UML (Unified Modeling Language)

UML (Unified Modeling Language) est un langage de modélisation visuel qui permet de représenter graphiquement la structure et le comportement des systèmes logiciels. Il est utilisé pour le développement de solutions logicielles orientées objet. UML permet de modéliser un projet de façon standard. Ce langage est devenu une référence en termes de modélisation objet. L'UML est utilisé pour spécifier, visualiser, modifier et construire les documents nécessaires au développement d'un logiciel orienté objet. UML permet d'obtenir une modélisation indépendante des langages de programmation.

1. Notion d'objet, de classe et d'association

Le but de la modélisation UML est de décrire les objets. L'objet permet la décomposition d'un problème. Il est donc très adapté à la réalisation d'un projet en le décomposant en plusieurs objets qui seront attribués à différents membres de l'équipe de développement.

L'objet est formé d'un état et d'un comportement :

L'état d'un objet est constitué par ses attributs. Les attributs sont des valeurs qui sont associées aux objets (ex : couleur est un attribut d'un objet Voiture).

Le comportement est décrit par ce que l'on appelle des méthodes. Les interactions entre objets se représentent au moyen de différents diagrammes.

Une classe est un élément de base de la modélisation orientée objet qui permet de représenter un ensemble d'objets partageant des caractéristiques communes. L'instanciation d'une classe permet de créer un objet.

Une association permet de modéliser la relation entre deux classes. Elle est représentée par une ligne reliant les deux classes et peut être accompagnée d'une multiplicité pour indiquer combien d'objets d'une classe peuvent être associés à un certain nombre d'objets de l'autre classe.

L'UML peut être utilisé pour modéliser et analyser les besoins d'un système. Les diagrammes de cas d'utilisation, par exemple, aident à comprendre comment les utilisateurs interagissent avec le système.

Les diagrammes d'états et de séquence UML sont utilisés pour planifier et concevoir des cas de test, ce qui facilite la validation et la vérification du logiciel.

En fournissant une vue claire de la structure du logiciel, l'UML aide les équipes de maintenance à comprendre rapidement le code existant et à prendre des décisions éclairées pour les mises à jour et les évolutions.

LE DIAGRAMME DE DÉPLOIEMENT

Le diagramme de déploiement est un diagramme structurel, c'est-à-dire qu'il décrit la structure du système.

Il décrit le déploiement physique des informations générées par le logiciel sur des composants matériels. Les diagrammes de déploiement sont constitués de plusieurs formes UML (Unified Modeling Language). Les boîtes en trois dimensions, appelées nœuds, représentent les composants du système, qu'ils soient logiciels ou matériels.

Les lignes entre les nœuds indiquent les relations ; les petites formes à l'intérieur des boîtes représentent les artefacts logiciels qui sont déployés.

Les diagrammes de déploiement sont notamment utilisés pour visualiser la topologie du système matériel.

1. Les éléments principaux d'un diagramme de déploiement

Les diagrammes de cas d'utilisation sont composés de nombreuses formes :

nœud : un nœud représente un élément matériel ou logiciel du système. Les nœuds sont utilisés pour montrer comment les différents éléments sont répartis sur le matériel physique ou les serveurs. Un nœud peut être un ordinateur, un serveur, un dispositif matériel par exemple. Il sert à indiquer où un composant logiciel particulier sera exécuté dans l'infrastructure physique. Les nœuds sont connectés par des liens pour montrer comment les composants logiciels ou matériels communiquent les uns avec les autres au sein du système ;

artefact : un artefact est un élément logiciel qui est déployé sur un nœud spécifique dans une architecture système. Il est symbolisé par un rectangle avec le nom et le mot « artefact » entourés de flèches doubles.

Les artefacts sont utilisés pour représenter des fichiers, des bibliothèques, des exécutables, etc.

Un artefact est associé à un nœud de déploiement particulier, ce qui signifie qu'il est installé et exécuté sur ce nœud ;

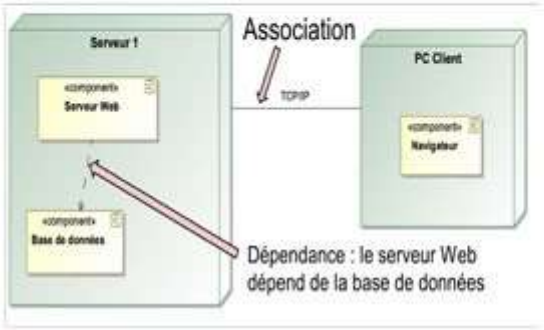
association : une association montre comment des nœuds différents interagissent les uns avec les autres pour réaliser une fonctionnalité ou un service dans un système. Elles indiquent comment les composants matériels (nœuds) sont connectés les uns aux autres. Elles peuvent représenter des connexions physiques, des liaisons réseau ou tout autre chemin de communication.

Par exemple, si on a un serveur web déployé sur un nœud de serveur et une base de données déployée sur un autre nœud, l'association entre ces deux nœuds montre par quelle connexion le serveur web communique avec la base de données ;

dépendance : une dépendance est une ligne en pointillés terminée par une flèche. Elle indique qu'un élément dépend d'un autre pour fonctionner correctement. Ces dépendances peuvent prendre différentes formes :– dépendance matérielle : un composant logiciel a besoin d'un matériel spécifique pour fonctionner. Par exemple, un serveur web peut dépendre d'un serveur de base de données pour stocker et récupérer des données ;– dépendance logicielle : un composant logiciel dépend d'un autre pour accéder à des fonctionnalités.

Capture d'écran du logiciel Magic Draw

Capture d'écran du logiciel Magic Draw



composant : un composant est une entité logicielle ou matérielle qui interagit avec le système. Les composants sont utilisés pour représenter les différentes parties d'une application ou d'un système informatique, ainsi que les ressources matérielles telles que des serveurs, des ordinateurs ou d'autres dispositifs. Les composants sont représentés sous forme de boîtes dans le diagramme de déploiement, et chaque boîte peut être associée à des artefacts, des fichiers ou des données. Les flèches ou les lignes reliant les composants indiquent les relations et les dépendances entre eux, montrant comment ils communiquent ou coopèrent au sein du système.

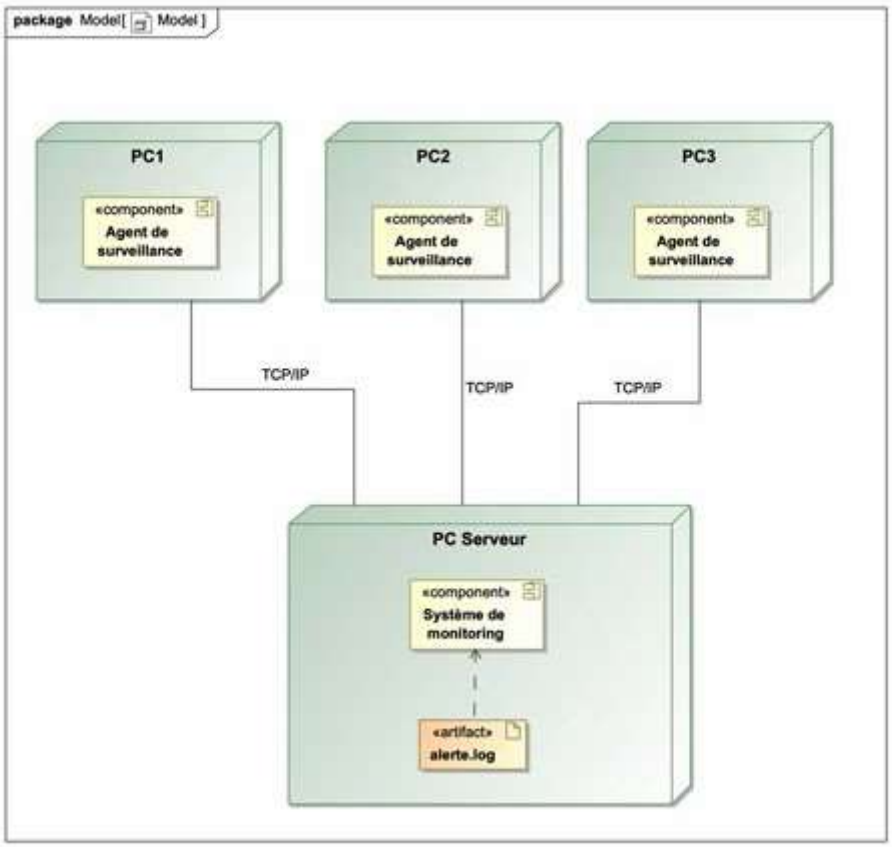
Le composant est le niveau inférieur du nœud ;

package (paquetage) : un package sert à grouper des éléments en un ensemble. Il est utilisé pour organiser et regrouper d'autres éléments de modélisation, tels que les nœuds, les composants, etc.

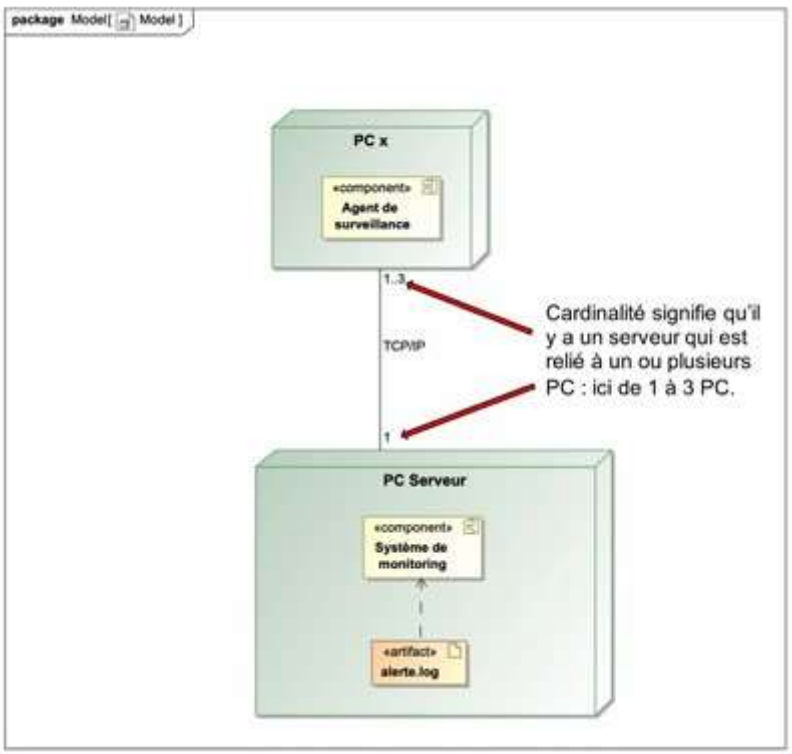
Cela permet de rendre la représentation visuelle plus claire pour les systèmes complexes ;

note : une note est un élément utilisé pour fournir des informations supplémentaires dans un diagramme. Les notes sont utilisées pour commenter des éléments du diagramme.

EXEMPLE :



Remarque On peut tout à fait mettre des cardinalités (=multiplicité) sur les associations. Pour l'exemple ci-joint, cela donnerait :



LE DIAGRAMME DE CAS D'UTILISATION

Le diagramme de cas d'utilisation est un diagramme comportemental.

Il constitue un moyen de décrire les besoins des acteurs (utilisateurs, entités extérieures…) du système. Il permet de décrire les fonctionnalités d'un système en se concentrant sur les actions que les acteurs effectuent en collaboration avec le système.

Un cas d'utilisation permet de décrire l'interaction entre les acteurs et le système. La description de l'interaction est réalisée suivant le point de vue de l'utilisateur.

Dans ce diagramme, on retrouve essentiellement des acteurs, des cas d'utilisation et l'interaction entre ceux-ci.

Les diagrammes de cas d'utilisation sont composés de :

Acteur : Un acteur représente une entité externe au système à modéliser. Ils interagissent avec le système avec des cas d'utilisation (use cases) pour accomplir certaines tâches. Les acteurs peuvent être des utilisateurs, d'autres systèmes, des composants matériels…

Les acteurs sont représentés par des icônes (forme humaine) à l'extérieur du diagramme de cas d'utilisation.

Il existe 2 types d'acteurs :

Des acteurs déclencheurs : ceux qui vont réaliser le cas d'utilisation, le système est conçu pour satisfaire les utilisateurs directs et non les autres.

Des acteurs secondaires : ceux qui ne font que recevoir des informations à l'issue de la réalisation du cas d'utilisation (exemple : client ou autre système informatique).

Lorsque l'on travaille sur les acteurs, on ne doit pas raisonner en termes d'entité physique (Monsieur X, système Z…) mais en termes de rôle que l'entité physique joue (administrateur, magasinier, serveur de fichiers…).

Cas d'utilisation (Use-case) : Un cas d'utilisation décrit ce que le système fait en réponse à une action de l'acteur. Il inclut un scénario nominal (principal) mais il peut aussi inclure des scénarios alternatifs qui sont les variantes du scénario nominal et enfin, les scénarios d'exception qui décrivent les cas d'erreurs.

Interaction : les interactions dans un diagramme de cas d'utilisation représentent les relations et les échanges d'informations entre les acteurs et les cas d'utilisation.

Il existe plusieurs types de relations dans un diagramme de cas d'utilisation :

– Les associations entre un acteur et un cas d'utilisation. Elles indiquent comment un acteur interagit avec un cas d'utilisation. Les associations sont représentées par des lignes reliant un acteur à un cas d'utilisation.

– Les extends (étendre) qui étendent la relation entre deux cas d'utilisation. Elles indiquent qu'un cas d'utilisation optionnel, appelé le cas d'utilisation étendu, peut être exécuté en plus. Les extends sont représentées par une flèche pointillée et le mot <<extend>> écrit.

– Les includes sont utilisés pour montrer comment un cas d'utilisation peut inclure le comportement

d'un autre cas d'utilisation sans pour autant dépendre directement de lui. Lorsqu'on utilise une relation "include" cela signifie que le cas d'utilisation inclus est obligatoire pour le cas d'utilisation qui l'inclut. Si le cas d'utilisation principal est déclenché, le cas d'utilisation inclus sera également exécuté. Les includes sont représentés par une flèche pointillée et le mot <<include>> écrit.

Les scénarios sont des descriptions textuelles des différents cas d'utilisation. Cela permet de mieux détailler et comprendre chaque cas d'utilisation.

Voici quelques scénarios typiques pour un diagramme de cas d'utilisation :

Scénario nominal ou principal (cas d'utilisation) : C'est le scénario principal qui décrit les étapes essentielles et les interactions entre les acteurs et le système. Il couvre le déroulement normal de l'utilisation du système.

Scénario d'extension : Les scénarios d'extension décrivent les fonctionnalités supplémentaires activées par un acteur pendant le scénario principal. Par exemple, un scénario d'extension peut décrire comment un administrateur peut modifier l'unité de température pendant l'affichage de celle-ci.

Scénario alternatif : les scénarios alternatifs décrivent une séquence d'étapes qui peut se produire à la place de certaines étapes du scénario principal du cas d'utilisation. Il est utilisé pour décrire des chemins différents ou des variations possibles du comportement principal du cas d'utilisation.

Scénario d'exception : Les scénarios d'exception décrivent ce qui se passe en cas d'erreurs.

Pour chaque scénario du cas d'utilisation, il faut :

Le nom du cas d'utilisation.

La description.

Les acteurs.

Le contexte.

Le scénario principal : étapes essentielles.

Le scénario alternatif ou d'extension : gestion des erreurs, activation d'options…

LE DIAGRAMME DE SEQUENCE

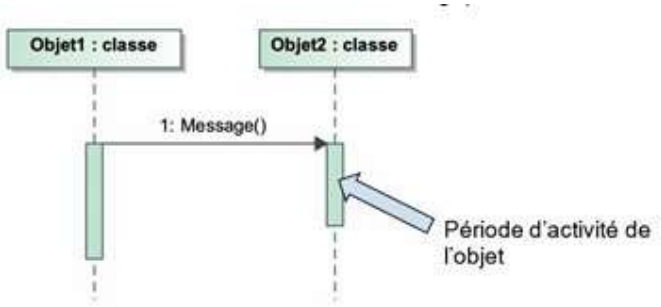
Le diagramme de séquence permet de représenter des échanges entre les différents objets et acteurs du système en fonction du temps. Il fait partie des diagrammes de comportement. Il décrit comment les éléments du système interagissent entre eux et avec les acteurs : les objets au cœur d'un système interagissent en s'échangent des messages ; les acteurs interagissent avec le système au moyen d'IHM (Interfaces Homme-Machine).

Les diagrammes de séquence sont composés de :

- ligne de vie : elle est utilisée pour montrer comment les objets (ou acteurs) interagissent au fil du temps. Une ligne de vie est représentée par une ligne pointillée verticale, et les messages échangés entre les objets sont affichés horizontalement sur cette ligne. Elle permet de suivre le déroulement

des scénarios d'exécution et de modéliser le comportement des objets au sein d'un système logiciel. A chaque objet est associée une ligne de vie ;

- messages : les principales informations contenues dans un diagramme de séquence sont les messages échangés entre les lignes de vie. Ils sont représentés par des flèches. Les messages sont numérotés et ordonnancés de manière verticale (chronologique).



À noter que sur la ligne de vie, on fait apparaître la période d'activité qui correspond au temps pendant lequel un objet fait une action.

Un message définit une communication particulière entre des lignes de vie (objets ou acteurs).

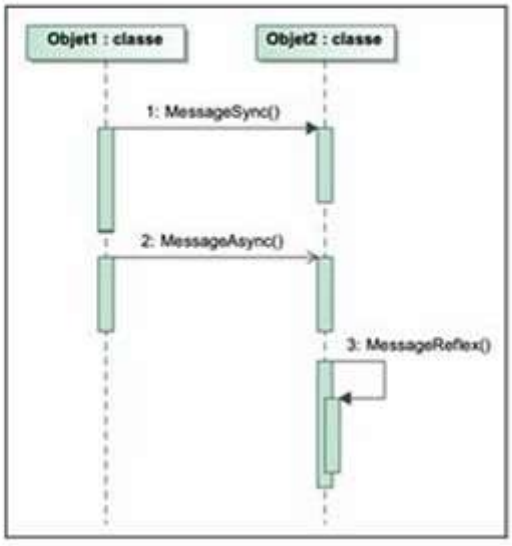
Plusieurs types de messages existent :

Le message synchrone : le message synchrone bloque l'expéditeur jusqu'à ce que le récepteur ait terminé son traitement (représenté par une flèche pleine) ;

Le message synchrone ne renvoie généralement pas de réponse.

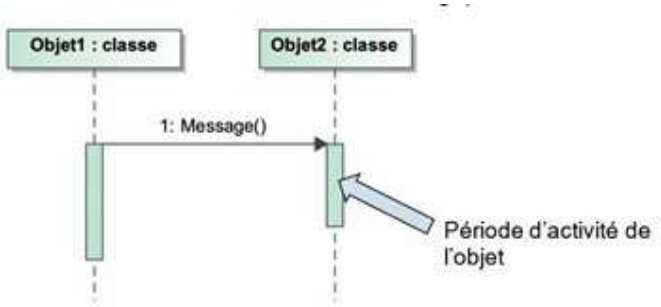
Le message asynchrone : le message asynchrone n'interrompt pas l'expéditeur qui peut émettre un autre message sans attendre la réponse du récepteur (représenté par une flèche creuse).

Le message réflexif : le message est envoyé d'un objet vers lui-même.



Le récepteur d'un message synchrone rend la main à l'émetteur du message en lui envoyant un message de retour. Les messages de retour sont optionnels : la fin de la période d'activité marque également la fin de l'exécution d'une méthode. Les messages de retour (spécification d'exécution dans le cas d'un appel de méthode) sont représentés en pointillés.

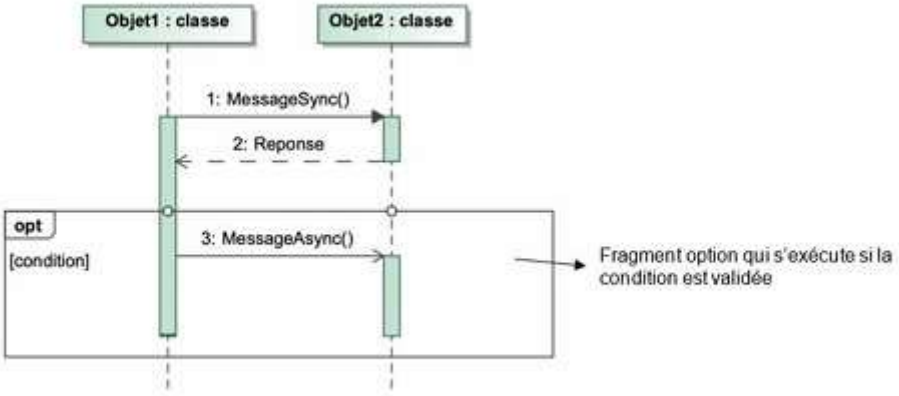
La réception des messages provoque une période d'activité (rectangle vertical sur la ligne de vie) marquant le traitement du message Des repères temporels avec des contraintes peuvent être placés le long de la ligne de vie.



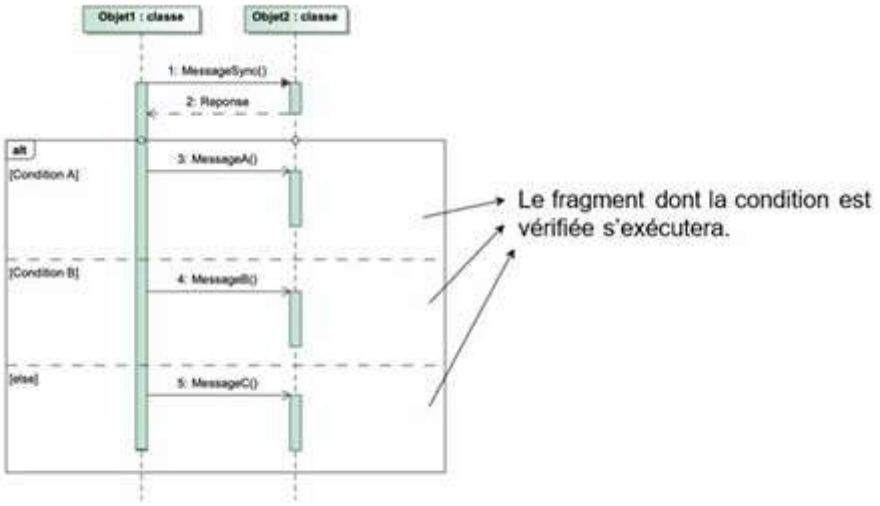
À noter que sur la ligne de vie, on fait apparaître la période d'activité qui correspond au temps pendant lequel un objet fait une action.

Les fragments d'interactions combinés : un fragment d'interactions est une partie du diagramme de séquence (délimitée par un rectangle) associée à une étiquette. L'étiquette contient la condition qui permet de décrire des modalités d'exécution des messages à l'intérieur du cadre.

– Fragment d'interaction avec opérateur opt : l'opérateur option contient une séquence qui peut ou non se produire. Vous pouvez spécifier la condition [ ] sous laquelle elle se produit.



– Fragment d'interaction avec opérateur alt : l'opérateur alternatives (alt) contient une liste des fragments dans lesquels se trouvent d'autres séquences de messages. Une seule séquence peut se produire à la fois. C'est l'équivalent d'une exécution à choix multiples. Seul le sous-fragment dont la condition est vraie est exécuté. La condition else est exécutée que si aucune autre condition n'est valide.



– Fragment d'interaction avec opérateur loop : l'opérateur de boucle (loop) est répété un certain nombre de fois. Dans la protection, on indique la condition sous laquelle il doit être répété : loop [min, max, condition] : chaque paramètre (min, max et condition) est optionnel.