Projet de développement collaboratif basé sur UML/Java

L'objectif de ce projet est de comprendre comment mettre en œuvre un développement orienté objet à travers un travail d'équipe en appliquant le **Processus Unifié avec UML**.

1 Cahier Des Charges

1.1 Introduction

L'organisation de secours lors d'une catastrophe majeure est un problème important. La gestion des différentes équipes travaillant sur un incendie est très complexe. On cherche donc actuellement à développer des moyens informatiques permettant de faciliter et de guider le travail des sauveteurs sur le terrain. Un domaine de recherche est l'utilisation de robots pompiers (cf. Figure 1) collaborant avec des drones fournissant une carte comportant des informations sur le terrain et les incendies.

Figure 1 - Un robot pompier araignée OLE développé par l'université de Magdeburg-Stendal

La gestion de ces robots, en fonction de leurs capacités de déplacement, de leur puissance, de leur nombre, et des algorithmes de calcul de trajectoire, est complexe et nécessite de réaliser des simulations. Il vous est demandé de réaliser un logiciel permettant de simuler une équipe de robots pompiers. Les roboticiens qui souhaitent utiliser ce logiciel fournissent les informations présentées ci dessous sur les cartes manipulées, les robots, le manager de tâches et le simulateur.

1.2 La carte

Les données fournies par les drones sont des cartes du terrain sous forme de matrice n x n (n pouvant aller de 20 à plusieurs milliers). Chaque case de la matrice représente une portion de terrain et est associée à une valeur entière ou réelle qui représente la « difficulté » du terrain : plus la valeur est élevée, plus le robot aura du mal à traverser le terrain et donc mettra de temps. Par exemple, si on considère un robot à chenilles, cette valeur pourrait représenter la pente du terrain. Voici donc quelques échelles de valeurs entières que l'on pourra utiliser :

- 0 – 255 : « difficulté » du terrain
- 1000 : il y a un incendie
- 2000 : on ne peut pas passer sur cette case

Bien sûr, d'autres solutions sont plus judicieuses ; celle ci vous est donnée à titre d'exemple mais vous êtes libres de choisir d'autres représentations.

1.3 Les robots

Les robots sont des « pompiers élémentaires » qui peuvent se déplacer, éteindre un feu et qui connaissent leur position à tout instant. Ils sont également capables de calculer le plus court chemin qu'ils peuvent trouver jusqu'à un point de la carte (en utilisant les valeurs des cases de la carte). Dès que le robot est sur une case adjacente à un incendie, il va l'éteindre et cela va prendre un certain temps.

Pour pouvoir déterminer quel est le plus court chemin d'un robot aux différents incendies sur la carte, on peut utiliser un algorithme de *pathfinding*. Ces algorithmes permettent de trouver le plus court chemin dans un graphe. De nombreux exemples se trouvent dans [1], et vous pourrez trouver sur Internet de nombreuses applications et exemples (algorithme de Djikstra, algorithme A*, …).
Plusieurs types de robots pourront être considérés : robots à chenilles, robots à pattes, etc. Les robots à chenilles pourraient par exemple être incapables de traverser un terrain de valeur supérieure à 100.

1.4 Le manager

Pour pouvoir affecter aux robots les tâches qu'ils auront à effectuer (i.e. quels incendies ils devront aller éteindre), nous considèrerons qu'un *manager* de tâches centralise l'aspect décisionnel (ce manager peut être embarqué dans un des robots pompier ou être dans un central communiquant avec les robots). Vous vous appuierez sur le protocole *Contract Net* [2]. Dans ce protocole, un manager envoie une demande à plusieurs robots. Chaque robot doit répondre dans un certain temps: soit il refuse la demande, soit il envoie une proposition.
Le manager sélectionne ensuite les différentes tâches à affecter aux robots. Un exemple de déroulement du protocole :

1. le manager propose aux robots les incendies à éteindre ;
2. les robots occupés à éteindre un incendie refusent la proposition. Les autres robots calculent leur plus court chemin et renvoient cette valeur au manager ;
3. pour chaque incendie, le manager sélectionne le robot le plus proche pour aller l'éteindre. En cas d'« égalité », une sélection arbitraire pourra être utilisée 1 ;
4. s'il reste des incendies non affectés, le manager attend un certain laps de temps et re-propose les incendies restants.

1.5 Le simulateur

L'objectif de ce projet est de fournir un simulateur permettant de visualiser la carte, les incendies et le déplacement des robots. Un exemple de scénario est :

1. lire un fichier décrivant une carte sous forme d'une image en niveaux de gris par exemple (le niveau de gris représentant la difficulté du terrain) ;
2. positionner les robots et les incendies sur la carte ;
3. lancer une simulation ;
4. visualiser le déplacement des robots sur la carte.

Ce logiciel devra être le plus extensible possible. Par exemple, on devra facilement pouvoir remplacer un algorithme de *pathfinding* par un autre. Vous pouvez également rajouter des extensions. On peut par exemple s'intéresser à :

- la dynamique de l'incendie durant la simulation (propagation de l'incendie due au vent)
- l'analyse de la topologie de l'incendie pour optimiser les choix du manager
- un interfaçage à Google Maps via KML ou autre API pour tracer les déplacements des robots

La simulation pourra être stoppée, réinitialisée ou être accélérée. Vous pourrez vous appuyer sur ce qui existe en simulateurs d'événements discrets. Un exemple est donné ici [3].

Le simulateur devra également permettre d'ajouter de nouveaux robots et de définir leur profil (nom, icône, vitesse, endroits non accessibles, etc.).

Il est également demandé de prévoir un système permettant de conserver dans un fichier les actions des robots pour une mission, le temps total de la mission, les distances parcourues (et tout ce qui vous semblera pertinent) afin de comparer les algorithmes et les choix de robots.

2 Contraintes sur Le processus de développement

2.1 Vue globale

Ce projet sera mené en appliquant un procédé de développement itératif inspiré du Processus Unifié. Pour chaque phase vous devrez déterminer les artefacts que vous jugerez nécessaire et les tenir à jour. Il vous est proposé de suivre les quatre phases en respectant le nombre d'itérations suivant :

- phase de lancement : une itération ;
- phase d'élaboration : au moins deux itérations ;
- phase de construction : autant d'itérations que de scénarios à implémenter
- phase de transition : une seule itération

Le projet sera mené par groupes de 4 ou 5 étudiants. Le rôle de chaque membre de l'équipe sera établi lors de la phase de planification. Il s'agit d'appliquer en fait une démarche de type "client fournisseur", l'équipe d'enseignants jouant le rôle du client. Une définition des rôles des membres du groupe devra être proposée (un étudiant pouvant jouer plusieurs rôles).

On rappelle ci-dessous quelques éléments importants du processus et l'on impose diverses contraintes techniques.

2.2 Gestion de projet

Un des objectifs de cette activité est de décrire les phases du projet, les rôles et les tâches des membres de l'équipe de développement. Cette description pourra être faite selon un formalisme vu en cours de Conduite de Projet (WBS, PBS) à l'aide de *MSProject* par exemple ou en utilisant les tickets de *Redmine* (qui permet ensuite de générer automatiquement les diagrammes de Gantt).
Pour la modélisation, la génération de code, l'implémentation, et la gestion de configuration vous pourrez utiliser RSA (Rational Software Architect), mais le choix est libre.

2.3 Gestion des exigences

Vous devrez intégrer les exigences présentées ci-dessus, mais également proposer tout type de changement qui vous semble pertinent (et le justifier). Le sujet décrit ici est une base sur laquelle vous devez ajouter toute exigence nécessaire à son bon fonctionnement.

2.4 Analyse et conception

Une analyse et conception par objet sera menée selon la notation UML. On privilégiera l'extensibilité de la solution (ajout d'une icône, d'une commande, d'un menu...) en exploitant judicieusement la généricité, l'héritage et le polymorphisme. On étudiera aussi l'aspect dynamique en exhibant quelques scénarii représentatifs, les diagrammes d'interaction associés, et en décrivant les diagrammes d'états-transitions des principaux objets manipulés.
La conception sera établie dans le but d'être réutilisable ! Elle devra donc être notamment structurée, concise et documentée. Elle réutilisera obligatoirement des patrons de conception vus dans le cours « design patterns » ou autres.

2.5 Implémentation/test

Le codage sera effectué en **Java**, et la bibliothèque graphique **SWING**. Vous devrez justifiez de tests unitaires (en utilisant JUnit par exemple) et d'intégration (cas de tests documentés).

2.6 Gestion de la configuration et des changements

Il vous est demandé d'utiliser l'environnement ***Redmine*****,** notamment **pour gérer votre *repository* SVN, créer votre page web (sur le wiki), et utiliser les « tickets » lors du développement.** Vous devrez désigner une personne responsable de la gestion des changements. Cette personne sera également responsable du site web du projet (voir le *template* en annexe B) à mettre dans le wiki de Redmine. Vous devrez obligatoirement maintenir à jour ce site car l'évaluation sera faite à partir des documents accessibles sur ce site.

3 Planning et évaluation du projet

L'évaluation du projet se fera via les documents accessibles sur les sites web (les dates précises d'évaluation vous seront données ultérieurement) et une présentation orale finale.

3.1 Planning

Le planning qui vous est proposé tient compte des séances avec des enseignants et des contraintes du projet. Cependant vous êtes libre d'en modifier le déroulé, tant que vous

respectez les jalons d'évaluation (voir section suivante). Vous aurez des séances de projet à toutes les périodes (P2 à P5) et un oral en dernière période (P6).

3.2 Rapports

L'évaluation ne requiert pas de rapport proprement dit. Elle se fera sur la consultation des artefacts produits lors du développement, mais à des dates fixées en fin de chaque période. Les artefacts devront être accessibles en ligne.

3.3 Recette client et présentation de réalisation technique

Le projet fera l'objet d'une présentation orale effectuée par les membres de l'équipe. L'objectif est de présenter globalement et synthétiquement le projet (sous Powerpoint) d'un point de vue recette client et bilan technique (choix technologiques, déroulement du processus de développement, bilan général). D'une durée de 30 minutes (20 minutes + 10 minutes de questions), elle doit permettre à chaque membre de l'équipe de s'exprimer. La présentation doit être auto-suffisante c.a.d. ne pas présupposer que l'auditoire connaît particulièrement votre projet. La présentation doit obligatoirement inclure un test de recette présentant les fonctionnalités du logiciel, les limites et les améliorations possibles.

3.4 Notation

Vous serez notés à la fin de chaque période, et lors de la soutenance orale. Une note globale sur l'application de RUP sera également donnée lors de ce projet.

4 Bibliographie

[1] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. Introduction à l'algorithmique. Dunod, 2nd edition, 2004.

[2] FIPA Contract Net Interaction Protocol specification, http://www.fipa.org/specs/fipa00029/SC00029H.html, consulté le 23/07/2010

[3] SSim - A Simple Discrete-Event Simulation Library, http://www.inf.usi.ch/carzaniga/ssim/index.html , consulté le 23/07/2010

ANNEXE A : Template du wiki de redmine

Nom du projet

[Avant-propos]

[mettre un logo ici]	**Equipe projet** Bob Reiener, bob@youpi.com Brigitte Clin, clin@cict.fr	**Liens** : Télécharger le logiciel *[mettre le nom du logiciel ici]* Documentation API javadoc

Rôles RUP *(plusieurs personnes jouent plusieurs rôles !)*	
Chef de projet	Bob, Brian
Architecte	Bruno

Documents *(les documents sont des liens vers des fichiers doc, html, pdf, ou images):*
Ces listes sont données à titre indicatif, vous aurez à les adapter en fonction de votre projet.

Documents créés lors de la phase de LANCEMENT			
Nom	Version	Dernière mise à jour	Auteur
Vision	1.2	12/09/2002	bob
Liste des risque			
Glossaire			
Planning des phases			
Modèle des C.U.			
Spécifications supplémentaires			
Ordres de priorités des C.U.			

Cas d'utilisation			
Nom	Version	Dernière mise à jour	Auteur
Diagramme des C.U.			
UC1 : Ouvrir un nouveau fichier			
UC2 : Sauvegarder la base de o			
Diag. Seq. UC1 ::scénario blabla			
Diag. Seq. UC2 ::scénario blabla			
Etc…			

Documents créés lors de la phase d'ELABORATION			
Nom	Version	Dernière mise à jour	Auteur
Plan d'itération1			
Architecture logicielle			
Diagramme de classes package1		..	..
Diagramme de classes package2		..	…
Diagramme de classes package n			
Planning Itération2			

Documents créés lors de la phase de CONSTRUCTION			
Nom	Version	Dernière mise à jour	Auteur
…	..	..	…

Documents créés lors de la phase de TRANSITION			
Nom	Version	Dernière mise à jour	Auteur
Tests et résultats			
Bilan Chef de projet			
Guide d'installation			
Manuel Utilisateur			