UNIVERSITÉ TOULOUSE III PAUL SABATIER

Université de Toulouse

MASTER DÉVELOPPEMENT LOGICIEL

iRobotSma

Clastrier & Debat

ARCHITECTURE

Recapitulatif des exigences

CODE	DESIGNATION	VALIDATION
ENF 1	De sa création jusqu'à son suicide, un agent répète un cycle d'exécution composé de trois opérations séquentielles Percevoir-Décider-Agir.	OK
ENF 2	Un agent peut créer d'autres agents.	OK
ENF 3	Un agent peut se suicider mais ne peut détruire un autre agent.	KO
ENF 4	Il doit être possible de visualiser l'exécution du système.	OK
ENF 5	Il doit être possible de visualiser l'état du système et/ou d'agents sélectionnés.	OK
ENF 6	Le système de visualisation doit être le plus découplé possible du reste du système.	OK
ENF 7	Le langage de programmation doit être Java.	OK
ENF 8	Le langage de description des composants et des assemblages doit être SpeADL.	OK
ENF 9	Vous devez quantifier le nombre de classes et de lignes de code écrites (et pas générées).	OK
ENF 10	La réutilisabilité/généricité de votre solution doit être évaluable objectivement, mesurée et la plus forte possible.	OK
ENF 11	La paramétrisation de votre solution pour le cas d'étude doit être complètement spécifiée.	OK
ENF 12	Votre architecture doit être décrite le plus précisément possible (vues C&C, module et allocation)	OK
ENF 13	Il doit être possible de contrôler l'exécution du système (mettre en pause, pas à pas , vitesse plus ou moins rapide).	OK
ENF 14	Il doit être possible d'obtenir une trace d'une exécution d'un agent sous forme de log.	KO
ENF 15	Il doit être possible de rejouer une exécution d'un scenario avec un état initial donné, y compris en présence de phénomènes aléatoires.	OK
ENF 16	Il doit être possible de persister l'état du système.	OK
ENF 17	Il doit être possible de répartir l'exécution du système.	KO

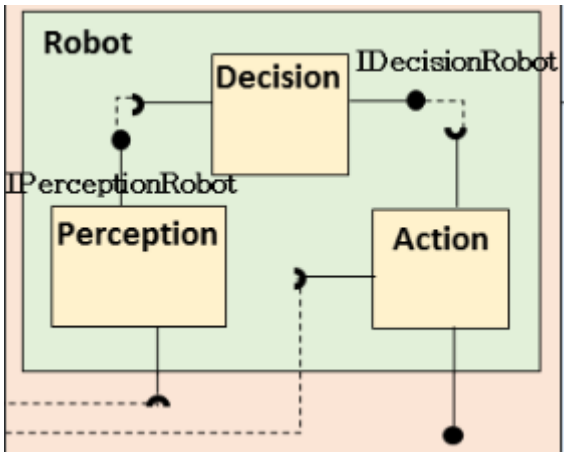
Source du projet

https://github.com/JohanD9/iRobotSMA

Exigences non fonctionnelles

ENF 1

Pour mettre en œuvre cette exigence nous avons opté pour le composant Robot soit composé de trois composants (Perception, Decision et Action).



Le Robot pourra donc agir via son composant Action, ce composant fera appel au composant décision qui fera à son tour appel au composant perception qui obtiendra, quant à lui, ses informations du composant Ihm.

Durant son cycle de vie Le composant Robot répètera à l'infini un appel à son composant Action ce qui permet d'assurer la bonne réalisation de cette exigence.

ENF 2

Cette exigence n'a pas été encore réalisée complètement puisqu'une fois le système lancé (en cours d'exécution après un clic sur le bouton Play) aucune espèce n'est créée dynamiquement.

Les espèces sont créées par leur écosystème lors de l'initialisation de notre composant Ihm. Cette création s'effectue grâce à l'interface ICreationEcosysteme.

ENF 3

Cette exigence n'a pas été traitée.

ENF 4

Pour visualiser l'exécution du système nous avons créé une interface graphique qui affichera la grille ainsi que les composants qui peuplent cette grille (Nid, Robot, Boite).
Cette interface fait partie du composant Ihm.

ENF 5

Grâce au composant Ihm, qui va lancer l'interface graphique, il sera alors possible, via cette interface, de connaitre la composition de la grille en cliquant sur les différentes cases.
Les informations apparaitront dans le coin inférieur droit de l'interface graphique.

ENF 6

Le système de visualisation et intégré à un seul composant (Ihm) qui est un simple composant et non un composant composite. De ce fait il est difficile d'isoler un peu plus l'interface graphique.

ENF 7

Le système à entièrement était réalisé en JAVA.

ENF 8

Le langage de description des composant est le speADL il est possible de consulter le fichier iRobotSma.speADL à la racine du projet.

Enf 9

Nombre de classe : 34

Nombre de ligne de code : 3500 environs

Metric	Total	Mean	Std. Dev.	Maxim...	Resource causing Maximum	Method
McCabe Cyclomatic Complexity (avg/max per		1,55	1,129	11	/iRobotSMA/src/ihm/XmlToPersist.java	saveGrilleInXmlFile
Number of Parameters (avg/max per method)		0,524	1,134	7	/iRobotSMA/src/ihm/Grille.java	move
Nested Block Depth (avg/max per method)		1,236	0,972	6	/iRobotSMA/src/ihm/XmlToPersist.java	loadGrilleFromXml
Afferent Coupling (avg/max per packageFragn		11,6	7,338	20	/iRobotSMA/src/interfaces	
Efferent Coupling (avg/max per packageFragm		7	3,847	11	/iRobotSMA/src/implementations	
Instability (avg/max per packageFragment)		0,507	0,283	1	/iRobotSMA/src/main	
Abstractness (avg/max per packageFragment)		0,343	0,43	1	/iRobotSMA/src/interfaces	
Normalized Distance (avg/max per packageFra		0,333	0,28	0,818	/iRobotSMA/src/ihm	
Depth of Inheritance Tree (avg/max per type)		0,724	1,006	6	/iRobotSMA/src/ihm/mainFraime.java	
Weighted methods per Class (avg/max per typ	592	4,813	7,059	39	/iRobotSMA/src/ihm/Case.java	
Number of Children (avg/max per type)	77	0,626	0,547	2	/iRobotSMA/speadl-gen/iRobotSMA/EcoBoite.java	
Number of Overridden Methods (avg/max per	6	0,049	0,334	3	/iRobotSMA/src/ihm/Composant.java	
Lack of Cohesion of Methods (avg/max per typ		0,182	0,302	1,033	/iRobotSMA/src/ihm/Grille.java	
Number of Attributes (avg/max per type)	196	1,593	3,782	36	/iRobotSMA/src/ihm/mainFraime.java	
Number of Static Attributes (avg/max per type	1	0,008	0,09	1	/iRobotSMA/src/ihm/Grille.java	
Number of Methods (avg/max per type)	379	3,081	3,726	16	/iRobotSMA/src/ihm/mainFraime.java	
Number of Static Methods (avg/max per type)	3	0,024	0,154	1	/iRobotSMA/src/ihm/Grille.java	
Specialization Index (avg/max per type)		0,01	0,063	0,5	/iRobotSMA/src/ihm/Composant.java	
Number of Classes (avg/max per packageFragr	123	24,6	31,96	88	/iRobotSMA/speadl-gen/iRobotSMA	
Number of Interfaces (avg/max per packageFra	62	12,4	20,175	52	/iRobotSMA/speadl-gen/iRobotSMA	
Number of Packages	5					
Total Lines of Code	3498					
Method Lines of Code (avg/max per method)	2082	5,45	19,537	298	/iRobotSMA/src/ihm/mainFraime.java	initComponents

ENF 10

Pour mettre en œuvre une généricité dans notre architecture nous avons fait en sorte que, quel que soit l'écosystème que nous souhaiterions afficher dans notre interface, chaque écosystème serait transmis à l'Ihm sous un type précis (type Composant).

L'ihm n'affichera ainsi que des composants dans la grille, peu importe s'il s'agit d'un robot ou bien d'une boîte.

Pour ce faire nous avons réalisé une interface générique (ICreationEcosystem) qui permettra de créer un composant à partir d'un écosystème avec un type défini.

L'ihm peut donc instancier des boites, des nids ou des robots grâce à cette interface. Cette solution a été mise en œuvre pour pouvoir, dans des évolutions futures, brancher d'autres écosystèmes à l'Ihm de façon transparente.

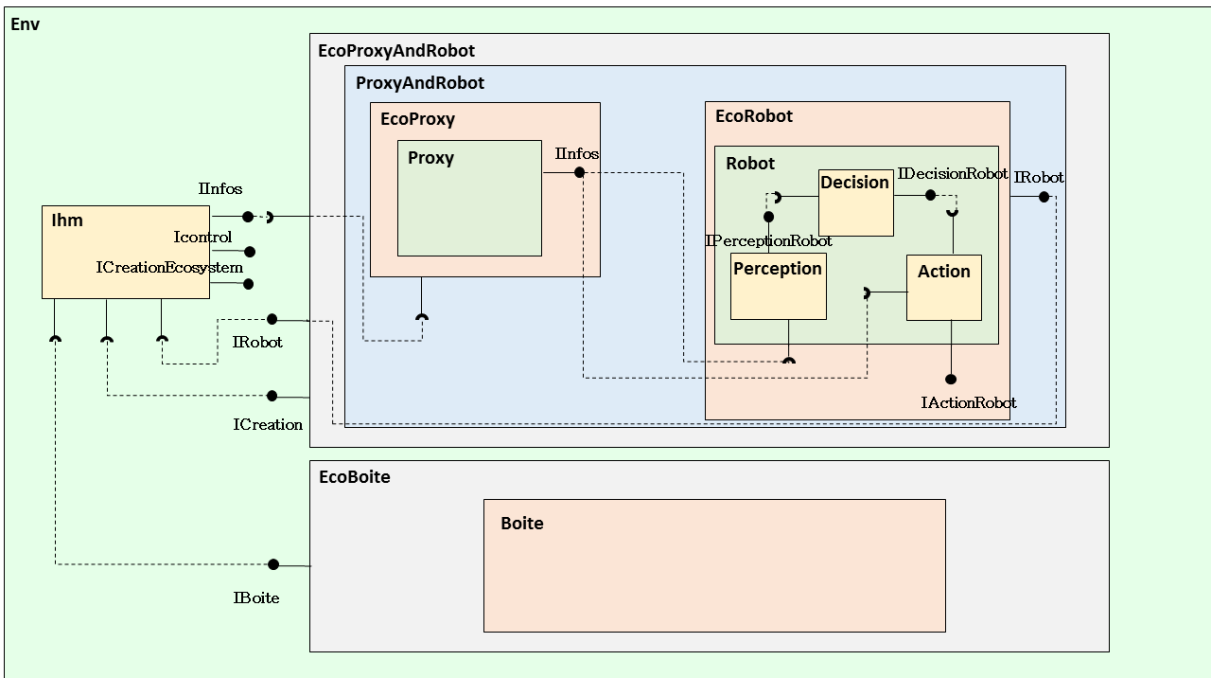
ENF 11

Lors du lancement du système et lorsque l'interface graphique apparait à l'utilisateur il est possible de choisir le nombre de boites et de robots que le système utilisera au démarrage de l'application (via le bouton Play). Cette fonctionnalité apparaitra sur la partie droite de l'interface graphique comme vous pouvez le voir dans le manuel utilisateur.

Env

L'architecture globale du système est composée des composants suivant :

- Env : ce composant et le composant qui englobe tous les autres composants du système, il englobe les composants Ihm, EcoProxyAndRobot et EcoBoite
- Ihm : ce composant nous permet de faire le lien entre le les différents agents qui seront affichés dans l'interface graphique
- EcoProxyAndRobot : ce composant contient des espèces de type ProxyAndRobot qui utilisent les écosystèmes EcoProxy et EcoRobot
    - EcoProxy contient des espèces de type Proxy qui font le lien entre le composant Ihm et les composants Robot
    - EcoRobot contient des espèces de type Robot
        - Robbot : contient les composants Perception, Decision et Action
            - Perception : permet au robot de percevoir ses alentours dans la grille
            - Decision : permet au robot de savoir quelle action il va réaliser
            - Action : permet au robot de réaliser une action
- EcoBoite : ce composant contient des espèces de type Boite



ENF 13

Sur la partie droite de l'interface graphique il est possible de sélectionner le mode pas à pas ou le mode automatique.

Dans le premier cas à chaque appui sur le bouton play chaque robot réalisera une seule action.

Dans le cas un simple appui sur le bouton play permettra de voir l'exécution totale et continue du système.

A noter qu'il est possible de passer d'un mode à l'autre une fois que le système est lancé sans aucun problème.

ENF 14

Aucun système de log n'a été mis en place jusqu'à présent.

ENF 15

Il est possible de charger un fichier XML contenant un état du système grâce au bouton prévu à cet effet dans la partie droite de l'interface graphique.

La structure de ce fichier n'est cependant pas générique elle est conforme au fichier généré par l'application pour la persistance d'un état du système.

Lors de son chargement, le fichier va initialiser la liste des composants que doit contenir la grille pour que l'état initial du système soit valide.

ENF 16

Il est possible de persister l'état du système grâce au bouton prévu à cet effet dans la partie droite de l'interface graphique.

Le système stocke alors l'état du système dans un fichier XML.

ENF 17

Cette exigence n'a pas été traitée.