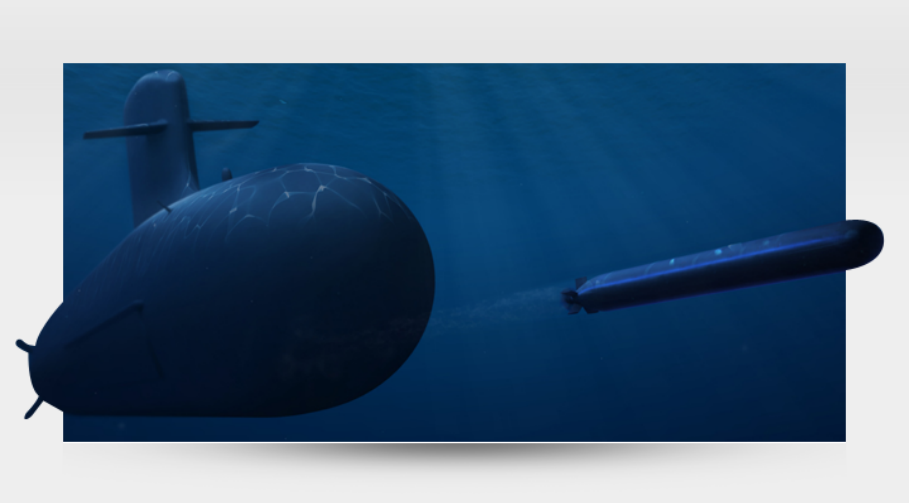
Projet d'UML : Submersibles

(IUT, département informatique, 1re année)

Laurent AUDIBERT



L'objectif du projet consiste à proposer un modèle UML d'une petite application permettant de mettre en œuvre des combats entres deux sous-marins, puis de proposer une implémentation en Java de cette application.

1 Description du jeu

1.1 Comment naissent les submersibles ?

L'application mettra en scène quatre types de sous-marins : le standard, l'agile, le cuirassé et le furtif. Construire un sous-marin se résume à choisir le type du sous-marin parmis les quatre types existants.

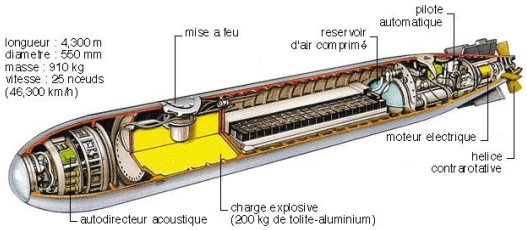
Le joueur doit ensuite équiper son sous-marin. Pour cela, il dispose d'un budget de 70 unités (U) pour acheter des torpilles, un sonar, une propulsion ainsi qu'un blindage. Le joueur peut acheter ce qu'il veut, mais il a tout intérêt de s'équiper au moins en torpilles, sans quoi il ne pourrait jamais attaquer son adversaire. La section 1.2 détaille l'ensemble des équipements disponibles.

Comme nous le verrons dans la section 1.3, le type du sous-marin peut moduler la performance de ses équipements.

1.2 Description des équipements d'un sous-marins

1.2.1 Torpilles

Description générale : Les torpilles constituent la seule arme offensive des sous-marins. Elles sont utilisées pour endommager, et à terme détruire, le sous-marin adverse. Quand une torpille explose suffisamment prêt d'un sous-marin, elle retourne une information sur la position du sous-marin potentiellement touché.



Caractéristique : Elles sont caractérisées par une puissance (5, 10, 15, 20 ou 30).

Coût unitaire : **30** (50 U), **20** (20 U), **15** (10 U), **10** (5 U), **5** (2 U).

Fonctionnement : Une torpille est programmée lors de son lancement pour toucher une cible en spécifiant ses coordonnées x et y. Une torpille d'une puissance x doit exploser à une distance inférieure à x unités de distance pour faire des dégâts sur le sous-marin adverse. D'une manière générale, les dégâts se calculent en soustrayant à la puissance de la torpille la distance à laquelle se trouve la cible ainsi que l'épaisseur du blindage de la cible. Par exemple, une torpille d'une puissance 20 explosant à une distance de 3,1 (arrondi à 3) fera 13 points de dégâts sur un sous-marin possédant une épaisseur de blindage de 4 ($20 - 3 - 4 = 13$).
D'autre part, quand une torpille d'une puissance x explose à une distance inférieure à x unités de distance d'un sous-marin adverse, elle retourne deux informations précieuses à son lanceur : le nombre de points de dégât encaissés par l'adversaire ainsi que la distance de cet adversaire par rapport au point d'impact.

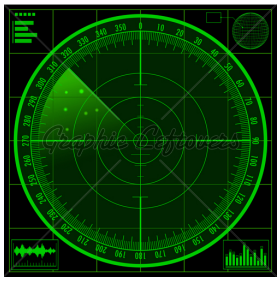
Particularités : Un sous-marin peut emporter autant de torpilles qu'il le veut (en fonction de ses finances). Un sous-marin garde ses torpilles non utilisées pour la partie suivante.

1.2.2 Sonar

Description générale : Le sonar est un équipement permettant de localiser avec plus ou moins de précision son adversaire.

Caractéristique : Un sonar est caractérisé par une imprécision (10, 6, 3 ou 0).

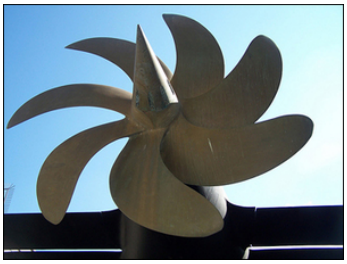
Coût : **0** (100 U), **3** (75 U), **6** (50 U), **10** (30 U)



Fonctionnement : Un sous-marin ne peut sonder qu'un quart du plan (quart NE, NO, SE ou SO) en prenant comme origine sa position. Le sous-marin doit donc choisir le quart qu'il désire sonder. Si son adversaire se trouve dans ce quart, sa position (coordonnées (x, y)) est retournée en ajoutant à chacune des coordonnées un nombre aléatoire compris entre $-précision$ et $+précision$. Par exemple, si la précision du sonar est de 6, la valeur de la coordonnée x retournée sera un nombre aléatoire entre $positionX - 6$ et $positionX + 6$ où $positionX$ est la coordonnée x de la position de l'adversaire sondé.

1.2.3 Propulsion

Description générale : La propulsion est l'équipement permettant à un sous-marin de se déplacer.



Caractéristique : Une propulsion est caractérisée par une puissance (2, 1, 6, 8 ou 10).

Coût : **10** (100 U), **8** (80 U), **6** (60 U), **4** (40 U), **2** (20 U)

Fonctionnement : Une propulsion d'une puissance de 8 permet de se déplacer de 8 unitées de distance par tour de jeux. Un joueur n'est pas obligé d'utiliser sa propulsion à plein régime.

1.2.4 Blindage

Description générale : Le blindage permet de réduire l'impact des torpilles adversses qui font mouche comme décrit dans la section concernant les *Torpilles*.

Caractéristique : Le blindage est caractérisé par une épaisseur (2, 4, 6, 8 ou 10).

Coût : **10** (100 U), **8** (80 U), **6** (60 U), **4** (40 U), **2** (20 U)

1.3 Description des quatres types de sous-marin

Le type du sous-marin peut moduler la performance de ses équipements.

Le standard : Il s'agit d'un sous-marin polyvalent ne comportant pas de capacité particulière.

L'agile : Il s'agit d'un sous-marin plus léger, donc plus rapide (+4 en propulsion par rapport à son équipement), mais également moins solide (−2 en épaisseur de blindage).

Le cuirassé : il s'agit d'un sous-marin au blindage renforcé (+2 en épaisseur de blindage), mais également plus lourd et donc moins rapide (−2 en propulsion par rapport à son équipement).

Le furtif : il s'agit d'un sous-marin plus difficile à répérer au sonnar (il faut ajouter +5 à l'imprecision du sonar de l'adversaire quand un sous-marin furtif est sondé), mais un peu moins rapide (−2 en propulsion par rapport à son équipement) et au blindage plus faible (−1 en épaisseur de blindage) pour rester plus discret.

Remarques – Un blindage négatif se traduit par des points de dégâts en plus quand un sous-marin est touché. Un sous-marin avec une puissance de propulsion inférieure ou égale à 0 ne peut pas se déplacer.

1.4 Déroulement d'un combat naval subaquatique

Au début d'un combat, chacun des participants se voit attribuer une somme de 40 unités pour s'équiper ainsi qu'une capacité à encaisser 20 points de dégâts. La position des sous-marin est tirée aléatoirement avec des coordonnées x et y compris entre -100 et $+100$. La vitesse initiale d'un sous-marin est nulle. Son cap est tiré aléatoirement parmis les huits caps possibles : N, NE, E, SE, S, SO, O et NO.

Un combat se déroule par tour de jeu. À chaque tour de jeu, chacun des deux joueurs peut faire l'une des actions suivantes :

- tirer une torpille ;
- moduler sa vitesse de une unité en plus ou en moins sans dépasser la puissance de sa propultion ;
- sonder un quart de l'océan ;
- changer de cap par pas de $\pm 45°$.
- capituler.

Une capitulation met fin à la partie sans qu'aucun sous-marin ne soit détruit. Quand un joueur capitule, son adversaire reçoit une somme égale au quart du coût-total des équipements embarqués par son adversaire au moment de sa capitulation. Le joueur qui capitule perd toutes ses torpilles. Quand un sous-marin est détruit, le vainqueur reçoit une somme égale à la moitié du coût-total des équipements embarqués par son adversaire au moment de sa destruction. Si les deux adversaires se retrouvent sans torpille, la partie est nulle et prend fin.

1.5 Fonctionnalités de l'application

L'application doit permettre de créer de nouveaux sous-marins et de les équiper avant un combat.

L'application doit, bien entendu, permettre de gérer des combats entre deux adversaires. Un combat doit se dérouler avec deux exécutions distinctes de l'application (une par joueur) communiquant par l'intermédiaire d'un fichier d'échange (dont l'emplacement est paramétrable) pour synchroniser le déroulement du combat.

La sauvegarde des sous-marins en fin de combat est automatique.

2 Partie UML du projet (Responsable : Laurent Audibert)

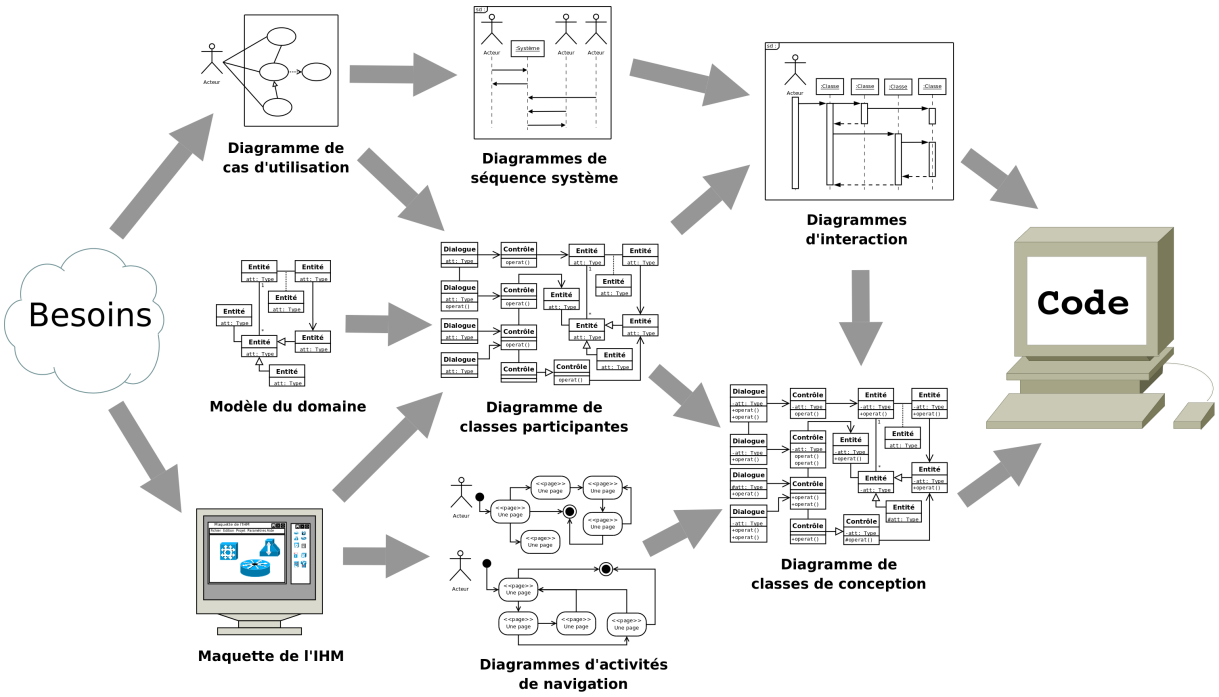


FIGURE 1 – Chaîne complète de la démarche de modélisation du besoin jusqu'au code.

2.1 Méthode de conception

Pour modéliser ce système, vous devrez utiliser la méthode minimale décrite en cours et synthétisée par la figure 1.

Dans le cadre du projet d'UML, vous n'irez pas jusqu'à l'écriture du code : l'implémentation concerne le projet de java. Toutes les autres étapes devront être prises en compte. Une manière de procéder pourrait donc être la suivante.

1. Identifiez et spécifiez les besoins en réalisant un *diagramme de cas d'utilisation* accompagné de la description textuelle des cas d'utilisation.
2. Illustrez les cas d'utilisation à l'aide de *diagrammes de séquence système*.
3. Analysez et modélisez le domaine en proposant un *modèle du domaine* sous forme de diagramme de classes.
4. Proposez une *maquette d'interface* graphique ou textuelle de l'application.
5. A l'aide d'un *diagramme d'activité de navigation* montrez la navigation dans l'interface graphique ou textuelle de l'application.
6. La première version du diagramme de classes (le modèle du domaine) doit être reprise pour y adjoindre les classes de contrôle et d'interface. Vous obtiendrez ainsi le *diagramme de classes participantes*
7. Chaque diagramme de séquence système doit être détaillé en prenant en compte les classes figurant dans le diagramme de classes participantes. Ces *diagrammes d'interaction* éclatent le système, qui n'est plus considéré comme une boîte noire, en plusieurs classes participantes de manière à montrer comment elles interagissent pour réaliser chacun des cas d'utilisation.

8. Les messages échangés entre les classes participantes dans les diagrammes d'interaction permettent de définir les opérations de chacune des classes participantes. Ces opérations seront spécifiées dans la nouvelle version du diagramme de classes : le *diagramme de classes de conception*. On veillera à ce que toutes les opérations soient réalisables, et que tout ce qui a été défini dans les activités de navigation soit présent.

Cette méthode est minimale et flexible. Si vous trouvez pertinent de spécifier certains aspects du logiciel en utilisant d'autres diagrammes, vous êtes naturellement libres de le faire.

Conseil : Pour ce projet, vous pouvez utiliser le patron de conception *Stratégie* très bien décrit dans l'article *Les limites de l'héritage : le pattern strategy* sur *le Site du Zéro* (`http://www.siteduzero.com/tutoriel-3-65563-les-limites-de-l-heritage-le-pattern-strategy.html`).

2.2 Modalités de remise du projet

Ce projet est à réaliser en binômes. Le rendu se fera impérativement sous forme de dossier papier à remettre le lundi 28 mai 2012 au secrétariat à l'attention de votre chargé de TD. Le 30 ou le 31 mai, vous recevrez par mail une confirmation de la bonne réception du projet. Si ce n'est pas le cas, inquiétez-vous !

3 Partie Java du projet (Responsable : Pierre Gérard)

3.1 Implémentation

L'implémentation java doit se faire obligatoirement après la modélisation UML et doit strictement respecter cette dernière. S'il s'avère que le modèle n'est pas adéquat, vous le modifierez avant de poursuivre l'implémentation. Toute rétro-analyse, consistant à concevoir le modèle UML à partir d'un programme java, sera très fortement pénalisée.

L'utilisation du polymophisme est obligatoire lorsqu'elle est possible. La clarté et la concision du programme seront considérées : le programme doit être commenté, les noms des classes, variables et méthodes seront évocateurs, enfin les méthodes contiendront peu de lignes.

L'affichage des menus doit être indépendant de la gestion du jeu.

3.2 Modalités de remise du projet

Le projet java sera remis le lundi 11 juin selon des modalités fournies ultérieurement. Une soutenance individuelle aura ensuite lieu où le modèle UML, l'implémentation java et le travail de l'étudiant au sein du binôme seront évalués. Deux étudiants d'un même binôme pourront avoir deux notes différentes en fonction de leur implication.