‘

Rapport TP

Rapport projet Systèmes Multi-Agents

S9 -2017

Yoann Fleytoux
Aurélien Bernier

Sommaire

Introduction

Ce projet à pour objectif de mener à bout la simulation d'un système multi-agent visant à automatiser la luminosité d'une salle de cours en vue de minimiser la consommation d'énergie et d'offrir à tout moment un degré de luminosité adéquate.

Les passages en anglais viennent du site https://www.irit.fr/ADELFE/, ils sont la pour aiguiller la démarche et faciliter nos futures révisions.

Besoins initiaux

Définition du besoin utilisateur

Functional requirements
A requirement that specifies an action that a system must be able to perform, without considering physical constraints; a requirement that specifies input/output behavior of a system.

Non functional requirements
A requirement that specifies system properties, such as environmental and implementation constraints, performance, platform dependencies, maintainability, extensibility, and reliability. A requirement that specifies physical constraints on a functional requirement

Besoins fonctionnels:

Les étudiants et enseignants ont besoin que le degré de luminosité soit toujours adéquat à l'utilisation de la salle de cours et que ce contrôle soit transparent pour eux. L'université a besoin d'assurer ce confort et de minimiser sa consommation d'énergie.

Besoins non fonctionnels:

Le système doit être facilement modifiable et peu coûteux ainsi que robuste aux erreurs (en cas de dysfonctionnement des capteurs ou actionneurs, un contrôle manuel est possible), ce qui ne sera pas forcément considéré dans la simulation.

Besoins consensuels:

Le système doit être capable de prendre en compte le feedback des utilisateurs (enseignant et étudiants), de s'adapter à l'agenda à horaires variables d'utilisation de la salle et d'être commandé manuellement.

Mots-clefs:

Consommation d'énergie, transparence, luminosité, capteurs, actionneurs, salle de cours, lumières intelligentes, volets automatiques, capteur de luminosités, feedback

Contraintes:

- Commande manuelle possible
- prise en compte du feedback utilisateur
- prise en compte de l'agenda d'utilisation de la salle
- prise en compte d'une limite de consommation

Limites:

-précisions des capteurs
-efficacité des lumières
-opinions divergentes du feedback utilisateur

Définition du de l'environnement

Environment
The environment of an agent refers to all that is external to the agent. One can distinguish the social environment (the agents its knows) from the physical environment (the material resources that can be perceived by the agent or by its own effectors).

Entités:

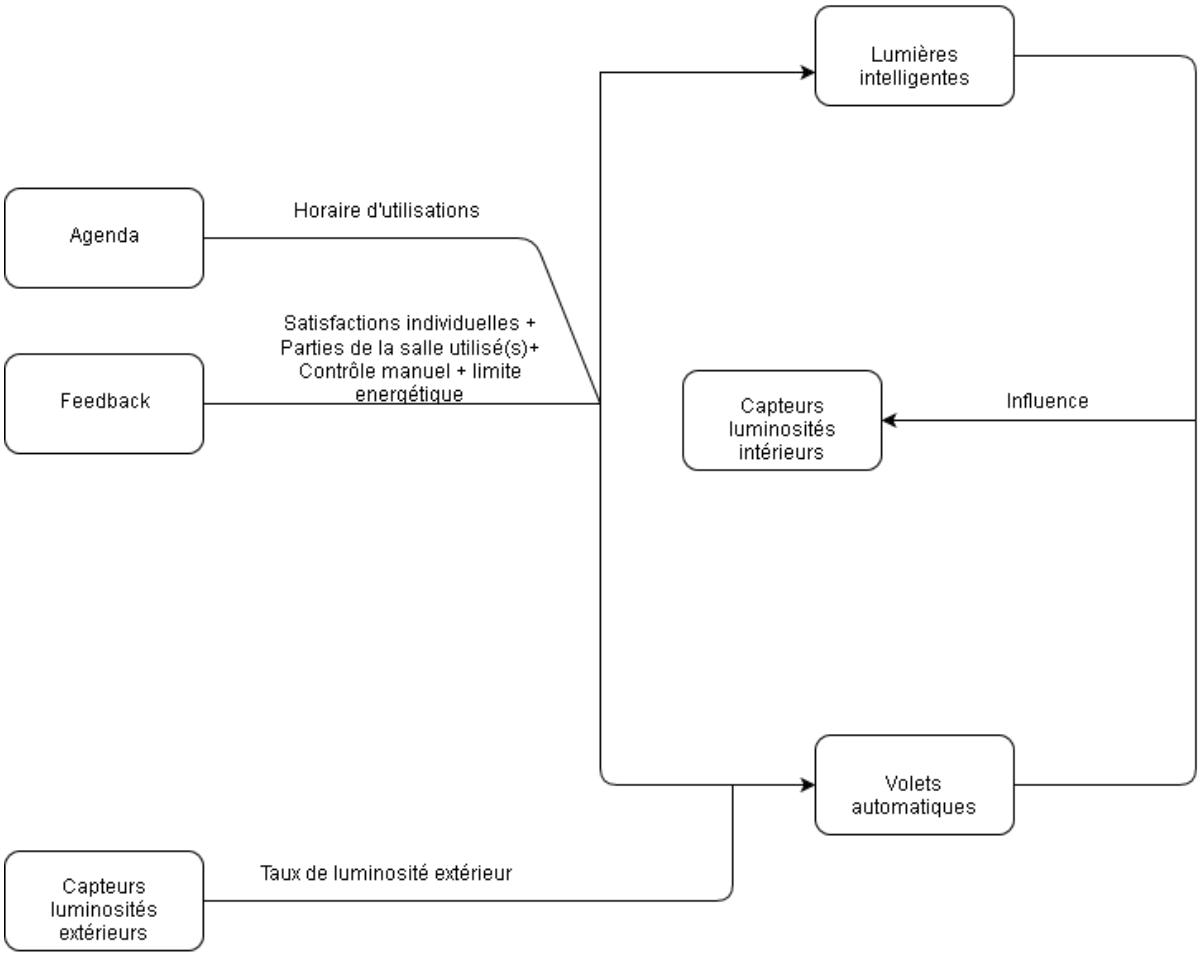
Entity
In ADELFE, two types of entities will be used:

- **Active** entities which may behave autonomously; for instance, by bringing modifications to their functioning constraints. They are able to act in a dynamical way with the system. Agents composing the system will be found among them,
- **Passive** entities which can be considered as resources by the system. Interactions with the system are restricted to data exchanges in order to realise the task the system must achieve. Passive entities may be used or modified by active ones but they do not change by themselves in an autonomous manner.

Entité	Type
Lumières intelligentes	actif
Volets automatiques	actif
Feedback	passif
Capteurs luminosités intérieurs	passif
Capteurs luminosités extérieurs	passif
Agenda	passif

Nous faisons plusieurs hypothèses:
- La présence ou non de la cloison peut être exprimé dans le feedback de l'utilisateur (en précisant quel parties de la salle sont utilisés, et ont donc besoin d'être éclairé), ainsi que la limite de consommation énergétique fixée. Le contrôle manuel également.
- Les capteurs sont autonomes, ils ne s'éteindront et ne s'allumeront pas en fonction d'autres agents.
- L'agenda ainsi que la météo seront considérés comme aléatoires.

Flots de données et interactions entités-systèmes:



Caractérisation de l'environnement :

Accessible environment (as opposed to "inaccessible").
The system can obtain complete, accurate and up-to-date information about the state of its environment. In an inaccessible environment, only partial information is available to the system. For example, an environment such as the Internet is not an accessible one because knowing all about it is impossible. A robot which evolves in an environment has sensors to perceive it and generally, these sensors do not allow it to know all about its environment. In this case, its environment is also considered as being inaccessible.

Le système est accessible: on a accès aux données qui caractérisent l'environnement (taux de luminosité, consommation énergétique, usage et satisfaction des utilisateurs)

Continuous environment (as opposed to "discrete").
In a continuous environment, the number of possible actions and perceptions in the environment is infinite. In a discrete environment, the system has distinct, clearly defined percepts that describe the environment. For example, in a real environment like the Internet,

the number of actions that can be performed by users can be unbounded. But, in a simulated environment, such as an ecosystem, the number of actions or perceptions a virtual entity (like an ant or a robot) can have is limited, the environment can then be discrete.

Le système est discret: les valeurs des consignes et des données sont numériques.

Deterministic environment (as opposed to "non deterministic").
In a determinist environment an action has a single and certain effect. If the system acts in its environment, there is no uncertainty about the effect of its action on the state of the environment. The next state of the environment is completely determined by the current state. In a non deterministic environment, an action does not have a single guaranteed effect. By nature, the real physical world is a non-deterministic environment.

Le système est déterminé: le système réagira tout le temps de la même manière avec les mêmes données en entrées.

Dynamic environment (as opposed to "static").
The state of a dynamic environment depends upon actions of the system that is within this environment but is also dependent on the actions of some other processes. So, changes cannot be predicted by system. A static environment cannot change while the system is not acting. For example, the Internet is a highly dynamic environment.

Le système est dynamique: Le système est influencé par les actions des utilisateurs (ce qui n'est pas prévisible).

Identification des agents

Agent

There is no universal definition of an "agent". The classical definition by Ferber can be used as a starting point.
An agent is a physical or virtual entity:

- which is capable of action in an environment
- which can communicate directly or not with other agents
- which is driven by a set of tendencies (in the form of individual objective or of a satisfaction/survival function which it tries to optimise)
- which possesses resources of its own
- which is capable of perceiving its environment (but to a limited extent)
- which has only a partial representation of this environment (and perhaps none at all)
- which possesses skills and can offer services
- which may be able to reproduce itself
- whose behavior tends towards satisfying its objective, taking account of the resources and skills available to it and depending of its perception, its representation and the communication it receives. [Ferber 99]

When working with co-operative agents to build Adaptive Multi-Agent Systems, the **autonomy** is the main property of an agent. **An agent is able to decide of its own behaviour**. This property distinguishes it from other entities such as objets.

Furthermore, agents in AMAS have a special social attitude: they must be **cooperative**. So, an agent must detect and process Non Cooperative Situations to always act to come back in a state it judges being cooperative from its own point of view. For instance, an agent that does not possess an information requested by another agent will do all it can to find another agent able to answer this request.

The lifecycle of an agent is: perceive -> decide -> act.

Autonomy

The autonomy of an agent can be expressed as following:

- An agent has its own life, independently of the existence of other agents,
- An agent is able to survive in dynamic environments without an external control,

- An agent takes internal decisions about its behaviour only considering the perceptions, knowledge and representations it possesses.

Goal

A goal is a set of states of the world that an agent is committed to achieve/maintain. Therefore a goal is a situation, but not all situations are goals. A set of states of the world is generally not a goal unless there is an agent committed to achieve/maintain this set of states [Eurecom 00].

Entité	Type	Autonome	But	Interaction	Négociation
Lumières intelligentes	actif	autonome	But local (satisfaire les contraintes de luminosités)	Capteurs luminosités intérieurs, Feedback, Agenda	Négocie
Volets automatiques	actif	autonome	But local (satisfaire les contraintes de luminosités)	Capteurs luminosités intérieurs, Capteurs luminosités extérieurs, Feedback, Agenda	Négocie
Feedback	passif	Non autonome (nécessite que les utilisateurs utilise la plateforme)	Pas de but local, communique ses informations	Lumières intelligentes, Volets automatiques	
Capteurs luminosités intérieurs	passif	autonome	Pas de but local, communique ses informations	Lumières intelligentes, Volets automatiques	Vue partielle
Capteurs luminosités extérieurs	passif	autonome	Pas de but local, communique ses informations	Volets automatiques	Vue partielle
Agenda	passif	Non autonome (nécessite d'être remplis manuellement)	Pas de but local, communique ses informations	Lumières intelligentes, Volets automatiques	Vue partielle

Les lumières intelligentes, et les volets automatiques valident les 3 premiers critères, ils sont donc potentiellement des agents.

Toutes ces entités interagissent avec un environnement dynamique (impacté par les utilisateurs au niveau du feedback et de l'agenda). Les cas de dysfonctionnements des capteurs et actionneurs ne sont pas considérés dans cette simulation.

Conception des agents

Compétences

Lumières intelligentes

Ajusté le niveau de luminosité délivré.

Volets automatiques

Ajusté la hauteur des volets

Aptitudes

Lumières intelligentes

Prend en compte les données des capteurs de luminosités intérieurs, du feedback et de l'agenda.

Volets automatiques

Prend en compte les données des capteurs de luminosités intérieurs et extérieurs, du feedback et de l'agenda.

Langage d'interaction

Lumières intelligentes

Appel de la fonction de régulation de la luminosité

Volets automatiques

Appel de la fonction de régulation de la hauteur des volets.

Représentation du monde

World representation
World representations of an agent - or beliefs - concern other agents (its social environment), the physical environment or the agent itself. The agent must always be able to access these representations to decide of its behaviour and, possibly, it must be able to modify them.

Lumières intelligentes

Capteurs luminosités intérieurs, Feedback, Agenda

Volets automatiques

Capteurs luminosités intérieurs, Capteurs luminosités extérieurs, Feedback, Agenda

Situation Non copératives

Lumières intelligentes & Volets automatiques

Perception:

Incompréhension : Le changement de la luminosité n'impacte pas la valeur perçue par les capteurs de luminosité intérieurs.
Ambiguïté : Si la valeur des capteurs de luminosité intérieurs est inappropriée, on satisfait le feedback utilisateur en priorité.

Décision:

Incompétence : Perte de fonctionnalité -> On utilise une autre lumière.
Improductivité : Adapter une lumière ne suffit pas pour donner la luminosité demandée par le feedback utilisateur -> On adapte les voisins aussi.

Action:

Concurrence : Quand deux lumières voisines ont le même niveau de criticalité.
Conflit :
Inutilité : Les stores quand la lumière du soleil est nulle.

Architecture

En suivant le framework donné:
AMAS.java : Système AMAS et gestion de l'IHM.
Classroom.java : L'environnement de l'AMAS.
BrightnessAgents.java : Les agents gérant la lumière, super-classe de Light et Blind.

- Light.java : Lumières intelligentes, agent.
- Blind.java : Volets intelligents, agent.

ExteriorLightSensor.java : Capteurs de luminosité extérieurs, gère également la météo.
InteriorLightSensor.java : Capteurs de luminosité intérieurs.
OccupiedHours.java : Gestion de l'agenda
Settings.java : Gestion des des variables
Launch.java : Lancement de l'application (main)

Simulation

Les entités extérieures au système :

La variation de la luminosité extérieure sera simulée entre 8h et 21h avec un pic à 12h.

La simulation sera codée en Java.

Afin de modéliser notre système, nous nous sommes basés sur une salle de l'université que nous ne connaissons que trop bien, la salle 108 du bâtiment U3. Vous pouvez voir ci-dessous les photos (en rouge les lumières et en bleu les capteurs) :

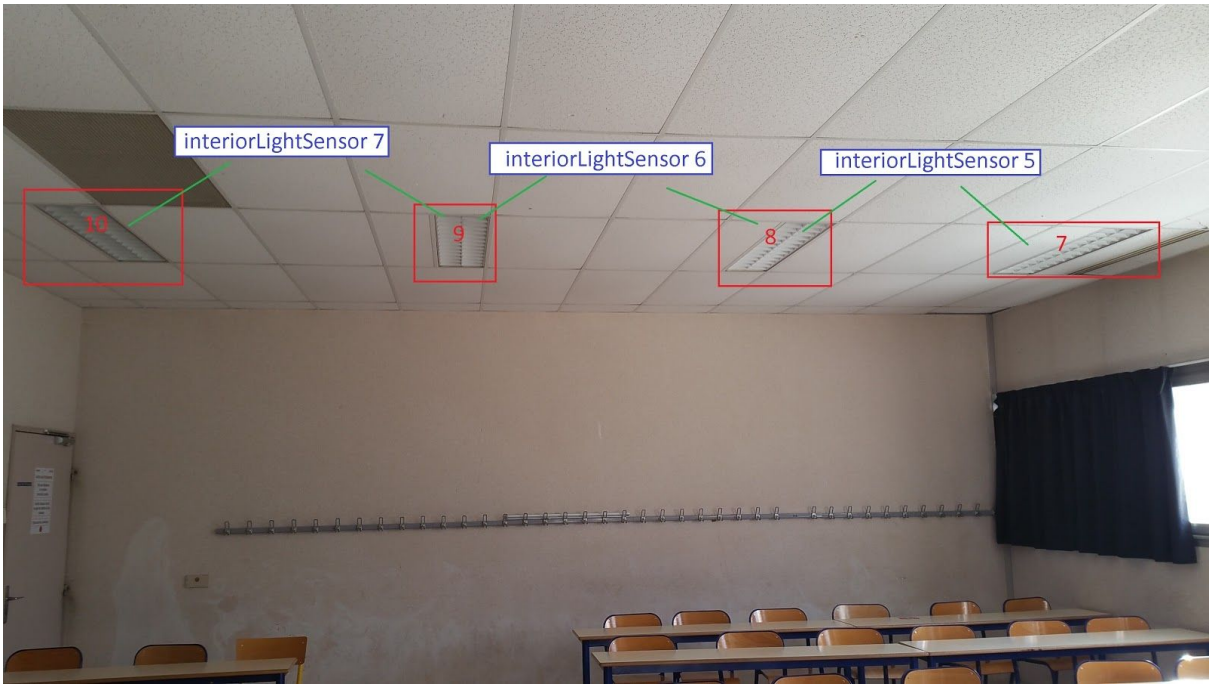


Image du fond de la salle

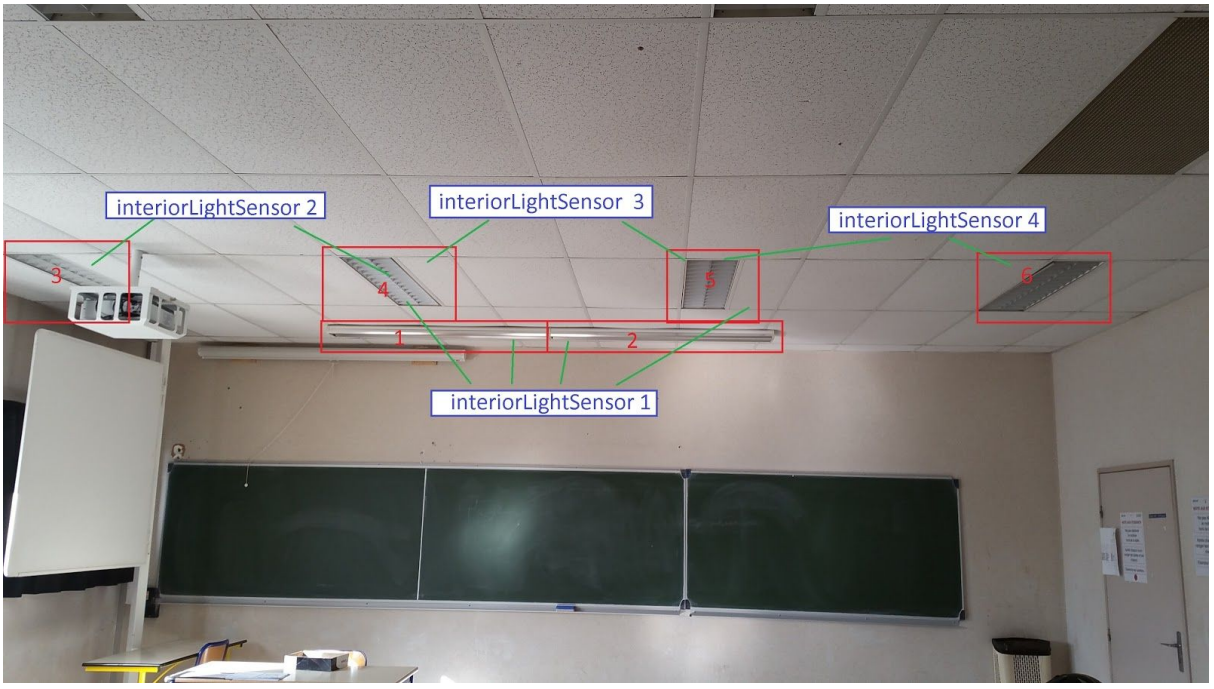


Image du devant de la salle (à noter la particularité des deux lumières sous le tableau)

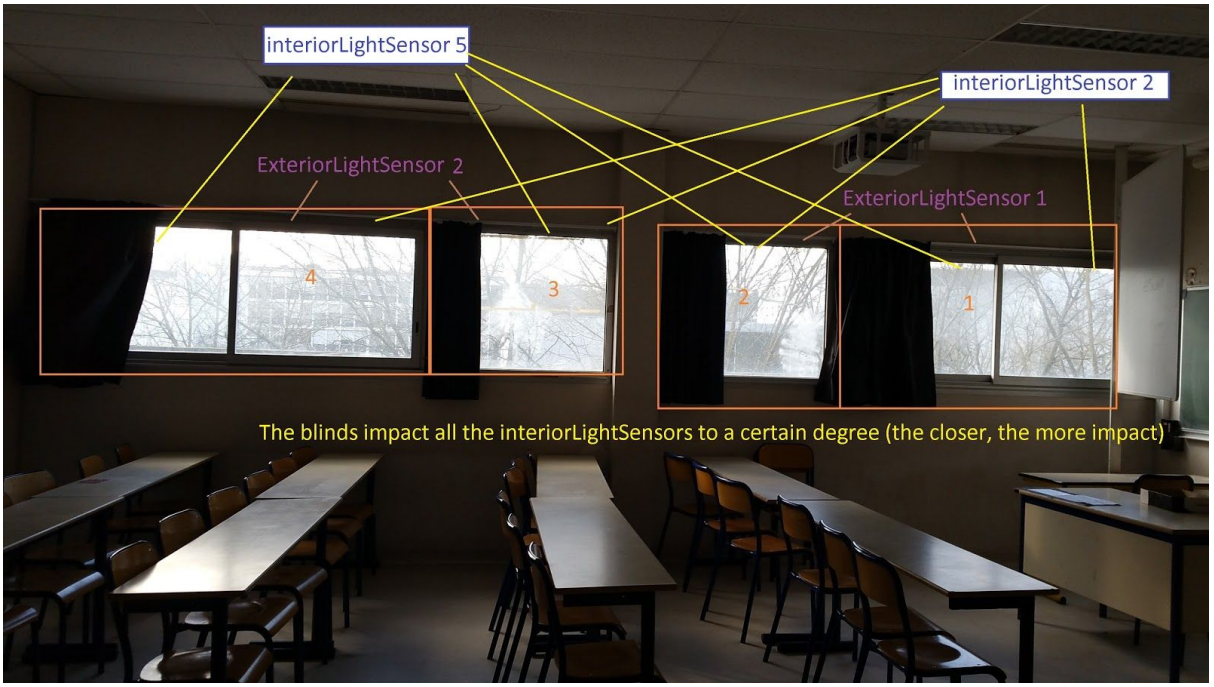
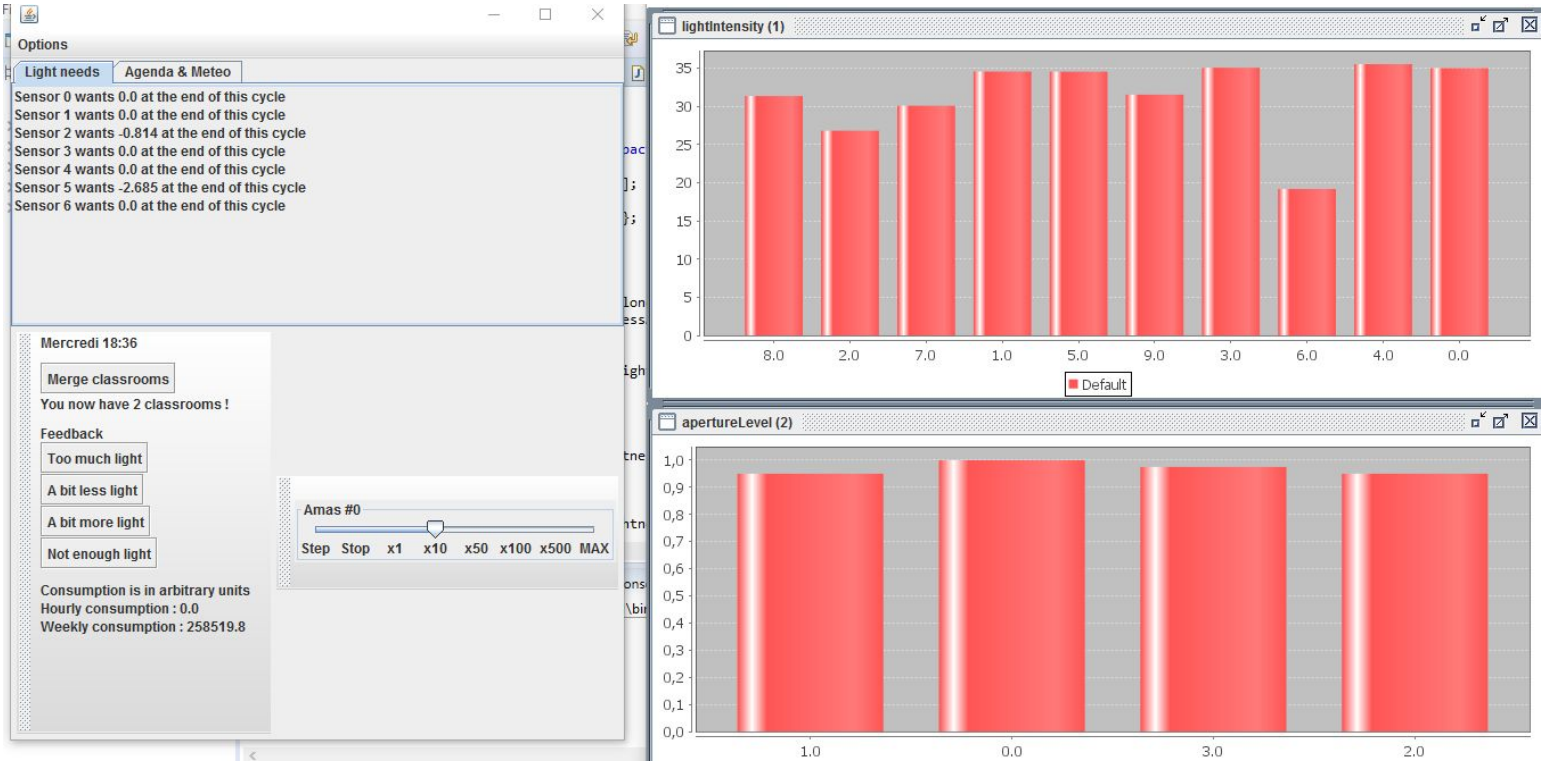


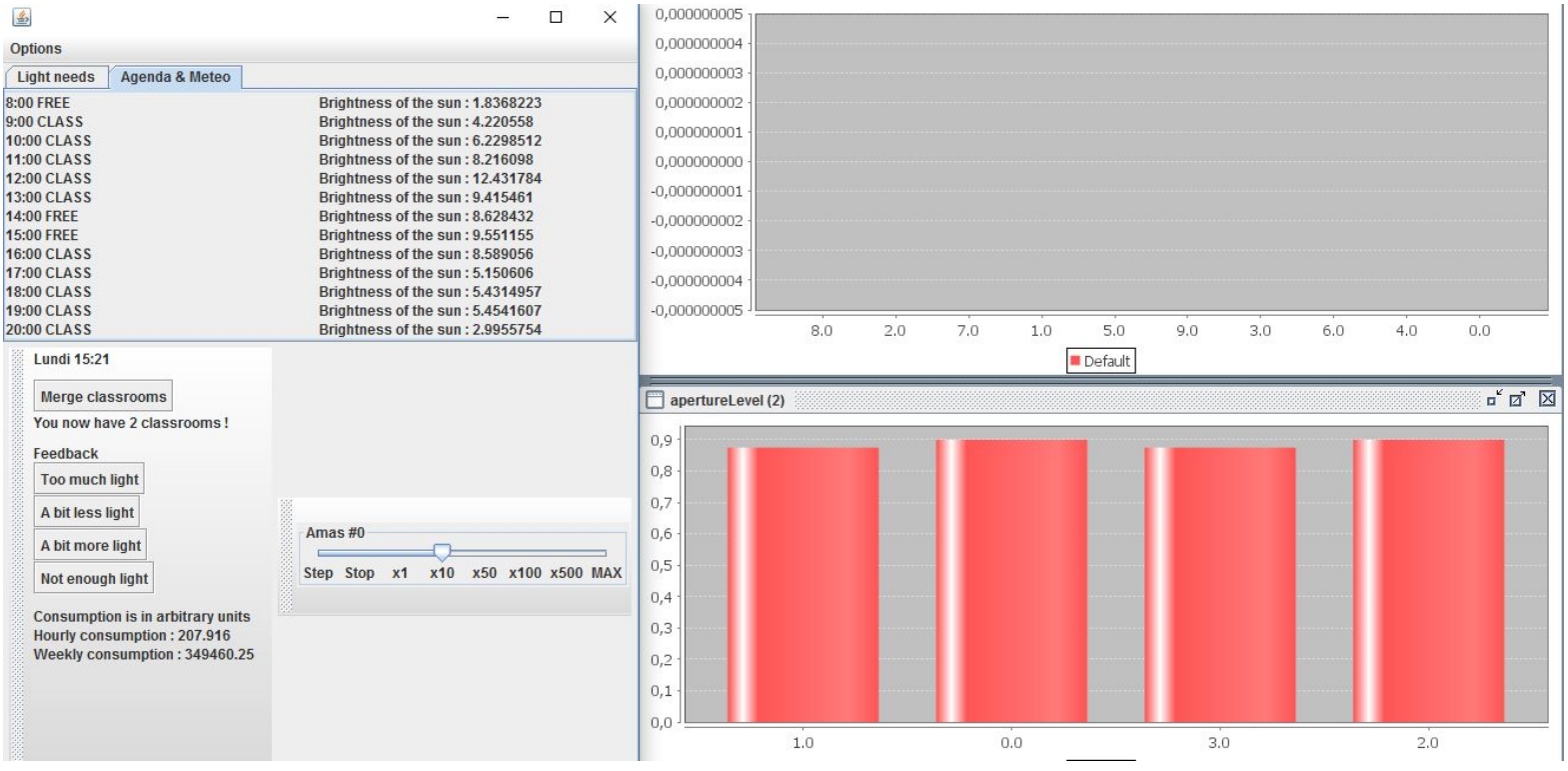
Image des stores (à noter que les stores impactent tous les capteurs intérieurs, ce qui se traduit dans le framework AMAS par le concept de voisin de tous les capteurs intérieurs)

Environnement -> classroom, 7 capteurs lumières intérieur, 2 extérieurs, 1 agenda

Interface



L'interface permet donc de gérer la séparation des classes, le feedback, et affiche l'intensité lumineuse (de 1 à 100) et l'ouverture des stores (de 0 à 1), on peut également voir sur le premier onglet les besoins des différents capteurs de luminosité ainsi que la consommation par heure et par semaine. La dernière consommation est affichée, on a donc 0 ici pour l'heure car il n'y avait pas cours Mercredi de 17 à 18.



On peut voir ici que le deuxième onglet affiche l’emploi du temps de la journée ainsi que la météo (les deux générés aléatoirement).

Choix techniques

Le feedback général affecte un ratio entre 0 et 1, qui est multiplié au besoin de toutes les lumières.

Il y a une difficulté de s'assurer que les volets aient priorité sur les lumières pour gérer la luminosité, en effet, il consommera probablement moins d'ouvrir les volets en entier si ils ne le sont pas que d'augmenter la puissance des lampes.

Criticalité des lumières : leurs intensités
Si les capteurs affectés par la lumière veulent globalement plus de lumière, elle augmente sa luminosité si ce n'est pas la plus lumineuse parmis ses voisins.
Si les capteurs affectés par la lumière veulent globalement moins de lumière, elle diminue sa luminosité si c'est la plus lumineuse parmis ses voisins.
Si il n'y a pas cours, les lumières sont éteintes.

Criticalité des volets : leurs ouvertures
Si les capteurs affecté par le volet veulent globalement plus de lumière, il augmente son ouverture si ce n'est pas le plus ouvert parmis ses voisins.
Si les capteurs affecté par le volet veulent globalement moins de lumière, elle diminue son ouverture si c'est le plus ouvert parmis ses voisins.
Si il n'y a pas cours durant le jour les volets restent ouvert comme ils étaient à l'heure précédentes.
Durant la nuit les volets se ferment (pour penser aux économies de chauffage).

L'agenda est aléatoire sauf pour les 3 soirs par semaine. Pas de cours Samedi ni Dimanche.
La météo suit une base définie dans le code avec des variations aléatoires sur la luminosité à chaque heure.

Critique de notre implémentation, nous n'avons pas pris en compte assez tôt le fait de diviser la classe en deux, donc le feedback reste global pour les 2 classes ce qui n'est pas optimal.

Comparaison de la consommation

En prenant cette agenda avec ce nombres d'heures de cours par jours:

Jour	Lundi	Mardi	Mercredi	Jeudi	Vendredi	Samedi	Dimanche
Nombres d'heures	6	5	7	5	9	0	0

Total des heures de cours: 32

Consommation de notre simulation Dynamique sans feedback:

Consommation théorique du Plan Statique

nombres_de_lumières*intensité_lumières_max*nombre_heures_semaine*nombre_de_minutes_par_heures+
nombres_de_volets*(consommation_lever_entierement_volet+consommation_baisser_entierement_volet)*nombre_de_jour_avec_cours
=10*100*32*60+4*(10+10)*5=1920400

287284.36/1920400~=0.15 =15%

Paramètres utilisés:
nombres_de_lumières=10
intensité_lumières_max=100
nombres_de_volets=4
consomation_lever_entierement_volet=consommation_baisser_entierement_volet=10

Théoriquement, on utilise seulement 15% des ressources avec notre Simulation.

Limites de cette comparaison:

La météo est aléatoire.

Pour le plan statique, on considère que l'on éteint les lumières quand il n'y a pas cours mais qu'on les allume toutes à l'intensité maximum si il y a cours.

La consommation des volets est dérisoire comparée à celle des lumières, ce qui avantage le modèle dynamique vu que celui-ci les fait agir plus souvent.

Conclusion

Ce projet nous aura permis d'appliquer l'architecture AMAS à un cas concret, la consommation d'électricité dans une salle de classe réduite via des agents intelligents autonomes, dans le cadre du projet NeoCampus.
Nous avons donc réalisé un modèle contenant les différentes variables permettant de modéliser ce système ainsi que les différents agents et capteurs, indépendants mais marchant en coopération.
Nous avons donc défini un agenda aléatoire, une météo aléatoire, avant de modéliser de la manière la plus cohérente pour nous la coopération entre les différents agents via leur visualisation de leurs voisins.

Pour les résultats des différents types de gestion de la luminosité de la salle :

- Le mode "statique" est évidemment le plus coûteux en terme d'énergie, mais ne nécessite pas d'installer du niveau matériel.

- Le mode dynamique sans feedback est le plus efficace énergétiquement mais ne remplit que la moitié du cahier des charges, il manque le confort des utilisateurs. Ce mode est basé sur une gestion de l'agenda des cours, de la météo.

- Le mode dynamique avec feedback est un peu moins efficace énergétiquement, sauf bien sûr si les utilisateurs jugent que le système donne toujours trop de lumière. De plus, moins le feedback est utilisé, plus on se rapproche des performances du mode dynamique sans feedback. Pour finir, c'est le seul mode gérant tout le cahier des charges.