Université de Montpellier
Faculté des Sciences

Licence 3 Informatique

HLIN613 Stage effectué dans le cadre du Cursus Master en Ingénierie

Création de tutoriels présentant les fonctionnalités de la plateforme MaDKit

---

Rapport de stage

LIRMM, équipe SMILE
du 23 mai 2018 au 20 juillet 2018

Étudiante :
Amandine Paillard

Tuteur de stage :
Fabien Michel



Remerciements

Avant d'entamer ce rapport, je tiens à remercier certaines personnes. Sans eux ce stage n'aurait peut être pas eu lieu ou ne se serait pas aussi bien passé.

Parmi les personnes qui ont rendu ce stage possible je voudrai remercier Anne-Élisabeth Baert pour m'avoir permis de faire un stage dans le domaine de la recherche. Merci également à Hinde Bouziane pour m'avoir conseillée et accompagnée dans mes recherches de stage tout au long de l'année.

Au sein de l'équipe dans laquelle j'ai travaillé, je suis bien évidemment reconnaissante envers Fabien Michel pour avoir accepté d'être mon tuteur et d'avoir consacré de son temps afin que je puisse découvrir les systèmes multi-agents et prendre en main les outils et technologies que j'allais être amenée à utiliser pendant mon stage. Je remercie également Anne Toulet, Nadira Boudjani, Abdelkader Gouaich et Elcio Abrahão pour leur accueil chaleureux et bienveillant au sein de l'équipe SMILE. Merci aussi à Iago pour la bonne humeur et la joie de vivre que tu apportes chaque jour dans le bureau.

De même, je tiens à remercier toutes les personnes que j'ai pu croiser au LIRMM et en particulier Laurie et Gladys pour leur accueil, leur disponibilité ainsi que la gentillesse dont elles font preuve tous les jours.

Enfin, pour des remerciements un peu plus personnels, je pense à Julie et à Thomas pour avoir été là tous les jours et pour nous être soutenus mutuellement dans nos stages respectifs. Je remercie également mes amis qui m'ont fait découvrir le LIRMM avant même que je songe à y faire mon stage, merci à Maëlle, Rémi, Mattéo et Florent. Enfin parmi les personnes qui n'ont pas encore été citées je souhaiterai remercier celles qui m'ont aidée dans la rédaction de ce rapport de part leurs corrections et leurs remarques -pas toujours pertinentes- : merci Félix, Christiane, Jean-Christophe et Camille.

À tous, je vous remercie.

Table des abréviations

CMI : Cursus Master en Ingénierie

LIRMM : Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier

SMILE : Système Multi-agents, Interaction, Langage, Évolution

API : interface de programmation (Application Programming Interface)

MaDKit : kit de développement multi-agents (Multi-agent Development Kit)

CNRS : Centre National de la Recherche Scientifique

MASQ : systèmes multi-agents basés sur les quadrants (Multi-Agent System based on Quadrants)

AGR : Agent / Groupe / Rôle

CGR : Communauté / Groupe / Rôle

Table des matières

Chapitre 1

Introduction

1.1 Contexte

Ce stage s'inscrit dans le cadre de ma troisième année de licence informatique en Cursus Master en Ingénierie à la Faculté des Sciences de l'Université de Montpellier. Cette formation préconise que les étudiants aient réalisé une cinquantaine de semaines de stage avant la fin de leur master et ceci non seulement dans le but de leur faire découvrir le monde du travail et de la recherche mais aussi de développer leur culture professionnelle. Pour mon stage j'ai donc décidé de découvrir le monde de la recherche qui m'était encore inconnu. Ainsi j'y ai vu l'occasion, peut être unique de m'approcher de ce milieu car je ne souhaitais pas m'y orienter pour mon futur professionnel. Ce stage a donc été l'opportunité de découvrir un autre milieu que celui de l'entreprise - milieu que je connaissais déjà - et l'occasion de réaliser cette expérience.

Concernant le choix du laboratoire de recherche dans lequel faire mon stage, le LIRMM m'est vite apparu comme la réponse évidente. Nombreux de mes camarades de promotion avaient déjà réalisé un stage dans ce laboratoire et la majeure partie de notre corps enseignant y travaille. Je me suis donc tournée vers cet organisme et je me suis renseignée sur le travail qu'effectuaient les différentes équipes de recherche qui le compose. Lors de ces recherches, l'équipe qui a le plus retenu mon attention est l'équipe SMILE. Cette équipe étudie les systèmes multi-agents et leur application pour divers domaines comme entre autre, les systèmes distribués ou bien les jeux vidéos sérieux ou ludiques. Dans le but de pouvoir travailler avec des systèmes multi-agents, des chercheurs de l'équipe ont créé une API (interface de programmation) permettant de modéliser ces systèmes.

Ainsi, l'enjeu de mon stage est d'améliorer la documentation existante par l'ajout de plusieurs tutoriels permettant de présenter le fonctionnement de diverses fonctionnalités de cette API.

Réaliser mon stage au sein de l'équipe SMILE du LIRMM a été motivé par la découverte du monde de la recherche, mais aussi la possibilité d'en apprendre plus sur les systèmes multi-agents, notion que je n'avais encore jamais étudiée et qui me semblait être très intéressante.

Dans ce rapport, après avoir présenté le LIRMM, le travail de l'équipe SMILE et le paradigme multi-agents, je détaillerai la mission qui m'a été confiée. Je préciserai également les divers outils avec lesquels j'ai pu travailler, et concluerai sur l'expérience que ce stage m'a apporté pendant ces deux mois.

1.2 Organisme d'accueil : le Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier

1.2.1 Présentation

Le LIRMM est un laboratoire de recherche issu à la fois de l'Université de Montpellier et du CNRS. Créé en 1992 et situé sur le campus Saint-Priest aux bâtiments quatre et cinq, il regroupe les départements informatique, robotique et microélectronique et est, une unité mixte de recherche. Depuis juillet 2015, il est dirigé par Philippe Poignet, assisté par Christophe Paul et Michel Renovell.

En février 2016 le laboratoire comptait pas moins de quatre cent quarante-quatre employés dont cent quatorze enseignants-chercheurs, quarante-neuf chercheurs et cent soixante-dix doctorants. Parmi ces employés du LIRMM il ne faut pas oublier les personnes s'occupant de la bonne gestion du laboratoire.

1.2.2 Activité

Les principales activités du LIRMM sont la parution internationale de publications, la création de matériels et logiciels mais aussi la création d'entreprises ou de partenariats industriels.

En plus des activités précédentes, les chercheurs du LIRMM déposent également de nombreux brevets et autres logiciels à l'Agence de Protection des Programmes. Enfin, les membres du laboratoire ont également l'occasion de travailler sur des projets de l'Agence Nationale de Recherche.

1.2.3 Organisation interne

Comme il a été vu un peu plus tôt, le LIRMM est composé de trois départements. Chacun de ces départements est divisé en équipe de recherche de la façon suivante.

Département Informatique : traite des thèmes allant des mathématiques au génie logiciel en passant par la bio-informatique.

- **Équipe ADVANSE** : ADVanced Analytics for data SciencE.
- **Équipe ALGCO** : Algorithmes, Graphes et Combinatoire.
- **Équipe COCONUT** : Agents, Apprentissage, Contraintes.
- **Équipe DALI** : Digits, architectures et logiciels informatiques.
- **Équipe ECO** : Exact Computing.
- **Équipe ESCAPE** : Systèmes complexes, automates et pavages.
- **Équipe FADO** : Fuzziness, Alignments, Data and Ontologies.
- **Équipe GraphiK** : Graphs for Inferences on Knowledge.
- **Équipe ICAR** : Image et Interaction.
- **Équipe MAB** : Méthodes et algorithmes pour la bio informatique.
- **Équipe MAORE** : Méthodes Algorithmes pour l'Ordonnancement et les Réseaux.
- **Équipe MAREL** : Models And Reuse Engineering, Languages.
- **Équipe SMILE** : Système Multi-agent, Interaction, Langage, Evolution.

Équipe TEXTE : Exploration et exploitation de données textuelles.

Équipe ZENITH : Gestion de données scientifiques.

Département Robotique : effectue des recherches sur des thèmes allant de la conception aux tests des systèmes intégrés et microsystèmes.

Équipe CAMIN : Control of Artificial Movement and Intuitive Neuroprosthesis.

Équipe DEXTER : Conception et commande de robots pour la manipulation.

Équipe EXPLORE : Robotique mobile pour l'exploration de l'environnement.

Équipe ICAR : Image et Interaction.

Équipe IDH : Interactive Digital Humans.

Département Microélectronique : étudie divers thèmes tels que la navigation, le traitement des images ou bien la supervision de systèmes dynamiques complexes.

Équipe ADAC : ADAptive Computing.

Équipe SmartIES : Smart Integrated Electronic Systems.

Équipe TEST : Test and dEpendability of microelectronic integrated SysTems.

Mon stage se déroule au sein de l'équipe SMILE du département Informatique. Nous allons voir plus en détails le fonctionnement et le travail de celle-ci.

1.3 Équipe SMILE : Système Multi-agent, Interaction, Langage, Évolution

1.3.1 Présentation

L'équipe SMILE étudie les systèmes multi-agents et leurs concepts, mais les chercheurs étudient également les applications que ces systèmes peuvent avoir. Ces dernières peuvent être catégorisées dans plusieurs domaines comme il est précisé dans la partie suivante.

1.3.2 Domaines de recherche

Tous les domaines de recherche de l'équipe SMILE concernent ou se basent sur le paradigme multi-agents.

Modèles organisationnels : les chercheurs développent de nouveaux modèles organisationnels relatifs aux systèmes multi-agents. Parmi les modèles étudiés, on peut citer AGR (Agents / Groupe / Rôle) ou MASQ (systèmes multi-agents basés sur les quadrants).

Simulation : un des enjeux de la simulation multi-agents -simulation de plus en plus utilisée de nos jours- est d'améliorer la façon dont les ressources de calcul sont utilisées afin de pouvoir simuler des populations toujours plus grandes dans des environnements virtuels plus grands eux aussi. On appelle ceci le calcul haute performance.

Web Service : ici on étudie le web comme un système composé d'une multitude d'agents et on en déduit des propriétés pour des futures architectures ou applications.

Jeux sérieux : ce thème implique l'application du paradigme multi-agents à l'implémentation de jeux sérieux. Le travail de ce domaine se compose non seulement de recherches mais également du développement de telles applications utilisées dans le domaine de la santé.

Intégration sémantique de données biomédicales et bio-ontologiques : création d'index sémantique, et d'autres outils se basant sur des connaissances médicales et ontologiques. La finalité de ces travaux est de permettre de nouvelles découvertes scientifiques grâce à une nouvelle organisation des données.

Maintenant que nous avons vu quels sont les domaines de travail de l'équipe SMILE, nous allons entrer dans le détail de la mission.

Chapitre 2

Présentation de la mission

2.1 Enjeu du stage

Les chercheurs de l'équipe SMILE développent depuis 1997 une API leur permettant de modéliser des systèmes multi-agents. Nous consacrerons une partie de ce rapport à la présentation de ces systèmes. MaDKit (kit de développement multi-agents) est réalisé en Java et se veut adaptable à n'importe quel projet. Afin qu'elle soit la plus facile à prendre en main il est capital que MaDKit soit suffisamment documenté. C'est cette idée qui va motiver la réalisation de ce stage.

À ce jour, MaDKit est documenté de diverses façons : son code est accessible sur GitHub, sa Javadoc ainsi que des tutoriels et autres démonstrations sont également disponibles sur son site web (http://www.madkit.net/madkit/). De plus, divers rapports et articles la concernant sont mis à disposition. Cependant si la Javadoc de MaDKit est assez complète, nous sommes loin d'avoir chacune de ses caractéristiques expliquées et illustrées. De ce fait enrichir cette documentation est toujours nécessaire.

L'intérêt d'un tutoriel est de pouvoir démontrer par des exemples de différents niveaux de difficulté une notion ou une fonctionnalité propre à MaDKit. Les tutoriels réalisés dans le cadre du stage devront avoir un niveau de difficulté croissant et être accessible au plus grand nombre. C'est pour répondre à cet objectif qu'ils seront en anglais et aussi documentés que possible sans que nous tombions dans des explications redondantes.

2.2 Présentation des outils et techniques employés

Ce stage m'a permis d'aborder des notions que je n'avais encore jamais étudiées.

2.2.1 Introduction aux systèmes multi-agents

Avant de parler de systèmes multi-agents, il convient de définir ce qu'est un agent. Un agent est une entité autonome et indépendante, qui peut être physique ou virtuelle et qui ne dispose que d'une vision partielle de son environnement. Par exemple, un individu, un robot ou même un programme peuvent être un agent. En fonction de ses *motivations*, un agent peut privilégier une action par rapport à une autre. Ces motivations peuvent être soit, choisies par une personne extérieure (le programmeur du système) soit, choisies par l'agent lui-même dans le cas où le-dit objectif s'accorde avec un objectif de survie ou de satisfaction de l'agent. Une action réalisée par un agent peut prendre différentes formes (interaction, collaboration avec d'autres agents, etc). Ce que nous appelons système multi-agents est tout simplement le cadre dans lequel plusieurs agents

vont évoluer. Selon les actions des agents constituant le système, ce dernier peut être modifié et, ainsi impacter à son tour les agents et leurs interactions. Notons toutefois que le système contient également des objets qui peuvent être différents des agents (avec lesquels les agents vont interagir), et qu'il est régi par un ensemble de relations, d'opérations et d'opérateurs.

On remarque que cette notion implique une répartition de l'intelligence. Dans son livre, (FERBER, 1995) Jacques FERBER explique l'intérêt de distribuer l'intelligence pour différents raisons. Par exemple, **les problèmes sont physiquement et fonctionnellement très distribués et hétérogènes** ; **les réseaux imposent une vision distribuée** ;**la complexité des problèmes imposent une vision locale** ou encore **les systèmes doivent pouvoir s'adapter à des modifications de structures ou d'environnement** et **le génie logiciel va dans le sens d'une conception en termes d'unités autonomes en interactions**. Toutes ces raisons justifient que nous nous tournions vers une répartition de l'intelligence.

Les systèmes multi-agents sont de nos jours utilisés dans divers domaines tels que la recherche, l'industrie que ce soit dans les jeux vidéos ou l'animation ou encore la finance pour la simulation de marchés.

2.2.2 MaDKit, une API pour modéliser des systèmes multi-agents

Pour rappel, MaDKit est une librairie en Java permettant de modéliser et de simuler des systèmes multi-agents. Contrairement à d'autres API du genre, MaDKit est basée sur une approche organisationnelle : l'AGR (Agent / Groupe / Rôle). La notion capitale de MaDKit est qu'elle ne propose pas un modèle organisationnel spécifique et se veut le plus hétérogène possible.

Existant depuis plus de vingt ans, MaDKit possède plusieurs versions. Depuis 2010 MaDKit en est à sa version 5.2. Nous pouvons remarquer un changement majeur entre cette version et les versions 1 à 4. Cette restructuration -plus que simple changement- est dû entre autre au fait que MaDKit v.4 fonctionnait plus comme un plug-in que comme une librairie ; ceci a été réglé avec la version 5. De plus, certaines parties de code constituant le noyau de MaDKit v.4 étaient trop vieilles et la librairie dépendait elle-même de plusieurs autres librairies. Cependant, la version 4 est toujours utilisée mais n'est plus maintenue.

Au total, on peut estimer au minimum le nombre de téléchargements de MaDKit entre 2011 et 2015 à un peu plus de 10 000. Ces téléchargements concernent aussi bien la version 4 que la version 5. Il reste néanmoins difficile d'avoir une idée plus précise du nombre d'utilisateurs réguliers de MaDKit.

Plus d'informations concernant MaDKit sont disponibles dans (MICHEL, 2000), (HANACHI et SIBERTIN-BLANC, 2000), (MICHEL, 2010) ou encore (MICHEL, 2015). Des informations supplémentaires peuvent également être consultées sur le site de MaDKit http://www.madkit.net/madkit/.

2.2.3 Outils de travail employés

Ce stage a également été l'occasion d'approfondir ma connaissance voire de découvrir de nouveaux outils. L'apprentissage de ces derniers s'est déroulé en deux parties : d'abord leur découverte ou rappel en début de stage, mais aussi un apprentissage sur le tas qui a parfois donné lieu à des situations assez cocasses, comme il est évoqué dans la partie de ce rapport traitant des difficultés rencontrées dans ce stage. Un exemple d'utilisation de ces outils est disponible en annexe 4.3.

Git

J'étais déjà familière avec Git (logiciel de gestion de version) avant ce stage. Cependant je suis encore loin de maîtriser toutes les possibilités qu'offrent cet outil. Cette expérience m'a permis d'en voir une utilisation plus professionnelle et plus complète.

Ant

Si j'avais déjà utilisé Maven (outil de gestion et d'automatisation pour les applications Java) par le passé, ce stage a marqué ma première rencontre avec Ant. Comme j'avais déjà réalisé des Makefiles pour mes projets en C et C++ et j'avais utilisé le XML par le passé, l'apprentissage de cet outil ne m'a pas posé plus de problème que cela. Par ailleurs un exemple de fichier de configuration est disponible sur le dépôt Git du projet ainsi que sur le site web. On peut considérer que mon utilisation de cet outil s'est limitée à l'ajout de cibles couvrant les nouvelles parties du projet créées par mes soins.

Java Web Start

Avant ce stage, je n'étais absolument pas au fait de cet outil, j'ai donc découvert ce dernier pendant mon séjour au LIRMM. Java Web Start est particulièrement utile et intéressant dans le cadre des tutoriels car il permet de lancer un exécutable Java à partir d'une page Web. Les utilisateurs n'ont donc qu'à appuyer sur un bouton pour voir l'exécution du code qu'ils ont sous les yeux.

Chapitre 3

Travail effectué

3.1 Organisation

Au cours de ces deux mois au LIRMM une certaine routine de travail a pris place. Après une période de découverte de la programmation orientée agents, de la plate-forme MaDKit ainsi que des différents outils, j'ai pu me lancer dans la réalisation des tutoriels.

3.1.1 Déroulement de la phase de prise en main

Cette phase a duré un peu plus d'une semaine. Personnellement je trouve que cette période a été la plus dense de mon stage car il m'a fallu voir (revoir parfois) plusieurs choses différentes dans un laps de temps le plus court possible pour commencer à vraiment travailler sur ma mission sans perdre trop de temps.

J'ai donc pris un peu plus d'une semaine pour me familiariser avec la programmation multi-agents. Dans cet objectif, les publications de (Michel, 2000), (Hanachi et Sibertin-Blanc, 2000) et (Michel, 2010) m'ont été particulièrement utiles et instructives.

De la même manière, je me suis intéressée aux articles de (Gutknecht, Ferber et Michel, 2000) et (Pradeilles et Hisler, 2010) afin d'en savoir plus sur MaDKit. Je me suis également servie des renseignements disponibles en ligne sur le site de MaDKit (Group, 2018) que ce soit la documentation, la Javadoc ou divers tutoriels et démonstrations disponibles. Concernant ces deux derniers points, j'ai pris un soin particulier à comprendre à la fois la façon dont ils étaient construits (ce qu'ils voulaient démontrer et la granularité de difficulté qui les différenciaient) mais ils m'ont également été utiles pour apprendre, moi aussi, à me servir de MaDKit.

Ne pas avoir été familière avec l'API ou même avec la programmation multi-agents m'a permis d'avoir un regard neuf par rapport à la documentation déjà existante et de repérer les points où elle n'était pas suffisamment claire pour un néophyte. Ceci nous a permis, mon maître de stage et moi d'apporter plus de précision dans la documentation existante mais surtout de repérer quelles notions requéraient un tutoriel afin d'être mieux expliquées. De manière plus générale, le but de cette phase préliminaire était d'obtenir une vision globale afin de me donner les notions nécessaires au bon déroulement de mon stage, tout en étant suffisamment technique pour s'intéresser à certains détails de l'API.

Mon travail consistant à enrichir la documentation et à faciliter la prise en main d'une API permettant de modéliser les systèmes multi-agents, il s'est avéré judicieux de comparer les informations mises à disposition pour MaDKit avec celles d'autres API ayant le même objectif. Sur les conseils de mon maître de stage je me suis donc renseignée sur les API : Jade, Jason, JaCaMo et SARL. Leur description et comparaison est disponible en annexe 4.3.

C'est également pendant cette période que j'ai étudié les outils de travail cités un peu plus tôt dans ce rapport.

3.1.2 Déroulement de la réalisation des tutoriels

Dans cette partie nous pouvons distinguer deux niveaux d'organisation distincts : celui de l'agencement entre les tutoriels et celui propre à la façon de travailler qui a été commune aux différents tutoriels. Commençons par l'organisation la plus générale avant d'entrer dans le détail.

Coordination des tutoriels

Tout au long de mon stage j'ai pu réaliser trois tutoriels : sur les codes de retour de MaDKit, ses propriétés et enfin sur ses options. Entre le moment où un thème de tutoriel se dessine et la fin de la réalisation de celui-ci, il est arrivé que le sujet principal du tutoriel change légèrement ou que sa structure soit grandement modifiée. J'ai pu me rendre compte de ceci pour chacun des travaux que j'ai réalisé.

Après avoir étudié différentes API de modélisation de systèmes multi-agents en plus de MaDKit, j'ai pu réaliser que les autres API, contrairement à la nôtre, présentaient assez souvent la façon dont les exceptions sont gérées. Mon idée de base était donc de faire un tutoriel sur la gestion de ces dernières dans MaDKit. Au fur et à mesure de l'avancement du tutoriel, il s'est avéré qu'étant faite en Java la plupart des exceptions que l'on rencontre dans MaDKit sont identiques aux exceptions Java ce qui n'est pas des plus pertinents à illustrer. En revanche, il existe une catégorie d'exceptions qui elles, sont propres à MaDKit et qui sont mises en relief par des codes retour. Ainsi ce tutoriel s'est beaucoup plus étendu sur la présentation des différents codes retour de MaDKit que sur les exceptions en elles-mêmes.

Si j'ai trouvé l'idée du premier tutoriel en regardant la documentation existante pour d'autres API, le sujet du second m'est apparu en me servant de la documentation de MaDKit. La notion de propriétés et d'options de MaDKit me semblait faiblement documentée et j'ai voulu en faire un tutoriel sans -à ce moment- saisir toutes les nuances qu'offraient ce thème. Ce n'est qu'en commençant sa réalisation que la nécessité de séparer les propriétés et les options de MaDKit s'est présentée. De ce fait les deux derniers tutoriels ont été créés conjointement.

Flux de travail

Tout le code de MaDKit ou des différents tutoriels et démonstrations étant libre, il est disponible sur GitHub : https://github.com/fmichel/MaDKit-tutorials. Cela a donc été l'outil de travail privilégié pour la réalisation des tutoriels. Cependant, son utilisation nécessite tout de même une certaine organisation :

- ***Fork* du projet originel**. Un *fork* est une copie d'un projet sur un dépot d'un contributeur afin que ce dernier ait sa propre version du projet et puisse travailler dessus sans conflit. Voici le lien de ma copie du projet : (https://github.com/amapai/MaDKit-tutorials).
- Travail sur un tutoriel dans une **branche** à part de mon dépôt. On divise ainsi un dépôt en différentes branches de manière à faire une branche par fonctionnalité.
- ***Commit*** au besoin. Un *commit* permet d'enregistrer le travail effectué sur un dépôt local. On distingue le dépôt local (hors ligne, sur l'ordinateur du programmeur) du dépôt global (ou dépôt distant) qui est en ligne, sur les serveurs de GitHub.
- ***Push*** sur le dépôt distant quand une fonctionnalité est complète. Cela permet d'envoyer les changements «*commités*» sur le dépôt distant.
- (Tant que le tutoriel n'est pas fini, on recommence les deux étapes précédentes tout en validant les avancements majeurs avec mon maître de stage.)
- Revue générale du tutoriel avec Fabien Michel.
- Ouverture d'une ***pull request*** une fois le tutoriel fini. Une *pull request* permet de fusionner (en réglant les éventuels conflits au préalable) la version d'un projet d'un collaborateur au projet originel.
- Vérification de la *pull request* puis intégration au projet initial si elle est correcte.
- Les fonctionnalités sont intégrées au projet initial, il faut mettre à jour le projet «*forké*» (ainsi que les différentes branches du dépôt du collaborateur).

3.2 Présentation du travail

3.2.1 Tutoriel sur les codes de retour dans MaDKit

Finalité

Ce tutoriel traite de la façon dont sont gérées les exceptions dans MaDKit. On distingue deux sortes d'erreur : celles liées à une mauvaise utilisation du langage Java et celles liées à une mauvaise utilisation de l'API. Les erreurs générées par une mauvaise utilisation de Java étant considérée comme connue des développeurs s'intéressant à notre API, nous allons accorder plus d'explications aux erreurs de l'API. Concernant ces dernières, nous allons principalement présenter une manière de sécuriser notre code de part les codes de retour. On peut considérer trois principales catégories parmi ces codes de retour : ceux concernant le lancement des agents, ceux concernant les CGR et enfin ceux concernant la communication.

On verra alors quels sont les scénarios où de telles erreurs peuvent surgir, et comment l'API nous permet d'y remédier et de sécuriser nos programmes. Nous montrerons d'abord comment l'API permet de ne pas bloquer toute une société si un de ces agents bogue, et comment sécuriser un maximum ces derniers avec les codes de retour.

Structure

Ce tutoriel suit l'architecture suivante :

Erreurs propres à Java : contient une classe générant une exception Java.

Erreurs propres à MaDKit : les classes de ce paquet illustre diverses mauvaises utilisations de MaDKit comme essayer de faire envoyer un message à un agent qui n'a pas encore été lancé par exemple.

Introduction aux codes de retour : ce pack ne contient qu'une seule classe. On y introduit le principe des codes de retour en testant si une méthode s'est bien déroulée (grâce notamment au code retour **SUCCESS**).

Codes de retour concernant le lancement des agents : présente notamment les codes de retour **NOT_YET_LAUNCHED**, **ALREADY_LAUNCHED**, **AGENT_CRASH**, **TIME_OUT** ou **ALREADY_KILLED**.

Codes de retour concernant le CGR : introduit les codes de retour **NOT_COMMUNITY**, **ALREADY_GROUP**, **NOT_GROUP**, **NOT_IN_GROUP**, **NOT_ROLE**, **ROLE_ALREADY_HANDLED** ou **ROLE_NOT_HANDLED**.

Codes de retour concernant la communication entre agents : ici, on voit les codes de retour **CANT_REPLY**, **NO_RECIPIENT_FOUND**, **INVALID_AGENT_ADDRESS**.

Utilisation concrète des codes de retour : l'exemple disponible dans ce paquet explique comment adapter son programme pour qu'il puisse traiter chaque type d'erreur.

utils : regroupe les classes utiles et communes à tout le tutoriel comme la classe représentant un agent type.

3.2.2 Tutoriel sur les propriétés dans MaDKit

Ce tutoriel est moins conséquent que les autres mais est tout aussi important. Il s'appuie sur la classe Property de Java qui est supposée connue pour les développeurs de MaDKit. Toutefois si ce n'est pas le cas, au début du tutoriel nous renvoyons les utilisateurs vers de la documentation officielle ainsi que vers un tutoriel pour s'assurer que ces notions ne leurs manquent pas.

Il a donc pour but -entre autre- de montrer à la communauté de MaDKit où sont stockées les propriétés par défaut, comment les faire apparaître, les modifier, en ajouter ou comment charger un fichier de propriétés XML (tout en étudiant l'apport de tels fichiers par rapport aux fichiers *properties*). Néanmoins ce tutoriel est capital pour la compréhension d'un autre guide : MaDKit options.

Ce tutoriel est relativement léger et ne comporte qu'un seul pack où sont illustrées les fonctionnalités décrites précédemment.

3.2.3 Tutoriel sur les options de MaDKit

Finalité

Ce tutoriel explique ce qu'est une option MaDKit. Il explique aussi que ces options sont réparties en trois grandes catégories : les options booléennes, sous forme de *Level* et les autres options.

Comme leur nom l'indique assez clairement, les options booléennes sont des fonctionnalités qui sont, soit activées, soit non activées. Les *options nivelées* elles, permettent de définir la précision avec laquelle une option est ou non activées. Elles se basent sur les *Levels* de Java et sont notamment utilisées pour définir un niveau d'affichage de messages : du moins précis au plus détaillé. Les autres options quant à elles prennent une valeur sous forme de chaîne de caractère.

Structure

Ce tutoriel suit l'architecture suivante :

BooleanOptions : ce paquet présente les options **console**, **cgrWarnings**, **debug**, **desktop**, **createLogFiles**, **noAgentConsoleLog**, **loadLocalDemos**, **network** et **autoConnectMadkitWebsite**.

LevelOptions : ici on introduit **madkitLogLevel**, **agentLogLevel**, **guiLogLevel**, **kernelLogLevel**, **networkLogLevel** et **warnignLogLevel**.

MadkitOptions : pour le dernier type d'options, on étudie **launchAgents**, **configFile**, **desktopFrameClass**, **agentFrame** et **logDirectory**.

utils : regroupe les classes utiles et communes à tout le tutoriel comme la classe représentant un agent type.

Chapitre 4

Bilans et perspectives

4.1 Présentation des résultats

Les tutoriels sont désormais en ligne et disponibles à cette adresse : http://www.madkit.net/madkit/tutorials/. Comme vous pouvez le constater sur les captures d'écran suivantes et notamment sur la figure 4.1, plusieurs tutoriels sont accessibles. Ainsi qu'il a été évoqué précédemment, j'ai pu réaliser ceux sur Exception, MaDKit options et MaDKit properties.

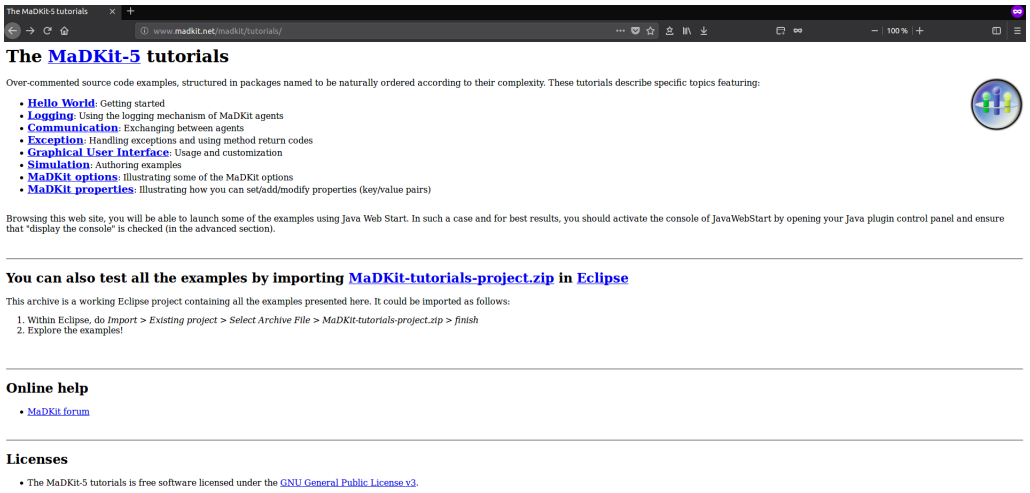

FIGURE 4.1 – **Menu des tutoriels**

Lorsqu'on choisit un tutoriel, on se retrouve sur une fenêtre comme celle disponible en figure 4.2. On dispose alors de toutes les classes, chacune illustrant en fait une particularité du sujet du tutoriel. Ces classes sont ordonnées au sein de leur pack ; les packs étant eux-mêmes ordonnés. De cette façon, on peut suivre le tutoriel avec une complexité croissante.

Sur la figure 4.2, nous pouvons également nous rendre compte de l'intérêt de Java Web Start. En appuyant sur le bouton *Launch*, le code que nous avons sous les yeux s'exécute, facilitant ainsi sa compréhension.

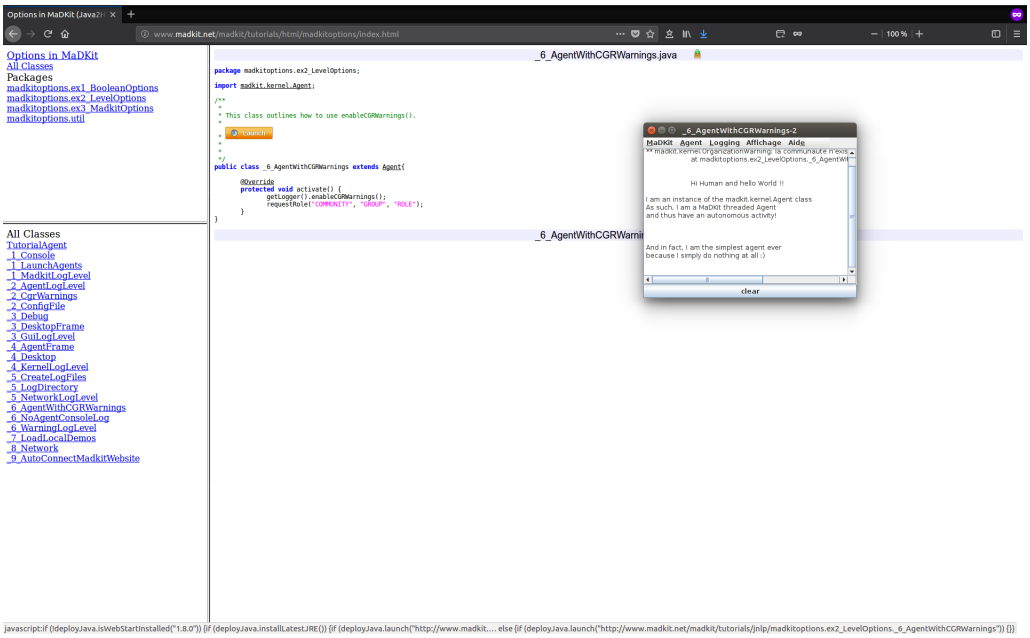

FIGURE 4.2 – **Interface type d'un tutoriel**

4.2 Retour sur les difficultés rencontrées

Ce stage m'a également permis de faire face à certains défis.

Le premier a été d'avoir choisi un stage dans un domaine de l'informatique que je n'avais pas encore étudié : la programmation multi-agents. Cette difficulté a largement été compensée avec le fait que la mission de mon stage était accessible. Ainsi j'ai donc pu pleinement profiter de cette occasion pour en apprendre plus sur cette notion avec un peu d'avance sur mes études. Par ailleurs, le fait d'étudier les systèmes multi-agents était clairement une des motivations qui m'ont poussée à choisir ce stage plutôt qu'un autre. De plus, découvrir les systèmes multi-agents en même temps que l'élaboration des tutoriels s'est avéré assez utile car cela m'a permis de conserver un regard neuf sur mon travail et m'a aidée dans l'élaboration des tutoriels.

Ce séjour au LIRMM m'a également permis d'étudier des outils que je n'avais encore jamais utilisé. Cela a mené à une période d'apprentissage où j'ai appris beaucoup de choses dans un temps qui se devait être le plus court possible. La découverte de ces outils couplée à celle de la programmation multi-agents m'a demandé d'assimiler beaucoup de choses à la fois mais avec l'usage, j'ai pu me familiariser avec ces technologies sans trop de problème.

Paradoxalement la plus grande difficulté que j'ai pu rencontrer dans mon stage n'est pas venue d'un outil avec lequel je n'avais jamais travaillé mais de Git. J'ai ainsi pu découvrir de façon un peu brutale la commande **git reset –hard**. Ceci m'a fait perdre un peu de temps dans l'élaboration des deux derniers tutoriels mais surtout j'ai pu me pencher plus sérieusement sur Git et étudier une utilisation plus professionnelle de cet outil.

4.3 Conclusion et perspectives

Ce stage a été enrichissant sur de nombreux points et je suis heureuse que mon travail puisse être utile à de nombreuses personnes.

Ces dernières semaines j'ai pu découvrir de nouvelles façons de travailler mais aussi enrichir ma connaissance de certains outils : programmation orientée agents, MaDKit, Java Web Start, Git etc. Concernant le dernier, ce stage m'a permis d'en avoir une utilisation plus concrète que ce que j'en avais auparavant. De plus, la réalisation de tutoriel est un travail tout à fait intéressant car il permet de se mettre à la place d'utilisateurs potentiellement néophytes et ainsi de prendre conscience à quel point la documentation d'un code est importante. De ce fait, ce travail est aussi instructif pour les utilisateurs que pour la personne à l'origine du-dit code.

Cette expérience a également été l'occasion pour moi de découvrir le monde de la recherche : j'ai pu comprendre un peu mieux le fonctionnement de cet univers et m'intéresser au métier de chercheur ainsi qu'aux études y menant. J'ai également eu l'opportunité d'assister aux soutenances de stages de recherche marquant la fin des études de certains étudiants de master de la faculté des sciences. J'ai pu y découvrir des notions que je ne connaissais pas tout en m'instruisant sur la façon dont pourrait se dérouler ma propre soutenance dans quelques années.

Mon travail est d'ores et déjà accessible et peut être consulté par n'importe quel utilisateur ou personne s'intéressant à MaDKit. De cette façon il est utile à toute une communauté et contribue à enrichir la documentation de cette librairie. Néanmoins ce travail ne fait que s'inscrire dans une longue lignée de tutoriels - existants ou futurs - qui permettront une meilleure accessibilité ainsi qu'une prise en main plus facile de MaDKit.

Bibliographie

Ferber, Jacques. *Les systèmes multi-agents : vers une intelligence collective*. InterEditions, 1995.

Group, MaDKit Developement. *MaDKit*. 2018. url : http://www.madkit.net/madkit/.

Gutknecht, Olivier, Jacques Ferber et Fabien Michel. « MaDKit : une expérience d'architecture de plateforme mulit-agents générique ». In : *Systèmes multi-agents : Méthodologie, technologie et expériences - JFIADSMA 00 - huitième journées francophones d'Intelligence Artificielle et systèmes multi-agents, Saint-Jean-la-Vêtre, Loire, France*. 2000.

Hanachi, C. et C. Sibertin-Blanc. *Introduction aux systèmes multi-agents*. 2000.

Michel, Fabien. « Approches environnement-centrées pour la simulation de systèmes multi-agents ». Thèse de doct. Université Montpellier II, 2015.

— « Programmation situationnelle : programmation visuelle de comportements agents pour non informaticiens ». In : *Systèmes Multi-agents, Défis Sociétaux - JFSMA 10 - Dix-huitièmes journées francophones sur les systèmes multi-agents*. 2010.

— « Une approche méthodologique pour la conception et l'analyse de simulateur multi-agents ». In : *Cinquièmes rencontres des Jeunes Chercheurs en Intelligence Artificielle, RJCIA'00, Lyon, France*. 2000.

Pradeilles, Vincent et Gaëlle Hisler. *Rédaction de tutoriels expliquant le fonctionnement d'un système multi-agents*. 2010.

Glossaire

GitHub : hébergeur en ligne comprenant un gestionnaire de version basé sur Git : https://github.com/.

Git : logiciel de gestion de version libre et décentralisé créé en 2005 par Linus Torvald.

Maven : outil de gestion et d'automatisation de production des projets logiciels Java. Apache Maven est géré par l'organisation Apache Software Foundation.

Foundation for Intelligent Physical Agents : organisation créée en 1996 ayant à coeur la création de normes contrôlant notamment l'interopérabilité des applications, services ou équipements informatiques. Voir http://www.fipa.org/ pour plus d'informations.

AgentSpeak : langage de programmation orienté agents. Se base sur l'architecture BDI (Belief, Desire, Intention).

Plug-in : logiciel ajoutant des fonctionnalités quand il est greffé à un programme de plus grande envergure.

Table des figures

Annexe A - Comparaison de différents outils de modélisation de système multi-agents

L'objectif de ce document est d'analyser des moyens d'implémenter des systèmes multi-agents existants, dans le but de voir les différences qui peuvent exister entre la façon dont est présentée MaDKit.

Jade

Jade tire son nom de Java Agent DEvelopment Framework. Comme son nom l'indique, il s'agit d'un logiciel intermédiaire qui permet de développer des applications multi-agents respectant les spécifications de FIPA. Le but de cet outil est de mettre à disposition des modèles d'agents et de services afin de simplifier le développement de tels systèmes.

La documentation de Jade est assez complète et peut compter sur des guides ainsi que sur une grande quantité de tutoriels explicitant diverses notions. Le seul bémol que j'ai trouvé à leur présentation est le fait que certains des tutoriels sont en PDF. Ceci rend leur compréhension un peu moins intuitive car on ne peut pas voir concrètement ce que fait le code utilisé. Parmi les guides et tutoriels existants on peut citer les guides du programmeur et de l'administrateur Jade ainsi que divers tutoriels comme ceux sur la configuration de Jade, sa sécurité ou ses protocoles de transport de message.

Jade met également à disposition divers outils tels que Jade RMA, Jade Dummy Agent, Jade Sniffer... Pour plus d'informations : http://jade.tilab.com/.

Jason

Jason est un plug-in sous Eclipse permettant d'interpréter une version "améliorée" d'AgentSpeak. AgentSpeak étant un langage de programmation orienté agent. Concernant leur documentation, ils proposent des exemples et de démonstrations qui sont assez similaires aux démonstrations et tutoriels de MaDKit. Parmi leur exemple d'utilisation, on peut compter sur un gold miners et un robot domestique. Les démonstrations disponibles couvrent les notions de communication entre agents, leur création, la gestion d'erreurs (...).

En revanche, concernant la documentation à proprement parler - et non plus des présentations de cas d'utilisation - la Javadoc ainsi qu'un vieux manuel sont accessibles en ligne. Un livre est vendu.

Pour plus d'informations : http://jason.sourceforge.net/wp/.

JaCaMo

JaCaMo est un framework fusionnant trois API existantes : **JAson** pour les agents autonomes, **CArthago** pour l'environnement et **MOise** pour l'organisation entre les différents agents.

Ils disposent de quelques tutoriels sur comment télécharger, configurer ou commencer un projet avec JaCaMo. Concernant les démonstrations, on retrouve un gold miners et un hello world. Pour plus d'informations : http://jacamo.sourceforge.net/.

SARL

SARL est un langage de programmation orienté agent. Sa syntaxe est intuitive et se base sur les Capacités et les Talents des agents ; le but étant de se concentrer plus sur notre système multi-agent que sur la façon de l'implémenter.

Globalement, la documentation de SARL est assez conséquente mais on regrette toute fois l'absence de démonstrations concrètes. Pour plus d'informations : http://www.sarl.io/index.html.

Annexe B - Génération des fichiers JNLP par Ant

Le but de ce document est d'expliquer comment les exécutables Java sont créés avec Ant et comment ils peuvent être lancés depuis un navigateur web.

C'est le fichier XML *build* qui a la charge de créer les exécutables mais aussi les pages HTML mises à disposition sur le site internet de MaDKit. Ce fichier permet également la création de fichiers JNLP (Java Network Launching Protocol) qui seront lancés par Java Web Start (JWS) et de publier le contenu ainsi généré sur le site de MaDKit (que ce soit les pages HTML contenant les fichiers JNLP ou les exécutables). Le format JNLP permet à Java Web Start de lancer des exécutables Java. Il s'agit d'un fichier qui décrit une application Java ainsi que ses dépendances. Ce fichier *pointe* sur l'exécutable de cette application pour la lancer via JWS.

Le fichier *build.xml* va donc créer tout cela à l'aide de ses nombreuses cibles. Étant assez conséquent nous ne présenterons pas ce fichier ici. En revanche il est intéressant de savoir comment en parcourant nos fichiers Java, Ant va pouvoir générer des fichiers JNLP. Pour que ces fichiers soient créés, nous avons inséré deux balises dans les fichiers Java, ces balises sont illustrées juste après. La première **#jws#** permet de spécifier le chemin accédant à ce fichier. La seconde **#args#** contient les arguments que l'on donne à MaDKit pour qu'il exécute ce fichier. À la fin de la construction du projet par le *build.xml* ces balises auront été remplacées par un bouton lançant JWS (le contenu des fichiers Java est intégré dans les fichiers HTML et forme les tutoriels).

```
package returncode.ex5_CGRReturnCodes;

import madkit.kernel.Message;
import returncode.utils.TutorialAgent;

/**
* This class exemplifies: NOT_ROLE.
* This ReturnCode indicates that a role does not exist.
*
*
* #jws# returncode.ex5_CGRReturnCodes._5_NotRole #jws#
* #args# --launchAgents returncode.ex5_CGRReturnCodes._5_NotRole, true
   #args#
*
*/

public class _5_NotRole extends TutorialAgent{
```

```
  /**
   *  We are initializing our _5_NotRole agent in its virtual society.
   */
  @Override
  protected void activate() {
    createGroupIfAbsent("myCommunity", "myGroup");
    requestRole("myCommunity", "myGroup", "myRole");
    pause(500);
  }

  /**
   * The _5_NotRole agent send a message to a role that does not exist.
   * Thus the method sendMessage() will fail and a warning is
       displayed.
   * While checking the returnCode we have decided to display a
       message explaining why the method has failed.
   *
   * A message informing that the sendMessage() has failed will also
       be displayed as a warning.
   */
  @Override
  protected void live(){
    ReturnCode sendFeedback;
    sendFeedback = sendMessage("myCommunity", "myGroup",
        "notExistingRole", new Message()); /* Note that the role is
        different then "myRole" */
    getLogger().info("\n\tThe ReturnCode is: \"" +
        sendFeedback.toString() + "\".\n\tIt means that I can not send
        a message to an agent whose role does not exist. \t\n");

    /* Then you do what you want with this agent */
  }

  /**
   * Launch a _5_NotRole agent.
   * @param argss
   */
  public static void main(String[] argss) {
    executeThisAgent();
  }
}
```

Ce qui nous intéresse dans ce code ce sont les deux balises dont nous avons parlé un peu plus tôt. Ces balises vont être traitées dans le fichier *build.xml* comme il est montré ci-dessous.

```
<target name="build_tuto_files" depends="define-ant-tasks">
  <property name="tuto.instruction" value="\&lt;h2\&gt;Examples are
      structured in packages explicitely named and ordered according
      to their complexity\&lt;/h2\&gt;" />
  <property name="html.break"
      value="\&lt;br/\&gt;\&lt;br/\&gt;\&lt;br/\&gt;\&lt;br/\&gt;\&lt;br/\&gt;\&lt;br/
      />
  <property name="tuto.download.html"
      value="&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp; &lt;a
      href='${download.url}'&gt;&lt;img alt='download project'
      style='border: 0px ; width: 17px; height: 17px;'
      src='http://www.madkit.net/madkit/tutorials/html/download.png'&gt;&lt;/a&gt;"
      />

  <fileset id="java.src.files" dir="${src.dir}/${tuto.src.dir}"
      includes="**/*.java" />

  <java2html title="${tuto.name}" simple="no" tabsize="7"
      marginsize="0" header="true" footer="true"
      destination="${release.dir}/${tuto.dir}/${tuto.destination}">
        <fileset refid="java.src.files" />
        <javadoc localRef="${jdk.api.dir}"
            httpRef="http://docs.oracle.com/javase/8/docs/api/" />
        <javadoc localRef="${mdk.api.dir}"
            httpRef="http://www.madkit.net/madkit/api" />
  </java2html>

  <!-- inserting jws -->
  <replace>
        <fileset dir="${release.dir}/${tuto.dir}/${tuto.destination}"
            includes="**/*java.html" />
        <replacefilter token="#jws# "
            value="&lt;script&gt;deployJava.createWebStartLaunchButton('http://www.mad
            />
        <replacefilter token=" #jws#" value="',
            '1.8.0');&lt;/script&gt;" />
        <replacefilter token=".java&lt;/font&gt;"
            value=".java&lt;/font\&gt;  ${tuto.download.html}" />
        <replacefilter token="/HEAD" value="script
            src='http://www.madkit.net/js/deployJava.js'&gt;&lt;/script&gt;&lt;/HEAD"
            />
  </replace>

  <replaceregexp match="#args#(.*)#args#" replace="">
        <fileset dir="${release.dir}/${tuto.dir}/${tuto.destination}"
```

```
            includes="**/*java.html" />
    </replaceregexp>

    <!-- modifying front pages -->
    <replace
        file="${release.dir}/${tuto.dir}/${tuto.destination}/front.html">
          <replacefilter token="Instructions:-" value="&lt;a
              href='http://www.madkit.org/'&gt;&lt;img alt='MDK'
                  src='../MaDKit.png' style='border: 0px solid ; width:
                      100px; height: 100px;'
                      align='right'&gt;&lt;/a&gt;&lt;h1&gt;${tuto.name}&lt;/h1&gt;"
                      />
          <replacefilter token="Top-Left Panel Selects a Package"
              value="${tuto.description}" />
    </replace>
    <replaceregexp match="\&lt;li\&gt;(Text.*?|Bott.*?)li\&gt;"
        replace="${html.break}" flags="gs">
          <fileset dir="${release.dir}/${tuto.dir}/${tuto.destination}"
              includes="front.html" />
    </replaceregexp>
    <replaceregexp match="Credits"
        replace="${tuto.instruction}${html.break}${html.break}\&lt;hr/\&gt;Credits"
        flags="gs">
          <fileset dir="${release.dir}/${tuto.dir}/${tuto.destination}"
              includes="front.html" />
    </replaceregexp>
    <replaceregexp match="\(If you like(.*)em&gt;"
        replace="\&lt;br/\&gt;" flags="gs">
          <fileset dir="${release.dir}/${tuto.dir}/${tuto.destination}"
              includes="front.html" />
    </replaceregexp>

    <!-- modifying index pages -->
    <replaceregexp match="cols=\p{Punct}30%, 70%\p{Punct}"
        replace="cols='20%, 80%'" flags="gs">
          <fileset dir="${release.dir}/${tuto.dir}/${tuto.destination}"
              includes="index.html" />
    </replaceregexp>

    <!-- create-jnlps -->
    <property name="files" refid="java.src.files" />
    <for list="${files}" delimiter=";" param="file">
          <sequential>
                  <createjnlpfromjavasource srcfile="@{file}"
                      jnlptemplate="${jnlp.template}"
                      destdir="${release.dir}/${jnlps.dir}" />
          </sequential>
    </for>
```

```
  <!-- remove jws tags from java source files -->
  <copy todir="${release.dir}/${src.dir}/${tuto.src.dir}">
        <fileset dir="${src.dir}/${tuto.src.dir}" />
        <filterchain>
                <linecontains negate="true">
                        <contains value="#jws#" />
                </linecontains>
        </filterchain>
        <filterchain>
                <linecontains negate="true">
                        <contains value="#args#" />
                </linecontains>
        </filterchain>
  </copy>
  <!-- add cleaned java sources to project zip file -->
  <zip destfile="${release.dir}/${project.zip.file}" update="true">
        <zipfileset dir="${release.dir}/${src.dir}/${tuto.src.dir}"
           prefix="${src.dir}/${tuto.src.dir}" />
  </zip>

</target>
```

La cible précédente se charge de créer les fichiers JNLP et de les insérer dans les fichiers HTML. Ce sont ces derniers que l'on retrouvera sur le site de MaDKit. C'est de cette façon que le bouton permettant de lancer l'exécutable par Java Web Start est intégré aux tutoriels.