Lycée LOUIS JOUVET Accompagner pour réussir	**Langage de modélisation** ***Cours UML***	***BTS CIEL*** ***Semestre 1*** ***2024_2025***

I/ Histoire et définitions de l'UML

UML (Unified Modeling Language ou langage unifié de modélisation) est un langage graphique destiné à la modélisation de systèmes et des processus. C'est un langage basé sur l'approche par objets.

UML est unifié car il provient de plusieurs notations qui l'ont procédé. Aujourd'hui l'UML est promu par l'OMG (Object Management Group), un consortium de plus de 800 sociétés et universités actives dans le domaine des technologies de l'objet.

UML est aujourd'hui devenu un langage de modélisation très répandu, notamment grâce à sa richesse sémantique qui le rend abstrait de nombreux aspects techniques.

Avec un langage de programmation, la description des objets est réalisée de façon formelle selon une syntaxe rigoureuse. Cette syntaxe présente l'inconvénient de ne pas être lisible par des non-programmeurs. Les humains, à la différence des machines, préfèrent les langages graphiques pour représenter des abstractions car ils peuvent ainsi les maîtriser plus facilement et plus rapidement, et obtenir une vue d'ensemble d'un système en un temps beaucoup plus court.

II/ Principes et définitions de base

1) Acteurs

Un acteur représente un rôle joué par une entité externe (utilisateur humain, dispositif matériel ou autre système) qui interagit directement avec le système étudié.

Un acteur peut consulter et/ou modifier directement l'état du système, en émettant et/ou en recevant des messages susceptibles d'être porteurs de données.

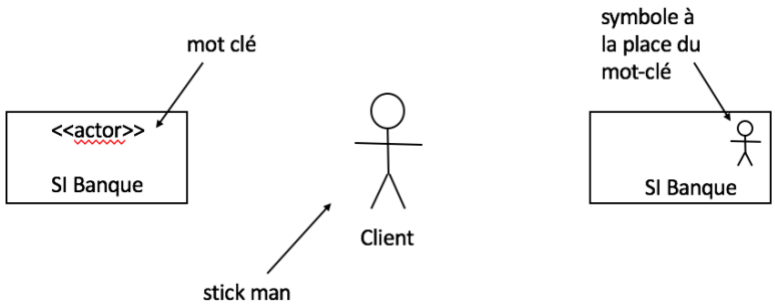
Comment les identifier ?

Les acteurs sont systématiquement :

- Les utilisateurs humains directs : il faut identifier tous les profils possibles, sans oublier l'administrateur, l'opérateur de maintenance, etc. …
- Les autres systèmes connexes qui interagissent aussi directement avec le système étudié, souvent par le biais de protocoles bidirectionnels.

Comment les représenter ?

La représentation graphique standard de l'acteur en UML est l'icône appelé *stickman*, avec le nom de l'acteur sous le dessin. On peut également figurer un acteur sous la forme rectangulaire d'une classe, avec le mot clé <<actor>>. Une troisième représentation (intermédiaire entre les deux premières) est également possible, comme cela est indiqué ci-après.



Une bonne recommandation consiste à faire prévaloir l'utilisation de la forme graphique du *stickman* pour les acteurs humains et une représentation rectangulaire pour les systèmes connectés.

2) Le diagramme des cas d'utilisation

Un cas d'utilisation (<< *use case* >>) représente un ensemble de séquences d'actions qui sont réalisées par le système et qui produisent un résultat observable intéressant pour un acteur particulier.

Chaque cas d'utilisation spécifie un comportement attendu du système considéré comme un tout, sans imposer le mode de réalisation de ce comportement. Il permet de décrire ce que le futur système devra faire, sans spécifier comment il le fera.

Comment les identifier ?

L'objectif est le suivant : l'ensemble des cas d'utilisation doit décrire exhaustivement les exigences fonctionnelles du système. Chaque cas d'utilisation correspond donc à une fonction métier du système, selon le point de vue d'un de ses acteurs.

Pour chaque acteur, il convient de :

- Rechercher les différentes intentions métier avec lesquelles il utilise le système
- Déterminer dans le cahier des charges les services fonctionnels attendus du système.

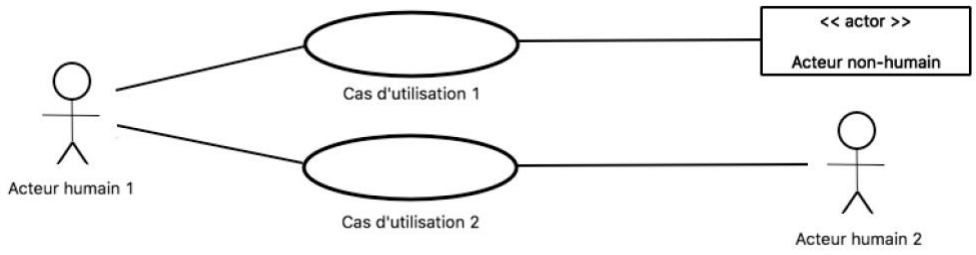
Il faut nommer les cas d'utilisation par un verbe à l'infinitif suivi d'un complément, du point de vue de l'acteur.

Comment les analyser ?

Pour détailler la dynamique du cas d'utilisation, la procédure la plus évidente consiste à recenser de façon textuelle toutes les interactions entre les acteurs et le système. Le cas d'utilisation doit avoir un début et une fin clairement identifiés. Il faut également préciser les variantes possibles, telles que les différents cas nominaux, les cas alternatifs, les cas d'erreurs, tout en essayant d'ordonner séquentiellement les descriptions, afin d'améliorer leur lisibilité.

Comment les représenter ?

Le diagramme de cas d'utilisation est un schéma qui montre les cas d'utilisation (ovales) reliés par des associations (lignes) à leurs acteurs (icône du « *stickman* », ou représentation graphique équivalente). Chaque association signifie simplement « participe à ». Un cas d'utilisation doit être relié à au moins un acteur.



III/ Les relations entre les cas d'utilisation

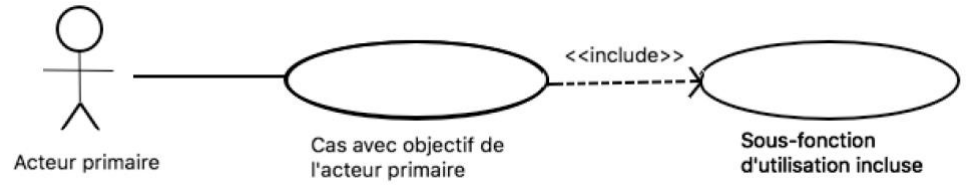
1) La relation d'inclusion

La relation d'inclusion sert à enrichir un cas d'utilisation par un autre cas d'utilisation. Cet enrichissement est réalisé par une inclusion impérative, il est donc systématique.

Le cas d'utilisation inclus existe uniquement dans ce but. En effet, il ne répond pas à un objectif d'un acteur primaire. Un tel cas d'utilisation est une sous-fonction.

L'inclusion sert à partager une fonctionnalité commune entre plusieurs cas d'utilisation. Elle peut également être employée pour structurer un cas d'utilisation en décrivant ses sous-fonctions.

Dans le diagramme des cas d'utilisation, cette relation est représentée par une flèche pointillée munie du stéréotype « include ».



2) La relation d'extension

Comme la relation d'inclusion, la relation d'extension enrichit un cas d'utilisation par un cas d'utilisation de sous-fonction. Cet enrichissement est analogue à celui de la relation d'inclusion, mais il est optionnel.

L'extension se fait dans le cas d'utilisation de base, en des points précis et prévus lors de la conception, appelés points d'extension.

L'application de chaque extension est décidée lors de déroulement d'un scénario. Par conséquent, le cas d'utilisation de base peut être employé sans être étendu.

Comme pour l'inclusion, l'extension sert à structurer un cas d'utilisation ou à partager un cas d'utilisation de sous-fonction.

IV/ La spécialisation et la généralisation des cas d'utilisation

Comme pour les classes d'objets, il est également possible de spécialiser un cas d'utilisation en un autre. On obtient ainsi un sous-cas d'utilisation.

De façon analogue aux classes, le sous-cas hérite du comportement du sur-cas d'utilisation. Un sous-cas d'utilisation hérite également des associations liant le sur-cas aux acteurs ainsi qu'aux relations d'inclusion et d'extension du sur-cas.

Le sur-cas d'utilisation est souvent abstrait, c'est à dire qu'il correspond à un comportement partiel complété dans les sous-cas d'utilisation.

Un sous-cas d'utilisation a le même niveau que son sur-cas. Si le sur-cas est un cas avec objectif d'acteur primaire, il en va de même pour le sous-cas. Si le sur-cas est un cas de sous-fonction, le sous-cas est lui aussi une sous-fonction.

Dans le diagramme des cas d'utilisation, la relation de spécialisation est représentée par une flèche pleine de spécialisation identique à celle qui relie les sous-classes aux surclasses. Le nom d'un cas d'utilisation abstrait est écrit en italique (ou accompagné du stéréotype <<abstract>>).