

Université de Nice - Sophia Antipolis

IUT de Nice - Côte d’Azur

Département Informatique

41 Boulevard Napoléon III - 06200 Nice

Rapport de stage pour l’obtention du

Diplôme Universitaire de Technologie

Session 2016-2017

**Modélisation et simulation de scénario de suivi de personnes**

**avec injection de données capteurs**

Présenté par Lucas Kacem

54 Nguyễn Lương Bằng,

Hòa Khánh Bắc, Liên Chiểu,

Đà Nẵng

Sous la direction de :   
Mme Marie-Agnès Peraldi-Frati - Tutrice

HUYNH Huu Hung - Tuteur

# 

# Remerciements

Je tiens à remercier **M. Nhan le Thanh** qui nous a permis de participer à ce stage à l’université de Da Nang au Vietnam.

Mes remerciements s’adressent aussi à **Mme Marie-Agnès Peraldi-Frati**, Tutrice IUT, et à **M. Huynh Huu Hung**, Tuteur de l’université de Da Nang, ainsi qu’à **M. Julien Deantoni**, chercheur, pour leur encadrement durant la totalité du stage.

Je remercie, également, **Mme Le Thi Hong Oanh** et **M. Tran Van Nam**,pour l’organisation de notre séjour lors de ce stage.

De plus, je remercie, AUF, et en particulier **M Nguyen** et **Le Duc Huy**, pour avoir mis à notre disposition leurs locaux lors de notre stage.

Enfin, je remercie tous les professeurs de l’université pour leur accompagnement et leur écoute.

**Introduction**

Le stage pour obtenir le Diplôme Universitaire de Technologie (DUT), objet de ce rapport, se déroule à l’Université de Technologie de Danang, au Vietnam,

en lien avec le Danang-Nice Institute of Technology (DNITT) et en partenariat avec L'Université de la Côte d'Azur (UCA) et avec la Maison de l'innovation et de la recherche (MIRE) dans le cadre du projet SLEGO du Centre de Recherche de Danang.

L'objectif de ce stage, effectué du 10 avril 2017 au 24 juin 2017 est le développement d’un modèle et la simulation de scénario de suivi de personnes et l'évaluation du scénario de suivi indoor/outdoor, en temps réel par injection de données capteurs. Nous considérons le cas d’usage où l’on souhaite tracer les activités quotidiennes d’une personne vivant à domicile.

Un réseau de capteurs (présence, humidité, luminosité, pression, camera, ...) est installé à l'intérieur et à l'extérieur du domicile.

Nous devons créer des scénarios correspondant d'abord à l'activité quotidienne normale de la personne. Puis des scénarios atypiques pour lesquels des alarmes doivent être transmises aux aidants / médecins.

Notre objectif est de simuler des scenarios tel que la prise de douche.

Premièrement, avec l'insertion de données de capteur de la part de l'utilisateur.

Dans un second temps, avec l'insertion de données stockées dans un fichier de trace.

Ce rapport présente tout d'abord l'entreprise et le contexte de travail.

Puis nous analyserons les besoins. Nous développerons, en particulier, la création de différents scenarios avec l'outil GEMOC.

Enfin, un nouveau paragraphe détaillera la modélisation de scénario en utilisant un logiciel fourni par l’Inria: GEMOC studio.

# Summary

The internship to obtain the « Diplôme Universitaire de Technologie (D.U.T) subject of this report, takes place at Da Nang University of Technology in Vietnam, in connection with the Da Nang-Nice Institute of Technology (DNITT) and in partnership with l’Universite de la Côte d’Azur (UCA) and La Maison de l’innovation et de la recherche (MIRE) as part of SLEGO project of Research Center of University of Danang.

The objective of this intership, done from 10 April 2017 to 24 June 2017 is the development of a model simulation of persons follow up scenario and evaluation of this follow up scenario Indoor/ outdoor in real time by injection of sensor data.

We consider the case where we want to trace the daily activities of a person living at home.

A network of sensor (presence humidity, brightness, pressure, camera, ...) is placed inside and outside home.

We need to create scenarios corresponding first at normal daily activity of the person.

Secondly atypical scenarios for which alarms must be transmitted to helpers / doctors.

One objective is to simulate scenarios, such as taking a shower.

First with insertion of sensors data of the user.

Then with insertion of data directly stocked in a file.

First this report presents the entreprise and the work context ; then we will analyse the needs.

We will particularly develop the creation of different scenarios with the tool GEMOC.

Finally, a new paragraph will detail modeling scenarios using the software provided by l’Inria : GEMOC

# Table des matières

Remerciements 2

Summary 4

Table des matières 5

1.1 Présentation de DNITT : UCA et Université de Danang 6

1.2 Présentation de MIRE 6

1.3 Présentation du projet SLEGO 6

2. Présentation et contexte du stage 7

2.1 Présentation du sujet 7

2.1.1 Objectifs 7

2.1.2 Description de l'existant 7

2.1.3 Critères d'acceptabilité du produit 7

2.1.4 Besoins fonctionnels 7

2.2 Présentation de l’outil GEMOC studio 8

2.2.1 Descriptif de l’environnement et outils utilisés 8

2.2.2 Environnement de développement GEMOC 9

2.2.3 SignPML - Domain Specific Modeling 10

2.2.4 Sirius 11

2.2.5 Swing (Java) 12

3. Analyse des besoins 13

3.1 Expression 13

3.2.1 Architecture de la solution : Dispatcher 14

3.1.2.2 Architecture de la solution : Les différents sensors 15

3.1.2.3 Architecture de la solution : l’agent contrôleur 16

3.1.2.3 Architecture de la solution : l’agent graphique 17

4. Travaux réalisés 18

4.1 Solution de simulation de douche avec valeurs entrées par l’utilisateur 18

4.2 Solution de simulation de douche avec valeur analysée par fichier .CSV 19

5. Evolution possible 25

5.1 Evolution possible : solution de simulation de douche avec valeur analysée par fichier .CSV 25

5.1.1 Evolution possible à court terme 25

5.1.2 Evolution possible à moyen terme 25

5.1.3 Evolution possible à long terme 25

5.2 Solution de simulation de douche avec valeur entrée par l’utilisateur 26

5.2.1 Evolution possible à court terme 26

5.2.2 Evolution possible à moyen terme 26

5.2.3 Evolution possible à long terme 26

6.0 Planification 27

6.1 Nombre d’heures effectuées 27

6.2 Planning prévisionnel 27

6.3 Rétrospective 28

Conclusion 29

Table des figures 30

Références 31

**1. Présentation de l’entreprise**

## 1.1 Présentation de DNITT : UCA et Université de Danang

Danang-Nice Institute of Technology (DNITT) s'inscrit dans la logique de déploiement de l'initiative d'excellence (IDEX) de UCA consistant à projeter à l'international, sur des sites miroirs, des projets de recherche et d'innovation sur lesquels viennent se greffer des créations et des projections de formation initiale et continue.

## 1.2 Présentation de MIRE

MIRE est un projet de collaboration entre l’Université de Da Nang (UD) et l’Université de Nice (UNS), afin de promouvoir le partage de connaissance, l’innovation et la recherche à l’Université de Da Nang (UD).

## 1.3 Présentation du projet SLEGO

La transmission des connaissances et du savoir-faire (appelé ici : expérience) est un des moteurs principaux du développement dans toutes les dimensions : sociale, économique, technologique, scientifique, etc.

La description d’une expérience nécessite la représentation de l’enchaînement de phases spécifiques au domaine applicatif ainsi que des données qui sont, soit des informations personnelles ou professionnelles de la personne qui détient cette expérience, soit les informations liées à l’expérience elle-même.

Dans plusieurs secteurs, tels que la santé, l’économie, la sécurité, ce mélange devient un obstacle pour tous ceux qui veulent partager leurs expériences mais aussi protéger les données concernant leur vie personnelle et/ou professionnelle.

SLEGO veut dépasser cet obstacle en fournissant des supports permettant à chacun de décrire et de transmettre ces expériences avec une séparation claire entre les éléments de l’expérience et les données qu’elle manipule ou qui lui sont associées.

Une fois l’expérience modélisée, SLEGO permettra de réaliser cette expérience en mode simulation dans l’environnement de développement et/ou de réaliser en temps réel cette expérience et d’effectuer un suivi en temps réel de son exécution.

# 2. Présentation et contexte du stage

## 2.1 Présentation du sujet

### 2.1.1 Objectifs

L’objectif principal est de réaliser une application qui permet le suivi de personnes âgées et isolées et qui permet de savoir en temps réel l’activité des personnes.

Le but de ce suivi est de prévenir les familles et aidants des comportements anormaux de ces personnes en remontant des alarmes et ainsi de prodiguer le meilleur soin possible et de s’occuper d’elles à tout moment, quand elles en ont besoin.

### 2.1.2 Description de l'existant

Nous disposons de plusieurs ordinateurs, du logiciel GEMOC, d’une salle de l’université de technologie de Da Nang, de raspberry, de capteurs d’humidité, de capteurs PIR et de capteurs de lumière pour pouvoir tester les différents scénarios.

### 2.1.3 Critères d'acceptabilité du produit

Lorsque les scénarios fonctionneront avec les valeurs envoyées par les capteurs, alors, nous pourrons considérer que la procédure est validée. Par ailleurs, si les données sont envoyées directement par l’utilisateur et non par les capteurs (sachant que nous considérons les sujets séparément et que, par la suite, nous n’avons aucune possibilité de connaître la validité sur Sujet 3), nous appliquerons comme critère d’acceptation des valeurs rentrées par nous-même se rapprochant fortement de celles habituellement envoyées par les capteurs

### 2.1.4 Besoins fonctionnels

Dans une première phase, nous utiliserons des artefacts déjà présents dans l’outil Gemoc Studio pour produire un modèle de l’application visée ; à savoir l’usage de diagrammes d’activités, de scénarios et d’architectures et leurs caractéristiques pour représenter les workflows, les différentes fonctions qui le composent, l’architecture physique et les scénarios à évaluer.

La description des workflows et des scénarios à détecter (normaux et d’exception) se fera par des diagrammes d’activités et de scénarios de Gemoc.

Les artefacts associés à ces diagrammes sont équipés d’une sémantique comportementale exprimée en CCSL ce qui permet la simulation des modèles et leur analyse. L’architecture matérielle sera décrite par le diagramme d’architecture.

Les éléments actuellement présents dans les librairies Gemoc devront être étendus pour couvrir les besoins applicatifs que nous considérons.

Enfin l’allocation des traitements sur les processeurs devra être également modélisée.

A l’issue de cette étape, il sera possible de simuler dans l’outil Gemoc les différents workflows et leurs scénarios pour vérifier la consistance des scénarios.

Nous finirons par la phase finale qui permettra de valider ou non notre produit. Elle consistera à exécuter les modèles par injection de véritables données.

## 2.2 Présentation de l’outil GEMOC studio

### 2.2.1 Descriptif de l’environnement et outils utilisés

Conformément aux développements en cours, le développement et la modélisation des différents scénarios devra être réalisée dans le cadre d’une approche par IDM en utilisant l’environnement GEMOC. Cet IDE basé sur Eclipse, intègre des technologies qui permettent d’avoir une approche haut niveau des modèles (Méta modélisation) qui définit les artefacts et la structure des modèles.

SigPML sera utilisé pour écrire et modéliser nos différents scénarios.

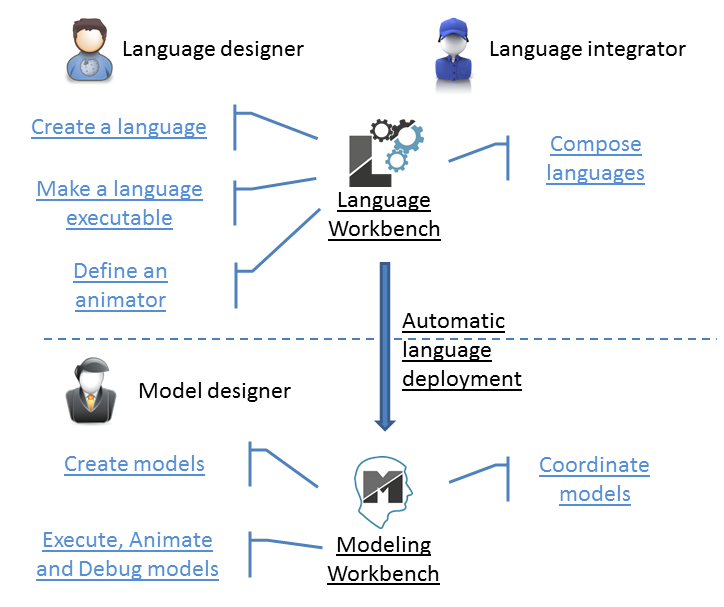
Nous utiliserons donc la partie modélisation de SigPML pour pouvoir réaliser notre projet.

### 

### 2.2.2 Environnement de développement GEMOC

GEMOC (Globalization of Modeling Languages) nous offre un environnement intégré permettant la définition de modèles graphique et/ou textuel, de langage et de leur sémantique et de leur lien avec des outils existant ; dans notre cas, la définition des différents langages de modélisation associés à différents ports pour créer directement différents flows d’événements.

GEMOC est un IDE développé par INRIA en partenariat avec plusieurs universités dans le monde. Cet outil dispose de la même interface graphique qu’Eclipse auquel il a été ajouté plusieurs plugins comme Xtext, Sirius, EMF et aussi TimeSquare ou encore SigPML.



*Figure 1 : vue d’ensemble de l’environnement de développement Eclipse réunissant les différents moyens d’utilisation de GEMOC Studio*

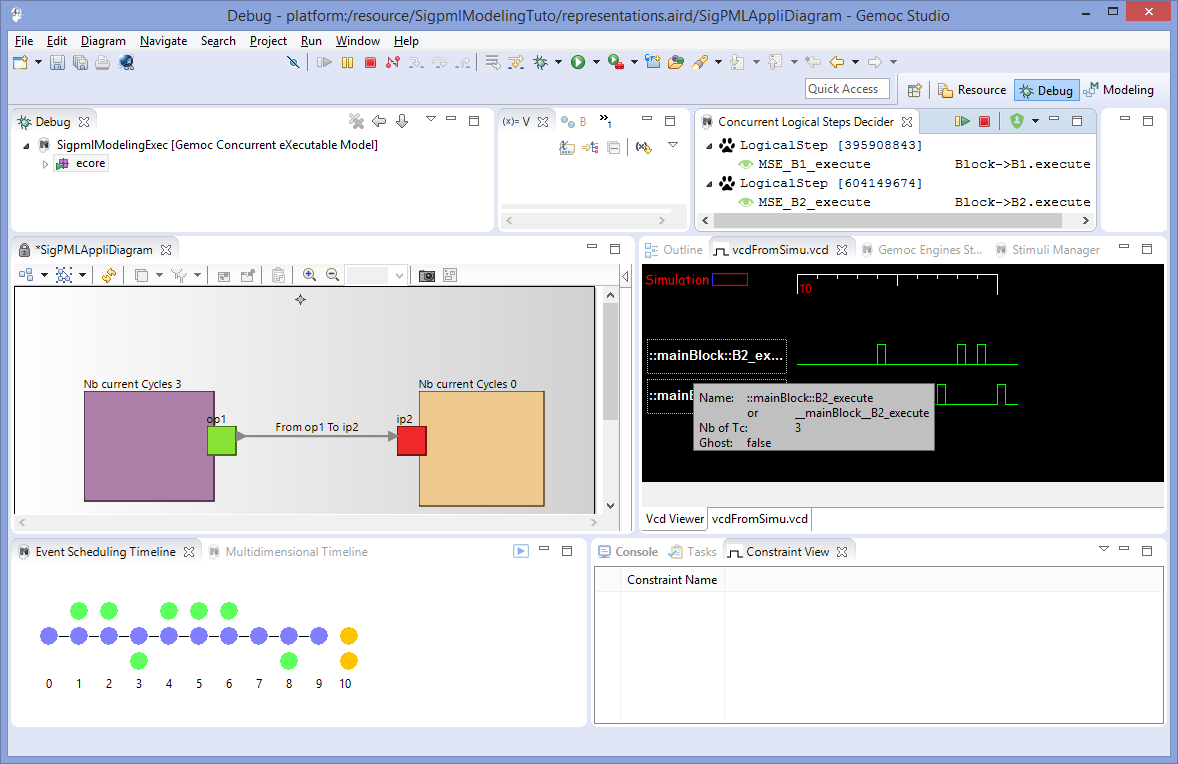
### 

### 2.2.3 SignPML - Domain Specific Modeling

SigPML est un langage de modélisation spécifique au domaine dédié au traitement des flux de données.

La syntaxe concrète est basée sur des blocs, des ports et des connecteurs qui associent des ports pour créer des flux dirigés.

La sémantique d'exécution est définit dans GEMOC qui permet ensuite de rendre le modèle exécutable et simulable Nous travaillons au niveau de la partie modeling workbench.



*Figure 2 : le résultat final de l'exécution du modèle SigPML est affiché sur l'image suivante avec un bloc de traitement pourpre qui est un bloc d'exécution et un bloc brun qui attend une synchronisation de connecteur après son accès au connecteur.*

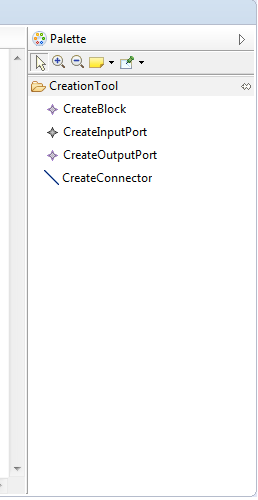
### 

### 2.2.4 Sirius

Sirius est un projet Open Source de la Fondation Eclipse. Cette technologie permet de concevoir un atelier de modélisation graphique sur mesure en s'appuyant sur les technologies Eclipse Modeling, en particulier EMF et GMF.

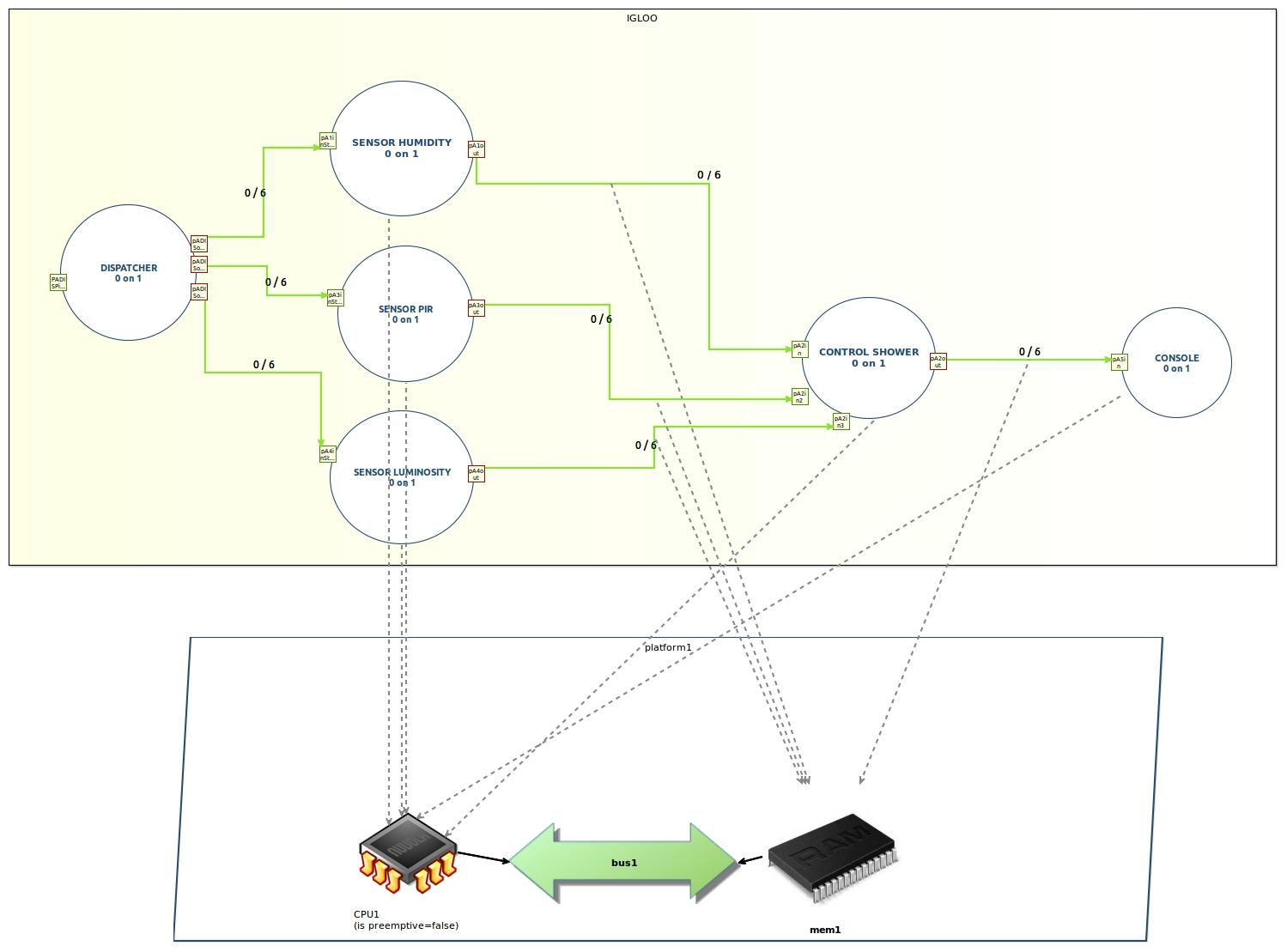
L'atelier de modélisation créé est composé d'un ensemble d'éditeurs Eclipse qui permettent aux utilisateurs de créer, éditer et visualiser des modèles EMF et GMF. Il nous permettra d’associer à des éléments de modèle une syntaxe graphique.

En effet grâce à SigPML Modeling, qui utilise le système Sirius, nous pouvons l’utiliser afin de modéliser différents capteurs, ports, etc..



*Figure 3 : exemple d’interface permettant la construction d’agent avec SigPML utilisant Sirius*

Ainsi que le montre la Figure 3, on peut créer une interface utilisateur (Palette), ce qui nous permet de modéliser des scénarios complexes plus facilement que si nous devions écrire le code.



*Figure 4 : Exemple de modélisation graphique avec SigPML utilisant Sirius*

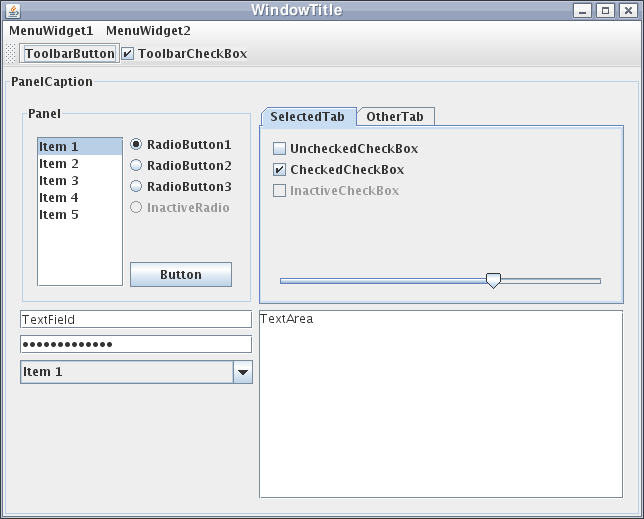
### 2.2.5 Swing (Java)

Swing est une bibliothèque graphique pour le langage de programmation Java.

Swing offre la possibilité de créer des interfaces graphiques identiques quel que soit le système d'exploitation sous-jacent.

Il utilise le principe Modèle-Vue-Contrôleur (MVC, les composants Swing jouent en fait le rôle de la vue au sens du MVC) et dispose de plusieurs choix d'apparence pour chacun des composants standards.

Pour ce projet nous avons donc utilisé cette bibliothèque compatible avec GEMOC studio.



*Figure 5 : exemple d’interface graphique possible avec Swing en Java*

# 3. Analyse des besoins

## 3.1 Expression

Conformément à la demande qui correspond aux différents scénarios à réaliser sur Gemoc Studio, nous avons dû au préalable analyser nos besoins pour en ressortir une architecture de la solution.

Puis nous avons examiné les différents composants afin de proposer la solution la plus adaptée au problème posé.

## 

### 3.2.1 Architecture de la solution : Dispatcher

Nous parlons ici de Dispatcher, un agent permettant de récolter des données brutes afin de les transformer pour les rendre compréhensibles par un système (ici, Gemoc Studio et en particulier le workbench SigPML).

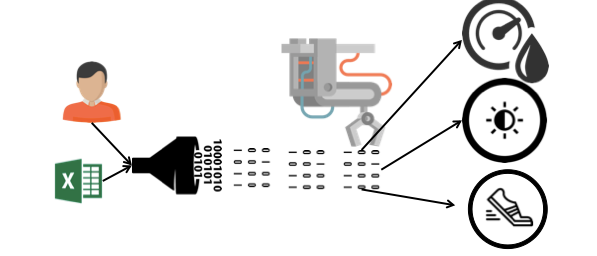
Il faut savoir que pour ce projet il y avait deux consignes ; la première étant la saisie par l’utilisateur et la seconde étant l’analyse d’un fichier brut en .csv.

Chacune des différentes méthodes pour la récupération des données devait être traitée par le Dispatcher.

Pour la première méthode, la saisie par l’utilisateur, le Dispatcher va traiter les données entrées par l’utilisateur, les convertir pour les rendre compréhensibles par le système et les envoyer dans les différents ports. Chaque port étant lié à différents capteurs, le traitement peut se faire et les différentes valeurs sont donc envoyées.

Pour la deuxième méthode, nous lisons un fichier .csv que l’utilisateur sélectionne grâce à une interface graphique dédiée.

Ensuite nous utilisons un algorithme qui nous permet de lire le fichier .csv et d’en extraire les informations nécessaires afin d’être transmises aux différents capteurs (sensor).



*Figure 6 : architecture de la solution du dispatcher*

### 

### 3.1.2.2 Architecture de la solution : Les différents sensors

L’architecture que nous proposons afin de répondre à la problématique du scénario de douche comprend 3 sensors (capteurs). Nous allons présenter un par un chacun des capteurs de l’architecture de solution.

Le premier est le capteur d’humidité. Celui-ci, comme son nom l’indique permet de capter l’humidité dans une salle. Les valeurs transmises sont comprises entre 0 et 100, sachant que 100 est la valeur maximum possible. Les valeurs transmises sont en pourcentage.

Le second est le sensor PIR, un capteur de mouvement, qui est donc capable de déterminer si une personne entre dans une salle ou non. Les valeurs sont comprises entre 0 et 1.

0 correspondant à un mouvement nul et 1 à un mouvement existant.

Le dernier est le capteur de lumière. Le capteur à des valeurs comprises entre 0 et 500 sachant qu’à partir de la valeur 120, on considère la lumière est allumée.



*Figure 7 : architecture de la solution des différents capteurs présents*

### 

### 3.1.2.3 Architecture de la solution : l’agent contrôleur

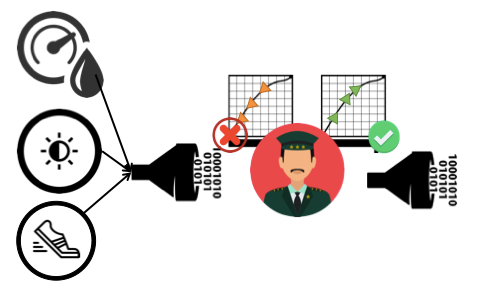
GEMOC est composé d’agents. Ces agents sont capables de s’exécuter à alpha temps dans cette architecture.

Nous avons constaté cela et nous avons imaginé un agent capable de contrôler les données émises par les différents capteurs et capable de vérifier si oui ou non une personne prend une douche.

Pour cela, nous récupérons les différentes valeurs transmises par les capteurs et, grâce à un algorithme présent dans l’agent, nous évaluons si les valeurs augmentent ou pas.

Si elles croissent, alors nous envoyons au prochain agent (l’agent qui s’occupe d’une partie de l’affichage) un réponse positive, donc 1, ou négative, 0.

De plus, l’agent, lors de la récupération des données affiche un graphique des différents sensors afin d’améliorer et de mieux transmettre les informations à l’utilisateur.



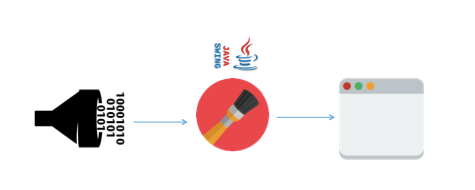
*Figure 8 : architecture de la solution de l’agent contrôleur*

### 

### 3.1.2.3 Architecture de la solution : l’agent graphique

Afin de rendre la modélisation et la compréhension plus simple pour l’utilisateur débutant, nous avons implémenté un agent utilisant la librairie de Java : Swing. En effet, nous affichons, selon la valeur envoyée par l’agent contrôlé, une fenêtre décrivant l’état de la simulation en cours (à savoir, si une personne prend sa douche ou pas).

Dans une version future, ces données pourraient être aussi envoyées à une base de données externe. Pour l’instant, l’état du projet ne permet pas encore d’évolution dans ce sens et reste locale.



*Figure 9 : architecture de la solution de l’agent graphique*

# 4. Travaux réalisés

Dans cette partie nous allons vous présenter les différents travaux réalisés lors de ce stage.

En particulier, les travaux réalisés sur la simulation d’une douche avec Gemoc, en utilisant les valeurs entrées par l’utilisateur et celle par un fichier .csv.

## 4.1 Solution de simulation de douche avec valeurs entrées par l’utilisateur

Nous basons notre simulation sur l’architecture vue précédemment.

Nous avons développé un code permettant au premier agent de demander des valeurs à l’utilisateur.

Pour cela, nous utilisons la librairie déjà présente dans java System.in.

Nous avons donc inséré dans l’agent un reader, ce qui nous permet de lire la valeur de l’utilisateur.

Ensuite, nous faisons un next.Double(), ce qui nous permet de convertir et de récupérer la valeur émise par l’utilisateur. Nous allons faire cela pour les 3 valeurs entrées par l’utilisateur. Ce qui nous permet de récupérer ces valeurs.

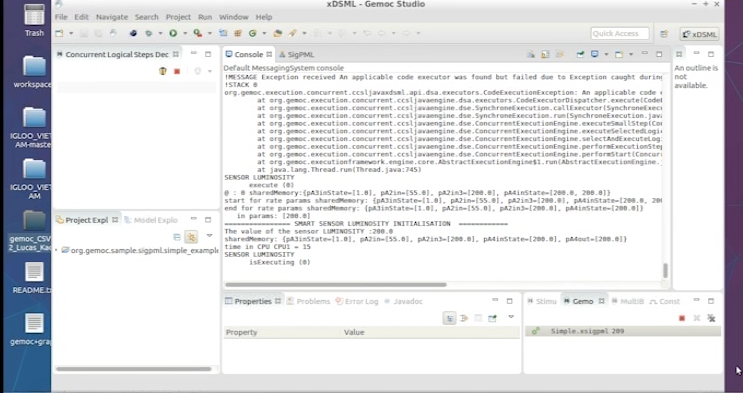
Ensuite, nous les transmettons à l’agent dispatcher qui va récupérer ces valeurs grâce à des ports directement modélisés sur l’agent. Il va s’occuper de les récupérer puis de les transmettre aux différents agents (qui correspondent à une simulation de capteur vue précédemment).

Puis, comme on peut le voir dans la figure ci-dessous, les différents agents sensor vont récupérer les valeurs par les ports, les afficher dans l’invite de commande et les transmettre au prochain agent : l’agent contrôleur.

Après avoir récupéré ces valeurs, il va les analyser pour savoir si oui ou non elles sont croissantes.

Si elle est croissante, elle enverra au port « 1 » et si elle est décroissante elle enverra « 0 ».

Ensuite, l’agent graphique affichera sur la console SHOWER ON ou SHOWER OFF, selon la valeur transmise par l’agent contrôleur.



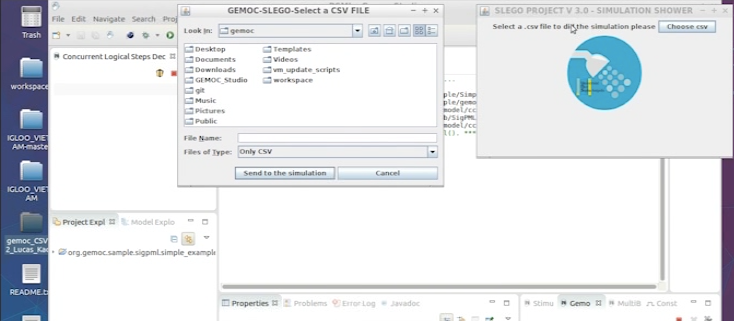
*Figure 10 : Exemple d’affichage de valeur d’un agent sensor (ici le sensor luminosity)*

## 4.2 Solution de simulation de douche avec valeur analysée par fichier .CSV

Ici, ce qui est différent par rapport à la première solution, c’est que nous récupérons les valeurs des différents capteurs en analysant un fichier .csv, défini par l’utilisateur.

Pour demander à l’utilisateur le fichier, nous utilisons une interface graphique avec Java Swing, comme le montre la Figure 11.

Une fois le fichier sélectionné par l’utilisateur, on stocke le PATH dans un document .txt, afin d’éviter tout problème avec GEMOC et que la variable soit permanente.

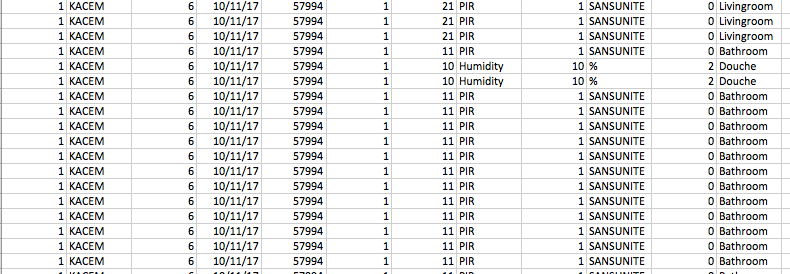


*Figure 11 : Affichage lors de la sélection de la trame MQTT par l’utilisateur.*

Puis, nous utilisons un BufferReader pour lire le premier fichier .txt afin de savoir où est le PATH du fichier .CSV que l’utilisateur veux analyser.

Une fois le fichier sélectionné, nous allons le lire et le trier pour que chaque ligne soit analysée et que les données liées aux différents capteurs soient envoyées. L’architecture de la trame MQTT (figure 12, exemple de trame MQTT) a été définie précédemment. Il nous est donc facile d’en extraire les données correspondantes.

Par la suite, nous l’analysons et nous regardons, par exemple, s’il s’agit d’un sensor Luminosity ou non et ainsi de suite, afin de trouver le bon sensor pour lui envoyer les bonnes valeurs.

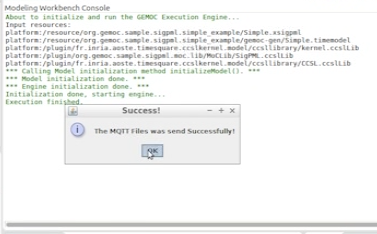


*Figure 12 : Exemple de fichier .CSV correspondant à une trame MQTT analysable par l’application réalisée.*

Comme on peut le voir dans cette figure, nous avons ajouté une sélection obligatoire de fichiers compatibles avec l’analyse (.csv).

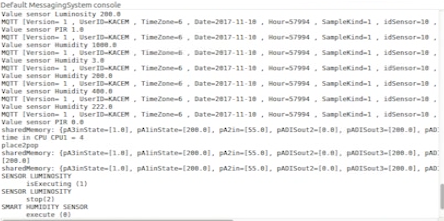
Une fois le fichier sélectionné, nous allons analyser, ligne par ligne (de 3 en 3), de façon constante, le fichier jusqu'à le trouver vide et stopper son analyse.

Une fois le fichier récupéré, nous affichons une confirmation de notre part et nous lançons le traitement de la trame MQTT, comme le montre la figure 13.



*Figure 13 : Affichage d’un message de succès après avoir récupéré le fichier à analyser*

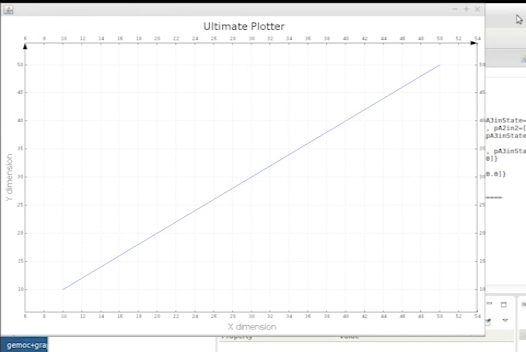
Un fois que la trame a été lue par le Reader et analysée, elle va être affichée ligne par ligne sur l’invite de commande et va ensuite envoyer au sensor correspondant les différentes valeurs par les différents ports, comme nous le montre la figure 14.



*Figure 14 : Affichage console après lecture du fichier .csv correspondant à une trame MQTT*

Puis la lecture va continuer et transférer à chaque sensor en affichant la valeur envoyée, comme on peut le voir dans la première solution (figure 10).

Après cela, il va lui-même transférer ces valeurs et le sensor dispatcher va donc vérifier si les valeurs récupérées sont croissantes et afficher, pour chaque sensor, un graphique correspondant.



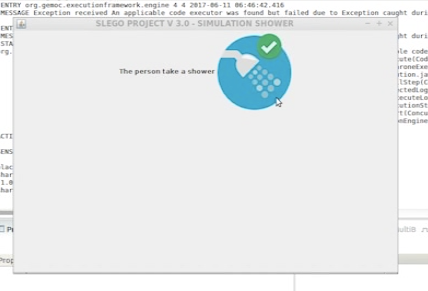
*Figure 15 : Représentation graphique de la valeur du sensor Humidity par le dispatcher*

Comme le montre cette figure pour le sensor humidity et luminosity, un graphique (modification en temps réel) va être généré et modifié au cours du temps et de l’analyse de la trame MQTT.

Ensuite, il va envoyer au sensor graphique, après avoir analysé les différentes valeurs, soit 0 (si les valeurs sont négatives) soit 1 (si les valeurs sont positives).

Puis le sensor graphique va faire un affichage graphique (comme le montre la Figure 16) et écrire si oui ou non la personne prend sa douche et cela à chacune des lignes (3 par 3).

Comme on peut le voir nous avons ajouté une image pour le rendre plus compréhensible par un utilisateur novice.



*Figure 16 : Affichage graphique après avoir reçu des valeurs positives des différents sensor, suite à l’analyse du fichier .CSV représentant une trame MQTT*

# 5. Evolution possible

## 5.1 Evolution possible : solution de simulation de douche avec valeur analysée par fichier .CSV

### 5.1.1 Evolution possible à court terme

Une implémentation rapide et qui permettrait de faire évoluer la solution serait l’affichage en mode graphique des différentes valeurs des différents sensor, comme le propose la deuxième solution (avec le fichier .csv).

De plus, lors de la sélection des données sur l’invite de commande, il faut sélectionner la première invite de commande, avant le lancement du workbench. On pourrait donc supprimer ce système et mettre dans la même console le System.In et le Sytem.out.

Enfin, plusieurs warnings dus à Gemoc sont présents dans cette même invite de commande. Nous pourrions donc concevoir une invite de commande spécifique à cette solution.

### 5.1.2 Evolution possible à moyen terme

On pourrait insérer un affichage graphique pour permettre la capture des valeurs utilisateurs. En clair, ne plus utiliser une console pour demander les valeurs de l’utilisateur mais créer une interface pour le demander.

### 5.1.3 Evolution possible à long terme

Implémentation des données sensors directement dans une base de données externe (MYSQL, par exemple) afin de les conserver et de pouvoir les traiter ultérieurement.

Utilisation de cette base de données pour la création d’un site web avec affichage graphique, permettant, par exemple, l’implémentation et la modification du contenu du site en temps réel.

# 5.2 Solution de simulation de douche avec valeur entrée par l’utilisateur

## 5.2.1 Evolution possible à court terme

L’invite de commande comporte plusieurs warnings. Nous pourrions donc concevoir une invite de commande spécifique à cette solution.

## 5.2.2 Evolution possible à moyen terme

Implémentation et configuration des données sensors directement dans une base de données externe afin de conserver et de pouvoir les traiter ultérieurement.

## 

## 5.2.3 Evolution possible à long terme

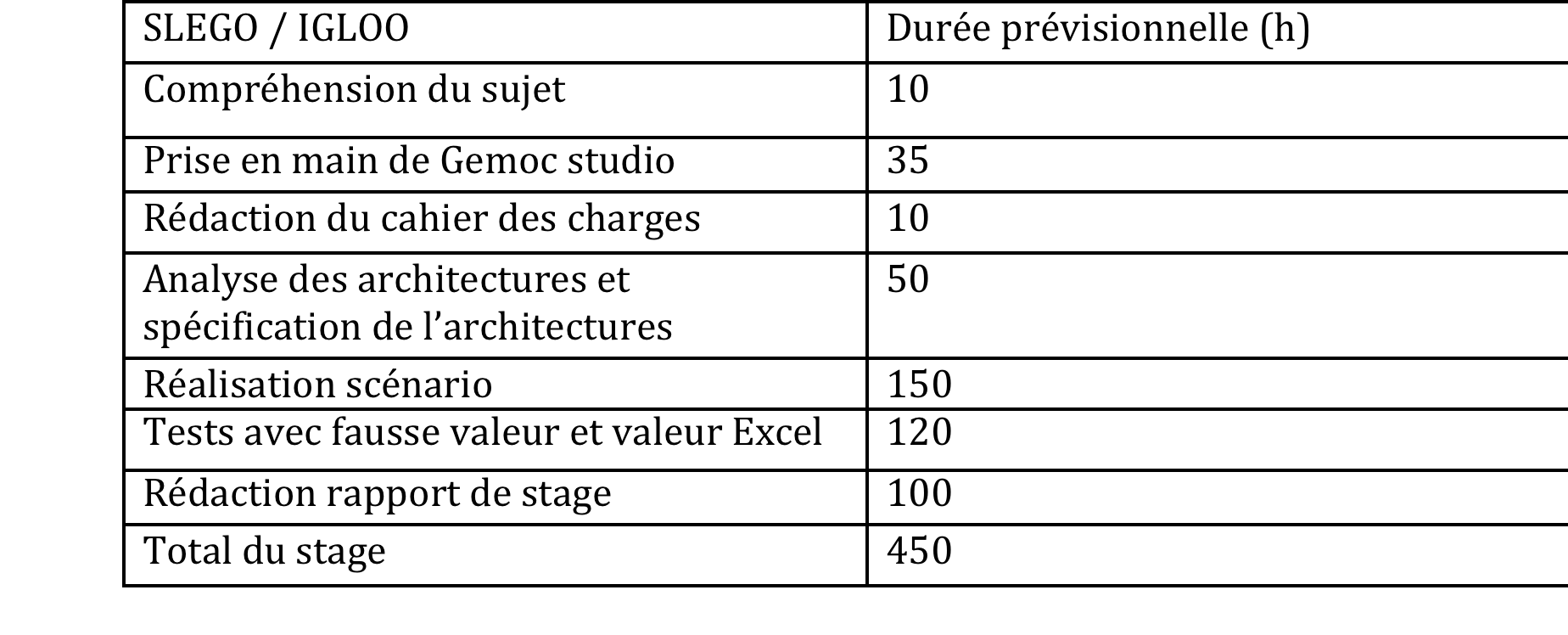
Utilisation de cette base de données pour la création d’un site web avec affichage graphique, permettant, par exemple, l’implémentation et la modification du contenu du site en temps réel.

# 6.0 Planification

## 6.1 Nombre d’heures effectuées

Ce tableau récapitule le nombre d’heures effectuées lors de ce stage.

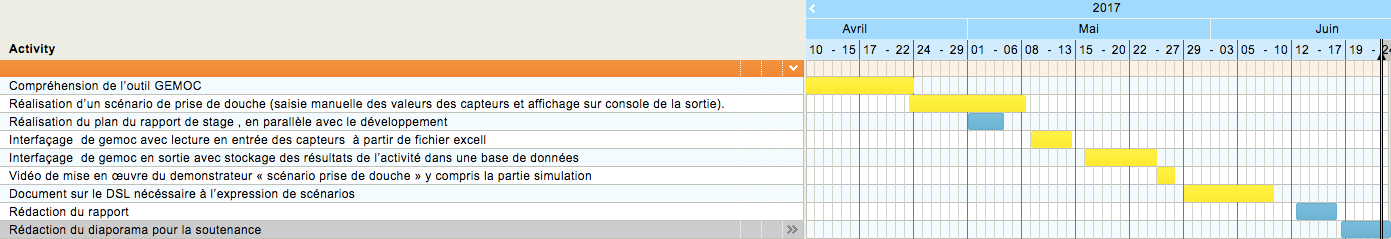
Il présente les principaux procédés réalisés :



*Figure 17 : Tableau récapitulatif du nombre d’heures effectuées*

## 6.2 Planning prévisionnel

Diagramme de gant représentant le planning prévisionnel de ce qui doit être réalisé au cours du stage :



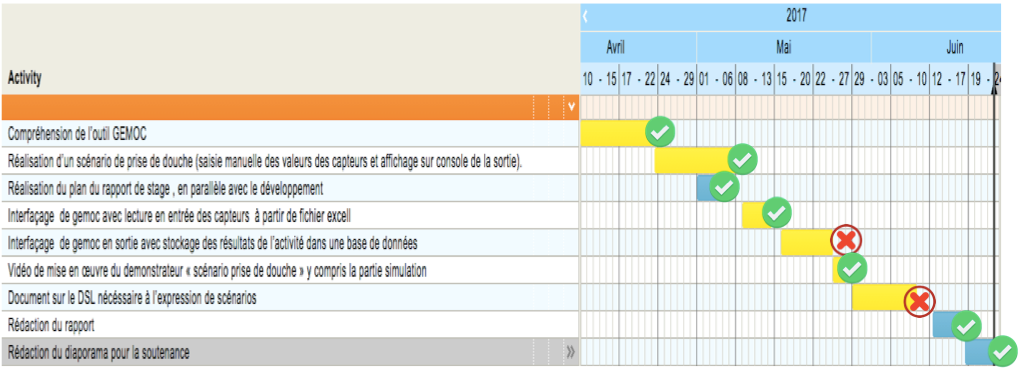
*Figure 18 : Diagramme de gant montrant les tâches à réaliser*

## 

## 6.3 Rétrospective

Suite à ce diagramme de gant, nous avons réalisé un certain nombre d’actions lors du stage.

Ce diagramme récapitule les actions réalisées ou non effectuées :



*Figure 19 : Diagramme de gant montrant les tâches réalisées*

# Conclusion

Au cours de ce stage, pour l’obtention du DUT, au sein de l’université de Da Nang, il m’a été donné pour objectif la conception et le développement d’une simulation de prise de douche sur l’outil GEMOC.

Afin d’arriver à cet objectif, j’ai dû apprendre un nouvel environnement qui est GEMOC.

La réalisation de cette simulation a été menée jusqu’au bout.

Un démonstrateur est disponible en ligne sur YouTube (voir en annexe) pour valider la réalisation.

Les performances de la simulation sont satisfaisantes.

Durant ce stage, j’ai découvert le domaine de l’électronique, les systèmes embarqués et IOT, tout en utilisant les compétences acquises pendant ma formation à l’IUT Informatique, en les mettant en application dans un milieu professionnel et scientifique.

J’ai aussi amélioré ma capacité à travailler en équipe et mon autonomie afin de faire aboutir un projet.

La thématique même de mon sujet de stage me permet maintenant de mesurer l’importance de l’élaboration et du suivi des éléments d’un cahier des charges et de mieux comprendre la thématique des objets connectés et leur importance dans notre monde actuel.

# Table des figures

Figure 1 : vue d’ensemble de l’environnement de développement Eclipse réunissant les différents moyens d’utilisation de GEMOC Studio

Figure 2 : le résultat final de l'exécution du modèle SigPML est affiché sur l'image suivante avec un bloc de traitement pourpre qui est un bloc d'exécution et un bloc brun qui attend une synchronisation de connecteur après son accès au connecteur.

Figure 3 : exemple d’interface permettant la construction d’agent avec SigPML utilisant Sirius

Figure 4 : Exemple de modélisation graphique avec SigPML utilisant Sirius

Figure 5 : exemple d’interface graphique possible avec Swing en Java

Figure 6 : architecture de la solution du dispatcher

Figure 7 : architecture de la solution des différents capteurs présents

Figure 8 : architecture de la solution de l’agent contrôleur

Figure 9 : architecture de la solution de l’agent graphique

Figure 10 : Exemple d’affichage de valeur d’un agent sensor (ici le sensor luminosity)

Figure 11 : Affichage lors de la sélection de la trame MQTT par l’utilisateur.

Figure 12 : Exemple de fichier .CSV correspondant à une trame MQTT analysable par l’application réalisée.

Figure 13 : Affichage d’un message de succès après avoir récupéré le fichier à analyser

Figure 14 : Affichage console après lecture du fichier .csv correspondant à une trame MQTT

Figure 15 : Représentation graphique de la valeur du sensor Humidity par le dispatcher

Figure 16 : Affichage graphique après avoir reçu des valeurs positives des différents sensor, suite à l’analyse du fichier .CSV représentant une trame MQTT

Figure 17 : Tableau récapitulatif du nombre d’heures effectuées

Figure 18 : Diagramme de gant montrant les tâches à réaliser

Figure 19 : Diagramme de gant montrant les tâches réalisées

# Références

[1] Guide Gemoc : <http://gemoc.org/gemoc-studio/publish/guide/html_single/Guide.html>

[2] Guide Swing de Oracle : https://docs.oracle.com/javase/tutorial/uiswing/

[3] Création de langage avec Gemoc: http://gemoc.org/gemoc-studio/publish/guide/html\_single/Guide.html#create-language-chapter

[4] Compréhension d’une trame MQTT : http://www.hivemq.com/blog/how-to-get-started-with-mqtt

[5] Création de langage executable avec Gemoc: <http://gemoc.org/gemoc-studio/publish/guide/html_single/Guide.html#make-language-executable-chapter>

[6] Edition modélisation de Gemoc : <http://gemoc.org/gemoc-studio/publish/guide/html_single/Guide.html#section-editing-model>

[7] Modélisation avec le workbench sur Gemoc : http://gemoc.org/gemoc-studio/publish/gurride/html\_single/Guide.html#modeling-workbench-executing-model-section