LOG1000 – Ingénierie logicielle TP #2 :

Objectifs de la modélisation UML :

• À partir des requis, concevoir le design d'un logiciel en utilisant la modélisation UML.

• Réaliser des diagrammes de cas d'utilisation, de séquence et de classe.

• Comprendre les concepts d'association, d'agrégation, de composition et d'héritage.

Déroulement du travail pratique (TP)

Le TP va couvrir une partie de la conception logicielle essentielle afin de transformer la vision d'un client en un produit logiciel fiable, maintenable et flexible. Vous devrez d'abord bien analyser la vision du client et en extraire des cas d'utilisation montrant les différents acteurs, opérations et relations. Ces cas d'utilisation refléteront chacun un scénario particulier. Par la suite, suite à l'obtention des cas, vous élaborez des diagrammes de séquence décrivant de manière chronologique le déroulement de certains cas. Ayant bien identifié les différents objets constituant le système, il sera par la suite possible de bâtir un diagramme de classe montrant les liens entre les classes essentielles et leurs méthodes.

Rédaction du rapport

Votre rapport sera un document PDF contenant les divers diagrammes et les réponses aux questions demandées. Le rapport sera remis dans un dossier nommé TP2 dans votre répertoire Git. N'oubliez pas de vérifier après la remise si votre rapport est beau et bien visible sur le serveur et pas seulement sur votre copie locale.

UMLet

Pour effectuer vos diagrammes, vous devez utiliser l'outil UMLet que vous devez l'installer dans votre dossier personnel. Pour vous renseigner aux directives d'installation et d'utilisation de cet

outil, veuillez visiter le lien suivant (important : téléchargez la version stand-alone) : http://www.umlet.com/. Voici un tutorial qui explique comment faire un diagramme de cas d'utilisation et un diagramme de séquence en utilisant l'outil UMLet : https://www.youtube.com/watch?v=HVSxE296JLc.

Trucs et astuces:

- Les acteurs ne sont pas des objets.
- Utilisez des verbes infinitifs au début des noms de cas d'utilisation et de fonctions.
- Les objets et les classes sont indépendants de hardware/plateforme.
- Voir aussi les astuces sur Moodle.

PARTIE 1 : Conception des diagrammes UML

Mise en Contexte :

Une petite équipe de contributeurs chez Savoir-faire Linux a eu l'idée de développer un application open source(Ring), qui permet les communications peer-to-peer tout en restant libre (sources ouvertes). Ring permet de passer des appels téléphoniques avec des fonctionnalités additionnelles comme la messagerie instantanée, le transfert de fichiers et la visioconférence, un peu comme Skype, mais sans y aller vers des technologies brevetées. Dans ce cadre l'équipe développement de RING a demandé de l'aide aux étudiants de l'École Polytechnique de Montréal.

Pour ce laboratoire, on vous demande de concevoir un système logiciel de communication peer-to-peer (Ring) https://ring.cx/.

Votre travail dans le cadre de ce TP est de réaliser en partie cette modélisation qui se divise en trois aspects (tel que décrit ci-dessus) :

- Aspect 1 : Le diagramme des cas d'utilisation;
- Aspect 2 : Le diagramme de séquence;

• Aspect 3 : Le diagramme de classes.

Afin que vous puissiez vous familiariser avec le fonctionnement de ce genre d'application, vous avez ci-dessous quelques informations supplémentaires. Ainsi, les objectifs de cette application sont :

• La gestion des contacts, ce qui permet à l'utilisateur de

1. Partager son Ring ID
2. Ajouter de nouveaux contacts
3. Créer des groupes de contact
4. Supprimer un contact
5. Bloquer un contact

• La gestion de communication

1. Appeler un contact ou un groupe de contacts.
2. Accepter un appel
3. Créer un appel en groupe.
4. Refuser un appel
5. Terminer un appel
6. Envoyer un message à un contact
7. Envoyer un fichier à un contact
8. Gérer les historiques de conversations et des appels

• La gestion des profils

1. Créer un profil.
2. Changer les coordonnées.

Tout ce qui est décrit au-dessus sont les objectifs basics et essentiels qui doivent se trouver dans l'application de communication Ring.

Aspect 1 : Le diagramme des cas d'utilisation (35 pts)

Afin de réaliser le diagramme des cas d'utilisation du système, basez-vous sur la mise en situation et les exigences fonctionnelles que vous en extrayez.

Q1 : Faites un diagramme de cas d'utilisation basé sur les exigences discutées au-dessus.

Votre diagramme devrait contenir l'ensemble des acteurs identifiés et l'ensemble des cas d'utilisation qui vous semblent pertinents. N'oubliez pas les relations entre les cas !

Q2 : Décrivez textuellement les 3 cas d'utilisation qui vous semblent les plus essentiels en suivant le modèle ci-dessous (cf. exemple du cours), et expliquez aussi pourquoi vous avez choisi ces 3 cas :

Titre	
Acteurs impliqués	
Préconditions	
Post conditions	
Scénario principale	Liste numérotée
Scénarios d'exception	Liste numérotée

Aspect 2 : Le diagramme de séquence (35 pts)

Construire 3 diagrammes de séquence, selon les scénarios ci-dessous :

• **Scénario 1** : Créer un profil initial.

• **Scénario 2** : partager son Ring ID

• **Scénario 3** : Initier un appel en groupe, c.-à-d. ajouter les différentes personnes, établir une connexion et commencer l'enregistrement des messages de clavardage (« chat log »).

Le diagramme de séquence permet de visualiser le déroulement chronologique de chacun des cas d'utilisation. Il vous faut considérer :

• Tous les objets impliqués et les messages qu'ils s'envoient. Choisissez un nom clair pour chaque classe, méthode et argument dans vos diagrammes.

• L'ordre chronologique des événements doit se dérouler du haut vers le bas.

Aspect 3 : Diagramme de Classes (20 pts)

Maintenant que vous avez identifié les objets qui constituent le système, vous êtes en mesure de réaliser le diagramme de classe contenant toutes les classes pour implémenter votre part du système. Même pour les cas qu'ils n'ont pas été élaborés, il faut ajouter les classes nécessaires.

Q1 : Identifiez chacune des classes qui constituent votre système informatique.

Q2 : Dessinez le diagramme de classe représentant votre logiciel. Il vous faut considérer :

• Tous les attributs et méthodes essentiels de chacune des classes.

• Les relations entre les classes : agrégation, composition, association ou héritage.

• Les quantités pour chaque relation, s'il y a lieu (par exemple, un-à-plusieurs, plusieurs-à-plusieurs, etc.)

PARTIE 2 : Rétro-ingénierie de diagramme de classe à partir du code (20 pts)

Pour cette partie du TP, nous progressons à décortiquer le projet open source Ring que vous connaissez déjà de la partie 2 du TP1, où vous avez cloné la composante Ring-deamon. Maintenant, on vous demande de dessiner le diagramme de classe contenant les classes suivantes, dont le code source est dans le dossier src/ :

• Recordable, Conference, Config/Serializable, Account, Manager, PluginManager, Call, et Call_factory.

Il vous faut considérer :

• Les attributs et méthodes essentiels qui montrent les liens entres les classes.

• Les relations entre les classes : agrégation, composition, association ou héritage.

Références http://www.uml-diagrams.org/

Considérations importantes pour la fin du TP

Toujours faire un « **git add** » des nouveaux fichiers, un « **git commit** » et un « **git push** » de vos dernières modifications pour que l'on puisse voir la dernière version de votre travail lors de la correction. Si vous ne faites pas de commit, il se peut que l'on évalue une version différente de votre TP local sur le serveur Git.

Vérification que vos travaux sont présents dans le répertoire Git du serveur : Vous pouvez utiliser la commande « git ls-files » afin de voir les fichiers dans le dernier snapshot de l'entrepôt Git lui-même. Remplacez XX par le numéro de votre équipe. Ce que vous verrez dans cette liste correspond à ce que l'on verra pour la correction.

!! DATE LIMITE DE REMISE : 22 février 2017 à midi!!

Tout ce qui est « commit » après cette date ne sera pas considéré dans la correction. !! Pénalités pour retard : 10% par jour !!