TD2- Conception Avancée en UML

I. Check-List avant de passer au TD 2

Avant de passer au problème du TD 2, vérifiez que votre solution au TD 1 répond bien à tous les critères suivants.

Rappel : Votre modélisation UML est indépendante du langage ciblé. Vous devriez pouvoir implémenter cette solution dans le langage de votre choix, de préférence cependant orienté objet.

1. Erreurs fréquentes et "graves"

a. Le "SI" n'est ni un acteur, ni un cas d'utilisation (il est représenté par l'ensemble des cas d'utilisation).
b. La BD (base de données) n'est pas un acteur. La BD peut être un moyen d'implémentation du stockage des informations des objets.
c. Pour chaque cas d'utilisation, les acteurs sont bien les personnes ou systèmes informatiques qui interagissent directement avec le Système informatique, *i.e.*, la personne qui "tape" sur le clavier, pas celle qui regarde par-dessus l'épaule !
d. Si un cas d'utilisation ne fait pas intervenir le système informatique, ce n'est pas un cas d'utilisation (c'est une activité du métier qui n'est pas informatisée).
e. Les relations entre les cas d'utilisation ne sont jamais des relations d'ordre. Il n'y a que spécialisation, inclusion ou extension.
f. Attention vos cas d'utilisation ne doivent pas se recouvrir : si vous pensez qu'un UC est inclus dans un autre ou le spécialise, vous devez l'expliciter.
g. Attention, un acteur n'est jamais un objet du système informatique (il n'est pas représenté par une classe). Son compte pour un humain, un proxy pour un acteur système est peut-être un objet du système, mais pas l'acteur lui-même.
h. A ce stade, si, dans une classe, vous conservez un ID qui vous permet de retrouver un autre objet, c'est qu'il y a une association entre votre classe et la classe de cet autre objet (idem si vous conservez un tableau d'ID 😱). Attention vous anticipez peut-être beaucoup trop tôt sur la représentation informatique du problème que vous êtes en train de modéliser. Les attributs dans les classes UML ne devraient contenir que des types de base (integer, String, Date). Toutes les autres informations sont représentées par des associations.
i. Tout ce qui est représenté doit respecter la sémantique du langage UML, par exemples
    i. Sens et représentation des relations entre UC ou classes
    ii. Placement des rôles et multiplicité dans les associations ; en l'absence de ces informations, votre « trait » ne sert à rien !
    iii. Un message dans un diagramme de séquence ne part pas "tout seul".
    iv. Les lignes de vie commencent en haut (l'objet existe déjà) ou à la création et dans ce cas, elles commencent par le message de création de l'objet.

2. Bonnes Pratiques

a. Le glossaire doit essentiellement définir les termes du métier, donc ici ceux d'un système de commandes en ligne. Il permet de fixer le vocabulaire et d'éviter les ambiguïtés. S'il y a des termes différents pour parler d'un même concept dans la spécification, dans la modélisation, un seul terme fixé dans le glossaire est utilisé PARTOUT. On ne doit donc pas trouver dans les classes, les diagrammes de séquence ou les UC, un même concept sous des noms différents.
b. Dans le diagramme de séquence, s'il met en œuvre des interactions avec un utilisateur, vous représentez l'"objet informatique" qui supporte les interactions (souvent une interface graphique, mais il peut s'agir d'une manette de jeux).
c. Au lieu d'utiliser une ligne de vie 'System", vous modélisez plutôt une classe Facade (STEats ou un nom dans ce genre) qui capture la connexion à votre backend, qui est la seule partie de votre architecture qui nous intéresse pour l'instant. Ceci n'est pas une obligation en UML, c'est juste une facilité pour faire des diagrammes de séquence simples et cohérents.
d. Attention, une classe ne doit pas concentrer toute la logique, il faut distribuer intelligemment les responsabilités des traitements sur les autres objets (*rappelez-vous les patrons GRASP*).
e. Un diagramme de séquence ne porte que sur un seul cas d'utilisation.
f. Pensez à terminer vos diagrammes de séquence en interaction avec un acteur humain par un message à l'humain. (Un système informatique qui ne dit pas ce qu'il fait, c'est exaspérant!)

3. Périmètre et cohérence

a. Le niveau de diagramme de cas d'utilisation doit permettre d'identifier les grandes fonctionnalités, pas les détails des scénarii, et inversement être suffisamment détaillé pour bien identifier les acteurs et les fonctionnalités.
b. Un diagramme de séquence est une vision du déroulement d'un des scenario d'un cas d'utilisation. On doit naturellement y retrouver les acteurs liés au cas d'utilisation comme déclencheur ou récepteur de messages. Assurez que vos diagrammes de séquence ne traversent pas plusieurs cas d'utilisation.
c. Dans un diagramme de séquence,
    i. les envois de message ne sont possibles qu'entre des instances de classes liées entre elles (par association, agrégation), ou des objets obtenus par retour de méthodes.
    ii. À l'inverse, les méthodes utilisées dans les diagrammes de séquence doivent apparaitre dans le diagramme de classes (dans les classes des objets récepteurs !).

II. Travail à réaliser

1. Vérifiez vos modèles

- ❑ Commencez par vérifier que vous n'avez commis aucune des erreurs présentes dans la check-list et que vous avez respecté les bonnes pratiques.

2. Complétez vos diagrammes de cas d'utilisation

- ❑ Déterminer les ***relations entre les cas d'utilisation*** pour en particulier bien capturer les éléments communs (*cas d'héritage*), les facilités de navigation entre cas d'utilisation (*extends*) et les exigences (*include*). N'en ajoutez pas artificiellement.

3. Complétez votre diagramme de séquence

Diagramme de séquence : *passer une commande individuelle, dans le contexte d'une commande de groupe dont l'horaire de livraison est fixé* (cf. TD1)

Le diagramme de séquence doit inclure :

- ❑ *Respect de l'horaire de livraison :* Prenez en compte les temps de préparation et la charge des restaurants. Attribuez la recherche des restaurants et des menus à des objets dédiés (pas "System" ou "STEats").
- ❑ *Création et enregistrement de la commande* **:** Cette tâche doit être réalisée par un objet qui en a la responsabilité.
- ❑ *Appel au service externe de paiement :* Assurez-vous que le service de paiement est bien intégré.

De plus :

- ❑ Ne détaillez pas les cas d'erreur, mais posez des notes pour les identifier.
- ❑ N'intégrez pas la connexion au Système qui est une précondition à ce cas d'utilisation.
- ❑ Ne traversez pas d'autres cas d'utilisation que celui modélisé, mais n'oubliez pas de notifier les éléments de votre système qui ont besoin de savoir qu'une nouvelle commande a été passée.

4. Mettez à jour le glossaire si nécessaire

- ❑ Vous pouvez peut-être à cette étape préciser votre vocabulaire.

5. Complétez votre diagramme de classes au niveau du domaine.

Vos diagrammes de classes doivent vous permettre d'identifier les éléments qui jouent un rôle dans le système, y compris leurs comportements.
Une bonne base de modélisation facilite ensuite la programmation et évite des mécompréhensions entre les membres de l'équipe.

- ❑ En cohérence avec votre diagramme de séquence, complétez votre modélisation en représentant les concepts et objets impliqués.
- ❑ Vous avez peut-être déjà pressenti des "Design patterns" qui pourraient s'appliquer. Si c'est le cas, prenez des notes, même éventuellement directement dans vos diagrammes de classes.

6. Identifier les limites additionnelles

- ❑ S'il y en a que vous ne traiterez pas (**G.6**) ou les **hypothèses** que vous avez fait sur le projet (E.4)

III. RENDU A2

Nous mettons ici ce qui est attendu dans ce rendu sur moodle.

La structure du rapport en pdf doit suivre le plan suivant :

1. Le nom des étudiants impliqués dans l'équipe sur la première page avec leur rôle dans l'équipe
2. Une section d'une page maximum résumant
    1. vos hypothèses de travail (E.4)
    2. les limites identifiées (G.6)
    3. et si certaines exigences requises restent non étudiées.
3. Une section pour chacun des éléments suivants, avec éventuellement des explications supplémentaires propres aux diagrammes, si cela est nécessaire. Si vous le souhaitez, vous pouvez ajouter des diagrammes.
    1. Le glossaire
    2. Le diagramme de cas d'utilisation
    3. Le diagramme de classes
    4. Le diagramme de séquence
    5. La maquette

Le périmètre du rapport à rendre correspond à l'ensemble des fonctionnalités demandées dans les TD 1 et TD 2.

N'oubliez pas de vérifier les différents éléments de la check-list fournie avec le sujet du TD 2.