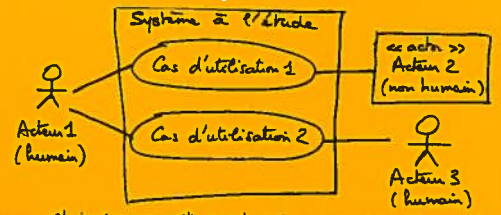
Diagramme de cas d'utilisation :

Montre les interactions fonctionnelles entre les acteurs et le système à l'étude



- Acteur : rôle joué par un utilisateur humain ou un autre système qui interagit directement avec le système étudié. Un acteur participe à au moins un cas d'utilisation.
- Cas d'utilisation (Use case) : ensemble de séquences d'actions réalisées par le système produisant un résultat observable intéressant pour un acteur particulier. Collection de scénarios reliés par un objectif utilisateur commun.
- Association : utilisée dans ce type de diagramme pour relier les acteurs et les cas d'utilisation par une relation qui signifie simplement : « participe à ».

Trucs et astuces

- Un cas d'utilisation peut faire participer plusieurs acteurs ; un acteur peut participer à plusieurs cas d'utilisation.
- Appliquez la recommandation suivante : un seul acteur principal par cas d'utilisation. Nous appelons acteur *principal* celui pour qui le cas d'utilisation produit un résultat observable. Par opposition, nous qualifions d'acteurs *secondaires* les autres participants du cas d'utilisation.
- Dans la mesure du possible, disposez les acteurs principaux à gauche des cas d'utilisation, et les acteurs secondaires à droite.
- Utilisez la forme graphique du stick man pour les acteurs humains, et la représentation rectangulaire avec le mot-clé <<actor>> ou un pictogramme pour les systèmes connectés.
- Éliminez les acteurs « physiques » au profit des acteurs « logiques » : l'acteur principal est celui qui bénéficie de l'utilisation du système.
- Répertoriez en tant qu'acteurs uniquement les entités externes *et non pas des composants internes* qui interagissent directement avec le système. Un moyen technique n'est pas un acteur pertinent.
- Ne confondez pas rôle et entité concrète. Une même entité externe concrète peut jouer successivement différents rôles par rapport au système étudié, et être modélisée par plusieurs acteurs. Réciproquement, le même rôle peut être tenu par plusieurs entités externes concrètes, qui sont alors modélisées par le même acteur.

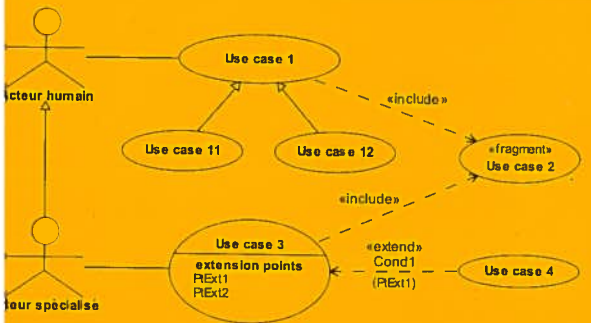


Diagramme de cas d'utilisation

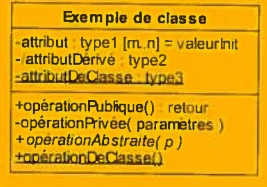
- Inclusion : le cas d'utilisation de base en incorpore explicitement un autre, de façon obligatoire, à un endroit spécifié dans son comportement.
- Extension : le cas d'utilisation de base en incorpore implicitement un autre, de façon optionnelle, à un endroit spécifié indirectement dans celui qui procède à l'extension.
- Généralisation : les cas d'utilisation descendants héritent de la description de leur parent commun. Chacun d'entre eux peut néanmoins comprendre des relations supplémentaires avec d'autres acteurs ou cas d'utilisation.

Trucs et astuces

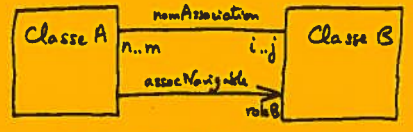
- Utilisez la relation d'inclusion entre cas d'utilisation pour éviter de devoir décrire plusieurs fois un comportement commun, en le factorisant dans un cas d'utilisation supplémentaire inclus de stéréotype « fragment ».
- N'abusez pas des relations entre cas d'utilisation *surtout extension et généralisation* : elles peuvent rendre les diagrammes de cas d'utilisation difficiles à décrypter pour les experts métier qui sont censés les valider.
- Limitez à 20 le nombre de vos cas d'utilisation de base *en dehors des cas inclus, spécialisés, ou des extensions*. Avec cette limite arbitraire, on reste synthétique et on ne tombe pas dans le piège de la granularité trop fine des cas d'utilisation. Un cas d'utilisation ne doit donc pas se réduire à une seule séquence, et encore moins à une simple action.
- Le diagramme de cas d'utilisation n'est qu'une sorte de table des matières graphique des exigences fonctionnelles. N'y passez pas trop de temps et d'énergie, mais concentrez-vous plutôt sur la description détaillée des cas d'utilisation (texte, diagrammes de séquence, etc.) !

Diagramme de classes

Montre les briques de base statiques : classes, associations, interfaces, attributs, opérations, généralisations, etc.



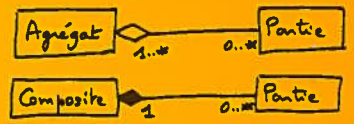
- Classe : description abstraite d'un ensemble d'objets qui partagent les mêmes propriétés attributs et associations et comportements opérations et états.
- Attribut : donnée déclarée au niveau d'une classe, éventuellement typée, et valorisée par chacun des objets de cette classe. Un attribut peut posséder une multiplicité et une valeur initiale.
- Opération : élément de comportement des objets, défini de manière globale dans leur classe. Une opération peut déclarer des paramètres eux-mêmes typés ainsi qu'un type de retour. Une méthode est une implémentation d'une opération.



- Association : relation sémantique durable entre deux classes qui décrit un ensemble de liens entre objets. Une association est décrite par un nom et des rôles, également optionnels. On spécifie sa navigabilité en ajoutant une flèche.
- Rôle : nom donné à une extrémité d'une association ; par extension, manière dont les instances d'une classe voient les instances d'une autre classe au travers d'une association.

Diagramme de classes

- Multiplicité : le nombre d'objets min..max de la classe cible qui peuvent participer à une relation avec un objet de la classe source dans le cadre d'une association. Multiplicités fréquentes :
  - 0..1 = optionnel mais pas multiple
  - 1 = exactement 1
  - 0..* = * = quelconque
  - 1..* = au moins 1



- Agrégation : cas particulier d'association non symétrique exprimant une relation de contenance, mais sans contrainte supplémentaire.
- Composition : forme forte d'agrégation, dans laquelle les parties ne peuvent appartenir à plusieurs agrégats à la fois et où le cycle de vie des parties est subordonné à celui de l'agrégat.

- Super-classe : classe plus générale reliée à une ou plusieurs autres classes plus spécialisées sous-classes par une relation de généralisation.
- Généralisation : relation entre classifieurs où les descendants héritent des propriétés de leur parent commun. Ils peuvent néanmoins comprendre chacun des propriétés spécifiques supplémentaires, mais aussi modifier les comportements hérités.

- Note : commentaire visible dans le diagramme. Peut être attaché à un élément du modèle ou plusieurs par un trait pointillé.

Trucs et astuces

- En analyse, évitez de confondre attribut et classe : si l'on ne peut demander à un élément que sa valeur, il s'agit d'un simple attribut; si l'on peut lui poser plusieurs questions, il s'agit plutôt d'une classe *qui possédera à son tour des attributs, ainsi que des associations avec d'autres classes.*
- On doit faire figurer une indication de multiplicité à chaque extrémité navigable d'une association.
- Un attribut optionnel peut être représenté en ajoutant une multiplicité [0..1] derrière son type.
- Un attribut dérivé est un attribut dont la valeur peut être déduite d'autres informations disponibles dans le modèle. Conservez cet attribut, qui pourrait être considéré comme redondant, uniquement s'il correspond à un concept important aux yeux de l'expert métier.
- Une association qui « reboucle » sur une seule classe permet de relier des objets de même type, en distinguant leurs rôles respectifs.
- Les agrégations et les compositions n'ont pas besoin d'être nommées : elles signifient implicitement « contient », « est composé de ».
- Utilisez la composition uniquement si la multiplicité est inférieure ou égale à 1 du côté du composite et si la destruction du composite entraîne celle de ses parties.
- La relation de généralisation doit pouvoir se lire ainsi : est-une-sorte-de. Il est également fortement recommandé que toutes les propriétés de la super-classe soient valables pour toutes les sous-classes *règle des 100 %*. Les sous-classes peuvent uniquement ajouter des propriétés et redéfinir des opérations.

- Interface : Ensemble nommé d'opérations qui caractérise le comportement d'un élément. Une classe peut réaliser plusieurs interfaces fournies et dépendre de plusieurs autres requises.