TME 2 – Composants

Frédéric Peschanski

22 janvier 2016

Dans ce TME, nous reprenons la démarche d'ingénierie logicielle qui consiste :

1. Á récupérer le code source d'un projet pré-existant et à en extraire l'architecture (*rétro-ingénierie*).
2. A concevoir et implanter des modifications/extensions

Nous illustrons en particulier que l'emploi du *design pattern Require/Provide* permet de faciliter la première étape de la démarche. En revanche, pour la deuxième étape on reste dans un cadre de programmation classique.

Le projet étudié cette semaine est une version simplifiée de *Maleva*[1] un modèle de simulation multi-agents à base de Composants. Les comportements des agents, ici de simples proies et prédateurs, simulés sont décrits par emboîtement de composants.

Exercice 1 : rétro-ingénierie

L'objectif de cet exercice est de parcourir les sources du projet *Maleva* pour en déduire une description de l'architecture. Puisque cette dernière est orientée composant, la description attendue correspond :

— à un **diagramme de composant** (en UMlet) pour chaque composant «simple» (non-composite).
— à un **diagramme de composite** pour chaque composant composite
— à un **diagramme de composition** pour le programme principal : MalevaTest2.

Remarque : attention à l'héritage, les composants sont des *classes concrètes*.

Exercice 2 :Déplacement aléatoire

L'objectif est de compléter le code source du composant de comportement maleva.agent.RandomMouvement. Le comportement attendu est un déplacement qui apparaît comme aléatoire dans la simulation. Pour cela, la méthode step() doit effectuer une déplacement élémentaire selon un angle calculé à partir de l'angle courant (outer.getAngle()) auquel on ajoute ou on retranche une valeur prise «au hasard» entre -angle_step et +angle_step, la variable angle_step étant un paramètre du comportement.

1. cf. `http://www-poleia.lip6.fr/~briot/cv/composants-agent-tsi01.ps.gz`

Exercice 3 : Agent trouillard

L'objectif est de compléter le code source du composant composite maleva.proiepred.Trouillard. Ce dernier correspond à un agent dont le comportement est :

— de fuir les proies détectées (donc de se déplacer selon un angle contraire à ces dernières),
— ou, si aucun proie n'est détectée, d'adopter le comportement de déplacement aléatoire de l'exercice précédent.

Exercice 4 : Simulateur «réaliste»

Le simulateur sera jugé «correct» si les affichages générés par la simulation (par ant run) semblent correspond à ce que l'on en attend. L'objectif de cet exercice est donc de comprendre le fonctionnement du simulateur pour éventuellement le corriger afin d'obtenir une simulation plus «réaliste» en terme de visualisation :

— les *méchants* sont bien méchants
— les *trouillards* sont vraiment trouillards.

Cette question illustre que connaître l'architecture du simulateur aide mais ne suffit pas à en comprendre le fonctionnement. La difficulté provient essentiellement des interactions (relativement) complexes entre composants.