, rien de catastrophique ne peut survenir, même en dehors des conditions d'utilisation prévues).

Métrique: Disponibilité : pourcentage du temps pendant lequel le système est utilisable) et taux d'erreur

Performance : le logiciel doit satisfaire aux contraintes de temps d'exécution
Métrique : temps d'exécution.

6. En quoi un modèle de cycle de vie divisé en phases aide-t-il à la gestion du développement d'un logiciel ?

Un cycle de vie décomposé en plusieurs phases permet d'avoir une meilleure visibilité du projet. Le cycle de vie permet de détecter les erreurs le plus tôt possible et ainsi de maîtriser la qualité du produit, les délais de sa réalisation et les coûts associés.

7. Recensez dans un tableau les principales forces et faiblesses de chacun des modèles en cascade, modèle en V et modèle en spiral.

(Consulter cours)

8. Pour chacun des documents ci-dessous indiquer sa phase de production

Manuel utilisateur final, Conception architecturale, Plan d'assurance de qualité, Code source, Cahier de charges, Plan de test, Manuel utilisateur préliminaire, Estimation des coûts, Calendrier du projet, Rapport des tests, Documentation

Document	Phase de production
Manuel utilisateur final	Implémentation
Conception architecturale	Conception
Plan d'assurance de qualité	Organisation du projet/ Planification
Code source	implémentation
Cahier de charges	Spécification
Plan de test	Spécification

Manuel utilisateur préliminaire	Spécification
Conception détaillée	Conception
Estimation des coûts	Organisation du projet/ planification
Calendrier du projet	Planification/ organisation du projet
Rapport des tests	Tests
Documentation	Implémentation

9. Citez quelques avantages offerts par les tests unitaires

- Facile à écrire ;
- Ne nécessite le plus souvent qu'un contexte minimal ;
- Il offre une granularité de test très fine et permet de valider exhaustivement le comportement de la partie du code testée
- Son exécution est rapide, ce qui permet de le lancer très fréquemment (*idéalement à chaque modification du code testé*) ;
- Il rassemble les cas d'utilisation possibles d'une portion d'un projet et représente donc une véritable documentation sur la manière de manipuler le code testé ;

10. Quand est-ce qu'un processus de développement est dit itératif et incrémental ?

Un processus de développement est dit itératif et incrémental si le projet logiciel est découpé en plusieurs itération et qu'à chaque itération on livre une version du logiciel. a chaque itération on surcharge le logiciel avec de nouvelles fonctionnalités (incréments)

11. Citez quelques critères généraux, internes et externes d'un bon logiciel.

Critères généraux : sont des principes et éléments de référence qui permettent de juger; d'estimer et de vérifier régulièrement si le processus de développement d'un logiciel possède ou non les différentes propriétés déterminées.

Exemple : Opérationnel, Transitionnel, Maintenance, etc.

Critères externes : expriment ce qu'est un bon logiciel du point de vue des **utilisateurs**.

Exemple : Robustesse et Sureté, Documentable , ergonomique, Intégrité

Critères internes: expriment ce qu'est un bon logiciel du point de **vue du développeur**. Ces critères sont essentiellement liés à la maintenance d'un logiciel.

Exemple : Lisibilité et Clarté, Portabilité, Extensibilité, Réutilisabilité, Traçabilité

12.Proposer le modèle de processus logiciel générique le plus appropriée qui pourrait être utilisé comme une base pour la gestion de développement des systèmes suivants:

- Un système pour contrôler le freinage anti-blocage dans une voiture.
C'est un système de sécurité critique qui exige beaucoup d'analyse à l'avance avant la mise en œuvre. Il a certainement besoin d'une approche de développement piloté par plan avec des exigences soigneusement analysées. Un modèle en cascade (voir aussi modèle en V) est donc le processus le plus approprié à utiliser.
- Un système de réalité virtuelle pour soutenir la maintenance des logiciels.
C'est un système où les données vont changer et il y'aura des composants d'interface utilisateur étendues. Le développement incrémental avec peut être des prototypes de l'interface utilisateur est le modèle le plus approprié. Un processus agile peut être aussi utilisé.
- Un système interactif de planification de voyage qui aide les utilisateurs à planifier leurs voyages avec le plus bas impact sur l'environnement.

Système avec une interface utilisateur complexe (plusieurs questions et réponses entre le système et l'utilisateur) mais qui doit être stable et fiable. Une approche de développement incrémental est la plus appropriée et avec le changement dans les exigences du système, l'expérience de l'utilisateur avec le système sera acquise.

13.Expliquer comment les principes fondamentaux des méthodes agiles conduisent au développement et au déploiement accélérés de logiciels.

(Consulter cours : fondements et principes des méthodes agiles)