Glossaire

UML

acteur
: Un acteur (*actor*) représente un rôle joué par une entité externe qui interagit avec le système. Un acteur est identifé par un nom et peut être : un humain, un dispositif matériel ou un autre système.

action
: Une action est le plus petit traitement qui puisse être exprimé en *Unified Modeling Language* (UML) : c'est une opération atomique ininterruptible. Une action a une incidence sur l'état du système ou en extrait une information.

activité
: Une activité définit un comportement décrit par un séquencement organisé d'actions. Le flot d'exécution est modélisé par des noeuds reliés par des transitions. Le flot de contrôle reste dans l'activité jusqu'à ce que les traitements soient terminés.

agrégation
: Une agrégation est un cas particulier d'association non symétrique exprimant une relation de contenance. Les agrégations n'ont pas besoin d'être nommées : implicitement elles signifient « contient » ou « est composé de ». Elle est représentée par trait plein avec ou sans flèche et un losange vide côté composite. Dans une agrégation, le composant peut être partagé entre plusieurs composites ce qui entraîne que, lorsque le composite sera détruit, le composant ne le sera pas forcément. La relation d'agrégation est une relation sémantique de type « avoir ».

artefact
: Un artefact est une manière de définir un élément concret (fichier, un programme, une bibliothèque ou une base de données) construit ou modifié dans un projet.

association
: Une association représente une relation sémantique durable entre deux classes. Les associations peuvent donc être nommées pour donner un sens précis à la relation. Elle est représentée par trait plein avec ou sans flèche. La relation d'association est une relation sémantique de type « avoir ».

cas d'utilisation
: Un cas d'utilisation (*use case*) représente une fonction offerte par le système et qui produit un résultat observable intéressant pour un acteur. Chaque cas d'utilisation spécifie un comportement attendu du système. Il permet de décrire ce que le système devra faire, sans spécifier comment il le fera. On nommera les cas d'utilisation par un verbe à l'infinitif suivi d'un complément, du point de vue de l'acteur (et non du système).

classe abstraite
: Une classe est dite abstraite lorsqu'on ne peut instancier d'objets à partir de cette classe. Le nom des classes abstraites est écrit en *italique*. On peut ajouter le stéréotype `«abstract»`.

classe active
: Une classe active est une classe qui possède une méthode qui s'exécute dans un flot de contrôle distinct. Ce flot est généralement le fil d'exécution d'un *thread*. Une instance d'une classe active sera nommée object actif. UML fournit un repère visuel (bord en trait épais ou double trait) qui permet de distinguer les éléments actifs des éléments passifs. Il est tout de même conseillé d'ajouter le stéréotype `«thread»`.

composition
: Une composition est une agrégation plus forte signifiant « est composée d'un » et impliquant : qu'une partie ne peut appartenir qu'à un seul composite (agrégation non partagée) et que la destruction du composite entraîne la destruction de toutes ses parties (il est responsable du cycle

de vie de ses parties). Elle est représentée par trait plein avec ou sans flèche et un losange plein côté composite. La relation de composition est une relation sémantique de type « avoir ».

dépendance
Une dépendance s'illustre par une flèche en pointillée en UML. Elle indique qu'un élément « a besoin » des services d'un autre élément. Lorsqu'un objet « utilise » temporairement les services d'un autre objet, cette relation d'utilisation est une dépendance entre classes. La plupart du temps, les dépendances servent à montrer qu'une classe utilise une autre comme argument dans la signature d'une méthode. On parle aussi de lien temporaire car il ne dure que le temps de l'exécution de la méthode. Cela peut être aussi le cas d'un objet local à une méthode. Généralement, les dépendances ne sont pas montrées dans un diagramme de classes car elles ne sont qu'une utilisation temporaire donc un détail de l'implémentation que l'on ne considère pas judicieux de mettre en avant.

diagramme
Les diagrammes sont des ensembles d'éléments graphiques (pictogrammes). Ils décrivent le contenu des vues et peuvent faire partie de plusieurs vues. Il en existe quatorze depuis UML 2.3. UML n'étant pas une méthode de développement, leur utilisation est laissée à l'appréciation de chacun. Des méthodologies, telles que l'*Unified Process* (UP), axent l'analyse en tout premier lieu sur les diagrammes de cas d'utilisation (*use case*).

diagramme d'activité
Un diagramme d'activité (*activity diagram*) est une représentation du comportement du système ou de ses composants sous forme de flux ou d'enchaînement d'activités. Il décrit les activités séquentielles et parallèles d'un système. Ils permettent ainsi de représenter graphiquement le comportement d'une méthode ou le déroulement d'un cas d'utilisation.

diagramme de classes
Le diagramme de classes (*class diagram*) représente les classes intervenant dans le système et leurs relations.

diagramme de composants
Le diagramme de composants (*component diagram*) décrit l'organisation du système du point de vue des éléments logiciels comme les modules (paquets, fichiers sources, bibliothèques, exécutables), des données (fichiers, bases de données) ou encore d'éléments de configuration (paramètres, scripts, fichiers de commandes). Ce diagramme permet notamment de mettre en évidence les dépendances entre les composants (qui utilise quoi).

diagramme de déploiement
Le diagramme de déploiement (*deployment diagram*) est une représentation des éléments matériels (ordinateurs, périphériques, réseaux, systèmes de stockage, ...) et la manière dont les composants du système sont répartis sur ces éléments matériels et interagissent entre eux. Il présente le déploiement des éléments logiciels sur l'architecture physique. Les caractéristiques des ressources matérielles physiques et des supports de communication peuvent être précisées par stéréotype (`«USB»`, `«RS485»`, ...).

diagramme de séquence
Un diagramme de séquence (*sequence diagram*) est un diagramme d'interaction dont le but est de décrire comment les objets collaborent au cours du temps et quelles responsabilités ils assument. Il décrit un scénario d'un cas d'utilisation. Un diagramme de séquence représente donc les interactions entre les éléments du système et/ou de ses acteurs, en insistant sur la chronologie des envois de message. C'est un diagramme qui représente la structure dynamique d'un système car il utilise une représentation temporelle. Les objets, intervenant dans l'interaction, sont matérialisés par une « ligne de vie », et les messages échangés au cours du temps sont mentionnés sous une forme textuelle.

diagramme de séquence système
Un Diagramme de Séquence Système (DSS) permet de décrire le comportement du système vu de l'extérieur (par les acteurs) sans préjuger de comment il sera réalisé. Le système est vu comme une « boîte noire » qui sera ouverte (décrite) seulement plus tard en conception..

diagramme des cas d'utilisation
Le diagramme des cas d'utilisation (*use-case diagram*) permet une représentation des possibilités d'interaction entre le système et les acteurs (intervenants extérieurs au système), c'est-à-dire de toutes les fonctionnalités que doit fournir le système.

diagramme états-transitions
Le diagramme d'états ou états-transitions (*state machine diagram*) représente le comportement du système ou de ses composants sous forme de machine à états finis. Il est souvent utilisé pour décrire le comportement interne d'un objet en représentant les séquences possibles d'états et d'actions. Il peut être aussi utiliser pour spécifier le comportement interne d'autres éléments tels que les cas d'utilisation, les sous-systèmes, les méthodes.

élément
Les modèles d'élément sont les éléments graphiques (pictogrammes) des diagrammes.

extension (use case)
Dans une relation d'extension (`«extend»`), le cas d'utilisation de base incorpore implicitement un autre cas de façon optionnelle.

fragment
Un fragment (ou cadre) permet d'identifier une sous-partie d'une interaction afin que celle-ci soit référencées par d'autres interactions ou de lui spécifier des conditions particulières d'exécution (boucle, optionnel, ...). Il existe par exemple les fragments suivants : `sd` (fragment du diagramme de séquence en entier), `alt` (fragment alternatif Si ... Alors ... Sinon ...), `opt` (fragment optionnel), `par` (fragment parallèle pour les traitements concurrents), `loop` (le fragment s'exécute plusieurs fois dans une boucle), `region` (région critique où un seul thread peut s'exécuter à la fois), `ref` (référence à une interaction dans un autre diagramme) ....

généralisation (use case)
Dans une relation de généralisation/spécialisation (héritage), les cas d'utilisation descendants héritent de leur parent commun. Il est également possible d'appliquer à un acteur la relation de généralisation. Cela se fait notamment lorsqu'un acteur est un sous-type d'une autre catégorie d'acteurs. Un acteur lié à un autre par un ce type de relation peut interagir avec le système de plus de manières que son parent.

héritage
L'héritage est un concept fondamental de la programmation orientée objet. Il se nomme ainsi car le principe est en quelque sorte le même que celui d'un arbre généalogique. Ce principe est fondé sur des classes « filles » qui héritent des caractéristiques des classses « mères ». L'héritage permet d'ajouter des propriétés à une classe existante pour en obtenir une nouvelle plus précise. Il permet donc la spécialisation ou la dérivation de types. L'héritage est représenté par un trait reliant les deux classes et dont l'origine (classe mère) se distingue de l'autre extrémité (classe fille) par un triangle vide. La relation d'héritage est une relation sémantique de type « être ».

inclusion (use case)
Dans une relation d'inclusion (`«include»`), le cas d'utilisation de base incorpore explicitement un autre cas de façon obligatoire.

message
Les objets interagissent entre eux en s'échangeant des messages. La réponse à la réception d'un message par un objet est appelée une méthode. Une méthode est donc la mise en oeuvre du message : elle décrit la réponse qui doit être donnée au message. Dans un diagramme de séquence, une activité représente l'exécution d'une méthode. On distingue deux types de message : synchrone où l'objet émetteur se bloque en attendant la réponse de l'objet récepteur du message et asynchrone où l'objet émetteur n'attend pas la réponse de l'objet récepteur du message et continue son activité.

modélisation
La modélisation consiste à créer une représentation simplifiée d'un problème : le modèle. Le modèle constitue ainsi une représentation possible du système pour un point de vue donné. La modélisation comporte deux composantes : l'analyse, c'est-à-dire l'étude du problème et des besoins (quoi faire ?) et la conception, qui l'élaboration d'une solution au problème et aux besoins (comment faire ?).

multiplicité
Aux extrémités d'une association, agrégation ou composition, il est possible d'y indiquer une multiplicité (ou cardinalité) : c'est pour préciser le nombre d'instances (objets) qui participent à la relation. Une multiplicité peut s'écrire : n (exactement n, un entier positif), n..m (n à m), n..* (n ou plus) ou * (plusieurs).

navigabilité
Aux extrémités d'une association, agrégation ou composition, il est possible d'ajouter une flèche sur la relation ce qui précise la navigabilité. La navigabilité permet de préciser « qui connaît qui » dans la relation. Les relations peuvent être bidirectionnelles (pas de flèche) ou unidirectionnelle (avec une flèche qui précise le sens).

paquet
Un paquet (*package*) est un mécanisme général de regroupement d'éléments (diagrammes, cas d'utilisation, acteurs, classes, ...).

relation (classes)
Étant donné que les objets logiciels interagissent entre eux, il existe donc des relations entre les classes. On distingue cinq différents types de relations de base entre les classes : l'association (trait plein avec ou sans flèche), la composition (trait plein avec ou sans flèche et un losange plein), l'agrégation (trait plein avec ou sans flèche et un losange vide), la relation de généralisation ou d'héritage (flèche fermée vide) et la dépendance (flèche pointillée). Il existe des relations durables (association, composition, agrégation, héritage) et des relations temporaires (dépendance).

relation (use case)
Pour affiner les diagrammes, il est possible d'ajouter des relations entre cas d'utilisation. UML définit trois types de relations standardisées : inclusion, extension et généralisation.

rôle
À l'extrémité d'une association, agrégation ou composition, on donne un nom : c'est le rôle de la relation. Par extension, c'est la manière dont les instances d'une classe voient les instances d'une autre classe au travers de la relation.

stéréotype
Un stéréotype est une marque de généralisation notée par des guillemets (« »), cela montre que l'objet est une variété d'un modèle.

transition
Une transition est le passage d'un état dans un autre si un événement déclencheur se produit (et que la condition de garde est vérifiée). Cela provoque l'exécution de certaines activités.

uml
UML est un langage de modélisation graphique à base de pictogrammes conçu pour fournir une méthode normalisée pour visualiser la conception d'un système. Il est couramment utilisé en développement logiciel et en conception orientée objet. UML est utilisé pour spécifier, visualiser, modifier et construire les documents nécessaires au bon développement d'un logiciel orienté objet. UML permet d'obtenir une modélisation indépendante des langages et des environnements.

visibilité
La notion de visibilité indique qui peut avoir accès à l'élément. La visibilité est précisée au sein des

classes (et non des objets) et elle peut être : `public` (`+`) quand toutes les autres classes ont accès, `protected` (`#`) quand seules la classe elle-même et les classes filles (héritage) ont accès, `private` (`-`) quand seule la classe elle-même a accès et `package` (`~`) quand la classe est visible uniquement dans le paquet.

vue

Les vues décrivent le système d'un point de vue donné, qui peut être organisationnel, dynamique, temporel, architectural, géographique, logique, etc. En combinant toutes ces vues, il est possible de définir (ou retrouver) le système complet. Les vues sont décrites par des diagrammes.

vue des cas d'utilisation

La vue des cas d'utilisation (*use-case view*) est la description du modèle vu par les acteurs du système. Elle correspond aux besoins attendus par chaque acteur.