TELECOM SudParis

Corrigé et Barème Exemple de questions en Spécification et Conception

Les seuls documents autorisés sont ceux distribués en cours et en TP, et mis à disposition sur le site Web du module, ainsi que vos notes personnelles.

Notes en préambule :

- les questions qui suivent constituent un entraînement sur les parties « spécification » et « conception » pour le contrôle final de la deuxième session du module CSC4102. Elles sont extraites d'un ancien sujet. Dans le cadre du module CSC4102, pour couvrir l'ensemble du programme, nous y ajouterions une question sur la « préparation des tests » et une question sur la « qualité du modèle » (par exemple sur les patrons de conception) ;
- la durée estimée pour répondre à toutes les questions qui suivent est 2h ;
- un barème est donné à titre indicatif ;
- soyez concis et précis, et justifiez vos réponses par des commentaires appropriés ;
- soyez rigoureux dans la syntaxe UML ;
- **veillez à rendre une copie propre et lisible, avec une marge à gauche.**

1 Sujet

Le système « GestionTâchesDéveloppement » [1] permet d'organiser le développement en mode agile d'une application informatique (un projet). Un développeur doit s'inscrire pour être enregistré dans le système. Pour simplifier l'étude de cas, nous supposons que tous les développeurs sont déjà inscrits et nous ne réaliserons donc pas cette partie du système.

Un développeur crée le projet et organise le travail tout au long du projet : c'est l'organisateur du projet. Autrement dit, un des développeurs du projet est l'organisateur du projet. L'organisateur ajoute les membres.

Le développement est décomposé en tâches. Une tâche est modélisée par une carte : par exemple, une carte aura le titre « construire le diagramme de machine à états de la classe Document de la médiathèque ». Tout projet contient six types de listes de cartes : ÀFaire, EnCours, Fait, BesoinAide (un membre du projet déplace une carte dans cette liste pour « appeler à l'aide » les autres développeurs du projet), Abandonné (une carte dont la tâche devient non pertinente peut être abandonnée ; seul un développeur travaillant sur la carte ou l'organisateur peuvent décider d'abandonner une carte), et Archives (cette dernière liste permet de stocker les cartes anciennes ; seul l'oganisateur peut archiver une carte). Les listes de cartes sont créées automatiquement lors de la création du projet.

Une carte possède un titre, une estimation des efforts nécessaires (un entier) et une mesure des efforts fournis (un entier). Le travail d'une carte est réalisé par un ou plusieurs développeurs du projet. Ce travail est expliqué dans des commentaires de la carte : chaque commentaire est écrit par un développeur et correspond à une carte. De nouveaux commentaires peuvent être ajoutés au fur et à mesure de l'avancée des travaux.

Le déroulement du projet est divisé en sprints, c'est-à-dire que le développement de l'application est organisé en incréments, avec chaque incrément qui ajoute quelques fonctionnalités à l'application. Les sprints sont par convention nommés Sprint1, Sprint2, etc. Lorsqu'une carte est créée, elle est affectée à un sprint. Si le travail sur une carte n'est pas terminé dans ce sprint, la carte pourra être affectée à d'autres sprints ; la carte appartiendra alors à plusieurs sprints.

En début de sprint, les développeurs affectent des cartes existantes au sprint et créent de nouvelles cartes. Les nouvelles cartes sont dans la liste ÀFaire. Chaque développeur indique aussi les cartes sur lesquelles il souhaite travailler. Le début du travail sur une carte consiste à déplacer la carte vers la liste EnCours. La fin du travail sur une carte consiste à la déplacer vers la liste Fait et à indiquer l'effort fourni. Un développeur peut demander de l'aide sur une carte, la carte étant déplacée dans la liste BesoinAide. Lorsque le problème est résolu, la carte revient dans la liste EnCours. Pendant le sprint, l'organisateur ou le développeur d'une carte des listes ÀFaire, EnCours, BesoinAide peut décider de la déplacer vers la liste Abandonné, après avoir ajouté un commentaire pour expliquer l'abandon.

À chaque fin de sprint, les développeurs participent à la validation des cartes de la liste Fait. Valider une carte a pour effet le déplacement de la carte vers la liste Archives. Cette opération est effectuée par l'organisateur du projet. Si la validation est refusée, la carte est soit abandonnée (l'organisateur ajoute un commentaire à la carte et la déplace dans la liste Abandonné, puis dans la liste Archives) soit remise dans la liste EnCours ou dans la liste ÀFaire (l'organisateur ajoute un commentaire et déplace la carte). Enfin, l'organisateur déplace les cartes de la liste Abandonné vers la liste Archives si l'abandon est accepté par tous, sinon vers la liste ÀFaire.

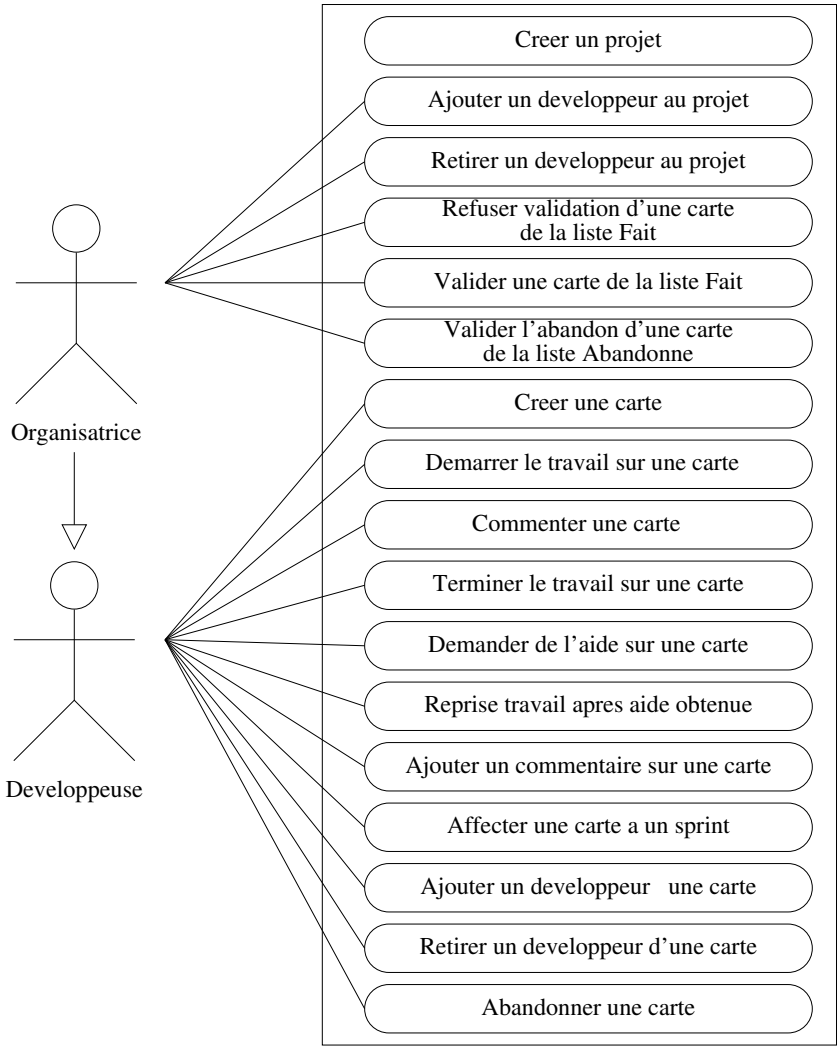
1. Ce sujet est largement inspiré des applications Web comme Trello (`trello.com`) ou Wekan (`github.com/wekan/wekan`), ainsi que des méthodes agiles comme Scrum ou Kanban. Aucune connaissance préalable sur le développement en mode agile n'est nécessaire pour réaliser cette étude de cas.

2 Questions

Question 1 : 3pts Décrivez les acteurs et les fonctionnalités significatives du système par un diagramme de cas d'utilisation.

Réponse :

La figure suivante présente les acteurs et les cas d'utilisation du système.



Erreurs ou faiblesses fréquemment observées :

- oubli de certains cas d'utilisation importants
- oubli de la généralisation spécialisation entre acteurs alors que les rôles sont explicitement reliés dans l'énoncé

Barème de correction sur 3 points :

```
Répartition :
1   respect de la notation UML (délimitation du système, icone de l'acteur,
    lien de communication, phrase verbale, cadre ovale autour des cas d'utilisation)
0,5 deux acteurs
0,5 généralisation spécialisation
0,5 au moins 4 cas d'utilisation pour l'organisateur
0,5 au moins 5 cas pour le développeur
```

Question 2 : 2pts Choisissez les classes qui vous semblent devoir faire partie de la modélisation de ce problème. Décrire textuellement ces classes et les attributs associés à chaque classe : donner le nom des classes et de leurs principaux attributs (et des explications textuelles uniquement quand cela vous semble nécessaire).

Réponse :

Les **classes** et — attributs — obtenus après analyse du texte sont les suivants :

- **GestionTâchesDéveloppement** : classe « interface » du système, patron de conception Façade ;
- **Projet** : — nomProjet — ;
- **Développeur** : — nom, prénom — ;
- **Sprint** : — nomSprint — ;
- **Carte** : — titre, description, effortEstimé, effortFourni — ;
- **Commentaire** : — texte — ;
- **Liste** : — — peut être une classe abstraite ;
- **À Faire** : — — classe fille de Liste ;
- **EnCours** : — — classe fille de Liste ;
- **Fait** : — — classe fille de Liste ;
- **BesoinAide** : — — classe fille de Liste ;
- **Abandonné** : — — classe fille de Liste ;
- **Archives** : — — classe fille de Liste.

Erreurs ou faiblesses fréquemment observées :

- acteur Organisateur comme classe du système
- commentaire en attribut
- dans la classe Commentaire, attributs nom et prénom ainsi que nom de la carte

Variantes possibles :

-

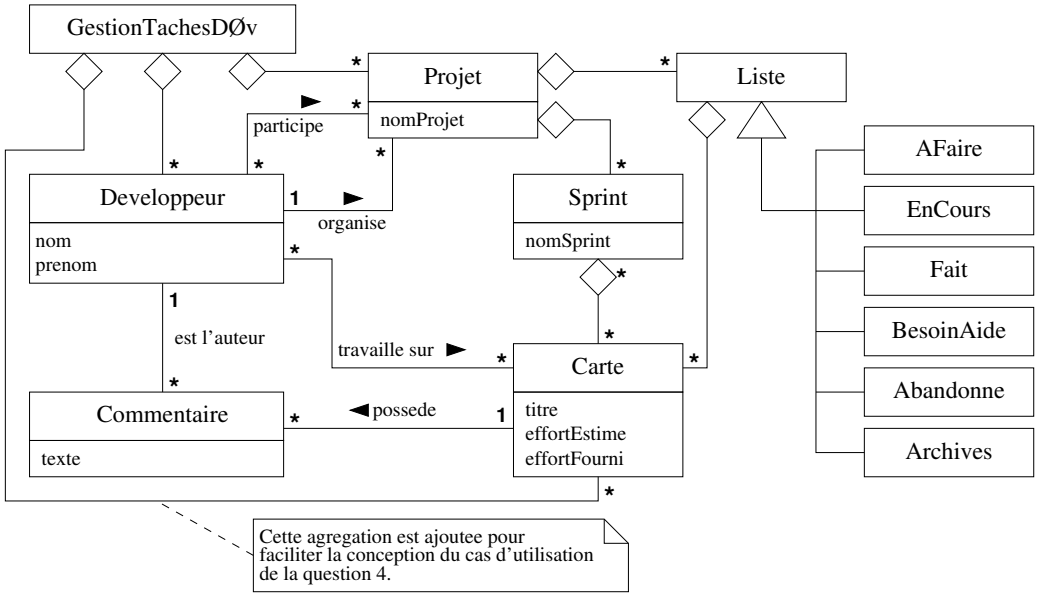
Barème de correction sur 2 points :

```
Répartition :
0,5 classes Projet, Développeur, Sprint, Liste, Carte (0,25 par manque)
0,5 pas de classe Organisateur, ni d'attribut organisateur
0,5 classe Commentaire
0,5 au moins les attributs titre, effortEstimé, effortFourni pour la classe Carte
```

Question 3 : 5pts Construisez un diagramme de classes qui représente le système. Dans ce diagramme, prenez soin de préciser les noms des associations, les rôles, les multiplicités et les sens de navigation des associations lorsque cela s'avère nécessaire. Vous n'avez pas besoin de remettre les attributs dans les classes du diagramme de classes.

Réponse :

La figure suivante présente le diagramme de classes proposé comme corrigé-type.



Erreurs ou faiblesses fréquemment observées :

- inversion du sens des agrégations ;
- aucune généralisation spécialisation ;
- multiplicités inversées ou absentes ;
- multiplicité pour exprimer qu'un développeur participe à plusieurs projets ;
- multiplicité pour exprimer qu'une carte peut être affectée à plusieurs sprints ;
- une association (erronnée) entre les classes Liste et Sprint
- ne pas factoriser l'association entre Sprint et la classe parente Liste
- pas de classe Liste, mais une généralisation spécialisation à partir de la class Carte : le concept « métier » Liste est très prégnant dans l'énoncé

Variantes possibles :

- classes d'association Commentaire ;
- plus d'agrégations de la façade vers les classes du système

Barème de correction sur 5 points :

```
Répartition :
1   qualité UML du diagramme
(notation UML: classe, nom association, généralisation spécialisation,
agrégation, -0,25 par association qui correspond à un cas d'utilisation) ;
0,5 la classe du patron Façade GestionTâcheDéveleppoment et son utilisation ;
0,5 associations participe à et organise entre Développeur et Projet
0,5 association entre Développeur et Carte
0,5 généralisation spécialisation Liste...
0,5 association entre Projet et Sprint
0,5 association entre Projet et Liste
0,5 classe Commentaire avec deux associations vers Développeur et vers Carte
0,5 la plupart des multiplicités sont correctes (-0.25 par erreur)
-0,5 (malus) non-propreté du diagramme (difficulté de lecture et de compréhension)
```

Question 4 : 3pts Nous nous intéressons à la conception de la classe correspondant à une carte. Donnez tous les attributs de cette classe avec leur visibilité, leur type, et leur éventuelle valeur par défaut.

Réponse :

- `titre : String`
- `effortEstimé : integer`

- effortFourni : integer
- sprints : collection @Sprint
- développeurs : collection @Développeur
- commentaires : collection @Commentaire
- liste : @Liste

Erreurs ou maladresses fréquemment observées :
— non conforme à la liste des attributs donnée à la question 2
— non conforme au diagramme de classes : navigations ou multiplicités
Variantes possibles :
— selon la réponse à la question 2,
— selon le diagramme de classes donné dans la question 3.

Barème de correction sur 3 points :

```
Répartition :
0,5 attributs privés
0.5 bon typage des attributs hors traduction des associations
1   conformité avec les attributs donnés dans la question 2
1   conformité avec les associations du diagramme de classes
    MAIS moins 0,5 par erreur (donc, max 1 et pas de min)
```

Question 5 : 5pts Donnez le diagramme de séquence qui correspond au cas d'utilisation : « Refuser la validation d'une carte de la liste Fait » dans le cas où la carte est remise dans la liste EnCours. Les arguments de l'opération refuserValidationCarteFait() sont la référence sur l'organisateur, la référence sur la carte à valider, la référence sur la liste EnCours, et le commentaire du refus de la validation.

Nous supposons que les références sont non nulles et correctes : il n'est pas nécessaire de faire des tests pour ces arguments.

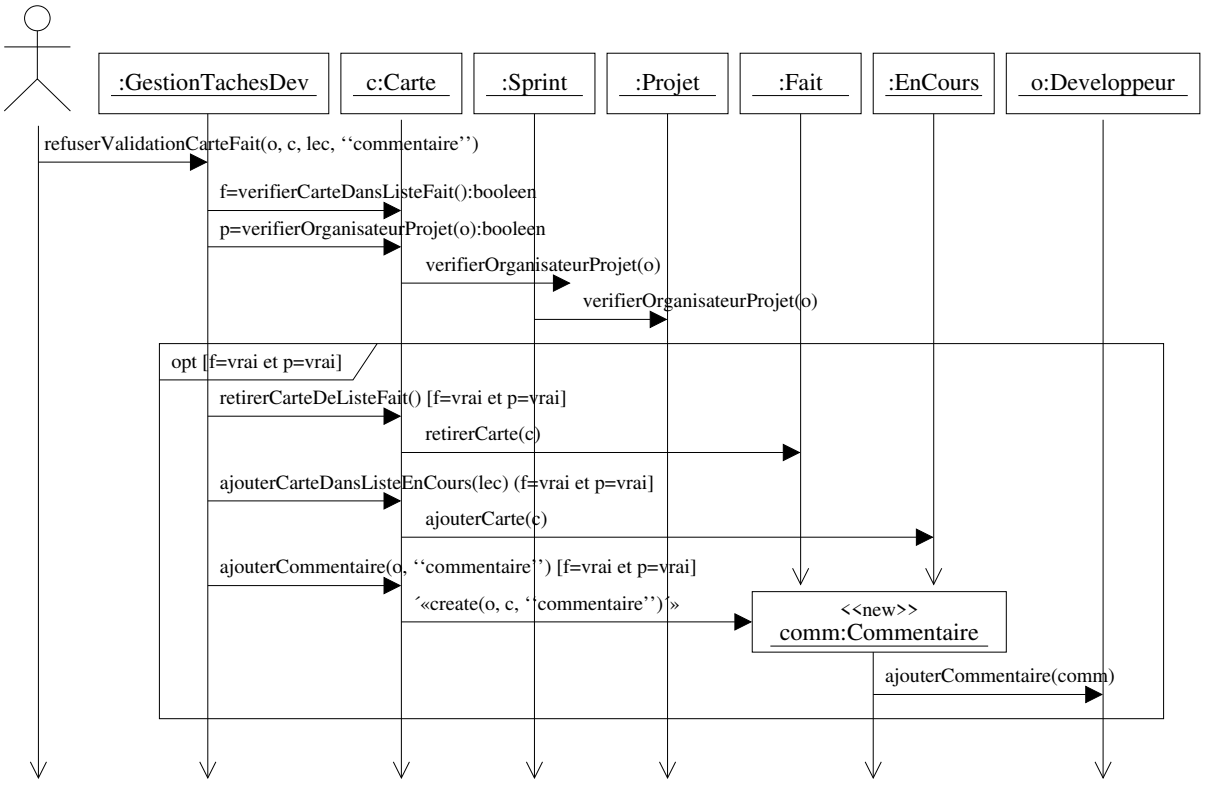
Nous vous demandons de détailler les actions correspondant à ce cas d'utilisation sous forme textuelle avant de réaliser le diagramme.

Réponse :

La figure suivante présente le diagramme de séquence associé au cas d'utilisation. La description informelle du cas d'utilisation est la suivante :

— 0. appel de l'acteur organisateur sur la façade
— 1. appel vérifierCarteDansListeFait() sur la carte
— 2. appel vérifierOrganisateurProjet(o) sur la carte
— 2.1. appel vérifierOrganisateurProjet(o) sur le sprint
— 2.1.1. appel vérifierOrganisateurProjet(o) sur le projet
— 3. appel retirerCarteDeListeFait() sur la carte
— 3.1 retirerCarte(c) de la liste Fait
— 4. appel ajouterCarteDansListeEnCours() sur la carte
— 4.1. ajouterCarte(c) de la liste EnCours
— 5. appel ajouterCommentaire(o, "commentaire") sur la carte
— 5.1. appel create(o, c, "commentaire") de la classe Commentaire, instance comm
— 5.1.1. appel ajoutCommentaire(comm) sur o

À faire : passer du diag. de communication au diag. de séquence, car c'est plutôt un diag. de séquence qui est attendu.



Erreurs fréquemment observées :

- non respect de la notation UML : objet, etc.
- non respect des relations du diagramme de classes
- non respect du séquencement ou des numérotations
- pas de test de valeurs retournées ou pas d'utilisation des valeurs retournées
- pas de fragment avec condition ou pas de condition sur les messages
- non respect des associations bidirectionnelles

Variantes possibles :

- diagramme de séquence
- autre expression selon le diagramme de classes

Barème de correction sur 5 points :

```
Répartition :
1   logique correcte, même si seulement exprimée dans l'explication
    textuelle (partie tests et partie actions du cas d'utilisation)
1   conformité au diagramme de classes (parcours d'associations existantes et navigabilité possible)
    + qualité de l'UML
0.5 vérifier carte dans la liste Fait
0.5 vérifier que c'est l'organisateur du projet
0.5 retrait de la liste Fait
0.5 ajout dans la liste EnCours
0.5 construction du nouveau Commentaire
0.5 appel sur le développeur organisateur pour l'assocation bi-directionnelle
```

Question 6 : 2pts Nous nous intéressons à l'opération de la classe GestionTâchesDéveloppment du cas d'utilisation de la question 5 : « Refuser la validation d'une carte de la liste Fait » dans le cas où la carte est remise dans la liste EnCours. Donnez l'algorithme de cette opération.

Réponse :

```
refuserValidationCarteFait(@Développeur o, @Liste l, string commentaire) {
        si (c.vérifierCarteDansListeFait() et c.vérifierOrganisateurProjet(o)) {
                retirerCarteDeListeFait()
                ajouterCarteDansListeEnCours()
                ajouterCommentairee(o, commentaire)
        } sinon {
                cas à étudier avec le client, car rien dans le cahier des charges
}
```

Erreurs ou maladresses fréquemment observées :

- non conforme au diagramme de séquence
- oubli vérifications
- oubli appels pour les associations bi-directionnelles

Variantes possibles :

- selon la réponse à la question du diagramme de séquence ou de séquence,
- selon le diagramme de classes.

Barème de correction sur 2 points :

```
Répartition :
barème indicatif à adapter selon la solution (questions~3 et~5)
0.5 correspond au diagramme de séquence
0,5 vérifierCarteDansListeFait et vérifierOrganisateurProjet
0,5 retirerCarteDeListeFait et ajouterCarteDansListeEnCours
0,5 ajouterCommentaire
```