M1 : Ingénierie du Logiciel – 4I 502

SORBONNE UNIVERSITE – FACULTE DES SCIENCES

Examen Réparti 1ere partie

14 novembre 2018 (2 heures avec documents : tous SAUF ANNALES CORRIGEES). Barème indicatif sur 20 points.

1. Questions de cours [5 Pts]

Répondez de façon précise et concise aux questions.

Barème : VALABLE sur toutes les questions de cours : -25 à -50% si la réponse inclut la bonne idée, mais qu'elle est noyée dans des infos ou autres réponses fausses/inappropriées.

Q1.1(1 point) : Dans le diagramme de cas d'utilisation d'un réseau social type FaceBook, on a modélisé que le use case « Poster Message » *inclut* (<<include>>) le cas d'utilisation « se loger ». Critiquer cette modélisation.

A priori, non, on ne doit pas se « re »-loger à chaque fois qu'on poste, on cherche ici une relation de précédence, qui ne s'exprime pas directement sur ces diagrammes.

On préfére une modélisation avec deux acteurs, l'un étant authentifié et pouvant « poster », l'autre ne pouvant que « se loger » dans un premier temps.

Barème :

20% le sens (on doit se reloger à chaque post) est compris

50% : include ne traduit pas la précédence, ce n'est pas ce qu'on veut

30% : on suggère une autre modélisation, avec des acteurs différents, ou à l'aide de pré-conditions.

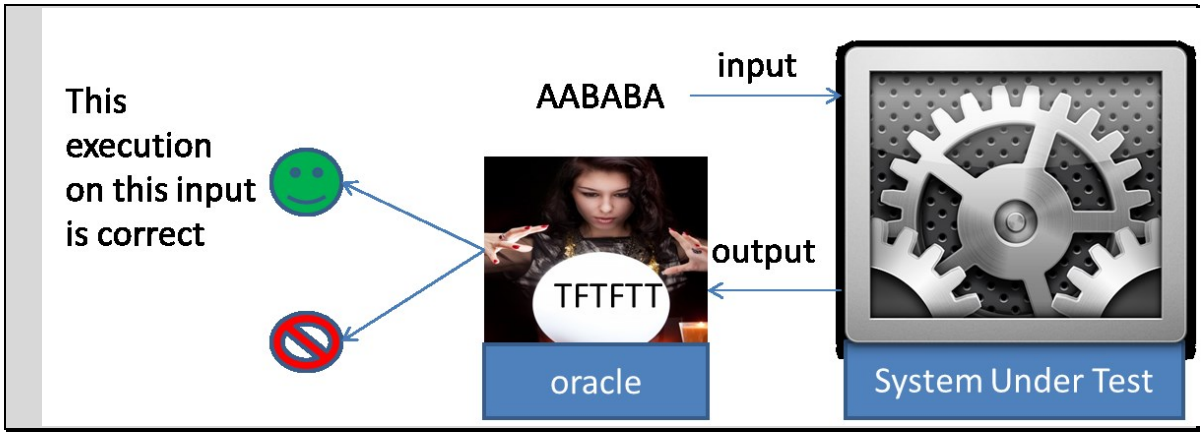
Q1.2(1.5 point) : Dans une approche en boite noire basée sur les tests, expliquez les termes : oracle, verdict.

Oracle : interprète la sortie fournie par le système et détermine le verdict. Dans les cas simples il s'agit simplement de comparer la sortie obtenue à une valeur de contrôle, dans les cas plus complexes (e.g. indéterminisme…) l'oracle peut être plus complexe (e.g. comparer la sortie avec celle d'un outil de référence, autoriser des réordonnancements…).

Verdict : La conclusion de l'Oracle, pour un test donné sur un système donné. Le verdict est le plus souvent : PASS/FAIL, mais on peut rencontrer parfois d'autres valeurs comme TIMEOUT, INCONCLUSIVE…

50% par notion

La figure du cours donne les points.



Q1.3 (1 point) : Pourquoi les classes métier ne contiennent-elles pas d'opérations ? Pourquoi ne précise-t-on pas la navigabilité des associations ?

Les classes métier ne *sont PAS* des classes au sens OO habituel, c'est une modélisation du domaine métier. Ces « classes métier » ne sont pas des artefacts techniques liés à la solution logicielle, mais bien un artefact qui spécifie en analyse les données et leurs liens.

L'attribution de méthodes aux classes est une tâche de Conception. De même, la navigabilité des associations est une problématique de conception, pas d'analyse.

70% car ce ne sont pas des « vraies » classes, ni des associations au sens OO implantés par des attributs (donc où la navigabilité prend son sens). La description de l'étudiant fait apparaître que cela n'a pas vraiment de sens, on cite une base de donnée, …

30% ces préoccupations sont du ressort de la conception (et/ou conception détaillée), mais pas des choix à poser en Analyse.

On note assez large cette question, la variabilité des réponses y est assez importante.

Q1.4 (1,5 point): Quels sont les trois éléments qu'un langage de description d'architecture doit définir séparément ?

1. Les connecteurs (e.g. interfaces UML)
2. Les composants, offrent/requièrent des connecteurs (e.g. composants UML)
3. La topologie d'instanciation, la description d'une configuration des composants instanciés (e.g. dia de structure interne UML)

Barème :

35% chaque point, max 100%. Il suffit de citer les trois termes « composant, connecteur, instanciation ».

En pratique ça donne du 0 ou 100 binaire.

2. Problème: Analyse de GitBus [15 Pts]

GitBus est une application offerte par le web dédiée aux développeurs de logiciel open source. Elle facilite la construction et l'administration de dépôts git, ainsi que la communication autour de projets open source.

Chaque utilisateur peut créer un compte personnel. Pour cela il doit choisir un identifiant, fournir un email (valide), et un mot de passe. La création du compte n'est confirmée que quand l'utilisateur clique sur le lien reçu par email. Un utilisateur authentifié peut créer une ou plusieurs *organisations,* une organisation comportant des membres et des propriétaires.

L'utilisateur qui crée l'organisation en est automatiquement propriétaire. Le propriétaire d'une organisation peut ajouter d'autres utilisateurs à l'organisation en tant que membres ou en tant que propriétaire. Tout propriétaire d'une organisation en est également membre.

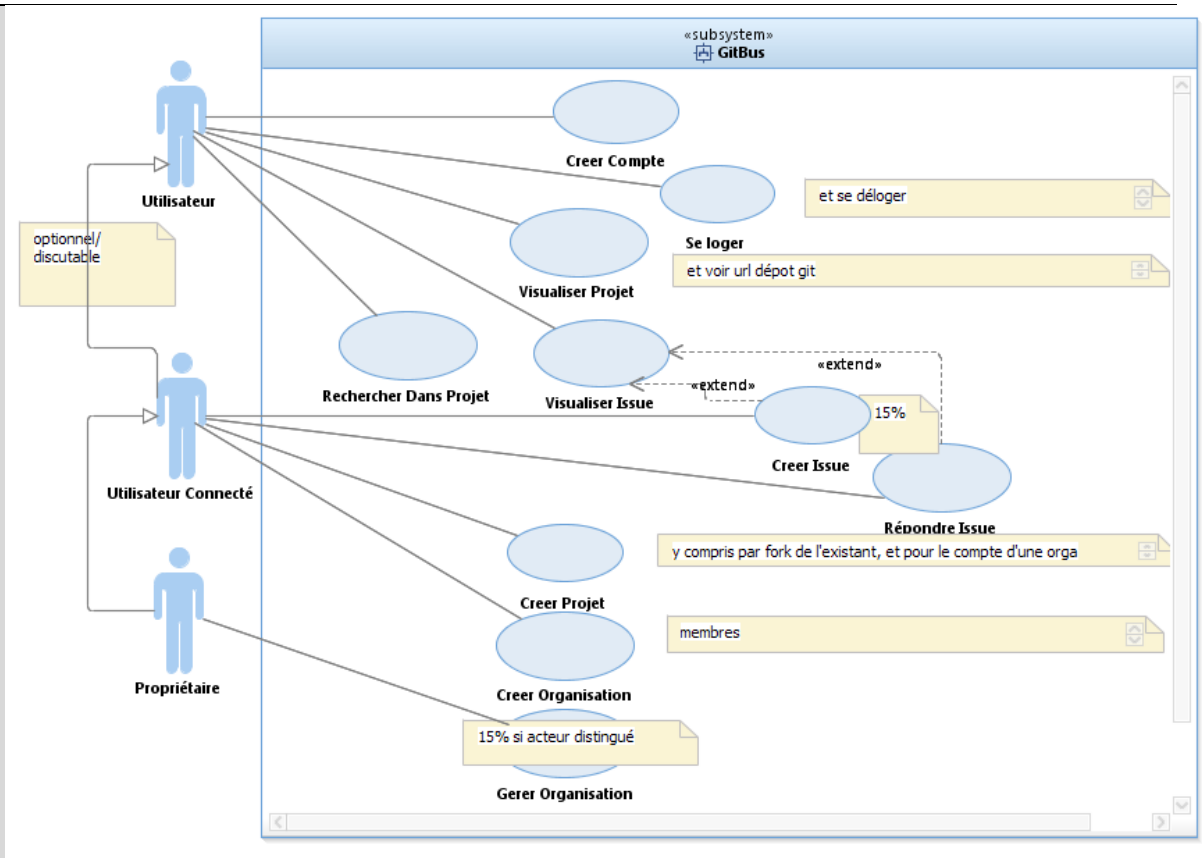
GitBus associe à chaque compte (personnel ou organisation) un ensemble de *projets*. Chaque projet a un nom unique au sein des projets de son propriétaire. Il est lié à l'url d'un dépôt git. Pour créer un nouveau projet, il faut préciser s'il sera attaché au compte personnel de l'utilisateur ou à l'une des organisations dont il est propriétaire. Si le propriétaire du projet est une organisation, tous les membres de l'organisation peuvent y accéder en écriture.

Chaque projet dispose de sa propre page d'accueil, qui permet de visualiser le code du projet, stocké dans un dépôt git hébergé par GitBus. On peut également récupérer l'url du dépôt git, qui servira aux utilisateurs à invoquer l'outil git, par exemple depuis leur IDE préféré. Si le projet appartient à une organisation, tous ses membres possèdent le droit de modifier le dépôt. Si le projet est lié à un compte personnel, seul le propriétaire du compte peut modifier le dépôt.

On dispose également d'une barre de recherche pour trouver des fichiers particuliers ou contenant un texte en particulier au sein du projet. Ces pages sont accessibles par tous, sans authentification préalable. Si le visiteur est authentifié il peut en plus choisir de « fork » un projet qu'il visite, ce qui aura pour effet de créer un nouveau projet (à son nom ou celui d'une organisation dont il est propriétaire, au choix) initialisé avec le contenu du dépôt git courant.

Les utilisateurs authentifiés peuvent également discuter des problèmes identifiés dans un projet à l'aide des « *issues* ». N'importe quel utilisateur peut créer une issue sur n'importe quel projet. La création d'une issue nécessite de fournir un titre pour le problème, et un texte décrivant le problème. Depuis la page d'accueil du projet, on peut visualiser les issues existantes, et y répondre. Chaque issue consiste en un fil de conversation, où les différents messages des participants sont visibles en ordre chronologique.

Question 2.1 : (3,5 pts) Réalisez le diagramme de cas d'utilisation de la phase d'analyse. Vous justifierez tous vos choix, par un texte ou des annotations sur le diagramme.



Barème :

10% par use case * 7 use case du diagramme du corrigé. Il doit être lié à un acteur « raisonnable ». On accepte que seuls les « connecté » puissent visualiser les issues, le sujet étant un peu ambigü. Si l'acteur n'est pas raisonnable 0% pour ce use case.

Les use case suivants sont ok, mais ne donnent pas de points : « forker projet », « récupérer URL dépôt ».

15% pour creer/répondre issue (souvent deux use case). Les extends du corrigé ne sont pas demandés/notés explicitement ici, ce sera évalué sur la fiche détaillée.

15% pour use case « gérer orga/ajouter membre… » s'il a un acteur distingué (qui hérite de l'utilisateur « normal »), sinon seulement 5% pour ce use case.

On accepte des extends et/ou includes raisonnables (un minimum commentés/justifiés) par exemple un « rechercher » qui étend « visualiser projet ». Les foires d'include/extends non justifiés

L'héritage de l'acteur « propriétaire » vers « util connecté » est attendu, l'héritage de « util connecté » vers « util non connecté/visiteur » n'est pas vraiment nécessaire.

Fautes fréquentes :

Le use case : « confirmer email » est toléré sans pénalité même si c'est un peu trop fin.

-5% Le use case « modifier dépôt » est hors système (c'est git qui gère ça), pénalité de -5% pas très méchante s'il apparaît.

-10% autres use cases hors cadre

-5% si c'est TROP détaillé (créer projet dans orga distingué de créer projet personnel, « fournir email » include depuis créer comptes (par contre « confirmer email » est ok, pas de pénalité)

-5 à -15% pour les fautes

-5 par use case mal formulé : on veut un verbe qui exprime l'action du point de vue de l'acteur.

-10% aucun commentaire/aucun texte pour accompagner le diagramme, diagramme sec

Jusqu'à -20% par héritage, include ou extend injustifiable ou autre incohérence ou mésusage d'UML.

-10% si on ne précise pas qui fait l'action dans le scenario (use case sans acteur lié)

Question 2.2 : (3,5 pts) Précisez la ou les fiches détaillée(s) (acteurs concernés, pré-conditions, post-conditions, scénario nominal, alternatives, exceptions) du (ou des) cas d'utilisation(s) correspondant aux interactions permettant de créer une « issue » et d'y répondre.

2 use case a priori : créer et répondre à une Issue.

Ou jusque trois use case si on a séparé visu, création et réponse à un issue.

UC1 : Visualiser Issue

Acteur : Utilisateur

Pré : L'utilisateur connecté est sur la page d'un projet

Scénario :

1. L'utilisateur clique sur « issues »
2. Le système affiche l'ensemble des issues liées au projet : pour chacune on peut voir le titre, le nombre de réponses.
3. L'utilisateur clique sur une issue en particulier
4. Le système affiche le fil de conversation de l'issue : tous les messages sont visibles par ordre chronologique

Post : aucune

ALT A1 : Créer Issue

A1.1 : en SN2, l'utilisateur **connecté** choisit de créer une issue, cf. UC2, « créer issue », puis retour en SN2.

ALT A2 : Répondre

A2.1 : en SN4, l'utilisateur **connecté** choisit de répondre, cf. UC3, « répondre issue », puis retour en SN4.

****

UC2 : Créer Issue

Acteur : Utilisateur connecté

Pré : L'utilisateur connecté et est sur la page « issues » d'un projet

1. L'utilisateur choisit de créer une issue
2. Le système lui affiche un formulaire permettant de saisir le titre de l'issue et le texte de son message
3. L'utilisateur saisit un titre, un texte et valide.
4. Le système enregistre le nouvel issue et l'ajoute au issues du projet

Post : Le nouveau issue est enregistré et ajouté au projet.

EXC E1 : en SN3, l'utilisateur peut annuler, fin de l'interaction.

****

UC3 : Répondre Issue

Acteur : Utilisateur connecté

Pré : L'utilisateur est sur le fil de discussion d'une des page « issues » d'un projet

1. L'utilisateur choisit de répondre
2. Le système lui affiche un formulaire permettant de saisir le texte de son message
3. L'utilisateur saisit un texte et valide.
4. Le système enregistre le nouveau message l'ajoute au fil de discussion de l'issue

Post : Le nouveau message est enregistré et ajouté au issue.

EXC E1 : en SN3, l'utilisateur peut annuler, fin de l'interaction.

Barême :sur 100%

10% Cohérent avec le diagramme de use case, une à trois fiches. On doit voir les include/extends du diagramme dans les fiches (include => étape du SN, extends = ALT).

10% Précondition sur l'utilisateur est connecté pour pouvoir créer/répondre, ou ALT dans le scenario sinon

10% Post condition le message/issue etc… est enregistré

10% on peut annuler aux moments opportuns

20% On décrit des échanges permettant effectivement à l'acteur de voir les messages, en ordre chronologique (5% sur *chronologique*)

20% La saisie d'une nouvelle issue est possible et raisonnable

20% la saisie d'une réponse à une issue est possible et raisonnable.

On peut éventuellement donner 10% pour des ALT ou exception bien spécifiées qui traitent d'autres cas pertinents. Attention à la cohérence avec les dia de use case, il y a 10% dans le barème pour la cohérence.

Cette question est très délicate à corriger. Il faut donc vérifier les points suivants.

-10 pour incohérence globale du texte, utilisation incorrecte des champs Pré/Post/Scenario etc... En particulier, -10% si les préconditions/hypothèses sont testées dans le scenario et ou si les étapes ne sont pas bien affectées à acteur ou système

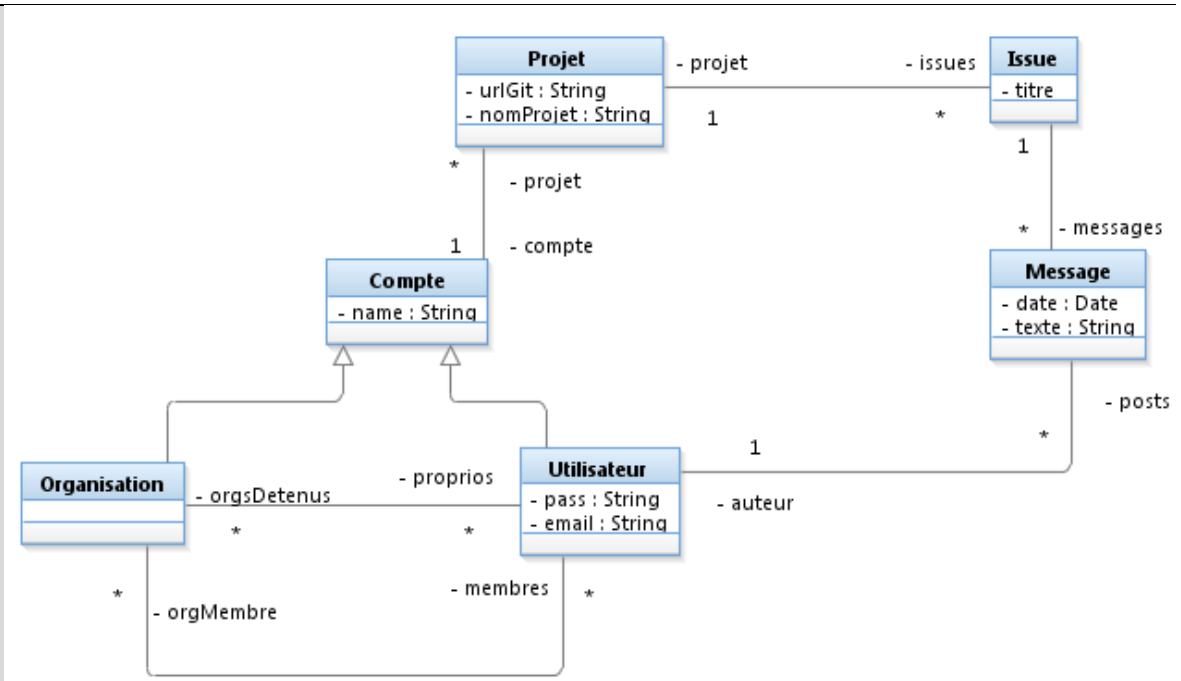
Erreurs fréquentes :

-5% à -10% on spécifie des pré/post conditions qui ne parlent de l'état du système

-10% on ne sait pas clairement qui du système ou de l'acteur fait l'action dans une étape du scenario

-15% :Spécification d'étapes hors système comme étapes du scenario

Question 2.3 : (3,5 points) Réalisez le diagramme de classes métier de la phase d'analyse. Vous justifierez tous vos choix, par un texte ou des annotations sur le diagramme. Ne modélisez pas la classe représentant le « Système », introduite dans l'approche en V du module.



Voici ma correction : si 10% pour un élément, ne donner que 5 si la formulation est incomplète (e.g. cardianlités mal renseignées, il manque des attributs parmi ceux recherchés,…).

Sur 100%: (s'il manque une partie des éléments donner 5%)

Projets (sur 30%) :

10 % : Un projet a un titre

10 % Un projet a une url dépôt git

10% Un projet a un propriétaire = compte perso ou orga

Issues (sur 30%):

10% issue a un titre, issue lié à un projet, projet lié * issue

10% Message a une date et un texte

10% message a un auteur = compte perso

Comptes/Orga (sur 40%):

10% Comptes (les deux types) ont des noms

10% Compte perso a password et email

10% Orga a des membres

10% Orga a des propriétaires

-10% à -20% éléments dynamiques qui n'ont pas à apparaitre, e.g. Acteurs.

-10% si associations orientées, compositions etc…

-10% si opérations sur les classes

-10% à 20% pour toute autre faute ou aberration

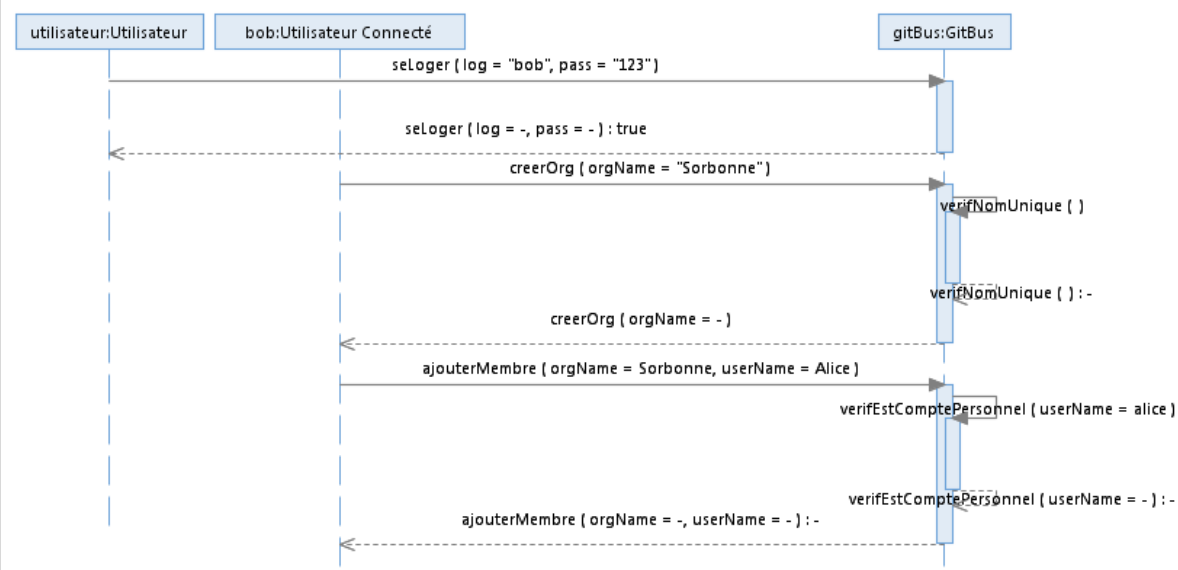
Question 2.4 : (2,25 pts)

A) Réalisez un diagramme de séquence de niveau analyse présentant le déroulement (scénario **nominal**) des étapes permettant à un utilisateur possédant un compte mais non connecté de créer une organisation et d'y ajouter un membre. On évitera de sur-spécifier les actions privées du système.

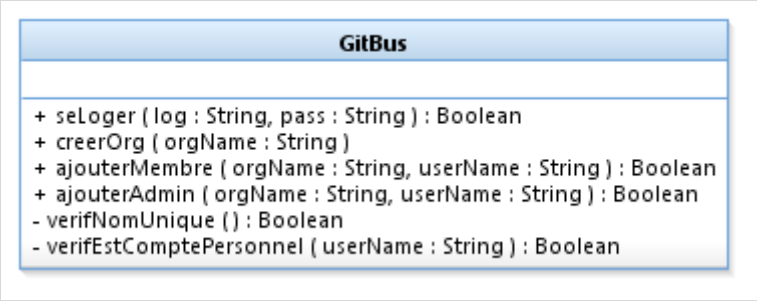
B) Dessinez la classe « système » afin de préciser les opérations identifiées dans cette séquence (signature, visibilité).

Essentiellement, on doit voir sur ce diagramme toute l'information circuler de l'acteur vers le système, sous une forme ou une autre. On ne doit pas nécessairement voir les opérations privées en self call

A priori on a un séquence très simple ici. Le corrigé est minimal mais suffisant.



Donc :



Barême :

A) 80%

+10% lignes de vie correctes : acteur (un ou deux, on n'exige pas les deux lignes de vie du corrigé) vs système

On cherche de l'information qu'on voit circuler de l'acteur vers le système

20 % Login

20% Création de l'org

20% une API raisonnable pour ajouter un membre (nom de la personne à ajouter + nom de l'org au minimum).

10% : modélisation de vérifications ou de l'affichage en self loop sur le système, modélisation d'actions pertinentes comme « enregistrer » du système, commentaires …

-10% on ne voit pas clairement que les lignes de vie sont des instances (notation o:Obj)

-20% si appel du système à une opération de l'acteur Agent. L'envoi asynchrone d'un message, ou une note expliquant qu'on considère que Joe représente l'acteur et son IHM => -10%. Cela reste incorrect. On cherche les responsabilités du système, pas des acteurs (donc externes au système).

-20% si on affecte des méthodes à des classes métier

B)

20% : signature(s) cohérentes avec le diag de séquence et réalisables, 0% dès qu'une incohérence est constatée. Les méthodes correspondant à d'autres use case sont tolérées mais ne donnent pas de points (hors sujet)

les self calls doivent être private (-10% si ce n'est pas le cas).

Question 2.5 : (2,25 pts) Ecrivez un test de validation couvrant la recherche d'un fichier particulier dans un repository.

TV042 : Test recherche

Contexte : Le projet « Test » est crée, il contient toto.txt.

Entrée : fichier à trouver : « toto »

Scenario :

1. L'utilisateur navigue vers la page du projet « Test »
2. Il clique saisit « toto » dans la barre de recherche et clique « par nom de fichier »

Résultat attendu : le système rapporte avoir trouvé un fichier contenant « toto » dans son nom, le fichier toto.txt. Le fichier est visible.

Moyens de vérification : Visuel.

Barême :

20% le contexte précise que ce qu'on recherche existe

10% le contexte ou le scenario amène dans la page du projet

20% on voit l'étape de saisie de la recherche

20% on voit les données de saisie, précisées dans l'entrée : toto. On ne choisit pas une recherche au hasard.

20% résultat attendu précise un ou des fichiers attendus dans la réponse

10% le moyen de vérification « visuel », a priori pas d'autre moyen de vérification, d'où l'importance de bien spécifier le RA.

-10% il existe des données saisies qui ne sont pas mentionnées dans la section « entrée », ou des données dans Entrée qui ne sont pas utilisées

-20% scenario difficilement réalisable, mentionne autre que les actions utilisateur, imprécis (il doit être reproductible sans réfléchir)

-25% le scenario mentionne des actions du système (autre que résultat attendu)