Compte rendu ISST

Service spatio temporal : gère les évènements historiques pour éviter d'interférer avec sous peine de créer de nouveaux flux temporels

La SST possède une méta base de donnée contenant : date d'évènements,localisations, personnages importants concernés

La SST possède une flotte de vaisseau pour accéder à différents endroits du flux.
Un vaisseau possède une classe(A,B,C,X) qui définit sa capacité de saut temporel (amplitude, capacité de rebond)
A : aller/retour , centaine d'année en amplitude.
B : nombre limité de voyage + 1 retour d'urgence, millier d'année en amplitude
C:Nombre limité de voyage + 1 retour d'urgence, aucune limite d'amplitude.
X : Aucune limite tant qu'il reste de l'énergie.

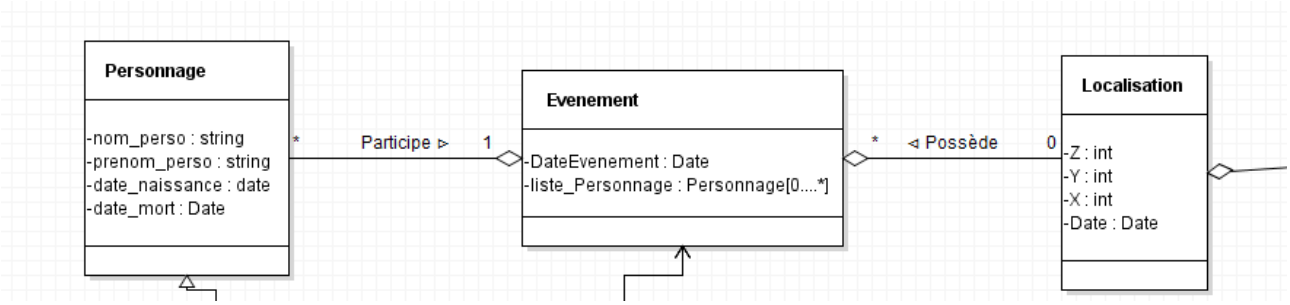
La SST contient une troupe d'agents accrédités pour les différentes classes de missions existantes et pour les classes de vaisseaux associées .

Element : Agent, troupe d'agent , capacité de rebond, vaisseau, vaisseau A,B,C,X, flotte de vaisseau , Evenement (contenant une date, localisation, des personnages importants)
 Et personnage.
Mission,
Rapport de Mission .

Objectif : Être capable de confier une mission à un agent sur un fait historique à l'aide d'un vaisseau, l'agent fera donc, un rapport de mission.

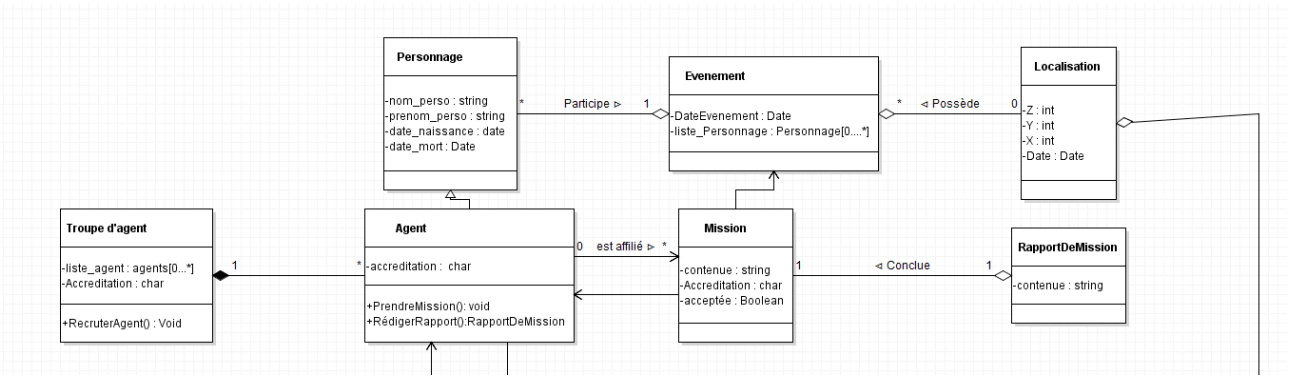
Pour répondre à la problématique posée par ce sujet, mon UML peut être diviser en trois parties, bien qu'elle soit lié ensemble:

La gestion des événements



Ici, il s'agit d'un UML simple, un Personnage participe à un événement en même temps, caractérisé par un lieu (Une Localisation) ainsi qu'une multitude de personnages y participant . Nous avons créer directement un objet Localisation pour que ce soit plus simple à implémenter aussi avec les Vaisseaux qui pourront se mouvoir .

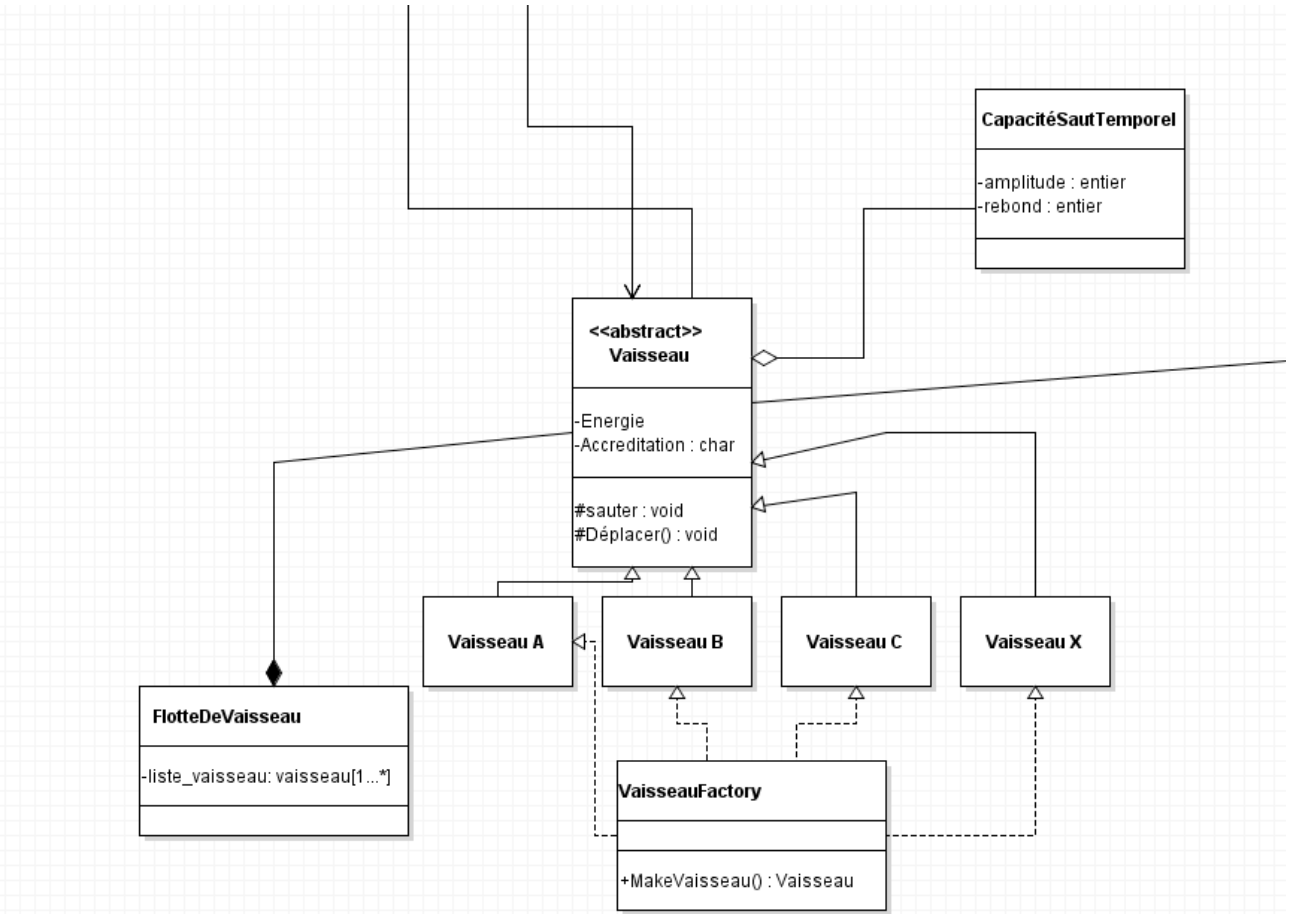
La gestion des Agents / Mission



Ici, l'UML contient la partie des Evenements.

Un agent hérite de personnage, il pourrait très bien ne pas en hériter et avoir ses propres attributs, c'est afin de préciser qu'un agent est un type spécifique de personnage .
Un agent fait partie d'une troupe d'agent, une troupe d'agent n'existe pas si elle est vide, ici c'est un choix qui est fait, mais qui n'est pas obligatoire. Un agent est affilié à 1 mission maximum, et une mission est affilié à plusieurs agents (Par la troupe, ici), et un un rapport de Mission conclue une mission (Dans l'implémentation, nous pourrons vérifier qu'une Mission a été rempli via la présence d'un rapport de Mission, et qu'elle a été accepté ou non (Via son attribut «acceptée»)) .

La gestion des vaisseaux :



Etant donnée la variété de vaisseaux existants , un choix a été fait ici, celui de l'utilisation du patron de conception «Factory» permettant d'instancier différents types d'objets sans savoir ce qu'ils sont à l'avances, et nous permettra aussi d'ajouter de nouveaux types de vaisseaux à partir d'une base de donnée ou d'un fichier texte par exemple.

Pour ce faire, notre usine à vaisseau implémente plusieurs types de vaisseaux, qui eux, hérite d'un objet abstrait vaisseau, possédant une énergie et une accréditation, et ayant une capacité de capacité temporelle . Lorsqu'un vaisseau sera créer à partir de l'usine à vaisseau, il prendra dans son constructeur une lettre (Définissant donc son type, s'il est A,B,C,X ou autre), et un objet correspondant sera créer en fonction des attributs renseignées dans le .txt

A noté qu'un Vaisseau compose une flotte de vaisseau, et qu'une flotte vide n'existe pas. (On aurait pu prendre un agrégat, ici aussi.)

Dans l'UML final, un vaisseau est lié à un agent qui le pilote , et possède une localisation pour savoir s'il est en place sur l'évènement. (Dans l'implémentation, on pourra savoir dans une fonction en comparant le lieu d'un événement donné, ainsi que le lieu du Vaisseau) .