Livrable n°1

Programmation Système

Groupe 02

DEGHFEL Walid Mebarek

VALOIS Aymerick

DREMAUX Valentin

Sommaire

Introduction

Notre équipe vient d'intégrer l'éditeur de logiciels ProSoft. Sous la responsabilité du DSI, nous avons la responsabilité de gérer le projet "EasySave" qui consiste à développer un logiciel de sauvegarde. Comme tout logiciel de la suite ProSoft, le logiciel s'intégrera à la politique tarifaire. Lors de ce projet, notre équipe devra assurer le développement, la gestion des versions majeures et mineures, mais aussi les documentations. Pour garantir une reprise de notre travail par d'autres équipes, la direction nous impose de travailler dans le respect des contraintes. Nous devons donc conduire ce projet de manière à réduire les coûts de développement des futures versions et surtout d'être capable de réagir rapidement à la remontée éventuelle d'un dysfonctionnement.

Environnement de travail

IDE : Visual Studio

L'IDE utilisé est Visual Studio 2019. C'est un ensemble complet d'outils de développement permettant de générer des applications web ASP.NET, des services web XML, des applications bureautiques et des applications mobiles. Visual Basic, Visual C++, Visual C# utilisent tous le même environnement de développement intégré (IDE), qui leur permet de partager des outils et facilite la création de solutions faisant appel à plusieurs langages. Par ailleurs, ces langages permettent de mieux tirer parti des fonctionnalités du Framework .NET, qui fournit un accès à des technologies clés simplifiant le développement d'applications web ASP et de services web XML grâce à Visual Web Developer.

AGL : Visual Paradigm

Pour la modélisation de l'application, nous avons choisi Visual Paradigm. Un logiciel de création de diagrammes dans le cadre d'une programmation. Tout en un, il possède plusieurs options permettant une large possibilité de modélisation en UML. Il permet de générer des codes sources en divers langages dont le C++ à partir du modèle créé. Inversement, il permet de produire un modèle à partir de codes sources.

Langage C#

Le langage de programmation qu'on utilisera est le C#. C'est un langage de programmation compilé permettant la programmation sous de multiples paradigmes.

Il utilise les concepts de la programmation orientée objet et permet entre autres :

- la classification (classes).
- l'encapsulation.
- des relations entre les classes :
  - la composition de classes (composition dans un diagramme de classes).
  - l'association de classes (en) (association dans un diagramme de classes).
  - l'agrégation de classes (agrégation dans un diagramme de classes).
  - la dépendance (dépendance dans un diagramme de classes).
  - l'héritage simple et multiple (héritage dans un diagramme de classes).

- le polymorphisme.
- l'abstraction.
- la généricité.
- la métaprogrammation.

Intégration continue : GitHub

Pour le partage de notre travail, nous utiliserons GitHub. GitHub est un service web d'hébergement et de gestion de développement de logiciels, utilisant le logiciel de gestion de versions Git. Le site assure également un contrôle d'accès et des fonctionnalités destinées à la collaboration comme le suivi des bugs, les demandes de fonctionnalités, la gestion de tâches et un wiki pour chaque projet.

Modélisation de l'application

Diagramme de classes

1) Qu'est-ce que le diagramme de classes ?

Les diagrammes de classes permettent de spécifier la structure et les liens entre les objets dont le système est composé : ils spécifient QUI sera à l'œuvre dans le système pour réaliser les fonctionnalités décrites par les diagrammes de cas d'utilisation.

1.1) Les classes :

Une classe est une représentation abstraite d'un d'ensemble d'objets, elle contient les informations nécessaires à la construction de l'objet (c'est-à-dire la définition des attributs et des méthodes). La classe peut donc être considérée comme le modèle, le moule ou la notice qui va permette la construction d'un objet. Nous pouvons encore parler de type (comme pour une donnée). On dit également qu'un objet est l'instance d'une classe (la concrétisation d'une classe).

1.2) Rôles du diagramme de classes :

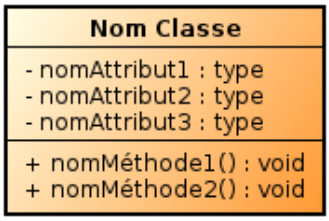
Le diagramme de classes est un diagramme structurel (statique) qui permet de représenter :

- les classes (attributs + méthodes)
- les associations (relations) entre les classes.

Le diagramme de classes est le plus important des diagrammes UML, c'est le seul qui soit obligatoire lors de la modélisation objet d'un système.

2) Représentation des Classes :

Une classe est représentée par un rectangle (**appelé aussi classeur**) divisé en 3 compartiments.



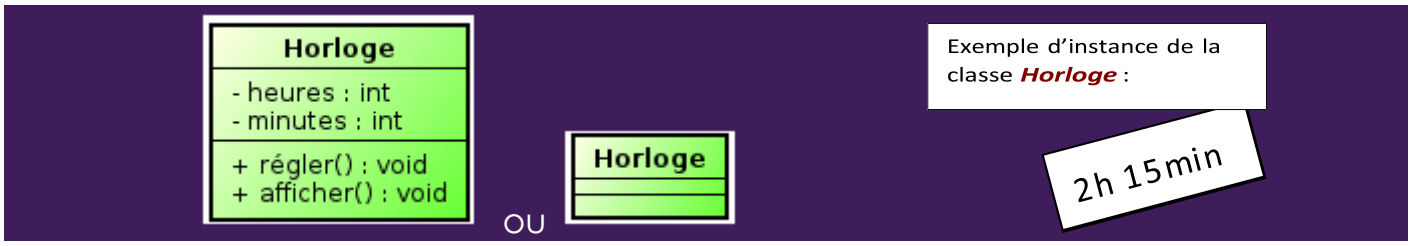
Le premier compartiment contient le *nom de la classe qui :*

- représente le type d'objet instancié.
- débute par une lettre majuscule.
- il est centré dans le compartiment supérieur de la classe. - il est écrit en caractère gras.
- il est en italique si la classe est abstraite (IMPOSSIBLE d'instancié un objet).

Le deuxième compartiment contient les *attributs*.

Le troisième compartiment contient les *méthodes*.

Si la modélisation ne s'intéresse qu'aux relations entre les différentes classe du système (et pas au contenu des classes), nous pouvons ne pas représenter les attributs et les méthodes de chaque classe (nous ne mettons rien dans le deuxième et troisième compartiment).



Exemples : *- La classe* ***Horloge***

3) Les relations entre classes :

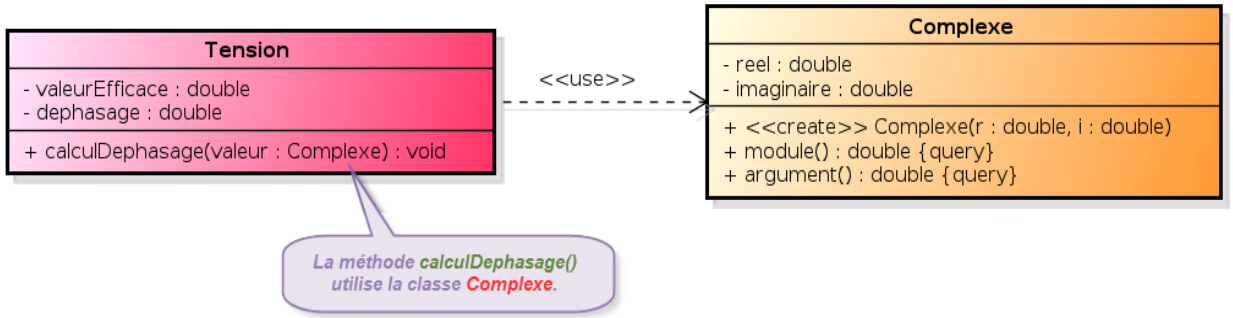
3-1) La relation de dépendance :

La dépendance est la forme la plus faible de relation entre classes. Une dépendance entre deux classes signifie que l'une des deux utilise l'autre. Typiquement, il s'agit d'une relation transitoire, au sens où la première interagit brièvement avec la seconde sans conserver à terme de relation avec elle (***liaison ponctuelle***).

Une dépendance peut s'interpréter comme une relation de type «utilise un ». Elle est habituellement utilisée lorsqu'une classe utilise un objet d'une autre classe comme argument dans la signature d'une méthode ou alors lorsque l'objet de l'autre classe est créé à l'intérieur de la méthode. Dans les deux cas la durée de vie de l'objet est très courte, elle correspond à la durée d'exécution de la méthode.

Notation : Elle est représentée par un trait discontinu orienté, reliant les deux classes. La dépendance est souvent stéréotypée (« use ») pour mieux expliciter le lien sémantique entre les éléments du modèle.

Exemple



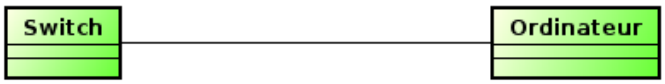
3-2) Les associations :

Alors que la dépendance autorise simplement une classe à utiliser des objets d'une autre classe, l'association signifie qu'une classe contiendra une référence (ou un pointeur) de l'objet de la classe associée sous la forme d'un attribut.

Cette relation est plus forte. Elle indique qu'une classe est en relation avec une autre pendant un certain laps de temps. La ligne de vie des deux objets concernés ne sont cependant pas associés étroitement (un objet peut être détruit sans que l'autre le soit nécessairement).

L'association est représentée par un simple trait continu, reliant les deux classes. Le fait que deux instances soient ainsi liées permet la navigation d'une instance vers l'autre, et vice versa (chaque classe possède un attribut qui fait référence à l'autre classe).

Exemple



3-3) Relation d'héritage :

Le mécanisme d'héritage permet de mettre en relation des classes ayant des caractéristiques communes (***attributs*** et ***comportements***) en respectant une certaine filiation.

L'héritage indique qu'une classes B est une spécialisation d'une classe A. La classe B (appelé classe fille, classe dérivée ou sous classe) hérite des ***attributs*** et des ***méthodes*** de la classe A (appelée classe mère, classe de base ou super classe). ***Il se représente par un triangle vide afin d'indiquer le sens de la généralisation (inverse de la spécialisation).***

Notre diagramme :

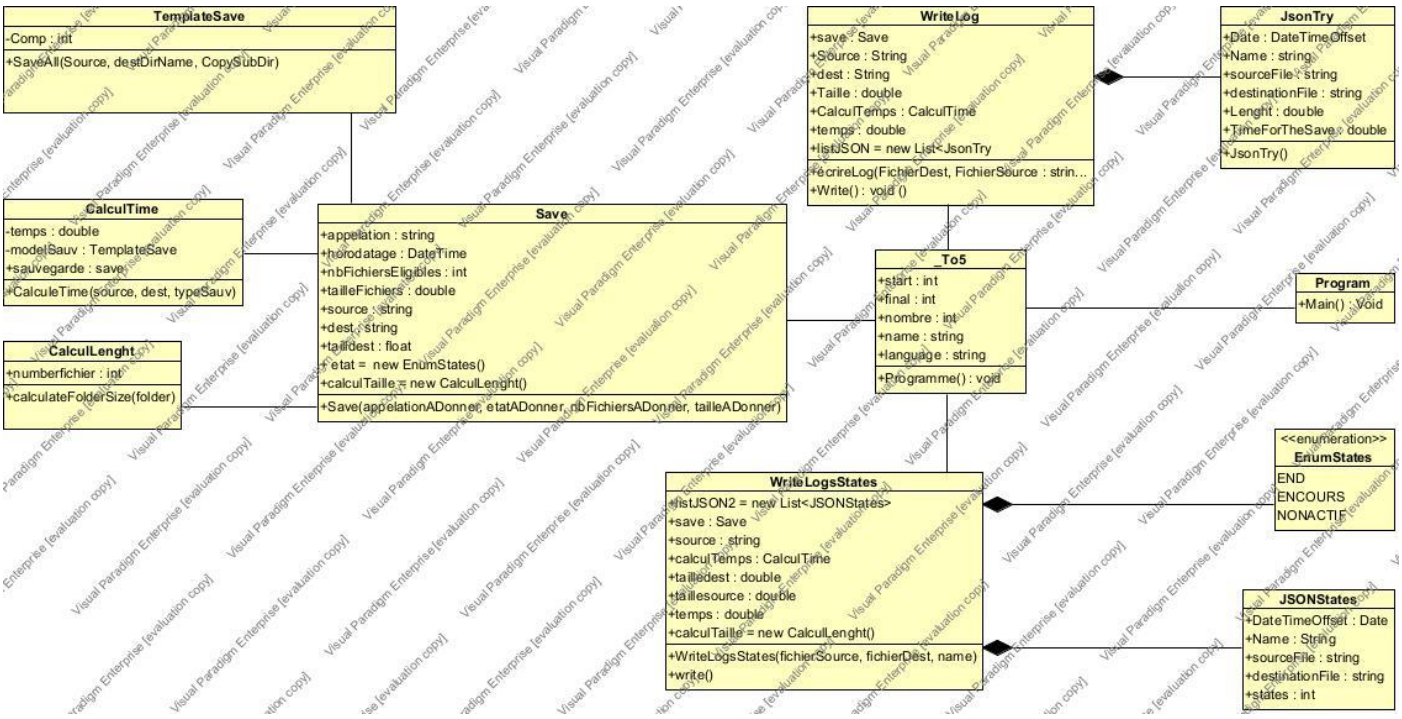


Diagramme de cas d'utilisation

Qu'est-ce que le diagramme de cas d'utilisation ?

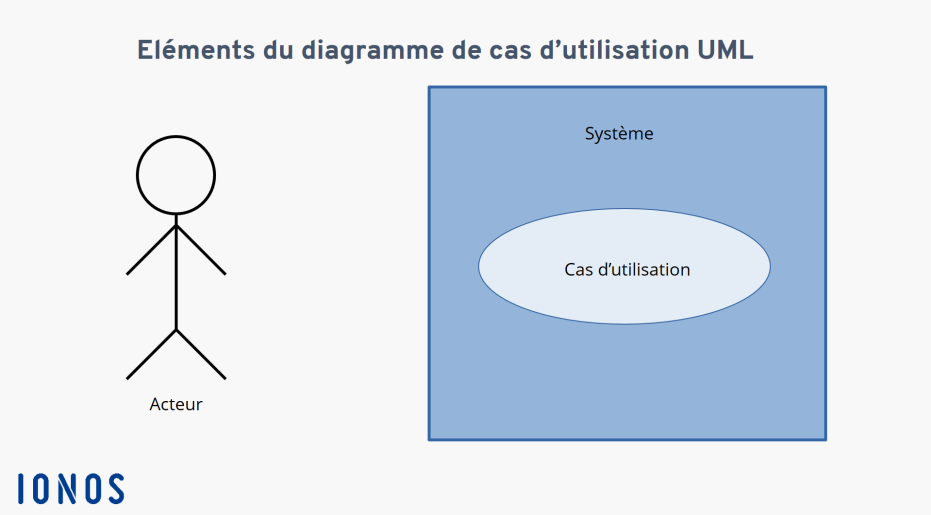
Un diagramme de cas d'utilisation permet de représenter les **fonctions d'un système du point de vue de l'utilisateur** (appelé « acteur » en UML). Cet acteur ne doit pas nécessairement être un utilisateur humain. Le rôle peut également être attribué à un système externe qui accède à un autre système. Le diagramme de cas d'utilisation montre en fait la **relation entre un acteur et ses demandes ou attentes vis-à-vis du système**, sans décrire les actions en cours ni les mettre dans un ordre logique.
Avec ce type de diagramme, on va pouvoir représenter les fonctionnalités, les dépendances entre elles ainsi que les différents acteurs interagissant avec la solution. De cette façon on s'assure une communication aisée avec la maitrise d'ouvrage en vue de se mettre d'accord sur les personnes en relation avec la solution ainsi que sur la nature de ces relation.

Structure du diagramme de cas d'utilisation

En pratique, cette structure est bien adaptée pour décrire de manière claire les fonctions et/ou les objectifs les plus importants d'un système. C'est pour cette raison que l'élaboration d'un diagramme de cas d'utilisation est souvent l'une des premières étapes **lors de la conception de logiciels ou de la planification de nouveaux processus métier**. Cela permet de visualiser facilement et clairement quels cas d'utilisation doivent être pris en compte dans la conception pour que les acteurs (et au sens large également l'opérateur ou le client) atteignent les objectifs escomptés – sans tenir compte dans un premier temps de la faisabilité technique.

Pour garantir que le diagramme de cas d'utilisation soit immédiatement compréhensible pour tout un chacun, on utilise des briques de base standardisées pour la représentation. Il y a en premier lieu trois éléments principaux :

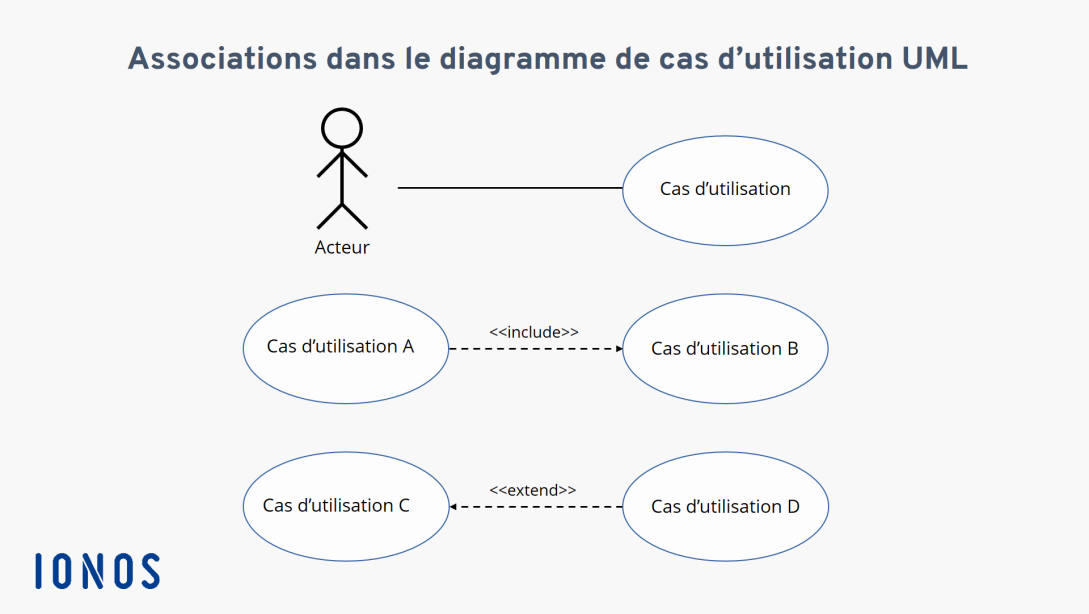
- **Acteur :** les acteurs, que ce soit des personnes ou des systèmes, sont représentés par des bonhommes en fil de fer.
- **Système :** le système auquel se rapporte le cas d'utilisation est représenté par un rectangle.
- **Use Case :** le cas d'utilisation lui-même est représenté par une ellipse, dans laquelle il y a généralement une courte phrase décrivant le processus**.**



Symboles utilisés pour représenter les éléments acteur, système et cas d'utilisation

Les relations entre ces éléments sont représentées par des lignes de connexion, appelées associations. Une **ligne continue entre un acteur et un cas d'utilisation** indique clairement que l'acteur et le cas d'utilisation décrits dans l'ellipse sont liés. Il y a aussi des **lignes pointillées pour représenter une relation entre différents cas d'utilisation**. Comme il existe deux types différents d'association entre les cas d'utilisation, les lignes sont complétées par un mot-clé appelé « stéréotype » en UML et représenté entre crochets. Une pointe de flèche montre par ailleurs une relation de dépendance entre cas d'utilisation. Une distinction est faite entre ces deux stéréotypes :

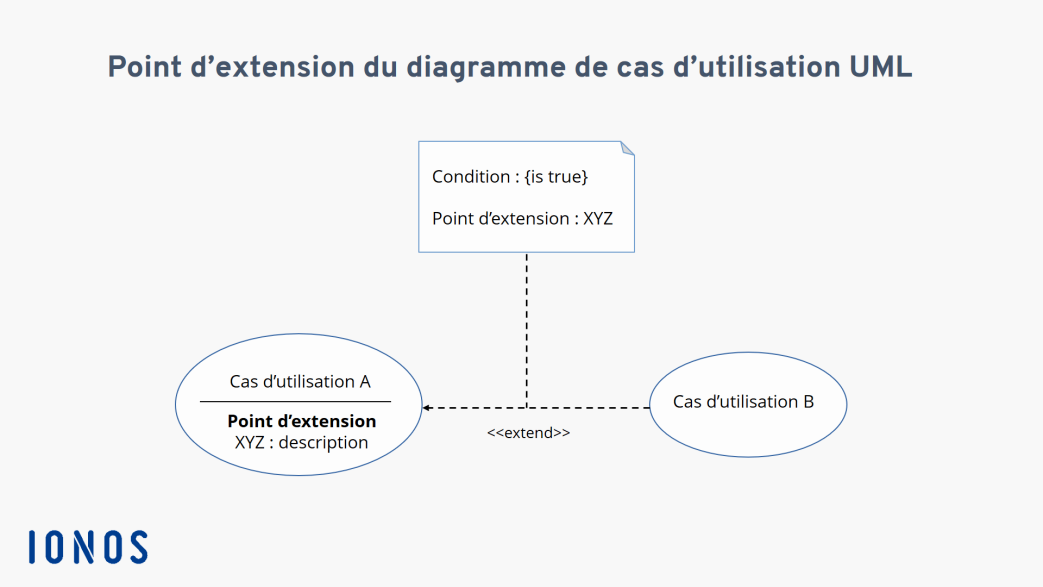
- **L'association d'inclusion (include) :** le cas d'utilisation d'où part la ligne de connexion en pointillés inclut un deuxième cas d'utilisation, pointé par la pointe de flèche.
- **L'association d'extension :** le cas d'utilisation d'où part la ligne de connexion en pointillés commence peut, sous certaines conditions, étendre le cas d'utilisation pointé par la tête de flèche. Mais ce n'est pas toujours le cas.



Les lignes mettent en évidence les relations entre les éléments individuels par le biais des pointes de flèche et des stéréotypes.

Alors que l'association d'inclusion nécessite la prise en compte des deux cas d'utilisation, la prise en compte du deuxième cas d'utilisation, dans l'association d'extension, dépend de certaines conditions. Ces conditions sont spécifiées dans le diagramme de cas d'utilisation UML par le biais d'un **point d'extension**. Ceci est représenté de deux manières différentes :

- **Extension de l'ellipse du cas d'utilisation :** le point d'extension possible est nommé et brièvement décrit sous le nom du cas d'utilisation.
- **Note** : le stéréotype d'extension est relié par une ligne en pointillés à une note stylisée (rectangle avec un coin plié) intitulée « Condition » et « Extension ». Derrière « condition », on spécifie entre accolades quelle condition doit être remplie pour que le deuxième cas d'utilisation soit pris en compte. Derrière le point d'extension, on renvoie à son nom dans l'ellipse du cas d'utilisation afin que l'extension puisse être attribuée sans équivoque.



Si les conditions définies dans le point d'extension s'appliquent, le cas d'utilisation B se produira et modifiera le cours du cas d'utilisation A.

Notre diagramme :

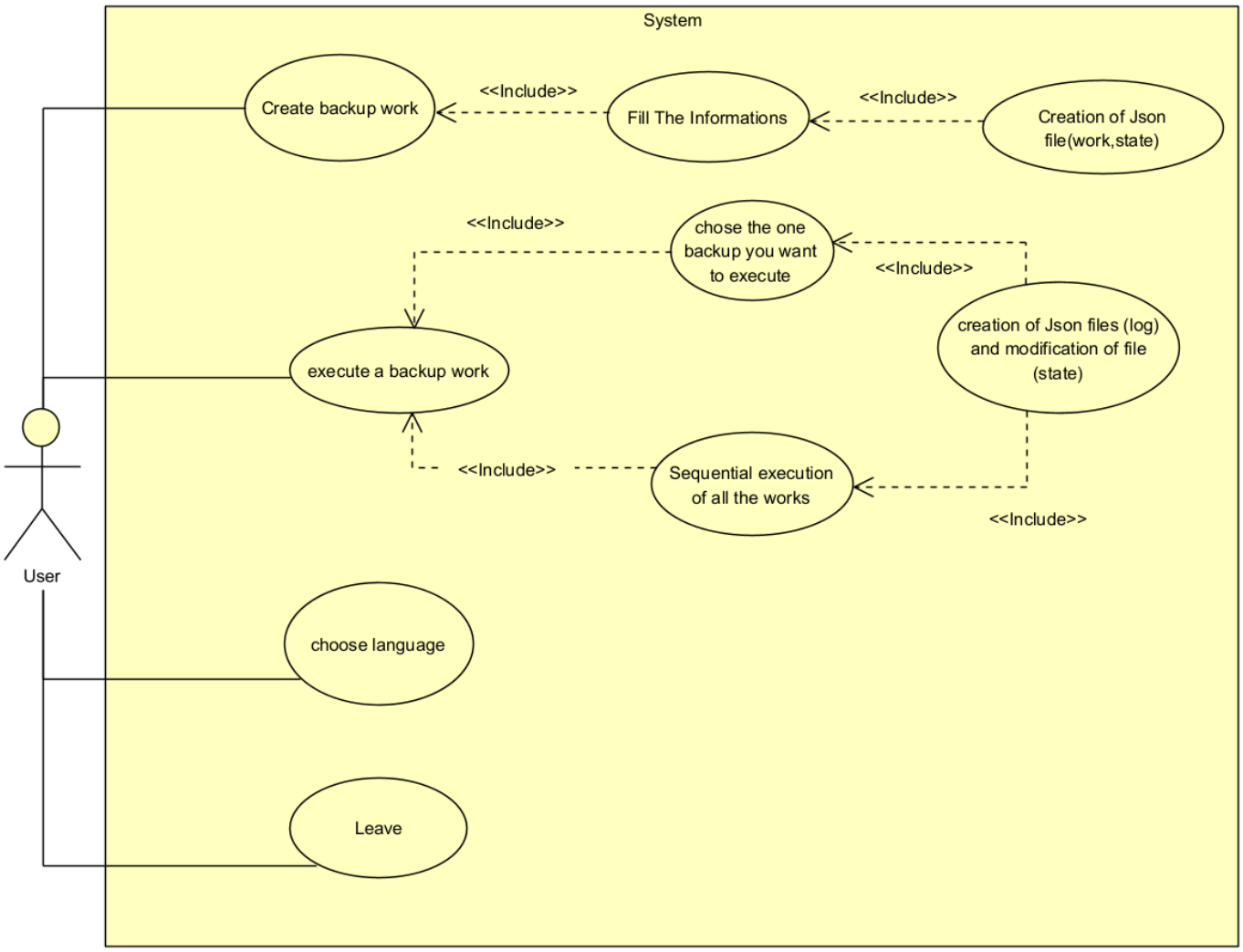


Diagramme de séquences

Qu'est-ce que le diagramme de séquences ?

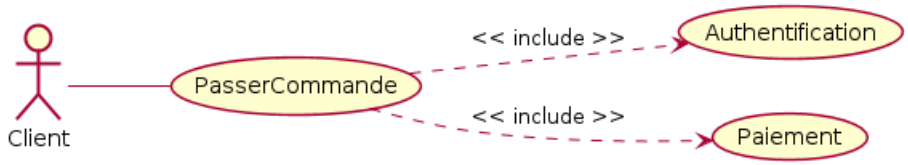
Les diagrammes de cas d'utilisation modélisent à QUOI sert le système, en organisant les interactions possibles avec les acteurs. Ils permettent de décrire COMMENT les éléments du système interagissent entre eux et avec les acteurs :

- 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).

Utilisation des diagrammes de séquences :

Les diagrammes de séquences sont principalement utilisés pour :

- Documenter des cas d'utilisation. Dans ce cas, un acteur est toujours présent.



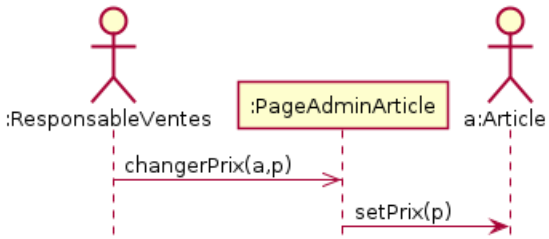
- Définir des opérations. Dans ce cas, on initie souvent le diagramme par un message trouvé et on est particulièrement rigoureux dans la définition des éléments du modèle.

Interaction :

Pour être complètement spécifiée, une interaction doit être décrite dans plusieurs diagrammes UML :

- Cas d'utilisation

- Séquences



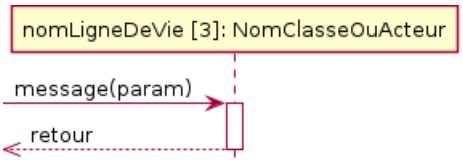
Ligne de vie :

Une ligne de vie représente un participant à une interaction (objet ou acteur). La syntaxe de son libellé est :

```
nomLigneDeVie {[selecteur]}: NomClasseOuActeur
```

Une ligne de vie est une instance, donc il y a nécessairement les *deux points* (*:*) dans son libellé.

Dans le cas d'une collection de participants, un sélecteur permet de choisir un objet parmi n (par exemple `objets[2]`).



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
- Ils sont présentés du haut vers le bas le long des lignes de vie, dans un ordre chronologique

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

Plusieurs types de messages existent, dont les plus courants :

- l'envoi d'un signal ;
- l'invocation d'une opération (appel de méthode) ;
- la création ou la destruction d'un objet.

La réception des messages provoque une période d'activité (rectangle vertical sur la ligne de vie) marquant le traitement du message (spécification d'exécution dans le cas d'un appel de méthode).

Fragment combiné :

Un fragment combiné permet de décomposer une interaction complexe en fragments suffisamment simples pour être compris.

- Recombiner les fragments restitue la complexité.
- Syntaxe complète avec UML 2 : représentation complète de processus avec un langage simple (ex : processus parallèles).

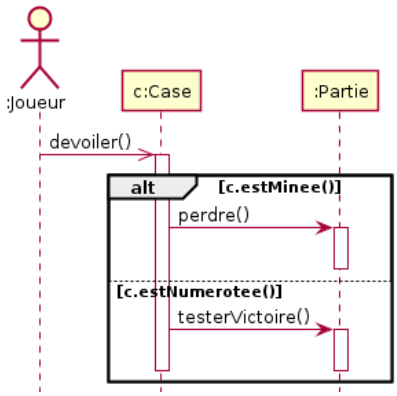
Un fragment combiné se représente de la même façon qu'une interaction. Il est représenté un rectangle dont le coin supérieur gauche contient un pentagone.

Dans le pentagone figure le type de la combinaison (appelé *opérateur d'interaction*).

Fragment alt : opérateur conditionnel

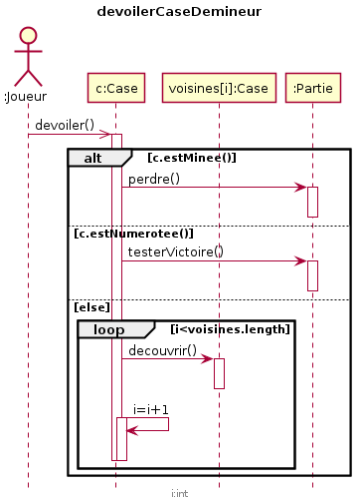
Les différentes alternatives sont spécifiées dans des zones délimitées par des pointillés.

- Les conditions sont spécifiées entre crochets dans chaque zones.
- On peut utiliser une clause `[else]`



Fragment loop : opérateur d'itération

Le fragment `loop` permet de répéter ce qui se trouve en son sein.On peut spécifier entre crochets à quelle condition continuer.



Notre diagramme :

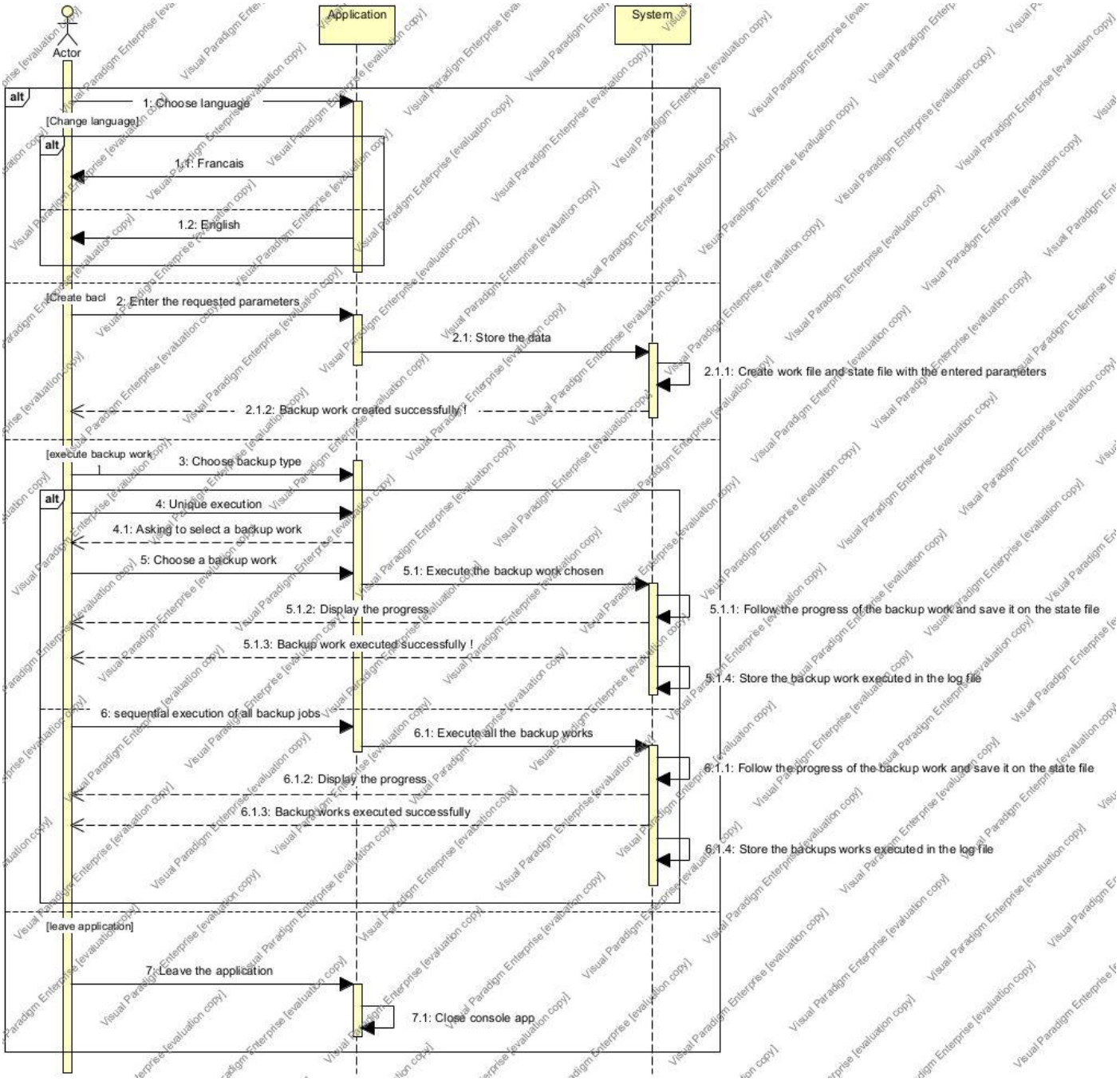


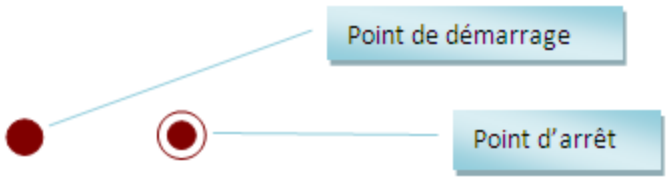
Diagramme d'activité

Qu'est-ce que le diagramme d'activité ?

Un diagramme d'activité est utilisé pour afficher la séquence des activités. Les diagrammes d'activité représentent le flux de travail à partir d'un point de départ au point d'arrivée. Détaillant les nombreux sentiers de décision qui existent dans la progression des événements contenus dans l'activité. Ils peuvent être utilisés à des situations de détail, où le traitement parallèle peut survenir dans l'exécution de certaines activités. Les diagrammes d'activités sont utiles pour la modélisation d'entreprise où ils sont utilisés pour détailler les processus impliqués dans des activités commerciales.
Ce type de diagramme représente les processus et leur enchainement logique dans leur système respectif ainsi que les interactions entre les systèmes et composants de la solution. On s'assurera ainsi d'avoir modéliser correctement un processus, l'ensemble de ses étapes, les données collectées ou produites ou simplement utilisés.

Point de démarrage et d'arrêt

Le diagramme est composé d'un point de démarrage, d'un point arrêt et d'action, qui sont représenté par des cercles rouges.

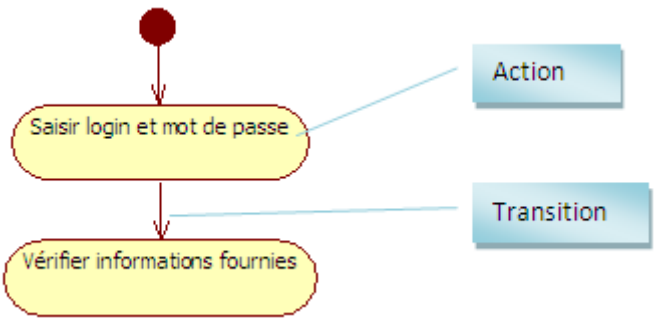


Point d'arrêt et point de démarrage

Les actions et les transitions

Un diagramme d'activité est une formalisation graphique des actions qui sont réalisées dans un cas d'utilisation.

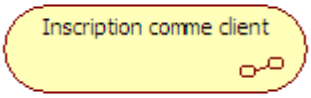
Le diagramme est donc organisé en actions réalisées soit par un acteur, soit par le système, relié par une flèche indiquant l'enchaînement des actions.



Action / transition

Le lot d'actions, ou autre cas d'utilisation

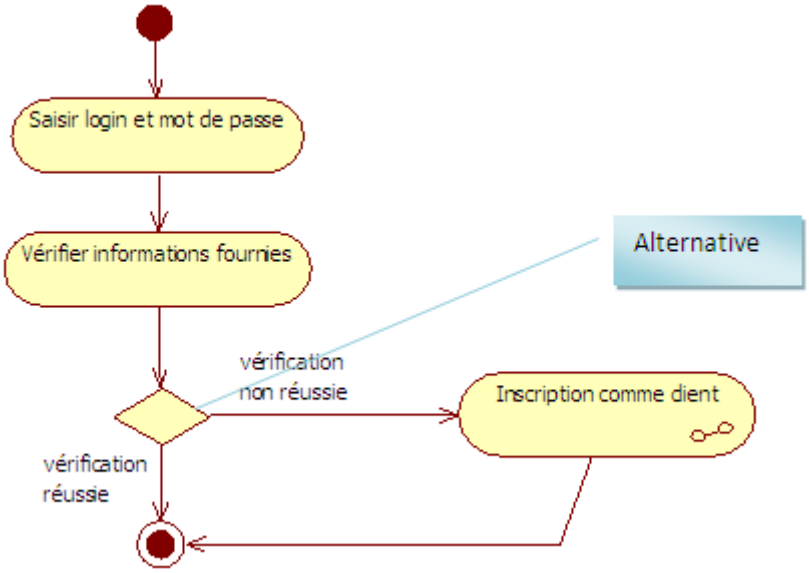
Si une action du cas d'utilisation correspond à l'appel d'un cas d'utilisation interne (lié par une relation de type « include » ou « extend ») ; elle est représentée par une action contenant un signe spécial : deux cercles reliés par un trait.



Cas d'utilisation inclus

L'alternative

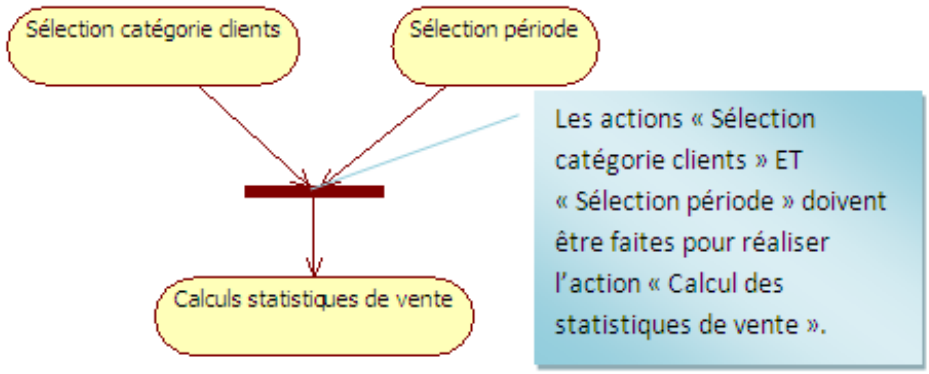
Elle permet d'indiquer les différents scénarios du cas d'utilisation dans un même diagramme. Dans l'exemple, il s'agit de la condition d'après laquelle le cas d'utilisation « Inscription comme client » serait appelé.



L'alternative

La synchronisation

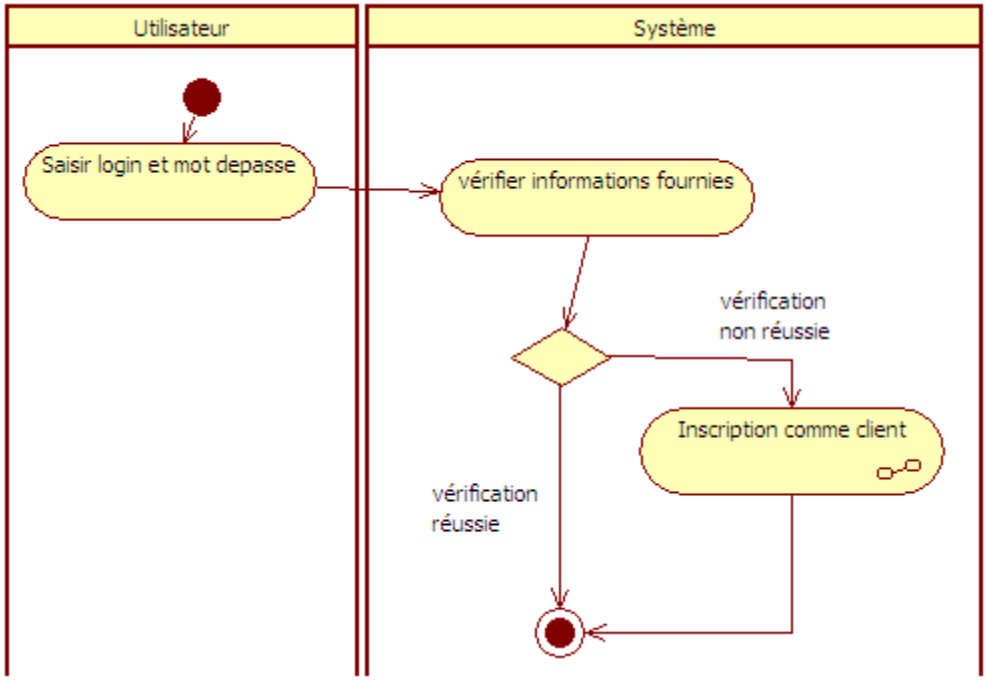
Elle indique qu'il faut avoir réalisé deux actions pour pouvoir réaliser la troisième en-dessous.



La synchronisation

Les couloirs « swimlanes »

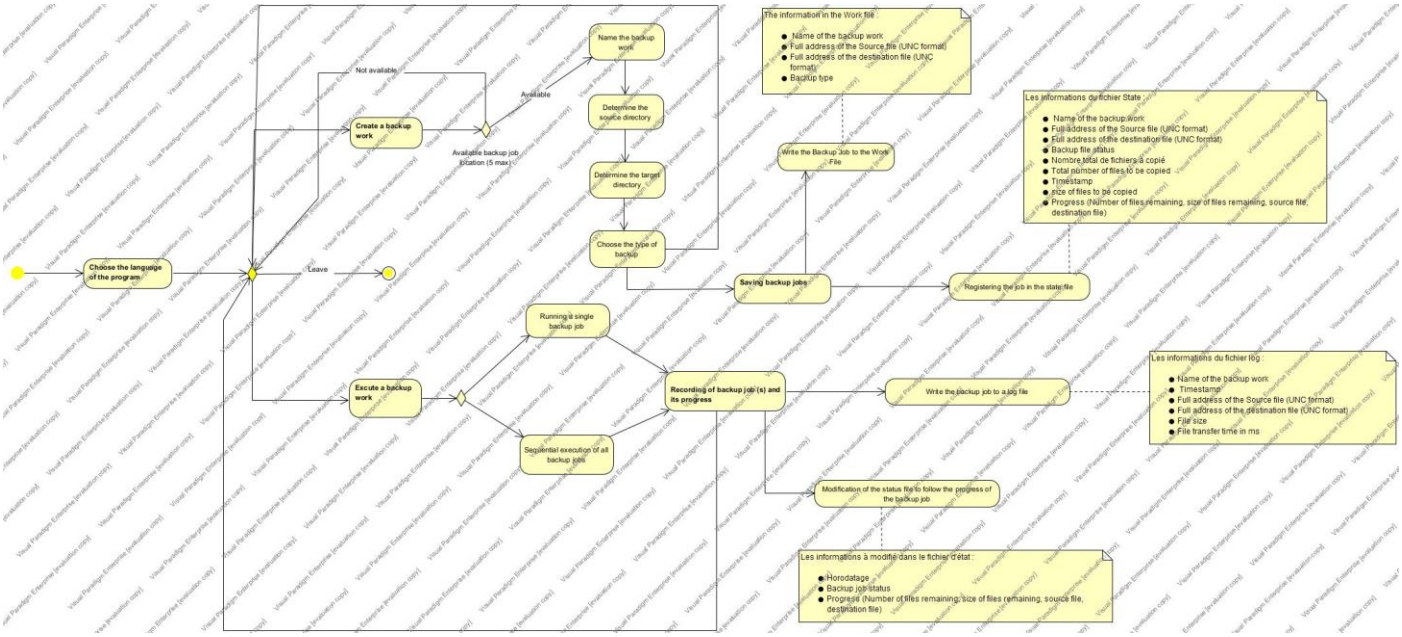
Ils permettent d'indiquer qui (de l'utilisateur ou du système) réalise les actions.



Les 'swimlanes'

Un diagramme d'activité est donc un bon complément à la fiche descriptive d'un cas d'utilisation complexe. Si un cas d'utilisation contient de nombreux scénarios, le diagramme d'activité permet de donner une vision globale de l'ensemble des scénarios possibles.

Notre diagramme :



Conclusion

Pour notre première version du logiciel nous avons donc produit une application console en C# qui nous permet de créer jusqu'à 5 travaux de sauvegarde, nous avons aussi le choix de la mettre en anglais ou en Français. Notre logiciel permet aussi l'écrire d'un fichier log journalier qui contient l'historiques des actions des travaux de sauvegardes, avec ceci on enregistrer en temps réel, dans un fichier unique, l'état d'avancement des travaux de sauvegarde. La version 2.0 de notre logiciel contiendra une architecture MVVM pour notre programme avec une interface graphique pour un meilleur visuel