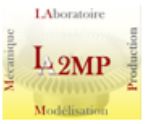
La conception bio-inspirée des systèmes de robots en essaim

Khalil Aloui








Université de Cergy-Pontoise
Université de Sfax
Thèse de Doctorat - École doctorale n○ 417

La conception bio-inspirée des systèmes de robots en essaim

Presentée par **Khalil ALOUI**

Thèse de doctorat en Génie mécanique

Soutenue à ISAE-SUPMECA le 25/01/2023

Membres du jury :

Mme. Christine PRELLE	Professeur, Université de Technologie de Compiègne	Présidente
M. Omar HAMMAMI	Professeur, ENSTA-Institut Polytechnique – Paris	Rapporteur
M. Mohamed Najib ICHCHOU	Professeur, Ecole Centrale de Lyon	Rapporteur
M. Hassen TRABELSI	Professeur, Institut Sup des Systèmes Indus de Gabes	Examinateur
M. Thierry SORIANO	Professeur, Seatech Université de Toulon	Directeur de thèse
M. Mohamed HADDAR	Professeur, ENIS, Sfax	Directeur de thèse
M. Moncef HAMMADI	Maître de Conférences – HDR, ISAE-SUPMECA, Paris	Encadrant
M. Amir GUIZANI	Maître de Conférences, Faculté des Sciences de Gafsa	Encadrant

À mes parents, mes premiers modèles et mes premiers soutiens...

Résumé

Titre : **La conception bio-inspirée des systèmes de robots en essaim.**

Les robots en essaim sont considérés comme des systèmes mécatroniques complexes caractérisés par de nombreux composants indépendants dont les actions de bas niveau produisent des résultats collectifs de haut niveau. La prévision de résultats de haut niveau (niveau système en essaim) à l'aide de règles de bas niveau (niveau des robots indépendants) constitue un défi majeur connu et ouvert. De même, trouver des règles de bas niveau donnant des résultats spécifiques de haut niveau, est en général encore moins compris. Les concepteurs trouvent ainsi des difficultés pour résoudre des problèmes complexes exploitant le parallélisme et l'auto-organisation. Dans cette thèse, nous avons développé une méthodologie de conception basée sur la méthode d'ingénierie des systèmes basée sur des modèles (MBSE) pour spécifier les exigences et les comportements collectifs des essaims, puis sur la vérification des modèles développés et enfin sur la validation du système en essaim par prototypage physique avec des robots réels en utilisant Robot Operating System (ROS). Cette méthodologie comprend deux phases : une phase descendante basée sur la méthode MBSE, de la spécification des exigences à la modélisation fonctionnelle et structurelle basée sur SysML et le langage spécifique au domaine (DSL), et une phase ascendante pour l'intégration du modèle et l'implémentation dans ROS. Pour valider notre méthodologie, nous l'avons appliquée à trois études de cas différentes, un cas d'agrégation d'un système robotique en essaim, un cas de conception d'un système de véhicules guidés automatisés (AGV) et enfin un cas de Multi-SLAM dans un environnement industriel.

Mots clés : Robotique en essaim, Méthode MBSE, Méthodologie de conception, Robot Operating Systems (ROS), Modélisation avec SysML/DSL, Agrégation, Multi-SLAM.

Title : **Bio-inspired design of swarm robot systems**

Swarm robots are considered as complex mechatronic systems characterized by many independent components whose low-level actions produce collective high-level results. Predicting high-level results (swarm system level) using low-level rules (independent robots level) is a major known and open challenge. Similarly, finding low-level rules that give specific high-level results is generally even less understood. Designers thus find it difficult to solve complex problems exploiting parallelism and self-organization. In this thesis, we have developed a design methodology based on the Model-Based Systems Engineering method (MBSE) to specify the requirements and the collective behaviors of the swarms, then on the verification of the developed models and finally on the validation of the swarm system by physical prototyping with real robots using Robot Operating System (ROS). This methodology includes two phases : a top-down phase based on the MBSE method, from the requirements specification to the functional and structural modeling based on SysML and the Domain Specific Language (DSL), and a bottom-up phase for the model integration and implementation in ROS. To validate our methodology, we applied it to three different case studies, a case of aggregation of a swarm robotic system, a case of design of an Automated Guided Vehicle (AGV) system and finally a case of Multi-SLAM in an industrial environment.

Keywords : Swarm Robotics, MBSE Method, Design Methodology, Robot Operating Systems (ROS), Modeling with SysML/DSL, Aggregation, Multi-SLAM.

Remerciements

Les travaux présentés dans cette thèse ont été réalisés en cotutelle entre le laboratoire QUARTZ à l'Institut Supérieur de Mécanique de Paris (ISAE-SUPMECA) et le Laboratoire de Mécanique, Modélisation et Productique (LA2MP) à l'Ecole Nationale d'Ingénieurs de Sfax (ENIS).

Tout d'abord, j'exprime ma sincère reconnaissance aux Professeurs Thierry SORIANO et Mohamed HADDAR, mes directeurs de thèses. Je les remercie pour m'avoir fait confiance et m'avoir accueilli dans leurs laboratoires et pour leurs multiples aides. J'ai tiens à exprimer ma sincère gratitude à mon co-encadrant Professeur Moncef HAMMADI pour m'avoir aidé durant cette thèse, pour ses suivis méthodiques de l'avancement de mes travaux de recherche, ses conseils avisés ainsi que sa rigueur scientifique. Je ne le remercierai jamais assez. Je remercie également mon encadrant Amir GUIZANI de m'avoir aidé durant la thèse.

Je remercie vivement les rapporteurs Omar HAMMAMI et Mohamed Najib ICHCHOU d'avoir accepté la lourde tâche d'évaluer mon manuscrit de thèse. Je remercie également le Professeur Mme Christine PRELLE de me faire l'honneur de présider mon jury de thèse et le Professeur Hassen TRABELSI d'avoir accepté d'être membre du jury de cette thèse.

Mes sincères remerciements à tout le personnel du laboratoire QUARTZ et laboratoire LA2MP qui m'ont aidé à mener à bien mes travaux de thèse par leur bonne humeur, leur encouragement et leur soutien durant toutes ces années.

Enfin, je dédie ce travail à l'âme de mon grand père (Amor), mes grandes mères (Fatma, Mbarka) et pour toute ma famille sans qui je ne serais jamais arrivé là.

Remercier est le début d'une grande sagesse. En remerciant, on désavoue à la fois l'ignorance et l'arrogance.

– Donachy Ladouceur

Table des matières

Table des figures

Introduction Générale

Les progrès rapides des technologies de fabrication et des applications dans les industries augmentent la productivité. L'industrie d'aujourd'hui a besoin de la numérisation et de l'intelligence des processus de fabrication [1]. L'industrie 4.0 représente la quatrième révolution industrielle qui se définit comme un nouveau niveau d'organisation et de contrôle de l'ensemble de la chaîne de valeur du cycle de vie des produits ; elle est orientée vers les besoins des clients. L'industrie 4.0 est un concept réaliste qui comprend l'internet industriel, l'internet des objets et la fabrication intelligente. Ces dernières années, les robots collaboratifs sont devenus l'un des principaux moteurs de l'industrie 4.0. En effet, la robotique collaborative a été considérée comme l'une des technologies habilitantes de la quatrième révolution industrielle, dans le cadre du programme Industrie 4.0. L'introduction de tels systèmes robotiques dans les applications industrielles pose plusieurs problèmes qui ne peuvent être ignorés.

Par rapport aux robots industriels, les systèmes robotiques en essaim sont plus productifs, flexibles, polyvalents et plus sécurisés. Ils sont utilisés dans l'usine intelligente pour plusieurs applications telles que le transport de marchandises, la surveillance et l'inspection. De nos jours, l'utilisation des systèmes robotiques en essaim dans la logistique industrielle n'est pas encore très répandue dans les usines de fabrication. Pour cette raison, de nombreux fabricants et développeurs de robots industriels se sont lancés dans l'industrie des robots collaboratifs en essaim pour réduire ces problèmes. Cependant, il est nécessaire de souligner les principaux obstacles qui découragent aujourd'hui le secteur industriel d'intégrer les systèmes robotiques en essaim dans leurs lignes de production. La robotique en essaim a été définie comme "a novel approach to the coordination of large numbers of robots" et comme "the study of how large numbers of relatively simple physically embodied agents can be designed such that a desired collective behavior emerges from the local interactions among agents and between the agents and the environment." [122]. Les systèmes robotiques en essaim sont des systèmes complexes. Ils présentent une dynamique à deux niveaux différents : le niveau collectif, ou macroscopique, et le niveau individuel, ou microscopique. Le comportement collectif est le résultat des interactions des robots individuels entre eux et avec l'environnement. Afin d'obtenir un comportement collectif souhaité, les comportements individuels et les interactions des robots doivent être soigneusement conçus. Cependant, ce processus de conception n'est généralement pas trivial, car la dynamique des systèmes complexes est très souvent difficile à prévoir.

La robotique en essaim (Swarm robotics) est une approche de coordination d'un grand nombre de robots utilisant des contrôleurs de comportements collectifs. Le paradigme de l'intelligence bio-inspirée des robots en essaim s'est avéré avoir des propriétés très intéressants telles que la robustesse, la flexibilité et la capacité à résoudre des problèmes complexes exploitant le parallélisme et l'auto organisation. Comme le montre la Figure 1, les robots en essaim sont considérés comme des systèmes mécatroniques complexes caractérisés par de nombreux composants indépendants dont les actions de bas niveau produisent des résultats collectifs de haut niveau. La prévision de résultats de haut niveau (niveau système en essaim) à l'aide de règles de bas niveau (niveau des robots indépendants) constitue un défi majeur connu et ouvert. De même, trouver des règles de bas niveau donnant des résultats

spécifiques de haut niveau, est en général encore moins compris. Les concepteurs trouvent ainsi des difficultés pour résoudre des problèmes complexes exploitant le parallélisme et l'auto-organisation.

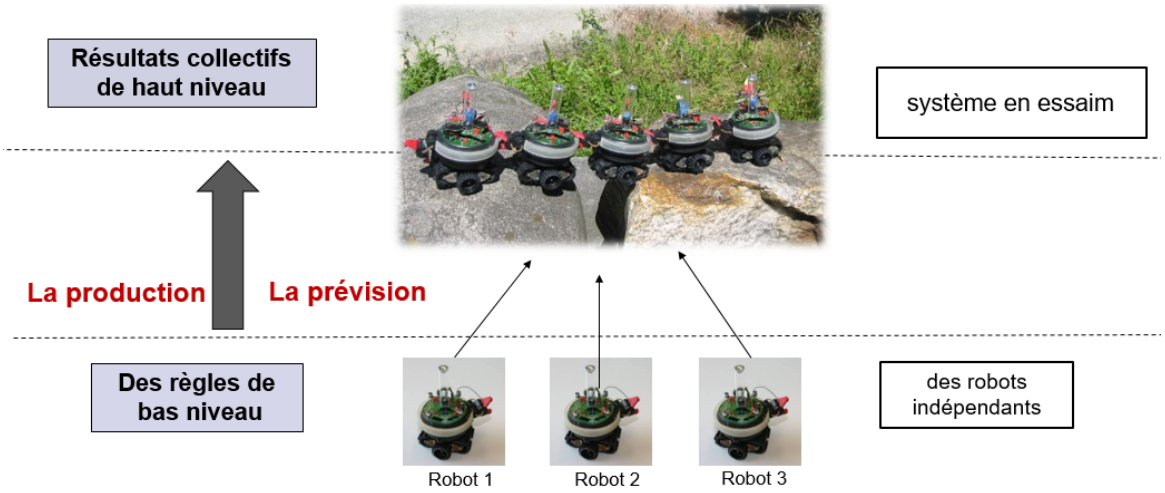


Figure 1 – La problématique de conception d'un système robotique en essaim

Dans cette thèse, nous traitons de deux problèmes fondamentaux des systèmes robotiques en essaim ; l'un purement industriel lié aux problèmes de modélisation, de simulation et d'implémentation des systèmes robotiques en essaim et l'autre scientifique lié aux méthodologies de conception et à la démarche à suivre pour concevoir ce type des robots.

1) Problématique industrielle :

Dans la robotique en essaim, le comportement des essaims émergeant d'interactions locales reste difficile à prévoir. Lorsque nous tentons de concevoir des systèmes robotiques en essaim pour des applications industrielles, nous sommes confrontés à un large éventail de défis logiciels et matériels tels que les communications entre les différents membres de l'essaim, l'intégration des systèmes de robots en essaim dans l'industrie, le problème de la consommation d'énergie, etc.

Le premier défi est lié à la communication. En général, les robots en essaim utilisent la technologie de communication sans fil pour échanger des informations entre ses membres mobiles. Cela signifie que tout dépend de la qualité et de la robustesse de la connexion WiFi. Cependant, avec l'arrivée de la technologie 5G, la connectivité sera théoriquement 10 fois plus élevée, et donc la 5G favorisera l'interconnexion de tous les réseaux et technologies existants, et leur permettra de communiquer entre eux plus efficacement. Le deuxième défi à l'intégration des systèmes des robots en essaim dans l'industrie et notamment dans les usines intelligentes est le problème de faire fonctionner ensemble des robots construits par différents fabricants. En effet, les différents systèmes des robots en essaim ne parlent pas vraiment le même langage. Donc, le bon choix de la méthodologie et des outils de développement des robots en essaim pourrait faciliter la collaboration de ces systèmes de différents fabricants. Les obstacles susmentionnés à l'intégration des systèmes robotiques en essaim dans les usines intelligentes peuvent être limités en adoptant la bonne méthodologie et les bons outils de développement. Cependant, il existe de nombreux défis à relever pour assurer

un développement efficace de ces systèmes. Par exemple, la complexité et la discontinuité du processus de conception sont des problèmes majeurs rencontrés par les concepteurs qui ralentissent l'intégration des robots en essaim dans les usines intelligentes. Le troisième défi est lié à l'énergie, car l'ensemble du système peut s'arrêter si les sources d'énergie sont épuisées.

2) Problématique Scientifique :

Dans le domaine de la robotique en essaim, il n'existe toujours pas de moyens formels ou précis de concevoir des comportements individuels qui produisent le comportement collectif souhaité. L'intuition du concepteur humain reste le principal ingrédient du développement des systèmes de robotique en essaim. La modélisation du système robotique en essaim représente un grand défi lié à la conception. En effet, la modélisation des essaims comprend plusieurs aspects tels que la cinématique non linéaire, le comportement dynamique du mouvement, le contrôle du système et la coordination pour évaluer les positions et les orientations des robots. En d'autres termes, il doit y avoir une continuité entre les données et les modèles utilisés tout au long du processus de conception pour éviter le risque d'erreurs et de reprises. En outre, le concepteur doit vérifier toutes les intégrations nécessaires (intégrations fonctionnelles, physiques et logicielles) pour garantir la cohérence de la conception. Les méthodologies de conception des systèmes robotiques en essaim doivent également offrir la possibilité d'automatiser le processus de vérification des exigences de conception, en particulier les exigences liées aux possibilités de contrôle décentralisé permettant de prédire la décision collaborative du système. Il est nécessaire de spécifier les exigences de conception d'un système robotique en essaim, de déterminer le comportement de chaque robot en fonction du comportement de l'essaim et de programmer ce comportement sur une plateforme logicielle. Des outils logiciels appropriés sont nécessaires pour maîtriser la complexité de la modélisation des systèmes en essaim. Par exemple, l'ingénierie des systèmes basée sur le modèle ou Model-Based Systems Engineering (MBSE) est l'application formalisée de la modélisation pour soutenir les différentes étapes de l'évolution du système, de la phase de conception à toutes les phases du cycle de vie qui suivent. Malgré les capacités de spécification offertes par SysML et les moyens de créer des relations de traçabilité entre les différents niveaux d'abstractions de modélisation, l'inconvénient majeur des outils basés sur SysML est les capacités de simulation limitées pour vérifier le développement. Il est donc nécessaire de combiner les outils de modélisation SysML avec d'autres outils d'intégration, de vérification et de validation. L'intégration, la vérification et la validation (IVV) sont des étapes décisives pour le développement de systèmes complexes. Dans le cas des systèmes robotiques en essaim, il est nécessaire de disposer d'une plateforme logicielle capable d'intégrer tous les codes logiciels liés aux composants matériels et de vérifier la coordination de tous les membres du système en essaim. Une telle plateforme logicielle doit pouvoir vérifier par simulation les actions de gestion assurant un mouvement sans conflit pour mettre en œuvre un système de navigation répondant aux exigences d'un système en essaim. Pour ce faire, l'environnement de simulation doit être capable de localiser les membres de l'essaim, de définir l'environnement et de planifier des trajectoires optimales dans l'environnement. De plus, pour permettre la communication avec les capteurs et les actionneurs, une couche d'abstraction matérielle est nécessaire.

3) L'objectif global de la thèse :

L'objectif de cette thèse est de développer une approche systématique bien fondée pour la spécification, la modélisation, la conception, la réalisation, la vérification, la validation et l'exploitation d'un système robotique en essaim. À ce but, nous présentons la principale contribution de cette thèse, à savoir le développement d'une nouvelle méthodologie pour la conception de systèmes robotiques en essaim. Cette méthodologie est basée sur la méthode MBSE utilisant les diagrammes SysML pour spécifier les exigences de l'essaim et identifier les différentes fonctions et comportements. Et finalement, nous utilisons Robot Operating System (ROS) pour simuler le système et l'implémenter sur des vrais robots.

Ce rapport est composé de 6 chapitres : Dans le premier chapitre, nous développons un état de l'art sur les robots en essaim, leurs caractéristiques, leurs avantages par rapport aux robots individuels et leurs comportements collectifs. Dans le deuxième chapitre, nous définissons dans une première partie, l'ingénierie en essaim, les types de modélisation et les différentes plateformes de simulation et d'implémentation de ces types de systèmes. Nous présentons dans la deuxième partie, quelques méthodes de conception de systèmes robotiques en essaim, les forces et les faiblesses de chaque méthode. Dans le troisième chapitre, nous présentons notre contribution à cette thèse ; il s'agit d'une méthodologie de conception développée pour la conception de robots en essaim basée sur l'ingénierie des systèmes, la méthode MBSE et l'implémentation sur ROS/ ROS2. Dans le quatrième chapitre, une étude de cas d'agrégation inspirée par les comportements collectifs des animaux sociaux a également été choisie pour valider et assurer la continuité de la méthodologie développée. Dans le cinquième chapitre, nous appliquons cette méthodologie à la conception d'un système des véhicules à guidage automatiques AGVs pour assurer la continuité de la méthodologie depuis la spécification des exigences du système, la modélisation, la simulation jusqu'à l'implémentation. Nous validons cette méthodologie également dans le dernier chapitre par une application de la technologie multi-SLAM dans un environnement industriel.

Première partie

L’état de l’art

Chapitre 1

Introduction à la robotique en essaim

La créativité individuelle peut-elle séduire l'intelligence du groupe ?

– Carl de Souza

1.1 Introduction

La recherche sur la robotique en essaim (RS) est née récemment de l'application des concepts d'intelligence en essaim aux systèmes multirobots, qui se concentre sur la concrétisation physique et les interactions réalistes entre les individus eux-mêmes ainsi qu'entre les individus et l'environnement. Le terme "essaim" fait référence à un grand groupe d'individus interagissant localement et ayant des objectifs communs. Il est utilisé pour décrire tous les types de comportements collectifs, même s'il est associé à des mouvements conjoints dans l'espace. L'intelligence en essaim est l'intelligence collective qui émerge des interactions entre de grands groupes d'individus autonomes. Ce terme a été utilisé pour la première fois par Beni et Wang [17] pour décrire un type particulier de système robotique cellulaire.

La robotique en essaim est une ancienne forme de multirobotique dans laquelle un grand nombre de robots sont coordonnés de manière distribuée et décentralisée. Elle est basée sur l'utilisation de règles locales, et de robots simples par rapport à la complexité de la tâche à accomplir, et s'inspire des insectes sociaux. Un grand nombre de robots simples peuvent accomplir des tâches complexes de manière plus efficace qu'un seul robot, ce qui confère robustesse et flexibilité au groupe.

Les principes de l'intelligence en essaim ont été largement étudiés et appliqués à un certain nombre de tâches différentes où un groupe de robots autonomes est utilisé pour résoudre un problème avec une approche distribuée, c'est-à-dire sans coordination centrale. Dans ce chapitre, nous définissons le concept de robots en essaim. Nous présentons ensuite

les avantages, les inconvénients et les comportements collectifs de ces types de robots.

1.2 Intelligence en essaim

1.2.1 Définition

En 1989, Beni et Wang ont utilisé le terme intelligence d'essaim dans le domaine des systèmes robotiques cellulaires pour définir une collection de robots autonomes, non synchronisés et non intelligents coopérant pour atteindre un objectif global, tandis que dans le domaine de l'intelligence artificielle, elle était définie comme le comportement collectif de systèmes décentralisés et auto-organisés (naturels ou artificiels)[17]. Autrement dit, la robotique en essaim est définie comme un ensemble de nombreux individus simples qui interagissent avec d'autres individus et l'environnement en utilisant une gestion décentralisée et auto-organisée pour atteindre leurs objectifs[38]. Bonabeau et al.[19]ont étendu leur définition pour inclure toute tentative de concevoir des algorithmes ou des dispositifs distribués de résolution de problèmes inspirés par le comportement collectif des colonies d'insectes sociales et d'autres sociétés animales. L'intelligence en essaim selon Charrier[35]est un domaine hautement bio-inspiré dont l'objectif est de modéliser au moyen de systèmes multi-agents, les mécanismes d'auto-organisation et d'adaptation observés dans les organismes vivants. Ces modèles donnent finalement naissance à des algorithmes qui simulent des phénomènes naturels dans des logiciels (sur un ordinateur) ou du matériel (avec des robots), ou servent de méta-heuristiques aux problèmes de l'intelligence artificielle. Li et Clerc ont défini l'intelligence en essaim comme la capacité de résolution de problèmes qui émerge dans les interactions de simples unités de traitement de l'information. Les unités de traitement de l'information qui composent un essaim peuvent être animées, mécaniques, informatiques ou mathématiques ; ils peuvent être des insectes, des oiseaux ou des êtres humains ; ils peuvent être réels ou imaginaires. Leur couplage peut avoir un large éventail de caractéristiques, mais il doit y avoir interaction entre les unités [82].

1.2.2 Motivation et inspiration

L'intelligence en essaim est une branche des techniques modernes d'intelligence artificielle étudiées et inspirées par les comportements collectifs des insectes sociaux, des animaux et des sociétés humaines [82], et traite de la conception de systèmes multi-agents tels que l'optimisation et la robotique [35]. Puisque l'intelligence est une entité collective et non une entité isolée, l'intelligence en essaim peut donc être considérée comme un concept plus large d'intelligence car elle met l'accent sur le fait que l'intelligence doit être modélisée dans un contexte social, en raison de l'interaction les uns avec les autres [82].

L'intelligence en essaim s'inspire souvent de systèmes biologiques tels que les colonies d'insectes [113], [96], les volées d'oiseaux [89], les bancs de poissons [6], les groupes d'amibes [31], les colonies de bactéries [114], et les cellules du corps humain ou animal [87]. L'inspiration est tirée de la nature parce qu'il a été démontré que l'étude des systèmes naturels favorise le développement de nouveaux ensembles de règles qui peuvent être utilisées

pour résoudre des problèmes difficiles qui pourraient être impossibles à résoudre avec les techniques traditionnelles [130]. Un autre avantage est que les nouvelles théories peuvent être étudiées, testées et mises à jour en les comparant directement à la source d'inspiration [2]. Quelques exemples de systèmes qualifiés d'intelligents en essaim sont présentés sur la Figure 1.1.

Figure 1.1 – La bio-inspiration de l'intelligence en essaim.

Les comportements collectifs des insectes sociaux, tels que la danse de l'abeille, la construction du nid de la guêpe, la construction de la termitière, ou le suivi des traces des fourmis, ont longtemps été considérés comme des aspects étranges et mystérieux de la biologie. Les chercheurs ont démontré au cours des dernières décennies que les individus n'ont pas besoin de représentation ou de connaissances sophistiquées pour produire des comportements aussi complexes [51]. Chez les insectes sociaux, les individus ne sont pas informés de l'état global de la colonie. Il n'existe pas de leader qui guide tous les autres individus afin d'accomplir leurs objectifs. La connaissance de l'essaim est distribuée à travers tous les agents, où un individu n'est pas capable d'accomplir sa tâche sans le reste de l'essaim. Les insectes sociaux sont capables d'échanger des informations et, par exemple, de communiquer la localisation d'une source de nourriture, d'une zone favorable à la recherche de nourriture ou la présence d'un danger pour leurs compagnons. L'interaction entre les individus est basée sur le concept de localité, où il n'y a aucune connaissance de la situation globale. La communication implicite par le biais des modifications apportées à l'environnement est appelée stigmergie [59, 106]. Les insectes modifient leurs comportements en fonction des changements précédents effectués par leurs compagnons dans l'environnement. Cela peut être observé dans la construction du nid des termites, où les changements de comportement des ouvriers sont déterminés par la structure du nid [19]. L'organisation émerge des interactions entre les individus et entre les individus et l'environnement. Ces interactions se propagent dans toute la colonie et celle-ci peut donc résoudre des tâches qui ne pourraient être résolues par un seul individu. Ces comportements collectifs sont ap-

pelés comportements d'auto-organisation. Les théories de l'auto-organisation, empruntées aux domaines de la physique et de la chimie, peuvent être utilisées pour expliquer comment les insectes sociaux présentent un comportement collectif complexe qui émerge des interactions d'individus se comportant simplement [19]. L'auto-organisation repose sur la combinaison des quatre règles de base suivantes : rétroaction positive, rétroaction négative, caractère aléatoire et interactions multiples [19].

1.2.3 Transferts d'information dans les systèmes d'intelligence en essaim

Dans la nature, les interactions représentent le canal de communication entre les insectes et entre eux et l'environnement, ce qui signifie que chaque société a sa propre manière d'interaction. Les interactions permettent aux insectes de partager et d'obtenir des informations sur les conditions environnementales et des colonies. Il existe deux types de communication [38], [50] :

Transfert direct d'information : Consiste en une communication locale où aucune modification de l'environnement n'est nécessaire. Les informations échangées par le biais d'interactions directes peuvent être de différents types, tels que le contact physique, l'échange de fluides, les signes visuels et acoustiques... etc. de sorte que l'information circule sous plusieurs formes : visuelle, tactile ou auditive, mais elle ne perdure pas dans l'environnement.

Transfert indirect d'information : Consiste en une communication entre les insectes médiés par l'environnement. Certains individus modifient l'environnement et d'autres perçoivent cette modification, ajustant leurs comportements en conséquence. Cette réaction est un exemple de stigmatisation. Ce terme signifie la forme de communication indirecte dans laquelle chacun travaille sur l'environnement et d'autres individus qui découvrent certains changements dans l'environnement interagissent avec la stimulation. Ce processus conduit à une coordination quasi complète du travail d'équipe et peut nous donner l'impression que la colonie suit un plan précis [38], [50].

De nombreux comportements d'auto-organisation dépendent des interactions directes entre les membres du groupe. Par exemple, des volées d'oiseaux se déplacent de manière cohérente et changent de direction soudainement mais simultanément, c'est le résultat d'une forte simulation comportementale couplée à un transfert direct d'informations [38], [50].

1.2.4 Fonctions assurées par les systèmes d'intelligence en essaim

Dans un contexte biologique, les processus d'auto-organisation remplissent souvent des fonctions qui permettent au groupe ou à la société animale d'appréhender les divers éléments de son environnement. Il est possible de regrouper ces fonctions en trois catégories : coordination, collaboration et délibération.

Coordination : C'est l'organisation des tâches nécessaires dans l'espace et le temps pour résoudre un problème [50]. Cette fonction affecte la séquence temporelle et / ou la distri-

bution spatiale des activités des individus pour atteindre un objectif donné. Par exemple, la coordination du travail pendant la construction du nid chez certaines espèces de guêpes sociales ou termites. En général, la coordination de l'activité des individus est la principale conséquence des processus d'auto-organisation qui sont au cœur des systèmes d'intelligence en essaim [52].

Collaboration : La collaboration se produit lorsque des individus accomplissent ensemble une tâche qui ne peut être accomplie par un seul. Les individus doivent conjuguer leurs efforts pour résoudre avec succès un problème qui dépasse leurs capacités individuelles [51], c'est-à-dire la répartition des activités entre individus ou groupes d'individus spécialisés dans la réalisation d'une de ces activités, qui peut être établie sur la base des différences morphologiques interindividuelles : des individus ayant des capacités physiques différentes effectueront différentes tâches. Il apparaît que ce mécanisme basé sur l'expérience des individus est à l'origine de l'organisation du travail chez plusieurs espèces d'insectes sociaux [50].

Délibération : La délibération fait référence aux mécanismes qui se produisent lorsqu'une colonie fait face à plusieurs opportunités, qui fera un choix collectif ou une décision collective. Par exemple, la fourmi des espèces de « Lasius Niger » est en mesure de choisir la plus riche parmi plusieurs sources de nourriture, et parmi plusieurs chemins menant au chemin le plus court vers la source [50].

1.2.5 Principaux algorithmes issus de l'intelligence des essaims

Il existe plusieurs approches inspirées du comportement social qui sont utilisées pour résoudre différents problèmes. Prenons par exemple : l'optimisation par essaims de particules (PSO), l'optimisation par colonies de fourmis (ACO), la colonie d'abeilles artificielles (ABC), l'algorithme de foraging bactérienne (BFA), l'algorithme lucioles (FA) et l'algorithme de libellule (DA).

Optimisation par Essaims de Particules (Particle swarm optimization- PSO)

En 1995, Kennedy et Eberhart ont développé une technique de calcul évolutionnaire qu'ils ont appelée "Optimisation par essaims de particules (PSO)". Il s'agit d'une approche évolutive qui s'inspire du comportement social projeté par les individus dans un banc de poissons ou dans une volée d'oiseaux. Ce comportement social des particules individuelles est déterminé par leur propre expérience passée ainsi que par l'expérience passée de leur voisinage [110]. L'optimiseur à essaims de particules se compose d'un certain nombre de particules, qui tournent en orbite et recherchent le meilleur emplacement dans l'espace. Les individus communiquent directement ou indirectement entre eux les directions de recherche. En général, PSO est une technique de recherche simple mais performante, avec peu de paramètres à régler et facile à mettre en œuvre.

Optimisation par Colonies de Fourmis (Ant Colony Optimization- ACO)

L'Optimisation par Colonies de Fourmis (ACO) est une solution méta-heuristique permettant de résoudre des problèmes d'optimisation combinatoire difficiles. La source d'inspiration de l'ACO est le comportement de pose et de suivi de pistes de phéromones des

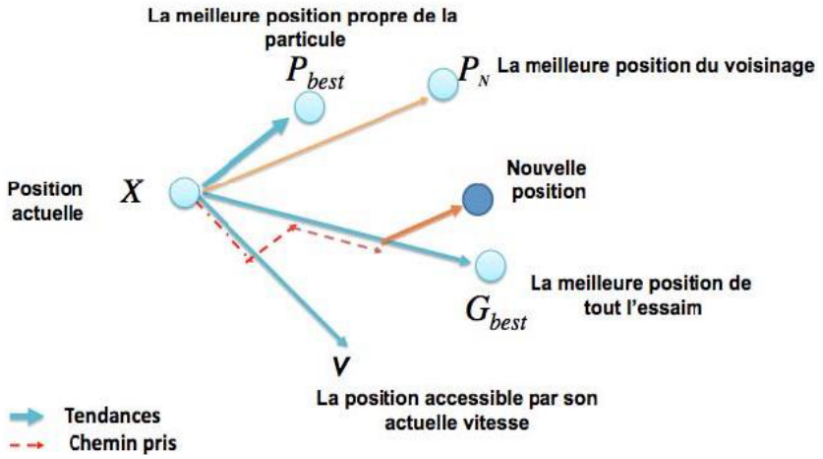


Figure 1.2 – Principe de fonctionnement du PSO [138].

fourmis réelles, qui utilisent les phéromones comme moyen de communication. Par analogie avec l'exemple biologique, l'ACO est basé sur la communication indirecte au sein d'une colonie d'agents simples, appelés fourmis (artificielles), par le biais de pistes de phéromones (artificielles). Les pistes de phéromones dans l'ACO servent d'informations numériques distribuées, que les fourmis utilisent pour construire de manière probabiliste des solutions au problème à résoudre et qu'elles adaptent pendant l'exécution de l'algorithme pour refléter leur expérience de recherche [42].

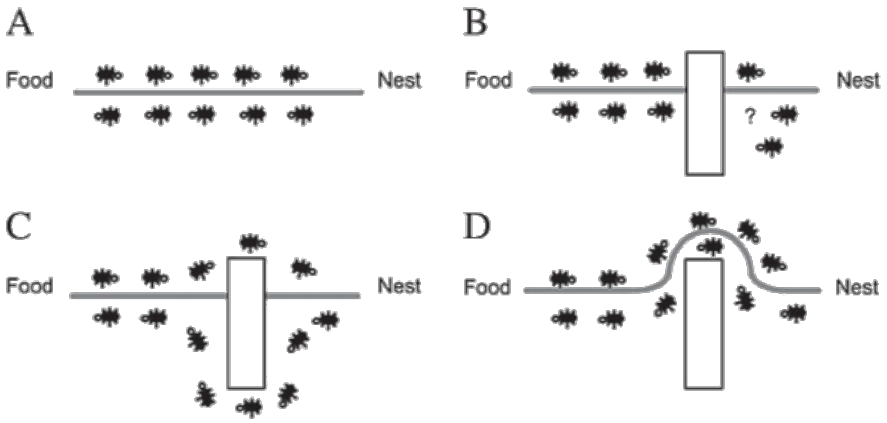


Figure 1.3 – Principe de fonctionnement du ACO [105].

Colonies d'Abeilles Artificielles (Artificial Bee Colony- ABC)

L'algorithme est défini comme une méta-heuristique qui adopte la technique employée par un essaim intelligent d'abeilles pour identifier leur source de nourriture. La nature des abeilles est étudiée en termes de communication, de sélection du site du nid, d'accouplement, d'attribution des tâches, de reproduction, de danse, de placement des phéromones et de mouvement pour ensuite modifier l'algorithme en fonction des exigences du problème. L'algorithme des colonies d'abeilles artificielles effectue une optimisation en recherchant de manière itérative la solution la plus adaptée parmi un grand nombre d'entrées tout en

essayant de résoudre des problèmes critiques.

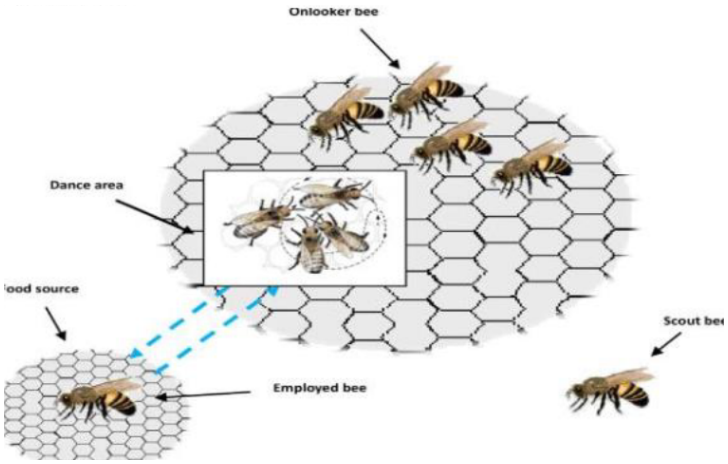


Figure 1.4 – Principe de fonctionnement du ABC [69].

Algorithme de Foraging Bactérienne (Bacterial Foraging Algorithm- BFA)

L'algorithme de recherche de nourriture bactérienne s'inspire du comportement de recherche de nourriture en groupe de bactéries telles que E. Coli et M. Xanthos. Le BFA s'inspire du comportement chimiotactique des bactéries qui perçoivent les gradients chimiques dans l'environnement (tels que les nutriments) et se rapprochent ou s'éloignent de signaux spécifiques [86].

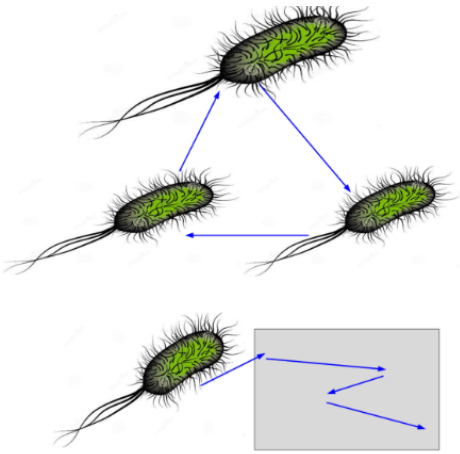


Figure 1.5 – Modèles de comportement de BFA [40]

Algorithme des Lucioles (Firefly Algorithm- FA)

L'algorithme Firefly (FA) a été développé pour la première fois par Xin-She Yang fin 2007 et 2008 à l'université de Cambridge sur la base du comportement de clignotement des lucioles [136]. FA suppose qu'une solution d'un problème d'optimisation est codée comme l'emplacement de l'agent / luciole, tandis que la fonction objectif est codée comme l'intensité lumineuse. Dans FA, il y a deux problèmes importants qui font le succès de l'algorithme : (1) les variations dans l'intensité lumineuse, et (2) la bonne formulation de l'attractivité, qui

est déterminée par la fonction de luminosité, qui à son tour est associée à la fonction objective.

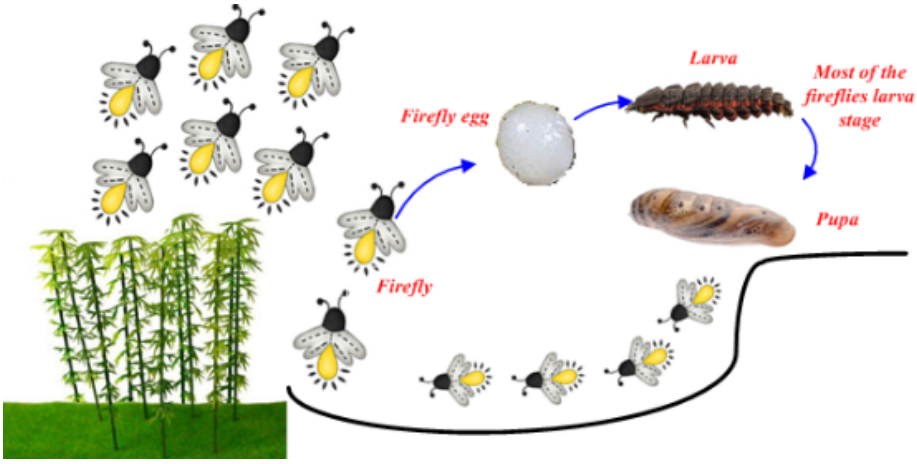


Figure 1.6 – Le cycle de vie de l'algorithme Firefly [68]

Algorithme de libellule (Dragonfly Algorithm « DA »)

Cet algorithme a été introduit par Seyedali Mirajalli. Son idée est dérivée des comportements statiques et dynamiques des libellules dans la nature. On peut dire que ces deux comportements sont très similaires aux deux phases importantes de l'optimisation, à savoir la recherche de proies (exploration) et l'attaque de proies (exploitation). Les étapes importantes du comportement d'essaimage des libellules sont décrites ci-dessous [1].

- Séparation (S), les essaims sont séparés des autres individus.

- Alignement (A), la vitesse de chaque individu est adaptée à celle des autres.

- La cohésion (C) concerne l'attraction de l'essaim vers le centre de l'équipe d'essaimage.

- L'attraction vers l'origine de la nourriture est représentée mathématiquement par (F).

- La distraction de l'ennemi (E).

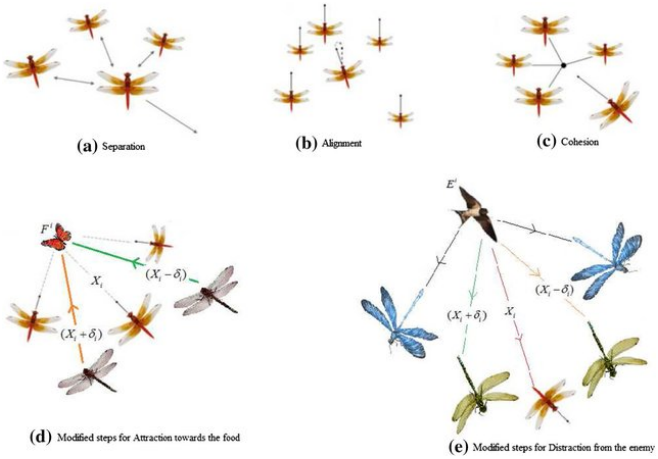


Figure 1.7 – Principe de fonctionnement de DA [128]

1.3 La robotique en essaim

1.3.1 Définition

La robotique en essaim a été définie comme "a novel approach to the coordination of large numbers of robots" (Sahin, 2005). Il s'agit d'une approche de la robotique collective, qui s'inspire du comportement auto-organisé des animaux sociaux. Les oiseaux, les fourmis, les poissons et les abeilles sont quelques exemples d'individus simples qui se rassemblent en groupes pour accomplir des tâches données [125]. C'est l'étude de la coordination de grands groupes de robots relativement simples par l'utilisation de règles locales. Elle s'inspire des sociétés d'insectes capables d'effectuer des tâches qui dépassent les capacités des individus. Beni [16] décrit ce type de coordination de robots comme "The group of robots is not just a group. It has some special characteristics, which are found in swarms of insects, that is, decentralised control, lack of synchronisation, simple and (quasi) identical members".

Selon Huang et al. [65], la robotique en essaim est définie avec les principes qu'un système d'essaim doit avoir un grand nombre de robots, les tâches doivent être résolues et améliorées en utilisant un système d'essaim et que les robots échangent des informations locales par des distances de communication limitées. La Figure 1.2 montre quelques exemples des essaims de robots.

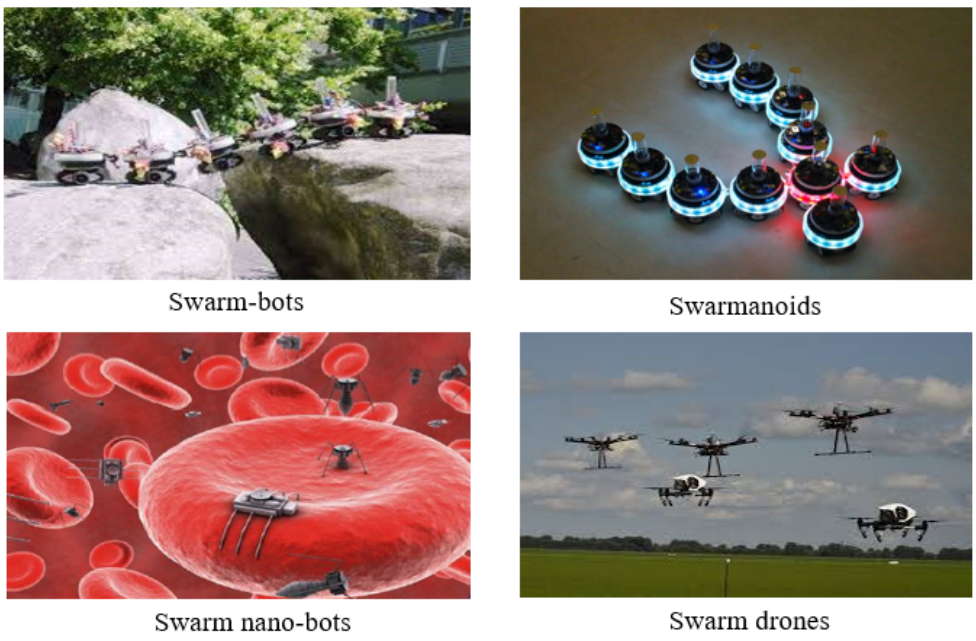


Figure 1.8 – Exemples des essaims de robots

1.3.2 Caractéristiques d'un système robotique en essaim

La recherche sur la robotique en essaim étudie comment utiliser des systèmes composés de plusieurs agents autonomes (robots) pour accomplir des tâches collectives lorsque celles-ci ne peuvent être accomplies par un robot individuel seul. Ce point de vue s'inspire du système des insectes sociaux qui se caractérise par : la robustesse, la flexibilité et l'évolutivité.

La robustesse : exige que le système robotique en essaim puisse continuer à fonctionner, bien qu'à un niveau de performance inférieur, malgré les défaillances des individus ou les perturbations de l'environnement. Cette robustesse peut être attribuée à plusieurs facteurs. Premièrement, la redondance du système, c'est-à-dire que toute perte ou dysfonctionnement d'un individu peut être compensé par un autre. Cela rend les individus dispensables. Deuxièmement, la coordination décentralisée ; c'est-à-dire que la destruction d'une certaine partie du système n'empêchera pas le fonctionnement du système. La coordination est une propriété émergente de l'ensemble du système. Troisièmement, la simplicité des individus ; c'est-à-dire que par rapport à un système complexe unique qui pourrait effectuer la même tâche, dans un système robotique en essaim, les individus seraient plus simples, ce qui les rendrait moins sujets aux défaillances. Quatrièmement, la multiplicité de la détection.

La flexibilité : exige que le système robotique en essaim ait la capacité de générer des solutions modulaires à des tâches différentes. Par exemple dans les colonies de fourmis, les individus participent des tâches de nature très différente, telles que la recherche de nourriture, la récupération de proies et la formation de chaînes. Au cours de la tâche de recherche de nourriture, les fourmis agissent indépendamment en cherchant de la nourriture dans l'environnement ; leur recherche est partiellement coordonnée par les phéromones déposées dans l'environnement. La tâche de récupération des proies exige que les fourmis génèrent une force beaucoup plus grande que celle d'un seul individu pour traîner une proie jusqu'au nid. Lorsqu'une grande proie est découverte, chaque fourmi la saisit avec sa mandibule et la tire dans différentes directions. On observe que les tractions apparemment aléatoires des fourmis sont coordonnées par la force intégrée sur la proie. Dans la tâche de formation de chaîne, les fourmis forment une structure physique semblable à une chaîne qui peut s'étendre au-delà de la portée d'une seule fourmi et exercer des forces importantes en tirant ensemble des feuilles. Au cours de cette tâche, les fourmis utilisent leur corps comme moyen de communication : les fourmis de la chaîne sont immobiles et chacune d'entre elles agrippe ou tient la patte des autres fourmis de la chaîne. Dans cette tâche, la coordination est assurée par le corps des fourmis. Les systèmes robotiques en essaim devraient également avoir la possibilité d'offrir des solutions aux tâches à accomplir en utilisant différentes stratégies de coordination en réponse aux changements de l'environnement.

L'évolutivité : exige qu'un système robotique en essaim soit capable de fonctionner dans une large gamme de tailles de groupe. En d'autres termes, les mécanismes de coordination qui assurent le fonctionnement de l'essaim doivent être relativement peu perturbés par les changements de taille des groupes.

Parmi les principales caractéristiques d'un système robotique en essaim, on trouve que les robots de l'essaim doivent être des robots autonomes, capables de détecter et d'agir dans un environnement réel. De plus, le nombre de robots dans l'essaim doit être important ou

du moins les règles de contrôle le permettent. Ainsi, les robots doivent être homogènes. Il peut exister différents types de robots dans l'essaim, mais ces groupes ne doivent pas être trop nombreux. En outre, les robots doivent être incapables ou inefficaces par rapport à la tâche principale qu'ils doivent résoudre, c'est-à-dire qu'ils doivent collaborer pour réussir ou améliorer leurs performances. Les robots n'ont que des capacités de communication et de détection locales. La coordination étant distribuée, l'évolutivité devient l'une des propriétés du système. Finalement, l'utilisation des plusieurs robots simples peut être plus facile, moins chère, plus flexible et plus tolérante aux pannes que d'avoir un seul robot puissant pour chaque tâche distincte.

1.3.3 Les avantages et les inconvénients d'un système robotique en essaim

Selon Ronald Arkin [8], les systèmes robotiques en essaim présentent plusieurs avantages par rapport aux systèmes mono-robots. Parmi ces avantages, on cite : Premièrement, l'amélioration des performances, c'est-à-dire, si les tâches peuvent être décomposées, alors en utilisant le parallélisme, les essaims peuvent faire en sorte que les tâches soient exécutées plus efficacement. Deuxièmement, l'activation des tâches, c'est-à-dire, les essaims de robots peuvent effectuer certaines tâches qui sont impossibles pour un seul robot. Troisièmement, la détection distribuée, c'est-à-dire, la portée de détection d'un essaim de robots est plus large que celle d'un seul robot. Quatrièmement, l'action distribuée, c'est-à-dire, un essaim de robots peut agir à différents endroits en même temps. Et finalement, la tolérance aux pannes, signifie que dans certaines conditions, la défaillance d'un seul robot au sein d'un essaim n'implique pas que la tâche donnée ne puisse être accomplie, grâce à la redondance du système.

Malgré tous ces avantages, il existe encore certains inconvénients tels que : l'interférence, qui signifie que les robots d'un essaim peuvent interférer les uns avec les autres, en raison de collisions, d'occlusions, etc. Si cela n'est pas clair, les robots peuvent se concurrencer au lieu de coopérer. Enfin, le coût global du système ; l'utilisation de plus d'un robot peut augmenter le coût économique. Ce n'est idéalement pas le cas des systèmes robotiques en essaim, qui prévoient d'utiliser de nombreux robots simples et peu coûteux dont le coût total est inférieur au coût d'un seul robot plus complexe effectuant la même tâche.

1.3.4 Domaines d'application d'un système robotique en essaim

Les systèmes robotiques en essaim sont appliqués dans plusieurs domaines. Nous présentons ci-dessous un certain nombre de domaines dans lesquels la robotique en essaim pourrait être appliquée [122].

Tâches qui couvrent une région

Les systèmes robotiques en essaim sont des systèmes distribués et sont bien adaptés aux tâches qui concernent l'état d'un espace. La surveillance de l'environnement (ou le suivi de l'état de santé) d'un lac constituerait un bon domaine d'application. La capacité de détection

distribuée d'un système robotique en essaim peut permettre la surveillance et la détection immédiate d'événements dangereux, tels que la fuite accidentelle d'un produit chimique. Dans ce cas, un système robotique en essaim présente deux avantages majeurs par rapport aux réseaux de capteurs, qui peuvent également être considérés comme des systèmes robotiques en essaim immobilisés. Premièrement, dans un tel cas, un système robotique en essaim a la capacité de se concentrer sur l'emplacement du problème en mobilisant ses membres vers la source du problème. Cette capacité permet à l'essaim de mieux localiser et identifier la nature du problème. Deuxièmement, l'essaim peut s'auto-assembler pour former un patch qui bloque la fuite.

Tâches trop dangereuses

Les individus qui créent un système robotique en essaim sont dispensables, ce qui rend le système adapté aux domaines qui contiennent des tâches dangereuses. Par exemple, le nettoyage d'un couloir sur un terrain minier peut être réalisé à moindre coût par un essaim de robots. Contrairement à un seul démineur robotique (plus complexe et plus coûteux) conçu pour la même tâche, les membres de l'essaim peuvent se permettre d'être "suicidaires" en marchant dans le champ. Nous pensons également qu'un couloir parcouru par un essaim de robots serait plus sûr que celui qui est contrôlé par un seul démineur robotique, car l'approche robotique de l'essaim marche physiquement sur les mines, simulant la marche des soldats.

Tâches qui évoluent dans le temps

Les systèmes robotiques en essaim ont le pouvoir d'augmenter ou de diminuer leur échelle en fonction de la tâche à accomplir. Par exemple, l'ampleur d'une fuite de pétrole, provenant d'un navire coulé, peut augmenter de façon spectaculaire à mesure que les réservoirs du navire tombent en panne. Un système robotique en essaim qui s'est assemblé tout seul pour contenir la fuite initiale dans une zone délimitée peut être étendu en déversant davantage de robots dans la zone.

Tâches nécessitant une redondance

La robustesse des systèmes robotiques en essaim provient de la redondance implicite de l'essaim. Cette redondance permet au système robotique en essaim de se dégrader tranquillement, ce qui le rend moins susceptible de subir des défaillances catastrophiques. Par exemple, les systèmes robotiques en essaim peuvent créer des réseaux de communication dynamiques sur le champ de bataille. Ces réseaux peuvent bénéficier de la robustesse obtenue par la reconfiguration des nœuds de communication lorsque certains d'entre eux sont touchés par le feu ennemi.

1.3.5 Comportements collectifs d'un système robotique en essaim

Les robots en essaim ont de nombreux comportements collectifs. Ces comportements collectifs sont des comportements de base d'un essaim qui pourraient être combinés pour aborder des applications complexes du monde réel comme, par exemple, la recherche de nourriture ou la construction. Brambilla et al [23] ont classé les comportements des robots en essaim en quatre classes : comportements d'organisation spatiale, comportements de na-

vigation, comportements de prise de décision collective et autres comportements collectifs. Cette classification est illustrée dans la Figure 1.9.

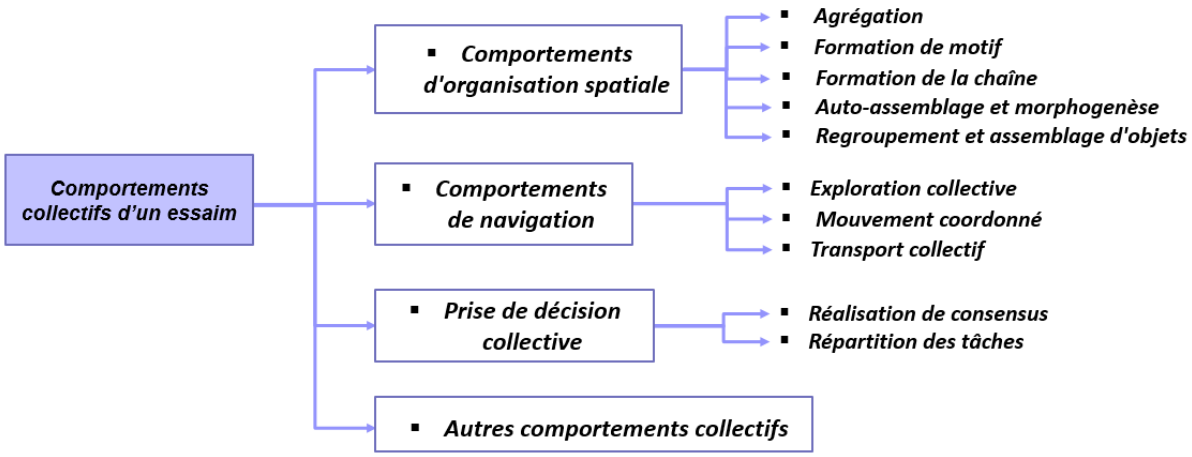


Figure 1.9 – Taxonomie des comportements des essaims [23]

Classe 1 - Comportements d'organisation spatiale

• *Agrégation*

Le but de l'agrégation est de regrouper tous les robots d'un essaim dans une région de l'environnement. Bien qu'il s'agisse d'un simple comportement collectif, l'agrégation est un élément très utile, car elle permet à un essaim de robots de se rapprocher suffisamment les uns des autres pour pouvoir interagir. L'agrégation est un comportement très courant dans la nature. Par exemple, l'agrégation peut être observée chez les bactéries, les cafards, les abeilles et les poissons. En robotique en essaim, l'agrégation est généralement abordée de deux manières : les machines probabilistes à états finis (PFSM) ou l'évolution artificielle.

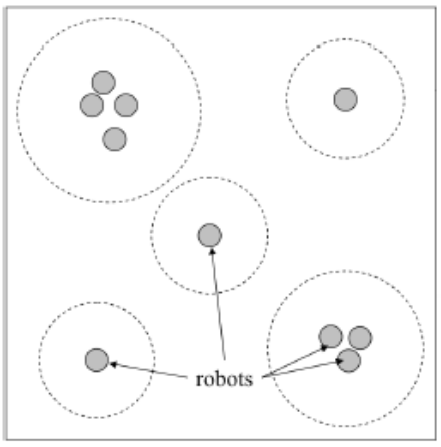


Figure 1.10 – Exemple du comportement collectif de l'agrégation [122]

• *Formation de motifs*

La formation de motifs vise à déployer les robots de manière régulière et répétitive. Les robots doivent généralement respecter des distances spécifiques entre eux afin de créer le motif souhaité. La formation de motifs se retrouve à la fois en biologie et en physique. Parmi les exemples biologiques, citons la disposition spatiale des colonies bactériennes et les motifs chromatiques de la fourrure de certains animaux [95]. La façon la plus courante de développer des comportements de formation de motifs dans des essaims de robots est d'utiliser la conception virtuelle basée sur la physique. La conception virtuelle basée sur la physique utilise des forces virtuelles pour coordonner les mouvements des robots.

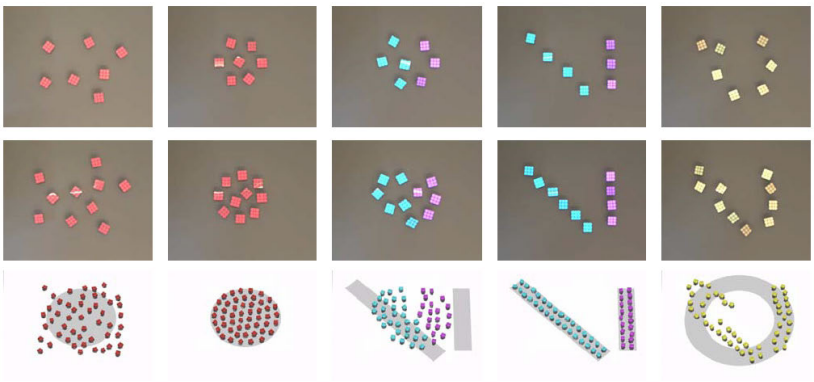


Figure 1.11 – Exemples du comportement collectif de formation de motif [4]

• *Auto-assemblage et morphogenèse*

En robotique, l'auto-assemblage est le processus par lequel les robots se connectent physiquement les uns aux autres. L'auto-assemblage peut être utilisé à différentes fins. Par exemple, pour accroître la stabilité lors de la navigation sur des terrains accidentés ou pour augmenter la puissance de traction des robots. La morphogenèse est le processus qui conduit un essaim de robots à s'auto-assembler en suivant un modèle particulier, et peut être utilisée par l'essaim pour s'auto-assembler en une structure particulièrement appropriée à une tâche donnée. Par exemple, l'auto-assemblage en une ligne peut permettre de passer sur un pont étroit, tandis qu'une forme en forme de goutte rendra le déplacement sur un terrain accidenté plus stable. L'auto-assemblage peut être observé chez plusieurs espèces de fourmis. Les fourmis sont capables de se connecter physiquement afin d'effectuer différentes tâches. En robotique en essaim, l'auto-assemblage et la morphogenèse sont généralement gérés soit par l'évolution artificielle, soit par des machines probabilistes à états finis.

• *Regroupement et assemblage d'objets*

Le but du regroupement et de l'assemblage d'objets est de regrouper des objets proches les uns des autres. La différence entre les clusters et les assemblages est que les clusters sont composés d'objets non connectés, alors que les assemblages sont composés d'objets physiquement liés. Les comportements de regroupement et d'assemblage d'objets sont des éléments fondamentaux de tout processus de construction. Les comportements de regroupement et d'assemblage d'objets sont affichés par de nombreux insectes sociaux tels que les fourmis et les termites. En robotique en essaim, le regroupement et l'assemblage d'objets sont généralement abordés à l'aide de machines à états finis probabilistes.

Figure 1.12 – Exemples du comportement collectif de formation de motif [4]

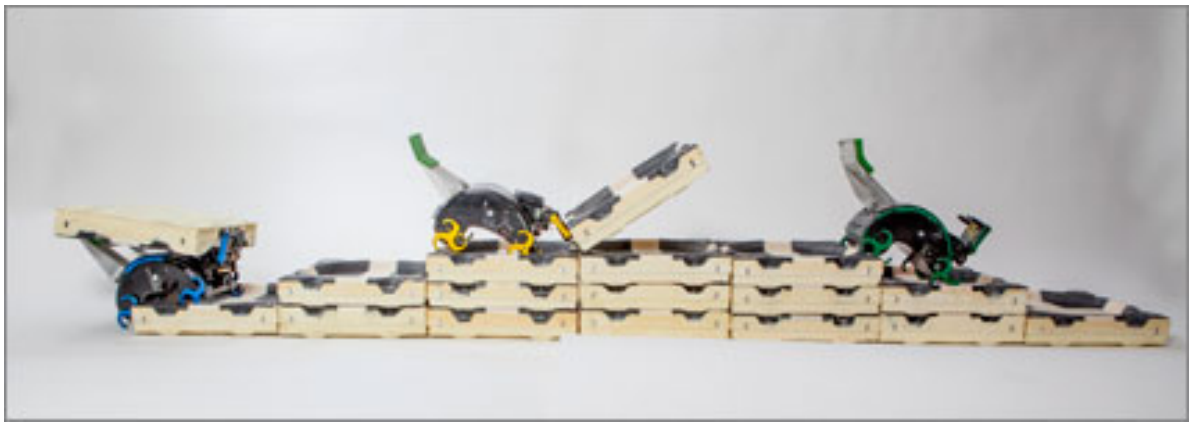


Figure 1.13 – Exemples du comportement de regroupement et d'assemblage d'objets [122]

Classe2 - Comportements de navigation

Ces comportements permettent de résoudre le problème de la coordination des mouvements d'un essaim de robots tels que l'exploration collective, le mouvement coordonné et le transport collectif.

• *Exploration collective*

L'exploration collective fait naviguer l'essaim de robots de manière coopérative dans l'environnement afin de l'explorer. Elle peut être utilisée pour obtenir un aperçu de la situation, rechercher des objets, surveiller l'environnement ou établir un réseau de communication. La couverture de zones et la navigation sont des comportements courants chez les animaux sociaux. Par exemple, les fourmis utilisent des pistes de phéromones pour trouver le chemin le plus court entre deux points et les abeilles communiquent directement les destinations dans l'environnement au moyen de danses. Dans le domaine de la robotique en essaim, la façon la plus courante d'aborder la couverture d'une zone consiste à utiliser une conception virtuelle basée sur la physique pour obtenir une grille couvrant l'environnement.

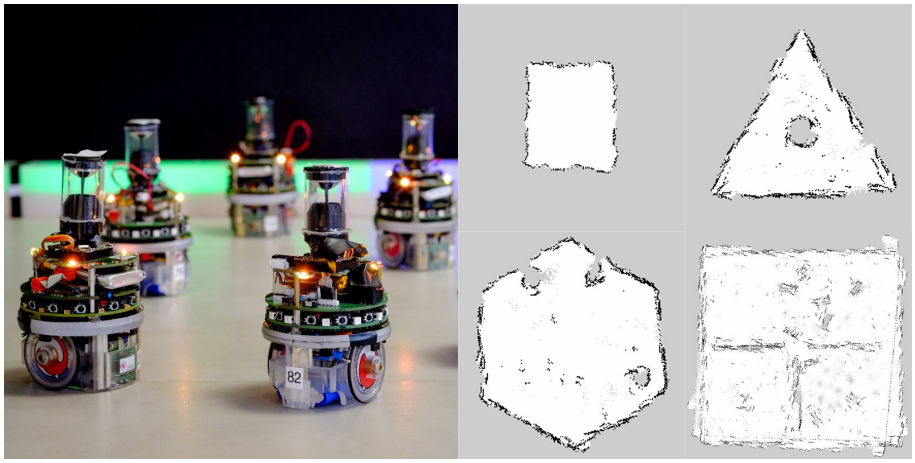


Figure 1.14 – Exemple de comportement d'exploration collective [72]

• *Mouvement coordonné*

Dans le mouvement coordonné, également connu sous le nom de flocage, les robots se déplacent en formation de manière similaire aux bancs de poissons ou aux volées d'oiseaux. Pour un groupe de robots autonomes, le mouvement coordonné peut être très utile pour naviguer dans un environnement où les collisions entre robots sont limitées ou inexistantes, et pour améliorer les capacités de détection de l'essaim. En robotique en essaim, les comportements de mouvement coordonnés sont généralement basés sur une conception virtuelle fondée sur la physique.

• *Transport collectif*

Le transport collectif, également appelé récupération de proies en groupe, est un comportement collectif dans lequel un groupe de robots doit coopérer afin de transporter un objet. En général, l'objet est lourd et ne peut être déplacé par un seul robot, ce qui rend la coopération nécessaire. Les robots doivent se mettre d'accord sur une direction commune afin de déplacer efficacement l'objet vers une cible. Les fourmis transportent souvent leurs proies de manière coopérative. Kube et Bonabeau [76] ont analysé comment le transport coopératif est réalisé dans les colonies de fourmis. Lorsque les fourmis trouvent leur cible, elles s'y attachent physiquement, puis commencent à tirer et à pousser. Si elles ne perçoivent aucun mouvement après un certain temps, elles modifient l'orientation de leur corps et réessayent. Si même cela ne fonctionne pas, elles se détachent, se rattachent à un autre endroit et essaient à nouveau. En robotique en essaim, les comportements de transport collectif sont obtenus en utilisant des machines à états finis probabilistes ou l'évolution artificielle. La coopération est obtenue soit par une communication explicite de la direction de mouvement souhaitée, soit par une communication indirecte, c'est-à-dire en mesurant la force appliquée à l'objet transporté par les autres robots.

Classe 3 - Prise de décision collective

La prise de décision collective traite de la manière dont des robots s'influencent mutuellement lorsqu'ils font des choix. Elle peut être utilisée pour répondre à deux besoins

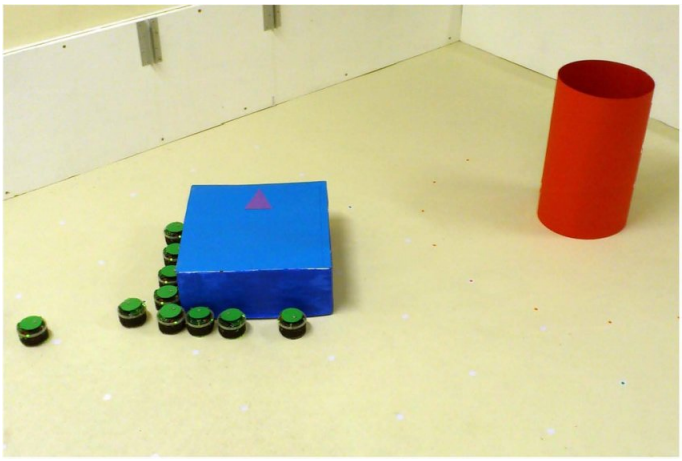


Figure 1.15 – Exemple de comportement de transport collectif [76]

opposés : l'accord et la spécialisation. Un exemple typique d'accord dans les systèmes de robotique en essaim est la réalisation du consensus. Le résultat souhaité de la réalisation du consensus est que tous les robots de l'essaim finissent par converger vers une décision unique parmi les alternatives possibles. En revanche, un exemple typique de spécialisation est la répartition des tâches. Le résultat souhaité de la répartition des tâches est que les robots de l'essaim se répartissent entre les différentes tâches possibles afin de maximiser les performances d'un système.

• *Réalisation d'un consensus*

La réalisation du consensus est un comportement collectif utilisé pour permettre à un essaim de robots d'atteindre un consensus sur un choix parmi différentes alternatives. Ce choix est généralement celui qui maximise les performances du système. Il est généralement difficile de parvenir à un consensus dans un essaim de robots car, très souvent, le meilleur choix peut changer au fil du temps ou ne pas être évident pour les robots en raison de leurs capacités de détection limitées. Ce comportement est affiché chez de nombreuses espèces d'insectes. Par exemple, les fourmis sont capables de décider entre le plus court de deux chemins en utilisant des phéromones. En robotique en essaim, les approches utilisées pour obtenir un consensus peuvent être divisées en deux catégories selon la manière dont la communication est utilisée. Dans la première catégorie, la communication directe est utilisée : chaque robot est capable de communiquer son choix préféré ou des informations connexes. Dans la deuxième catégorie, on utilise plutôt la communication indirecte : la décision est prise par le biais d'indices indirects, tels que la densité de la population de robots [36].

• *Répartition des tâches*

La répartition des tâches est un comportement collectif dans lequel les robots se répartissent entre différentes tâches. L'objectif est de maximiser les performances du système en laissant les robots choisir dynamiquement la tâche à effectuer. Ce comportement peut être observé dans des systèmes naturels tels que comme les colonies de fourmis et d'abeilles. En robotique en essaim, la répartition des tâches est principalement obtenue par l'utilisation de machines à états finis probabilistes. Pour promouvoir la spécialisation, les probabilités

de sélectionner une des tâches disponibles sont soit différentes entre les robots, soit elles peuvent changer en réponse à l'exécution d'une tâche ou à des messages d'autres robots. Dans le domaine de la robotique en essaim, la répartition des tâches a été étudiée principalement sur des robots en recherche de nourriture.

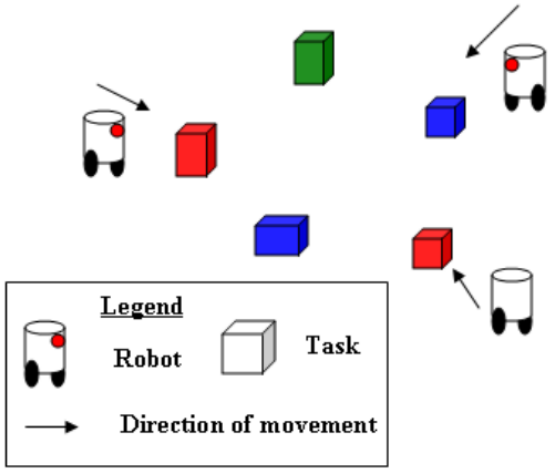


Figure 1.16 – Exemple de comportement collectif de répartition des tâches [41]

Classe 4 - Autres comportements collectifs

Il existe d'autres comportements des robots en essaim qui ne correspondent à aucune des catégories ci-dessus, comme la détection collective des défauts, la régulation de la taille du groupe et l'interaction homme-essaim, ...

• *Détection collective des défauts*

La fiabilité des robots autonomes est encore limitée. Même si la qualité et la robustesse du matériel augmentent, les défaillances matérielles sont encore assez fréquentes. Des techniques permettant aux robots de détecter de manière autonome les défaillances et les comportements défectueux ont été développées en exploitant la redondance naturelle des systèmes robotiques en essaim. Christensen et al. [37] ont développé un comportement de détection des pannes au niveau de l'essaim basé sur la synchronisation des lucioles. Tous les robots de l'essaim émettent un signal de manière synchrone. Les robots sont capables de percevoir si un autre robot est en état de défaillance en observant s'il est synchronisé avec eux. Si un robot n'est pas synchronisé, il est considéré comme défectueux et une réponse est initiée.

• *Régulation de la taille des groupes*

La régulation de la taille des groupes est la capacité collective de créer ou de sélectionner un groupe d'une taille souhaitée. Cela peut être utile pour de nombreuses raisons. Par exemple, Lerman et Galstyan [79] ont montré comment un nombre excessif de robots peut réduire les performances d'un système, et ont démontré pour différents comportements qu'il est possible d'identifier une taille de groupe qui maximise les performances de l'essaim.

• *Interaction entre l'homme et l'essaim*

Les systèmes de robotique en essaim sont conçus pour être autonomes et pour prendre des décisions de manière distribuée. Bien que ces caractéristiques soient généralement considérées comme positives, elles limitent également le degré de contrôle d'un opérateur humain sur le système. En effet, comme il n'y a ni leader ni contrôle centralisé, l'opérateur ne dispose pas d'un moyen simple de contrôler le comportement du système. L'interaction homme-essaim étudie comment un opérateur humain peut contrôler un essaim et recevoir des informations en retour de celui-ci.

1.4 Conclusion

Dans ce chapitre, nous avons d'abord défini l'intelligence en essaim avec ses différents algorithmes principaux qui s'inspirent du comportement social. Ensuite, nous avons présenté les robots en essaim avec ses différentes caractéristiques, avantages et inconvénients. Nous avons également détaillé chaque comportement collectif des robots en essaim. Généralement, l'ingénierie des essaims est une application systématique des connaissances scientifiques et techniques pour spécifier les exigences, modéliser, concevoir, construire, vérifier, valider, exploiter et maintenir un système d'intelligence artificielle. L'ingénierie des essaims est compliquée. Il est difficile de comprendre, d'analyser et de concevoir des systèmes de robots en essaim en raison de la difficulté de séparer les essaims en parties plus simples.

Dans le domaine de la robotique en essaim, il n'existe à ce jour aucune méthodologie bien structurée pour développer des systèmes robotiques en essaim. Généralement,les essaims de robots ont été principalement conçus manuellement : un comportement individuel est amélioré et testé de manière itérative jusqu'à ce que le comportement collectif souhaité soit atteint. Cette approche non systématique n'est ni fiable ni cohérente. La qualité de la solution obtenue dépend fortement de l'expérience et de l'intuition du concepteur. Pour éviter les inconvénients induits par le rôle crucial du concepteur humain, des approches de conception automatique et semi-automatique seront proposées dans le chapitre suivant. Nous définirons également les avantages et les inconvénients de ces approches.

Chapitre 2

Les méthodes de conception des systèmes robotiques en essaim

Le plus grand ennemi de la connaissance n'est pas l'ignorance, c'est l'illusion de la connaissance.

– Daniel J. Boorstin

2.1 Introduction

Dans la robotique en essaim, un grand nombre de robots effectuent une mission qui ne peut être accomplie par un seul robot. Le comportement collectif de l'essaim de robots est obtenu par la collaboration et la coopération des robots individuels. Ainsi, le comportement collectif d'un essaim de robots est le résultat des interactions locales entre le robot individuel, ses voisins et son environnement. Dans le cas général, la nature complexe de ces interactions est pratiquement impossible à traduire dans le comportement des robots individuels, ce qui crée un écart entre le comportement collectif que l'on souhaite obtenir et ce que chacun des robots individuels devrait faire. Combler cet écart est l'un des principaux défis de la robotique en essaim et l'absence d'une méthodologie générale pour combler cet écart influence la façon dont le logiciel de contrôle des essaims de robots est conçu et réalisé.

Jusqu'à présent, les essaims de robots ont surtout été conçus manuellement : un comportement individuel est amélioré et testé de manière itérative jusqu'à l'obtention du comportement collectif souhaité. Cette approche non systématique n'est ni fiable, ni cohérente. La qualité de la solution obtenue dépend fortement de l'expérience et de l'intuition du concepteur. Pour éviter, ou du moins réduire, les inconvénients induits par le rôle crucial du concepteur humain, des approches de conception automatiques et semi-automatiques ont été proposées. Bien qu'un certain nombre d'approches automatiques ont été proposées dans la littérature sur le génie logiciel et le génie des systèmes, elles n'ont pas été étudiées dans

le contexte de la robotique en essaim. Cela s'explique par le fait que ces approches, qui se concentrent sur le découplage et l'automatisation des différentes phases du cycle de vie du robot, semblent être inappropriées en robotique en essaim. En effet, elles modélisent le système à réaliser à un niveau d'abstraction trop élevé et négligent les interactions complexes robot-robot et robot-environnement qui caractérisent le fonctionnement d'un essaim de robots. Par exemple, ces approches supposent qu'il est possible d'établir une correspondance entre les objectifs collectifs de haut niveau de l'essaim et les comportements individuels de bas niveau des robots. Malheureusement, la pratique de la robotique en essaim indique que l'explicitation d'une telle correspondance n'est généralement pas possible.

Dans ce chapitre, nous commencerons par définir l'ingénierie des essaims et nous citerons les différentes plateformes utilisées en ingénierie des essaims pour modéliser et simuler les systèmes en essaim. Ensuite, nous présenterons les différentes méthodes et approches développées par les chercheurs en conception de systèmes robotiques en essaim. Nous définirons également les avantages et les inconvénients de ces approches, les limites de chaque méthode et les solutions que nous proposons pour surmonter ces limites.

2.2 Ingénierie des essaims

2.2.1 Définition

L'ingénierie des essaims est l'application systématique des connaissances scientifiques et techniques pour modéliser et spécifier les exigences, concevoir, réaliser, vérifier, valider, exploiter et maintenir un système d'intelligence en essaim. Le terme « ingénierie des essaims » a été introduit par Kazadi [71], qui a reconnu que la recherche sur l'intelligence des essaims s'oriente vers "the design of predictable, controllable swarms with well-defined global goals and provable minimal conditions". Il ajoute également que "to the swarm engineer, the important points in the design of a swarm are that the swarm will do precisely what it is designed to do, and that it will do so reliably and on time" [71].

2.2.2 La modélisation d'un système robotique en essaim

La modélisation est une méthode utilisée dans de nombreux domaines de recherche pour mieux comprendre les mécanismes internes du système étudié. La modélisation présente des avantages supplémentaires pour la robotique en essaim par rapport à d'autres domaines. L'existence de risques potentiels pour les robots et la puissance limitée des robots obligent le concepteur à réaliser périodiquement certaines expériences. Le temps consacré à ces expériences et le risque éventuel de perdre les robots deviennent un obstacle lorsque plusieurs expériences sont nécessaires pour valider les résultats des études. Pour éliminer ces problèmes, il est plus facile de modéliser les expériences et de les simuler sur un ordinateur. La modélisation est également importante pour les études sur la robotique en essaim lorsqu'il s'agit de tester l'évolutivité des expériences. La plupart du temps, l'évolutivité nécessite de tester les algorithmes de contrôle sur plus de quelques centaines de robots. Mais le coût d'un robot individuel interdit de tester les expériences sur plus de quelques dizaines de

robots dans l'état actuel de la technologie robotique. L'évolutivité étant un objectif important des systèmes de robots en essaim, il semble que les modèles seront nécessaires jusqu'à ce que des robots beaucoup moins chers soient fabriqués.

• *Modélisation basée sur les capteurs*

La modélisation basée sur les capteurs est une méthode de modélisation qui utilise les modèles des capteurs et des actionneurs des robots et des objets de l'environnement comme composants principaux du système modélisé. Après avoir modélisé ces composants principaux, les interactions des robots avec l'environnement et les interactions entre les robots sont modélisées. Cette méthode de modélisation est la plus utilisée et la plus ancienne pour la modélisation des expériences robotiques. Il existe deux approches principales pour la modélisation basée sur les capteurs : les simulations non physiques et les simulations physiques. Dans le premier cas, la dynamique des robots et des objets de l'environnement est ignorée et ils sont considérés comme des objets sans propriétés physiques, sauf à ajouter une certaine logique pour éliminer les collisions entre eux [58]. Dans le deuxième cas, la modélisation des interactions des robots et de l'environnement est réalisée sur la base de règles physiques en attribuant des propriétés physiques aux objets comme la masse et la force motrice pour pouvoir déplacer les robots. Pour obtenir des résultats réalistes, des moteurs physiques externes sont intégrés à la simulation. Cette approche ajoute beaucoup plus de complexité au modèle dans le but d'obtenir des résultats plus réalistes.

• *Modélisation microscopique*

Les modèles microscopiques permettent de modéliser des expériences robotiques en représentant mathématiquement chaque robot et leurs interactions. Dans cette méthode, les comportements des robots sont définis comme des états et la transition entre ces états est liée à des événements internes au robot et à des événements externes dans l'environnement. L'approche microscopique modélise les expériences en représentant chaque robot, tandis que l'approche macroscopique modélise directement le comportement global du système.

Dans les modèles microscopiques probabilistes [90], une unité de temps est définie sur la base d'un événement primitif afin de pouvoir faire avancer le modèle à chaque étape du modèle. Après avoir spécifié cette unité de temps, la probabilité de chaque transition d'état est calculée à l'aide d'expériences systématiques réalisées avec des robots réels. En d'autres termes, les probabilités de tous les événements sont calculées par unité de temps du modèle. Après avoir trouvé ces probabilités de transition d'état, le modèle mathématique est exécuté pour chaque robot en générant des nombres aléatoires entre 0 et 1 pour chaque transition d'événement possible du robot sélectionné et en comparant ces nombres aux probabilités de transition d'état. Si certains de ces nombres sont inférieurs aux probabilités de transition prédéfinies des événements associés, ces événements sont supposés s'être produits et l'état de ce robot est modifié [14].

• *Modélisation macroscopique*

Un autre type de méthode de modélisation mathématique des expériences robotiques est la modélisation macroscopique. Dans la modélisation macroscopique, le comportement du système est défini par des équations différentielles et chacun des états du système (variables des équations différentielles) représente le nombre moyen de robots dans un état particulier

à un certain pas de temps. Les modèles microscopiques permettent de saisir les variations dans les expériences. En d'autres termes, si les modèles macroscopiques permettent d'obtenir rapidement un comportement global approximatif du système robotique, les modèles microscopiques permettent d'obtenir lentement un comportement global plus réaliste.

• *Modélisation par automates cellulaires*

Les automates cellulaires (AC) font partie des modèles mathématiques les plus simples des systèmes complexes [66]. Les modèles d'AC contiennent un réseau discret de cellules dans une ou plusieurs dimensions où chaque cellule du réseau a un nombre fini d'états possibles. Chaque cellule interagit uniquement avec les cellules qui se trouvent dans son voisinage local et la dynamique du système est caractérisée par les règles locales exécutées localement sur les cellules dans des étapes de temps discrètes.

2.2.3 Plateformes de simulation d'un système robotique en essaim

Il existe plusieurs simulateurs qui peuvent être utilisés pour simuler des systèmes robotiques en essaim tels que ARGoS, Webots, Player / Stage et ROS/gazebo ...

• *Autonomous Robots Go Swarming ARGoS*

ARGoS est conçu pour simuler des expériences complexes impliquant de grands essaims de robots de différents types. C'est le premier simulateur multi-robots qui est à la fois efficace (performances rapides avec de nombreux robots) et flexible (hautement personnalisable pour des expériences spécifiques). De nouveaux choix de conception dans ARGoS ont permis cette découverte. Premièrement, dans ARGoS, il est possible de partitionner l'espace simulé en plusieurs sous-espaces, gérés par différents moteurs physiques fonctionnant en parallèle. Deuxièmement, l'architecture d'ARGoS est multiple, ce qui permet d'optimiser l'utilisation des processeurs multi-cœurs modernes. Enfin, l'architecture d'ARGoS est hautement modulaire, ce qui permet d'ajouter facilement des fonctionnalités personnalisées et d'allouer les ressources de calcul de manière appropriée [112].

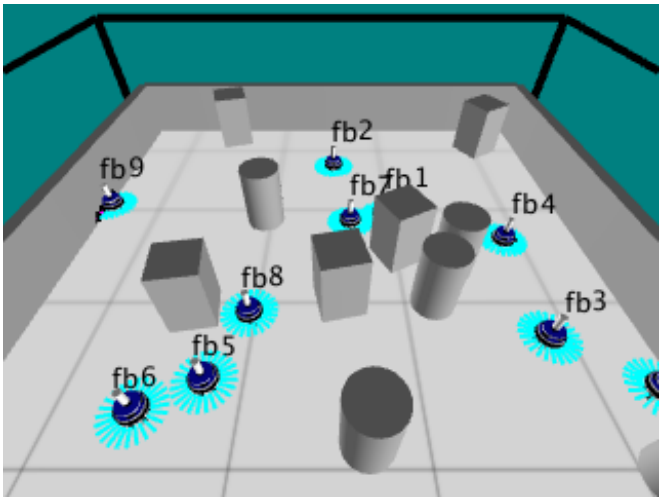


Figure 2.1 – l'interface du ARGoS [108]

• *Simulateur Webots*

Webots est un progiciel de simulation de robot mobile professionnel. L'utilisateur peut ajouter des objets passifs simples ou des objets actifs appelés robots mobiles. Ces robots peuvent avoir différents schémas de locomotion (robots à roues, robots à pattes ou robots volants). L'utilisateur peut programmer chaque robot individuellement pour présenter le comportement souhaité. Webots contient un grand nombre de modèles de robots et d'exemples de programme de contrôleur pour aider les Utilisateurs à démarrer [39].

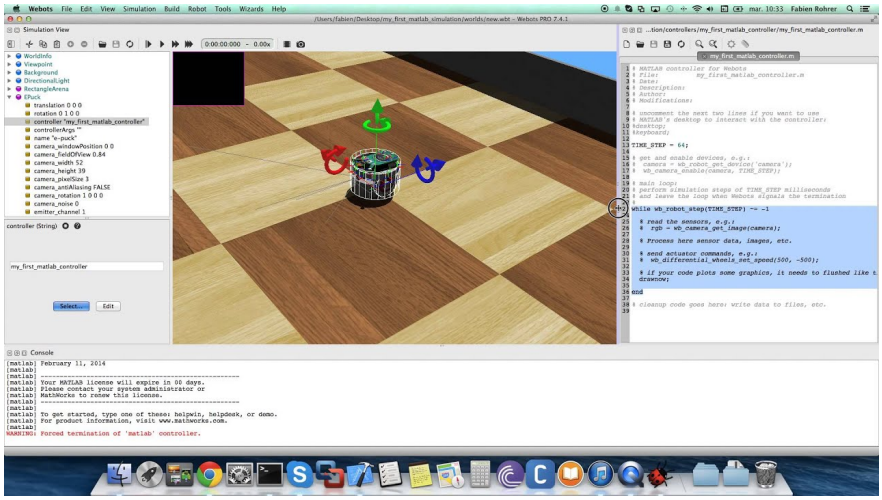


Figure 2.2 – l'interface du simulateur Webots [39]

• *Simulateur Player / Stage*

Stage est souvent utilisé comme module de plugin Player, fournissant des populations de périphériques virtuels pour Player. Les utilisateurs écrivent les contrôleurs de robot et les algorithmes de capteur en tant que « clients » sur le « serveur » du lecteur. Stage permet un prototypage rapide de contrôleurs destinés à de vrais robots. Stage permet également des expériences avec des appareils robotiques réalistes que vous n'avez pas [3].

• *Simulateur ROS/Gazebo*

Gazebo est un simulateur dynamique 3D ayant la capacité de simuler avec précision et efficacité des populations de robots dans des environnements intérieurs et extérieurs complexes. Bien que similaire aux moteurs de jeu, Gazebo fournit une simulation physique avec un degré de fidélité beaucoup plus élevé, une suite de capteurs et des interfaces utilisateur et programme. Gazebo est un logiciel libre, publié sous la licence publique GNU. Les utilisateurs sont libres de l'utiliser, de l'étendre et de le modifier selon leurs besoins [53].

2.2.4 Plateformes d'implémentation d'un système robotique en essaim

Il existe plusieurs plateformes de recherche développées à des buts éducatifs et scientifiques, résumées dans le tableau de la Figure 2.4. Elles permettent d'étudier l'application

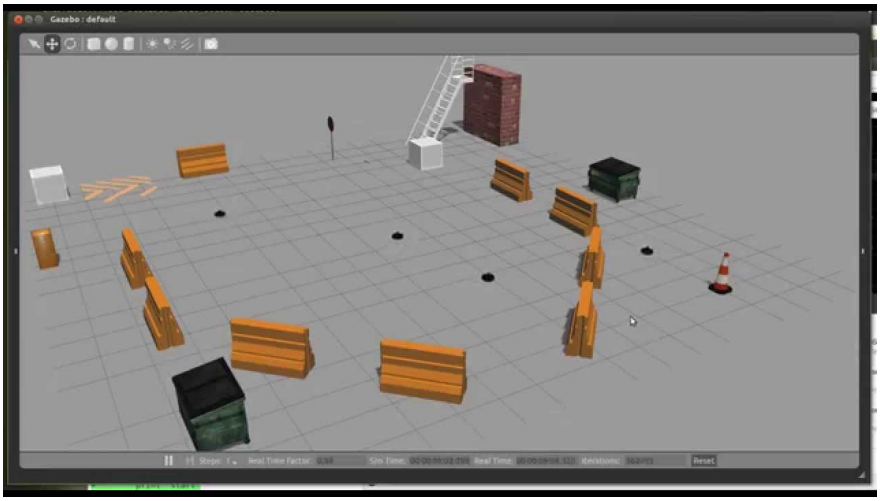


Figure 2.3 – l'interface du simulateur ROS/gazebo [53]

des algorithmes des robots en essaim. Il existe encore d'autres plateformes robotique sophistiquées qui ne sont pas incluses, car elles n'ont pas été développées dans l'intention de les utiliser dans des applications de recherche.

- *Le robot Khepera* [99], destiné à la recherche et à l'enseignement, développé par l'École Polytechnique Fédérale de Lausanne (EPFL, Suisse), largement utilisé dans le passé, est aujourd'hui tombé en panne.

- *Le robot Khepera III* [115], conçu par la K-Team en collaboration avec l'EPFL.

- *Le robot e-puck* [102], conçu à l'EPFL à des fins éducatives.

- *Le robot miniature Alice* [32], également développé à l'EPFL.

- *Le robot Jasmine* [75], développé dans le cadre du projet I-swarm.

- *Le robot I-Swarm* [132], très petit, également développé par le projet I-swarm.

- *S-Bot* [101], très polyvalent, développé dans le cadre du projet Swarmbots.

- *Kobot* [131], conçu par l'Université technique du Moyen-Orient (Turquie).

- *SwarmBot* [94], conçu par la société i-Robot pour la recherche.

2.3 Les défis de la conception de systèmes robotiques en essaims

2.3.1 Défis lié au processus de conception

Aujourd'hui, les systèmes robotiques en essaim représentent un défi de conception considérable d'un point de vue opérationnel. La plupart des chercheurs qui conçoivent des

Name	Size (mm) (diam.) or ($l \times w$)	Sensors	Communications	Relative positioning system	Development
Khepera	55	8 IR	RS232 Wired link	—	Research Commercial
Khepera III	120	11 IR 5 ultra sound	WIFI and bluetooth	Expansion (IR based)	Research Commercial
e-puck	75	11 infrared (IR) Contact ring Colour camera	Bluetooth	Expansion (IR based)	Open source Commercial Research and Edu.
Alice	20 × 20	IR proximity and light Linear camera	Radio (115 kbit/s)	—	Research Non commercial
Jasmine	23 × 23	8 IR	IR	Integrated (IR based)	Open source Research
I-Swarm robot	3 × 3	[a]	[a]	—	Research Non commercial
S-Bot	120	15 Proximity OmniCamera Microphone, Temp.	WIFI	Camera based	Research Non commercial
Kobot	120	8 IR Colour camera	Zigbee	Integrated (IR based)	Research Non commercial
SwarmBot	127 × 127	IR, light sensors Contact, camera	IR based (Local)	Integrated (IR based)	Research Non commercial

Figure 2.4 – Résumé des principales plateformes robotiques utilisées dans la robotique en essaim

systèmes en essaim ne sont pas issus de l'ingénierie des systèmes ; leur expertise se situe plutôt dans le domaine de l'informatique. Les chercheurs se concentrent souvent sur la conception détaillée, sans architecture de conception globale pour fournir le contexte de la mission. Il est essentiel de faire le lien entre la mission et la conception détaillée afin d'obtenir un produit adapté aux besoins opérationnels. En raison des propriétés autonomes inhérentes aux systèmes en essaim, le commandant d'un essaim devra gérer et contrôler les systèmes en essaim à un niveau d'abstraction plus élevé que ne le permettent les conceptions actuelles.

En robotique en essaim, il n'y a pas de mécanisme de coordination centralisé derrière le fonctionnement synchrone des robots en essaim, mais leur fonctionnement doit être robuste, flexible et évolutif. Dans [109], Parker a soulevé la question du choix de systèmes centralisés ou décentralisés pour contrôler facilement les systèmes de robots en essaim. Pour les systèmes centralisés, les modèles de communication et de contrôle ne s'adaptent pas bien à un nombre croissant d'individus et ils sont sensibles à la perte du leader central. En revanche, les systèmes décentralisés sont incapables de synthétiser ou d'accéder aux données globales à moins que tous les individus ne soient connectés, car il n'existe pas de mécanisme central capable de synthétiser les données de tous les membres de l'essaim. Le comportement global découle également des interactions entre les nombreux individus qui interagissent localement, il est donc difficile pour un contrôleur humain de prédire le comportement exact de ces systèmes. Tolstaya et al.[129] proposent une approche d'apprentissage des contrôleurs locaux qui ne nécessitent que des informations et des communications locales au moment du test en imitant la politique des contrôleurs centralisés qui utilisent des informations globales au moment de la formation.

2.3.2 Défis liés à la modélisation des systèmes en essaim

La modélisation fait partie des outils les plus utilisés pour analyser et valider les systèmes de robotique en essaim. En fait, plusieurs chercheurs ont proposé des méthodes de modélisation pour les systèmes en essaim. En général, les modèles peuvent être divisés en deux catégories : micro-scopique et macroscopique. Les descriptions microscopiques traitent du robot comme l'unité de base du modèle. Ces modèles décrivent les interactions du robot avec les autres robots et l'environnement. La résolution ou la simulation d'un système composé de nombreux agents de ce type permet aux chercheurs de comprendre le comportement global du système. En revanche, un modèle macroscopique, décrit directement le comportement collectif de l'essaim robotique. Il est efficace sur le plan informatique car il utilise moins de variables [80]. Auricchio [10] présente un modèle continu pour capturer le comportement d'un essaim d'agents de fabrication (par exemple, des drones, des imprimantes 3D, etc.) ainsi qu'une implémentation numérique très simple, mais efficace, pour simuler l'évolution d'un tel essaim. Il présente également des exemples unidimensionnels et bidimensionnels, montrant le potentiel de l'approche proposée pour prédire les comportements des essaims.

2.3.3 Défis liés à la simulation de systèmes en essaim

La simulation offre un moyen plus facile, plus rapide et même plus sûr de réaliser des expériences que l'utilisation de robots réels. Cependant, il n'est pas simple de choisir le meilleur logiciel pour simuler ou modéliser un essaim robotique. Erez et al. dans [45] ont établi quelques mesures de performance de simulation, basées sur des défis numériques typiques en robotique. Ils ont conclu que chaque plate-forme donne ses meilleures performances dans le type de système pour lequel elle a été conçue et optimisée. Il est donc difficile de comparer différents simulateurs multi-robots, car chacun a été développé avec des objectifs différents. Il existe de nombreux types de plates-formes et de logiciels de simulation de robots pour différents objectifs et types de robots. Par exemple, Swarm-bots est un simulateur robotique dans lequel l'interaction collective est mise à profit par le mécanisme d'intelligence en essaim. Son niveau de contrôle peut être étendu au niveau physique et son utilisation a été recommandée dans des expériences avec jusqu'à 40 robots [134]. C'est un outil puissant, mais il n'est pas disponible pour le public. Il peut être utilisé pour simuler des propriétés telles que la robustesse, la flexibilité, et a la capacité de résoudre des problèmes complexes en exploitant le parallélisme et l'auto-organisation [100]. ARGoS est un autre simulateur et multi-moteur spécialement pour la robotique en essaim hétérogène, il permet d'utiliser plusieurs moteurs physiques de différents types et les alloue à différentes parties de l'environnement. Les contrôleurs, les capteurs, les actionneurs, les moteurs physiques et les visualisations peuvent être intégrés dans le robot de conception [111]. Pour notre part, nous utiliserons Gazebo comme simulateur dans notre méthodologie de conception. En fait, Gazebo est un simulateur 3D d'environnements externes. Il produit un retour réaliste des capteurs et des interactions physiquement cohérentes entre les objets, permettant à l'utilisateur de choisir entre plusieurs moteurs dynamiques, mais il a l'inconvénient de fonctionner lentement avec de grandes populations [97]. Gazebo a été utilisé pour comparer des algorithmes de navigation et de préhension dans un environnement contrôlé.

2.3.4 Défis liés à l'implémentation réelle de systèmes en essaim

La mise en œuvre d'un système robotique en essaim sur une plateforme réelle reste une tâche compliquée. La recherche sur la robotique en essaim utilise des simulations et des implémentations dans le monde réel pour tester et évaluer les modèles de comportement des essaims. La plupart des robots en essaim utilisent plusieurs robots mobiles identiques comme agents en essaim. En fait, les robots mobiles tels que E-puck, Khepera et Kilobot ont été très populaires parmi les chercheurs en robotique en essaim, apparaissant dans des publications du monde entier et toujours présents aujourd'hui. Il convient de noter que ces robots ont la particularité d'être disponibles dans le commerce, ce qui les rend plus accessibles aux groupes de recherche [30]. En ce qui nous concerne, nous choisirons le robot mobile turtlebot3 comme plateforme pour nos recherches.

2.4 Les méthodes de conception existantes des systèmes en essaims

2.4.1 Introduction

L'ingénierie des essaims est compliquée ; il est difficile de comprendre, d'analyser et de concevoir des essaims car ils ne peuvent pas être facilement décomposés en éléments plus simples [15]. Le développement d'une méthode de conception structurée pour le développement des comportements collectifs des systèmes robotiques en essaim reste un défi ouvert [126]. Traditionnellement, une approche "code and fix" est employée pour concevoir et développer des systèmes robotiques en essaim [21]. Cette approche implique que le développeur conçoive, teste et modifie les comportements individuels des robots jusqu'à ce que le comportement collectif requis soit atteint. Cette approche prend du temps et dépend de l'ingéniosité et de l'expertise du développeur. Aujourd'hui, il existe deux types de méthodes fondamentales dans le développement des systèmes robotiques en essaim :

• *La vérification ascendante (Bottom-up method)* pour savoir si les règles locales mèneront à l'objectif global souhaité.

• *Le développement descendant (Top-down method)* de règles locales à partir d'un objectif global,

2.4.2 Conception ascendante de systèmes en essaim

Les approches de modélisation ascendantes se concentrent sur l'assemblage de sous-composants de systèmes pour construire des systèmes plus complexes. Cette approche est avantageuse du point de vue de la modularité et de la composabilité, mais elle risque de ne pas répondre aux exigences des systèmes de niveau supérieur si le processus de conception précède le développement de l'architecture du système de haut niveau. C'est pourquoi la combinaison de modèles ascendants et descendants est une heuristique bien établie en

matière de développement de logiciels [26]. La modélisation basée sur les agents, les machines à états finis (FSM) et le comportement des agents sont des méthodes de modélisation ascendantes fréquemment utilisées dans la conception de systèmes en essaim. [22, 92, 7].

a) Méthode de modélisation basée sur des agents

La modélisation des interactions au niveau des agents peut fournir des informations précieuses sur le comportement émergent inhérent aux systèmes en essaim. McCune et al.[93] ont utilisé la modélisation basée sur les agents pour étudier la commande et le contrôle des essaims de drones, Bonabeau a simulé des systèmes humains à l'aide de méthodes de modélisation basées sur les agents et Munoz [104] a étudié l'utilisation d'essaims de drones défensifs à l'aide de la modélisation basée sur les agents. Le comportement émergent d'un système en essaim est un attribut de conception clé à prendre en compte lors du développement d'un système en essaim, de la création de tactiques d'essaimage ou de la conception d'une méthodologie d'évaluation pour un système en essaim. Si la modélisation basée sur les agents peut fournir des informations utiles sur les interactions au sein d'un système, la variabilité de la définition des agents et la normalisation limitée des approches rendent la vérification des modèles difficile.

b) Méthode des machines à états finis (FSM)

Les FSM (ou automates à états finis) sont une approche courante pour la modélisation des architectures de systèmes autonomes multi-véhicules [91]. Dans une architecture FSM, chaque agent fonctionne dans l'un de plusieurs états à un moment donné. Des événements déclencheurs, générés par des capteurs embarqués ou des conditions environnementales, font passer l'agent d'un état à l'autre. Cette approche est applicable au développement de systèmes militaires en essaim car les états et les déclencheurs peuvent être définis de manière déterministe, ce qui est nécessaire pour les événements de mission à haut risque tels que les frappes de cibles. À l'inverse, il peut y avoir d'autres événements de mission, comme la recherche secrète, pour lesquels un certain degré d'imprévisibilité est souhaité. Dans ces cas, un FSM probabiliste peut être utilisé pour permettre différents comportements à l'intérieur d'un état ou permettre des transitions multiples entre les états en fonction de probabilités fixes ou variables [135]. Les comportements d'allocation et d'agrégation des tâches ont également été réalisés à l'aide d'approches FSM probabilistes.

c) Méthode de conception basée sur le comportement

La conception basée sur le comportement, dans laquelle le comportement de l'agent individuel est développé de manière itérative jusqu'à ce que le comportement souhaité de l'essaim soit acquis, est une méthode typique de conception de systèmes en essaim. Les comportements peuvent s'appliquer à des agents individuels, à leur environnement et à des groupes d'agents (souvent appelés comportements collectifs) [24]. Les comportements peuvent également être classés en comportements abstraits de niveau supérieur et en comportements primitifs de niveau inférieur, ou simplement en primitives. Le terme "primitives"

est emprunté à la discipline informatique et fonctionne de manière similaire dans la littérature robotique ; elles servent de blocs de construction pour la programmation de fonctions de niveau supérieur.

2.4.3 Conception descendante de systèmes en essaim

Les modèles de conception descendante, dans lesquels les éléments fonctionnels de haut niveau sont spécifiés avant de décomposer les fonctions de niveau inférieur, sont courants en ingénierie des systèmes. Les diagrammes d'activité, de séquence, de machine d'état et de cas d'utilisation sont des diagrammes traditionnels de modélisation du comportement en ingénierie des systèmes.13 Les approches de conception descendante sont moins importantes dans la conception des systèmes en essaim. Ces approches comprennent la méthode de conception axée sur les propriétés "Property Driven Design" développée par Brambilla et al. en 2012 [22] et la méthode Hamiltonienne développée par Kazadi et al. en 2007[70].

a) La méthode de conception "Trial-and-error design"

La conception d'un essaim de robots par essais et erreurs relève plus de l'art que de la science. Le concepteur opère de manière non structurée avec peu de bases scientifiques et d'outils techniques : il recherche un comportement individuel qui, par l'interaction complexe d'un grand nombre de robots, aboutirait au comportement collectif souhaité. Le processus de recherche s'effectue par le biais de suppositions éclairées qui reposent uniquement sur l'expertise et l'ingéniosité du concepteur. Le concepteur commence par définir une première implémentation du comportement individuel du robot. Il teste ensuite ce comportement, généralement au moyen de simulations informatiques, et l'ajuste de manière itérative jusqu'à ce que le comportement collectif résultant réponde aux exigences de l'essaim. Souvent, le concepteur s'inspire des systèmes biologiques : lorsque l'objectif est de concevoir un essaim de robots dont le comportement au niveau de l'essaim est similaire à celui d'un système biologique (par exemple, un essaim d'insectes, une volée d'oiseaux ou un troupeau de mammifères), le concepteur peut trouver pratique de concevoir le comportement du robot individuel en imitant celui du membre individuel du système biologique.

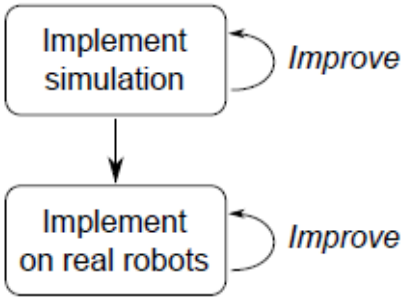


Figure 2.5 – Les phases de la méthode de conception Trial-and-error [22].

La relation entre les niveaux microscopique et macroscopique pose des problèmes dif-

ficiles à l'approche de conception par essais et erreurs. En particulier, le comportement du robot individuel ne peut pas être évalué directement et en soi : il doit être évalué indirectement en observant le comportement collectif d'un essaim composé d'un grand nombre de robots individuels qui exécutent le comportement analysé. Malgré ses limites, l'approche par essais et erreurs a été utilisée avec succès pour développer plusieurs comportements collectifs, notamment l'agrégation, la formation de chaînes et la répartition des tâches.

b) La méthode de conception "Property Driven Design"

L'idée de la conception basée sur les propriétés (Property Driven Design) est qu'un système de robotique en essaim peut être formellement décrit par un ensemble de propriétés. Ces propriétés sont les caractéristiques distinctives du système que le développeur veut atteindre. Elles peuvent être spécifiques à une tâche, par exemple, le système finit par accomplir la tâche X, ou elles peuvent exprimer des propriétés plus génériques, par exemple, le système continue de fonctionner tant qu'il y a au moins N robots ou le système n'entrera jamais dans l'état Y. Cette approche comporte quatre phases : Tout d'abord, les exigences de l'essaim de robots sont décrites formellement sous la forme de propriétés souhaitées. Ensuite, un modèle prescriptif de l'essaim de robots est créé. Puis ce modèle prescriptif est utilisé comme plan pour mettre en œuvre et améliorer une version simulée de l'essaim de robots souhaité. Enfin, l'essaim de robots final est mis en œuvre. La figure 2.6 présente un diagramme montrant les différentes phases de la conception basée sur les propriétés.

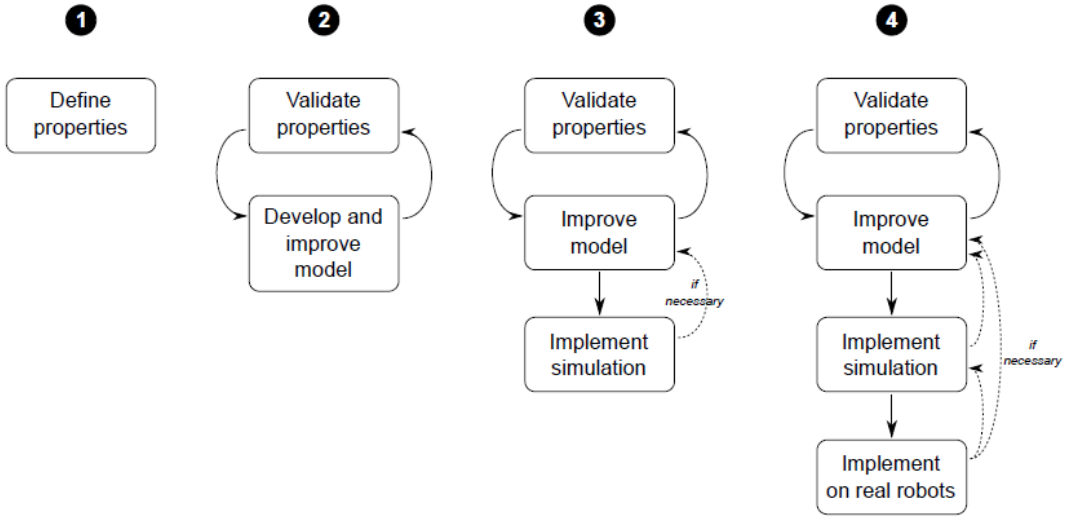


Figure 2.6 – Les quatre phases de la méthode property-driven design [22].

À chacune des phases de la conception pilotée par les propriétés, une nouvelle couche est ajoutée au système. Les couches diffèrent par leur niveau d'abstraction : la couche des propriétés est la plus abstraite, dans laquelle seules les caractéristiques de l'essaim de robots sont énoncées ; la couche des robots est la plus concrète, dans laquelle le logiciel des robots réels est développé et déployé. L'ajout d'une nouvelle couche rapproche le système de son état final. Chaque phase de cette approche est caractérisée par un cycle de développement/validation : le développeur se concentre sur la couche nouvellement introduite, mais

toutes les couches précédemment développées sont toujours actives, c'est-à-dire qu'elles peuvent encore être améliorées et étendues, si cela s'avère nécessaire pour garantir la cohérence de toutes les couches. La couche nouvellement introduite.

Première phase : Propriétés - Dans cette phase, le développeur spécifie formellement les exigences de l'essaim de robots sous la forme de propriétés souhaitées. Ces propriétés sont les caractéristiques distinctives de l'essaim de robots que le développeur veut réaliser. Elles peuvent être spécifiques à une tâche, par exemple le système finit par accomplir la tâche X, ou elles peuvent exprimer des propriétés plus génériques, par exemple le système continue de fonctionner tant qu'il y a au moins N robots ou le système ne sera jamais dans l'état Y. Plus ces propriétés sont claires et complètes dans cette phase, plus l'essaim de robots développé répondra aux attentes. Des exigences clairement définies permettent de réduire le risque de développer "le mauvais essaim de robots".

Deuxième phase : Modèle - Dans cette phase, le développeur crée un modèle prescriptif de l'essaim de robots. En général, le modèle prescriptif décrit comment les robots changent d'état au fil du temps, où un état est une description abstraite simplifiée des actions d'un robot. Le modèle prescriptif doit être suffisamment détaillé pour capturer le comportement des robots et leur interaction, mais ne doit pas être trop détaillé, afin d'éviter toute complication inutile. Une fois qu'une première version du modèle prescriptif est produite, les propriétés souhaitées énoncées dans la première phase sont vérifiées à l'aide du model checking. Au début, il est possible que le modèle prescriptif ne satisfasse pas toutes les propriétés souhaitées. Dans un processus itératif, le développeur étend et améliore le modèle prescriptif, jusqu'à ce que les propriétés soient satisfaites. Le résultat de ce processus est un modèle prescriptif du comportement collectif de l'essaim de robots qui satisfait aux propriétés énoncées.

Troisième phase : Simulation - Au cours de cette phase, le développeur utilise le modèle prescriptif comme plan directeur pour mettre en œuvre et améliorer l'essaim de robots à l'aide d'une simulation informatique basée sur la physique. Cela permet au développeur de se concentrer sur des aspects spécifiques et de négliger d'autres détails mineurs. De plus, le fait de se concentrer sur le modèle prescriptif au moment de la conception permet au développeur d'orienter ses efforts vers des décisions de haut niveau plutôt que sur la mise en œuvre. Il est possible que les choix de mise en œuvre ou d'autres aspects imprévus du système donnent lieu à un système simulé qui ne se comporte pas comme prévu par le modèle prescriptif. Dans ce cas, le développeur doit revenir aux phases précédentes, modifier le modèle prescriptif pour tenir compte des résultats obtenus par la simulation, et vérifier si les propriétés requises sont toujours vraies.

Quatrième phase : Robots - Dans la dernière phase, le développeur réalise l'essaim final de robots. Comme pour la transition entre le modèle prescriptif et la simulation, si la mise en œuvre sur les robots révèle que certaines hypothèses formulées au cours des phases précédentes ne tiennent pas, il peut être nécessaire de modifier la version simulée ou le modèle prescriptif, afin de conserver la cohérence de tous les niveaux.

c) La méthode de conception basée sur la physique virtuelle

Dans la conception basée sur la physique virtuelle [70], chaque robot est considéré comme une particule virtuelle qui exerce des forces sur d'autres particules, c'est-à-dire d'autres robots. Chaque robot est donc immergé dans un champ de forces qui dépend de la présence et de la distance des robots voisins. La force virtuelle agissant sur chaque robot est $f = \sum_{i=1}^{k} f_i(d_i)e^{j\Theta_i}$, où j désigne l'unité imaginaire, d_i et Θ_i sont la distance et la direction du ième robot voisin, et la fonction $f_i(d_i)$ est la dérivée d'une fonction potentielle artificielle. Le potentiel de Lennard-Jones (figure 2.7 (a)) est couramment utilisé dans ce contexte. Les figures 2.7 (b), (c) et (d) montrent trois exemples de la force virtuelle qui agit sur un robot en fonction de la position de ses pairs voisins. Le concepteur peut associer des forces répulsives virtuelles aux obstacles et autres objets de l'environnement pour éviter les collisions. Chaque robot estime les forces virtuelles qui agissent sur lui et les traduit en commandes de mouvement.

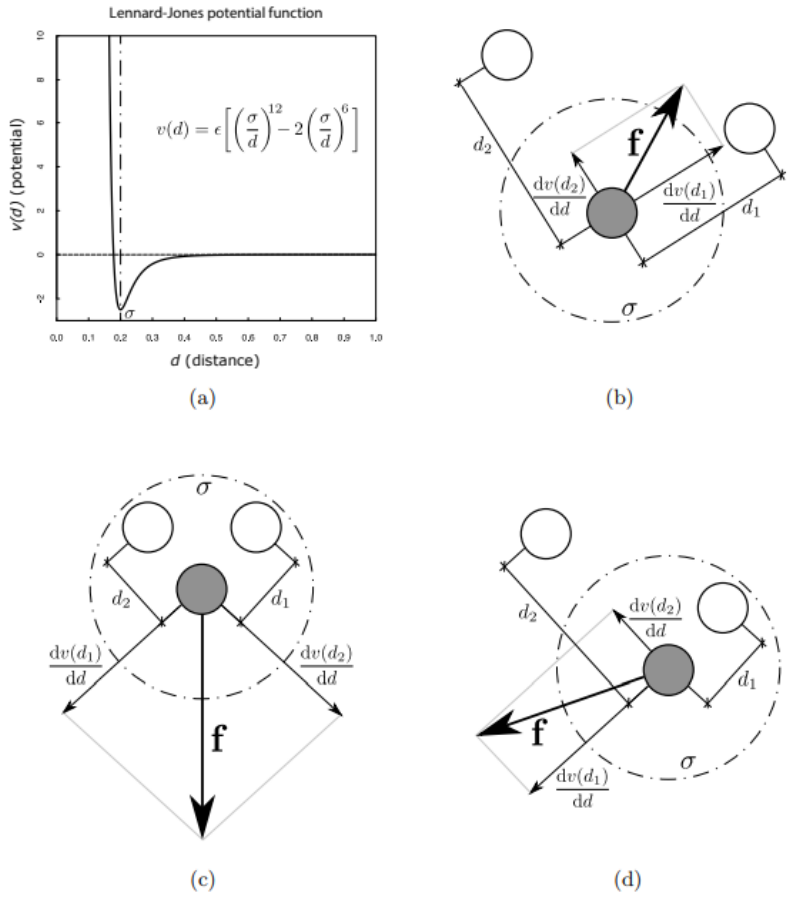


Figure 2.7 – La conception virtuelle basée sur la physique [49].

Le principal avantage de la conception basée sur la physique virtuelle est qu'elle permet au concepteur de prouver formellement les propriétés des comportements de l'essaim, notamment la stabilité, la convergence et la robustesse. Une extension de la conception virtuelle basée sur la physique est la méthode Hamiltonienne [70]. À partir d'une description mathématique de l'essaim au niveau macroscopique, la méthode hamiltonienne déduit

le comportement microscopique qui minimise ou maximise la valeur d'une quantité pertinente (par exemple, l'énergie potentielle virtuelle de l'état de l'essaim). Le principal inconvénient de la conception virtuelle basée sur la physique et de la méthode hamiltonienne est qu'elles ne conviennent que pour la conception de comportements collectifs à organisation spatiale.

d) Processus de conception automatique intégré pour les essaims de robots

C'est un processus de conception automatique hors ligne constitué de trois phases : la spécification des besoins, la conception par optimisation et le déploiement du logiciel de contrôle sur les robots [20].

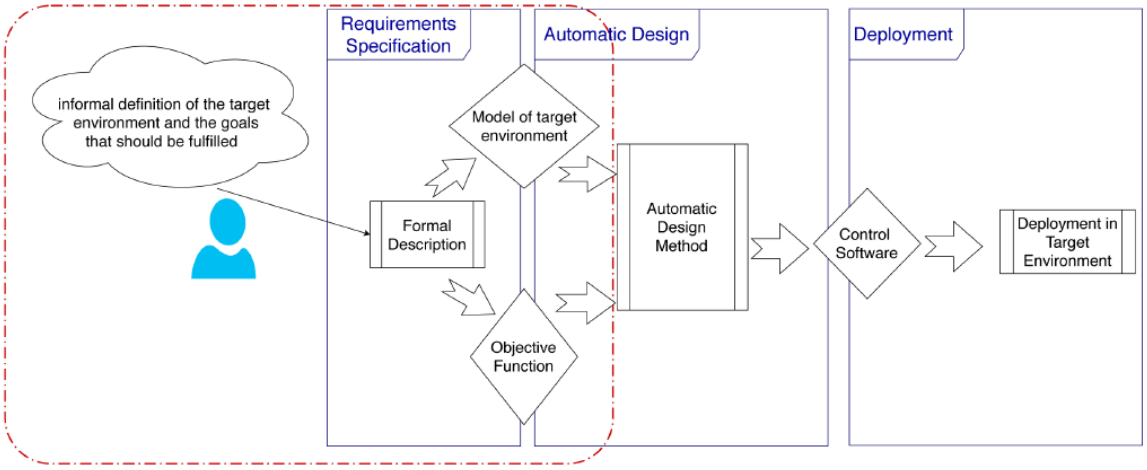


Figure 2.8 – Processus de conception automatique intégré pour les essaims de robots. [20].

Première phase - La spécification des exigences : Dans cette phase, le concepteur identifie et déclare toutes les caractéristiques de l'essaim de robots, l'environnement cible dans lequel il va opérer, la mission qu'il doit accomplir, l'objectif qu'il doit remplir, les contraintes éventuelles, etc. Dans l'état actuel de la technique, aucun processus standard n'a été défini pour la collecte des exigences. En général, les concepteurs spécifient les missions de manière informelle, ce qui rend les spécifications vagues et empêche finalement de vérifier si l'essaim développé satisfait ou non aux exigences. En partant des exigences, le concepteur définit une fonction objectif qui sera ensuite optimisée dans la deuxième phase. Comme les exigences sont spécifiées de manière informelle, cette étape doit être réalisée manuellement et est discrétionnaire, non reproductible et sujette à erreur.

Deuxième phase - La conception par optimisation : Dans cette phase, le logiciel de commande des robots individuels composant l'essaim est produit par une méthode de conception automatique. Une méthode de conception automatique est définie par : (i) un modèle de référence de la plateforme robotique pour laquelle elle peut concevoir un logiciel de commande ; (ii) un algorithme d'optimisation ; et (iii) l'espace des logiciels de commande qu'elle peut éventuellement produire. Le modèle de référence est une abstraction de la plateforme robotique qui spécifie en termes formels les caractéristiques et les capacités des robots ; l'algorithme d'optimisation est l'algorithme qui dirige le processus d'optimisation ;

et l'espace du logiciel de commande qui peut être produit est généralement exprimé par une architecture paramétrique et par l'ensemble des valeurs possibles de ses paramètres.

Troisième phase - Le déploiement du logiciel de contrôle sur les robots : Cette phase consiste en toutes les activités liées au transfert du logiciel de contrôle produit vers les robots dans l'environnement cible. Il existe des outils capables de générer du code qui peut être directement porté sur les robots. Par exemple, ARGoS, un simulateur multi-moteur pour les essaims de robots, peut actuellement générer du code pour un certain nombre de plateformes, notamment marXbot, e-puck, Thymio, Kilobot et Khepera IV. Grâce à sa nature modulaire, ARGoS peut être étendu pour générer du code pour une variété de plateformes robotiques.

2.5 Conclusion

Généralement, les essaims de robots ont été principalement conçus manuellement : le comportement individuel est amélioré et testé de manière itérative jusqu'à ce que le comportement collectif souhaité soit atteint. Cette approche non systématique n'est ni fiable ni cohérente. La qualité de la solution obtenue dépend fortement de l'expérience et de l'intuition du concepteur. Dans ce chapitre, nous avons présenté les différentes méthodes et essais développés par les chercheurs en conception de systèmes robotiques en essaim. Nous avons également défini les avantages et les inconvénients de ces approches, les limites de chaque méthode.

Pour éviter, ou du moins réduire, les inconvénients de ces approches, nous présenterons dans le chapitre suivant une méthodologie de conception bien structurée pour la conception des robots en essaim. Cette méthodologie de conception est basée sur la méthode MBSE pour spécifier les exigences et les comportements collectifs des essaims, puis sur la vérification des modèles développés et enfin sur la validation du système en essaim par prototypage physique avec des robots réels en utilisant Robot Operating System (ROS).

Deuxième partie

Contribution

Chapitre 3

Méthodologie de conception proposée pour la conception d'un système robotique en essaim

Rien ne va de soi. Rien n'est donné. Tout est construit.

– Gaston Bachelard

3.1 Introduction

Dans la robotique en essaim, les robots résolvent des problèmes en utilisant des comportements collectifs similaires à ceux observés dans les systèmes naturels, comme les oiseaux, les abeilles ou les poissons. Ils déterminent leur comportement collectif par le biais de plusieurs interactions simples. Cependant, le comportement des essaims émergeant d'interactions locales reste difficile à prévoir. Lorsqu'ils essaient de concevoir des systèmes robotiques en essaim pour des applications réelles, les chercheurs sont confrontés à un large éventail de défis logiciels et matériels dans les différentes phases du processus de conception ; des problèmes liés à la modélisation des systèmes en essaim, à la simulation, à l'implémentation sur des logiciels et même sur des systèmes réels, etc. Jusqu'à aujourd'hui, il n'existe pas de méthodologie complète ou d'approche systématique qui puisse être suivie pour concevoir un système robotique en essaim. Notre contribution consiste dans cette thèse en une nouvelle méthodologie intégrée pour le développement de systèmes robotiques en essaim. Cette méthodologie est basée sur la modélisation avec la méthode MBSE pour spécifier les exigences et les comportements collectifs des essaims, puis sur la vérification des modèles développés et enfin sur la validation du système en essaim par prototypage physique avec des robots réels. Notre contribution se concentre également sur le développement d'un nouveau profil SysML utilisant le Domain Specific Language (DSL) que nous

appelons *SwarmML* pour personnaliser la modélisation fonctionnelle et structurelle d'un système en essaim (propriétés et attributs de l'essaim). L'avantage de cette méthodologie est qu'elle permet d'assurer la continuité du processus de conception à chaque étape de la conception, c'est-à-dire que la transition entre la phase de spécification des exigences et la phase de modélisation comportementale et structurelle est vérifiée soit par des matrices d'allocation, soit par une comparaison des données présentées sur les diagrammes SysML. De plus, SysML est un outil de modélisation standard, il est incapable de décrire précisément les propriétés spécifiques du système robotique en essaim. Pour cette raison, nous utilisons au cours de cette méthodologie le DSL pour créer un profil SwarmML. Le but de ce profil est de modéliser les fonctions individuelles de chaque membre de l'essaim et le comportement global de l'essaim entier en montrant les caractéristiques de l'essaim. Un autre avantage de cette méthodologie est qu'elle facilite l'intégration du système modélisé avec la méthode MBSE sur ROS puisque les modèles développés avec DSL sont prêts et compatibles avec les concepts ROS (Topics, nodes, services, ...). La validation du modèle simulé est très simple dans notre méthodologie ; soit par visualisation directe sur les outils de simulation (RVIZ, Gazebo, Unity, ...), soit par comparaison des données modélisées sur les diagrammes SysML/SwarmML et les données trouvées sur les diagrammes générés par ROS (rqt_graph). Dans ce chapitre, nous allons décrire notre méthodologie de conception IDMSR développée pour la conception d'un système robotique en essaim. En effet, nous allons détailler chaque phase et chaque étape de cette méthodologie avec les différents outils utilisés.

3.2 Méthodologie de conception développée : IDMSR

3.2.1 Principe de la méthodologie IDMSR

Le principe de notre méthodologie de conception developpée - Integrated Design Methodology of Swarm Robots (IDMSR) - est qu'un système robotique en essaim peut être décrit par un ensemble de propriétés et d'exigences pour réaliser une telle mission. Cette méthodologie est basée sur la modélisation avec la méthode MBSE pour spécifier les exigences et les comportements collectifs des essaims, puis sur la vérification des modèles développés et enfin sur la validation du système en essaim par prototypage physique avec des robots réels. Comme SysML est un langage général multi-vues pour la modélisation de systèmes, un ensemble de diagrammes a été développé comme un nouveau profil que nous avons appelé "SwarmML". Nous utilisons un langage spécifique au domaine (DSL) comme nouveau métamodèle pour personnaliser les propriétés et les attributs du système robotique en essaim.

3.2.2 Les outils et les techniques utilisés

a - L'ingénierie des systèmes

L'ingénierie des systèmes - Systems engineering (SE)- est une approche et interdisciplinaires pour permettre la réalisation de systèmes performants. Elle se concentre sur la

définition des besoins du client et des fonctionnalités requises au début du cycle de développement, sur la documentation des exigences, puis sur la synthèse de la conception et la validation des systèmes tout en considérant le problème dans son ensemble (opérations, performances, essais, fabrication, coût et délais).

Selon International Council on Systems Engineering (INCOSE), un système est un arrangement de parties ou d'éléments qui, ensemble, présentent un comportement ou une signification que les constituants individuels ne présentent pas. Les systèmes peuvent être physiques ou conceptuels, ou une combinaison des deux. Les systèmes de l'univers physique sont composés de matière et d'énergie, peuvent contenir des informations codées dans des supports de matière et d'énergie, et présentent un comportement observable. Les systèmes conceptuels sont des systèmes abstraits d'informations pures, qui ne présentent pas directement un comportement, mais un "sens".[124].

b - La méthode MBSE

L'ingénierie des systèmes basée sur les modèles - Model-based systems engineering (MBSE) - selon l'INCOSE, est l'application formalisée de la modélisation pour soutenir les exigences du système, la conception, l'analyse, la vérification et la validation des activités commençant dans la phase de conception et se poursuivant tout au long du développement et des phases ultérieures du cycle de vie. [L'approche technique MBSE est une approche technique de l'ingénierie des systèmes qui se concentre sur la création et l'exploitation de modèles de domaine comme moyen principal d'échange d'informations, plutôt que sur l'échange d'informations basé sur des documents [46]. Le mot méthodologie est souvent considéré comme synonyme du mot processus. Pour les besoins de cette étude, les définitions suivantes de Martin [88] sont utilisées pour distinguer la méthodologie du processus, des méthodes et des outils :

- *Un processus (P)* est une séquence logique de tâches effectuées pour atteindre un objectif particulier. Un processus définit "QUOI" doit être fait, sans spécifier "COMMENT" chaque tâche est exécutée. La structure d'un processus fournit plusieurs niveaux d'agrégation pour permettre l'analyse et la définition à différents niveaux de détail pour soutenir différents besoins de prise de décision.

- *Une méthode (M)* consiste en des techniques pour réaliser une tâche, en d'autres termes, elle définit le "COMMENT" de chaque tâche. (Dans ce contexte, les mots "méthode", "technique", "pratique" et "procédure" sont souvent utilisés de manière interchangeable). À tout niveau, les tâches du processus sont exécutées à l'aide de méthodes. Cependant, chaque méthode est également un processus en soi, avec une séquence de tâches à réaliser pour cette méthode particulière. En d'autres termes, le "COMMENT" à un niveau d'abstraction devient le "QUOI" au niveau inférieur suivant.

- *Un outil (T)* est un instrument qui, lorsqu'il est appliqué à une méthode particulière, peut améliorer l'efficacité de la tâche, à condition qu'il soit appliqué correctement et par une personne possédant les compétences et la formation appropriées. L'objectif d'un outil doit être de faciliter l'accomplissement des "COMMENT". Dans un sens plus large, un outil améliore le "QUOI" et le "COMMENT". La plupart des outils utilisés pour soutenir l'ingénierie des systèmes sont basés sur des ordinateurs ou des logiciels, également connus sous le nom d'outils d'ingénierie assistée par ordinateur (IAO).

La figure 3-1 présente un graphique qui décrit la relation entre les éléments dits "PMTE" (processus, méthodes, outils et environnement), ainsi que les effets de la technologie et des personnes sur les éléments PMTE.

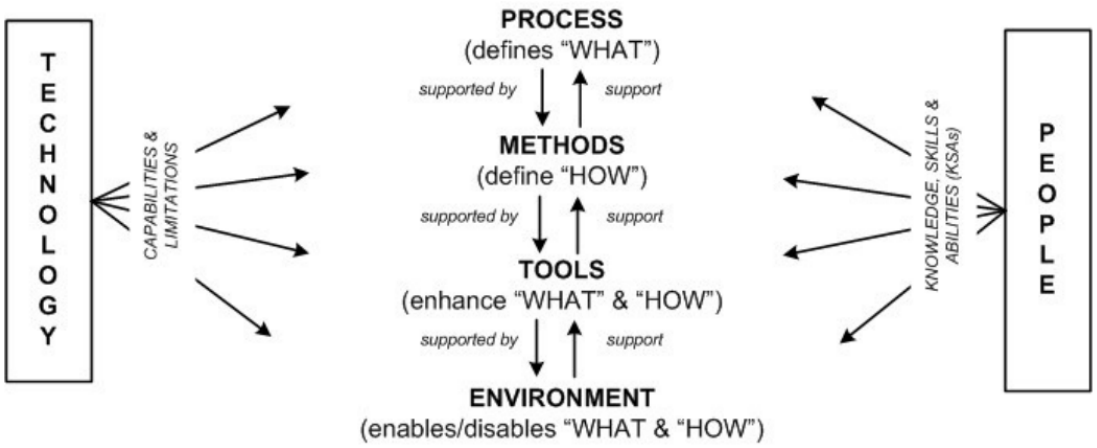


Figure 3.1 – Les Éléments PMTE et les effets de la technologie et des personnes [88].

c - Systems Modeling Language (SysML)

SysML est un langage de modélisation visuelle d'usage général pour les applications d'ingénierie des systèmes qui prend en charge la spécification, la conception, l'analyse et la vérification des systèmes qui peuvent inclure du matériel, des logiciels, des données, du personnel, des procédures et des installations. En générale, SysML est un langage de modélisation graphique avec une base sémantique pour représenter les exigences, le comportement, la structure et les propriétés du système et de ses composants [57].La taxonomie des diagrammes SysML est présentée à la figure 3.2.

La structure du système est représentée par des diagrammes de définition de blocs et des diagrammes de blocs internes. Un diagramme de définition de bloc décrit la hiérarchie du système et les classifications des systèmes/composants. Le diagramme de bloc interne décrit la structure interne d'un système en termes de ses parties, ports et connecteurs. Le diagramme de blocs est utilisé pour organiser le modèle.

Les diagrammes de comportement comprennent le diagramme de cas d'utilisation, le diagramme d'activité, le diagramme de séquence et le diagramme des états. Un diagramme de cas d'utilisation fournit une description de haut niveau de la fonctionnalité du système. Le diagramme d'activité représente le flux de données et de contrôle entre les activités. Un diagramme de séquence représente l'interaction entre les parties collaboratrices d'un système. Le diagramme des états décrit les transitions d'état et les actions qu'un système ou ses parties exécutent en réponse à des événements.

Le diagramme d'exigences capture les hiérarchies d'exigences et les relations de dérivation, de satisfaction, de vérification et de raffinement. Ces relations permettent de relier les exigences entre elles et de les relier aux modèles de conception du système et aux cas de test. Le diagramme d'exigences fournit un pont entre les outils typiques de gestion des exigences et les modèles de système. Le diagramme paramétrique représente les contraintes sur les valeurs des paramètres du système, telles que les propriétés de performance, de fiabilité et

de masse, afin de soutenir l'analyse technique. SysML comprend une relation d'allocation pour représenter divers types d'allocation, y compris l'allocation des fonctions aux composants, des composants logiques aux composants physiques. fonctions aux composants, des composants logiques aux composants physiques et du logiciel au matériel [57].

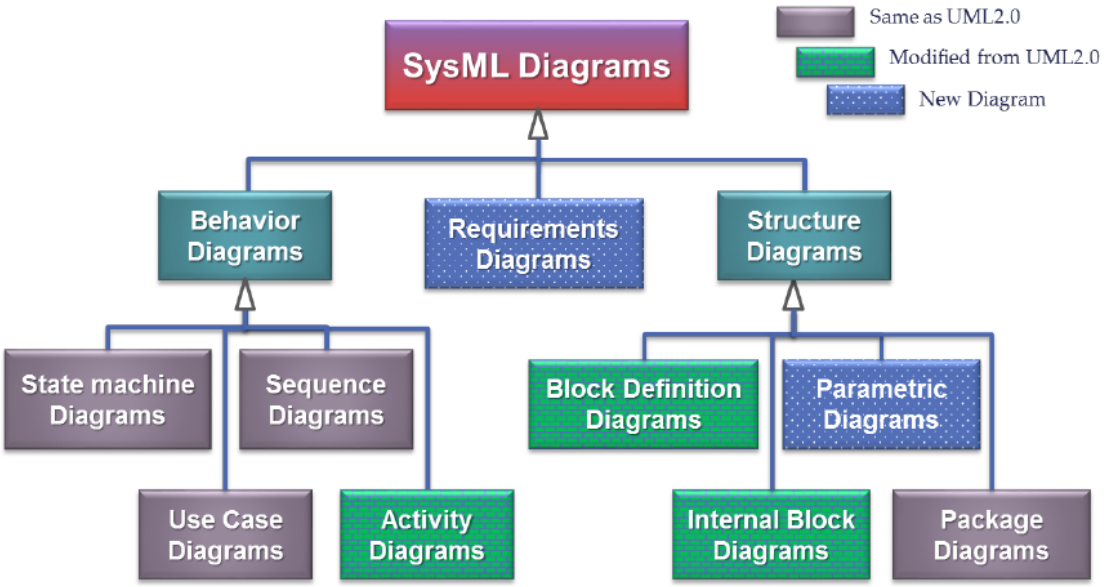


Figure 3.2 – Taxonomie des diagrammes SysML [57].

d - Domain Specific Language DSL

Un langage dédié - domain-specific language (DSL) - est un langage de programmation avec un niveau d'abstraction supérieur, optimisé pour une catégorie spécifique de problèmes. Un langage DSL utilise les concepts et les règles du secteur ou du domaine. Un langage DSL est généralement moins complexe qu'un langage générique comme Java ou C. Dans la plupart des cas, les DSL sont élaborés en étroite collaboration avec les experts du domaine pour lequel le DSL est conçu. Dans de nombreux cas, les langages DSL sont destinés à être utilisés non par des développeurs de logiciels, mais par des non-programmeurs qui maîtrisent le domaine auquel s'adresse le DSL. Dans notre cas, nous utilisons le DSL comme un nouveau métamodèle pour personnaliser les propriétés et les attributs du système robotique en essaim puisque SysML est un langage plus standard et général.

e - Robot Operating System ROS

Robot Operating System (ROS) est un méta-système d'exploitation à code source ouvert pour les robots. Il fournit les services attendus d'un système d'exploitation, notamment l'abstraction matérielle, le contrôle des périphériques de bas niveau, l'implémentation de fonctionnalités courantes, le passage de messages entre les processus et la gestion des paquets. Il fournit également des outils et des bibliothèques pour obtenir, construire, écrire et exécuter du code sur plusieurs ordinateurs [116].

ROS est caractérisé par trois types de communication : les services, les topics et les actions. Chaque nœud doit être chargé d'un objectif unique et modulaire (par exemple, un nœud pour contrôler un télémètre laser, un nœud pour contrôler les moteurs de roue, etc.)

En pratique, chaque nœud peut recevoir et envoyer des données aux autres nœuds par le biais de services, de sujets et d'actions. Les éditeurs de topics envoient des messages unidirectionnels à plusieurs abonnés. Les clients de services envoient une nouvelle demande à un serveur de services et reçoivent un résultat, sans disposer d'informations sur la progression. Comme les services, les clients d'action envoient une demande à un serveur d'action pour atteindre un certain objectif et reçoivent une réponse. Contrairement aux services, pendant l'exécution de l'action, un serveur d'action envoie un retour d'information sur la progression au client. Les actions permettent à un client d'annuler la demande avant qu'elle ne soit terminée, de suivre la progression d'une demande et d'obtenir le résultat final.

ROS peut être utilisé sur Windows, MAC, Ubuntu et Debian. La Figure 3.3 représente la différence entre les deux versions de ROS. Bien que ROS 1 et ROS 2 contiennent les mêmes fonctions de base, des différences existent entre les deux. Cependant, ROS 1 utilise un format de sérialisation personnalisé, un protocole de transport personnalisé ainsi qu'un mécanisme de découverte central personnalisé. ROS 2 dispose d'une interface intergiciel abstraite, par laquelle la sérialisation, le transport et la découverte sont fournis. Actuellement, toutes les implémentations de cette interface sont basées sur le standard DDS.

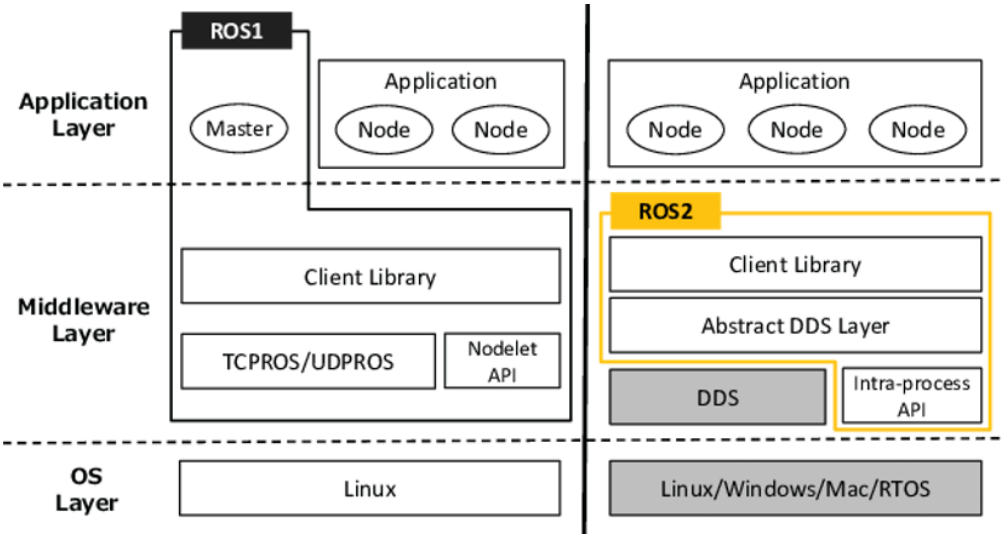


Figure 3.3 – Différence entre ROS1 et ROS2 [84].

3.2.3 Description de la méthodologie IDMSR

La méthodologie que nous proposons comprend deux phases : une phase descendante basée sur la méthode MBSE, de la spécification des besoins à la modélisation fonctionnelle et structurelle basée sur SysML et les nouveaux diagrammes de profil SwarmML, et une phase ascendante pour l'intégration des modèles et la mise en œuvre dans ROS/ROS2.

Le concepteur de l'essaim commence par spécifier les exigences de conception en utilisant le diagramme d'exigences SysML pour décrire les différents besoins du système de l'essaim. À partir de ces exigences, le concepteur identifie les différentes fonctions qui construisent les comportements collectifs de l'essaim en utilisant des diagrammes de profil

SwarmML personnalisés tels que le diagramme de comportement des robots individuels (*Individual Robot Behavior Diagram IRBD*) pour développer les fonctions individuelles de chaque robot, le diagramme de comportement de l'essaim collectif (*Collective Swarm Behavior Diagram CSBD*) pour modéliser les comportements collectifs de l'essaim et le diagramme de mission de l'essaim (*Swarm Mission Diagram SMD*) pour définir la mission de l'essaim dans ROS. Ces diagrammes sont créés à l'aide du langage défié DSL et seront décrits plus en détail ultérieurement. Pour assurer une traçabilité de haut niveau entre les exigences, les comportements et les fonctions, le concepteur utilise des matrices d'allocations (Exigence-Comportement, Comportement-Fonction). Ces matrices relient les exigences spécifiées aux fonctions que le système doit exécuter tout en respectant le comportement de l'essaim. Enfin, le concepteur détaille la structure du système en spécifiant les composants capables de réaliser les fonctions précédemment modélisées que le système doit exécuter.

Pour assurer l'intégration du code, le concepteur suit une approche ascendante guidée par le profil SysML/SwarmML pour mettre en œuvre le comportement de l'essaim avec ROS/ROS2. À ce niveau de la conception, la structure du système de l'essaim est modélisée par un fichier URDF (Unified Robot Description Format) basé sur de nouveaux diagrammes SysML personnalisés. En outre, l'environnement du système est décrit dans ROS/ROS2 par la création d'un fichier world basé sur *Swarm Mission Diagram SMD* . La simulation finale du comportement de l'essaim est performée avec ROS couplé avec un outil de visualisation tel que Gazebo, Unity ou Rviz pour répondre aux exigences décrites avec les diagrammes d'exigences SysML.

La vérification/validation (VV) est assurée à chaque étape du processus de conception soit par la comparaison entre les diagrammes développés avec SysML/SwarmML et les diagrammes générés par ROS/ ROS2 comme la comparaison du Swarm Mission Diagram avec le rqt-graph généré par ROS, soit par une validation visuelle comme la visualisation de la mission simulée sur Gazebo ou réalisée sur des robots réels. La Figure 3.4 représente les étapes de la méthodologie de conception intégrée proposée. Ces étapes sont détaillées dans les sections suivantes.

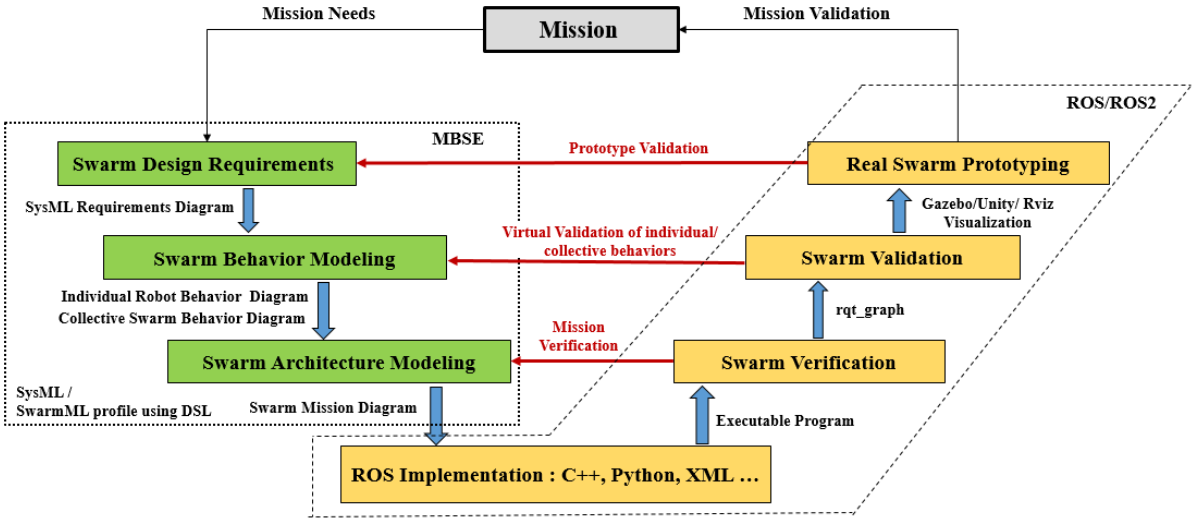


Figure 3.4 – Méthodologie de conception intégrée de systèmes en essaim : IDMSR.

• PHASE 1 : Modélisation avec la méthode MBSE

- Exigences de conception de l'essaim :

Dans la première étape, les diagrammes d'exigences SysML sont utilisés pour définir les exigences que l'essaim doit satisfaire. Une exigence décrit une capacité, une propriété ou un comportement que le système doit satisfaire. Le développeur de l'essaim doit donc s'intéresser aux exigences décrites par l'utilisateur, qu'elles soient complètes ou partielles (besoin du client). Ainsi, les exigences principales définissent la mission globale du système. Ces exigences peuvent être complétées par des exigences supplémentaires (exigences techniques) qui répondent aux besoins du client. Ces exigences sont divisées en deux sous exigences : (matériel et logiciel). Parmi les exigences relatives aux essaims matériels, on trouve la flexibilité, la robustesse, l'évolutivité, la maintenance et l'énergie. En outre, certaines exigences logicielles pour un système de robot en essaim sont citées, comme la conception de mécanismes qui prennent en charge la connectivité orientée vers un but, la sélection de systèmes pour une communication centralisée et décentralisée appropriée, l'auto-assemblage, la reconfigurabilité et l'autonomie.

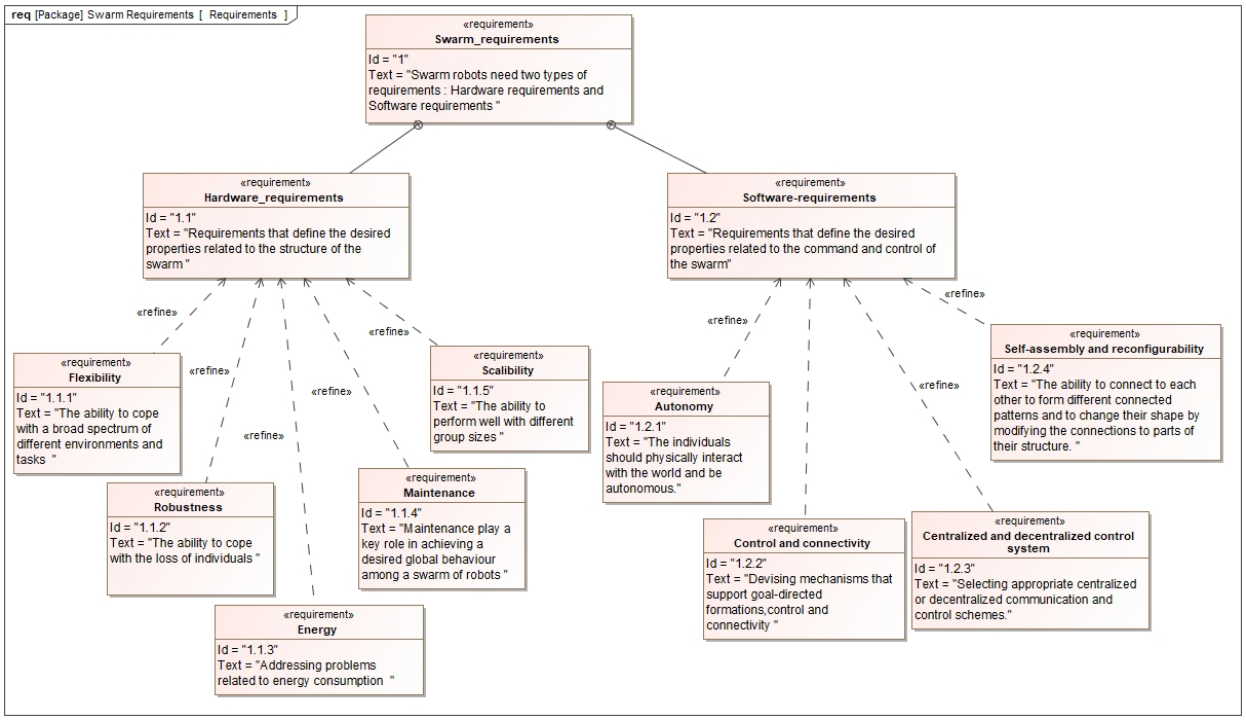


Figure 3.5 – Les exigences de conception d'un système robotique en essaim

- Modélisation du comportement de l'essaim :

En raison de leur flexibilité et de leur robustesse, les robots en essaim ont de nombreux comportements collectifs. Ce sont les comportements collectifs de base d'un essaim pour gérer des applications complexes du monde réel, comme la construction ou la recherche de nourriture. Ces comportements sont classés par Brambilla 2013 en quatre catégories : comportements de navigation, comportements d'organisation spatiale, prise de décision collec-

tive, et autres comportements collectifs. Dans cette étape, nous modélisons les différents comportements collectifs des robots de l'essaim. Ces comportements sont exprimés par un ensemble de fonctions exécutées par chaque robot du groupe. Le développeur peut utiliser le diagramme des états de SysML pour modéliser ces fonctions réalisées par les robots individuels. Nous appelons ce diagramme "Individual robot behavior diagram IRBD", illustré à la Figure 3.6. En effet, les différentes fonctions offertes par le système doivent être définies. Une fonction est une action réalisée par le système ou par l'une de ses parties. L'ensemble des fonctions des robots individuels peut produire un comportement complet de l'essaim. Chaque robot change d'état en déclenchant une transition pour créer un comportement collectif avec ses voisins du groupe.

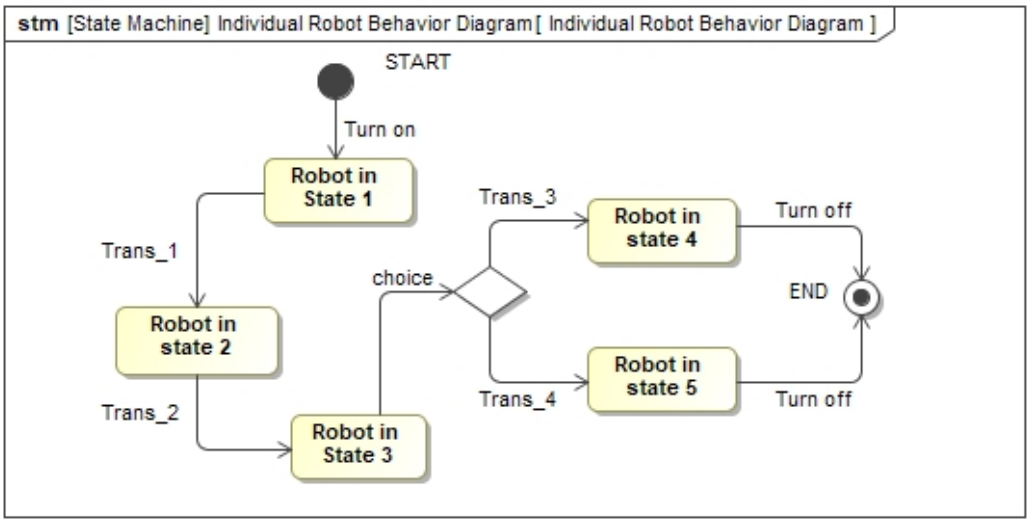


Figure 3.6 – Diagramme de comportement individuel du robot IRBD

Par exemple, pour assurer certains comportements collectifs tels que la formation de motifs, l'aggrégation ou l'auto-assemblage, le robot avance, repousse, attend et s'approche des autres robots de manière autonome. Les fonctions principales du système sont décomposées en sous-fonctions qui sont connectées afin de transformer les flux d'entrée en flux de sortie. Les comportements collectifs des robots sont définis par l'ensemble des fonctions offertes par le système. Pour assurer le comportement de recherche de nourriture, chaque robot du groupe doit passer par les états suivants : recherche, saisie, localisation et dépose.

Une fois les fonctions des robots individuels modélisées, le développeur peut utiliser le diagramme d'activité de SysML pour modéliser le comportement collectif de l'ensemble du groupe de robots. Nous appelons ce diagramme le "Collective Swarm Behavior Diagram CSBD" illustré à la Figure 3.7. Grâce à ce diagramme, le développeur décrit le rôle de chaque robot pour prédire le comportement collectif de l'essaim. Ce comportement est constitué de l'ensemble des actions réalisées par chaque robot de l'essaim au cours du temps. Par exemple, pour le comportement de recherche de nourriture, chaque robot de l'essaim doit suivre le processus décrit par le diagramme de comportement individuel du robot développé ci-dessus.

Pour assurer la continuité de la méthodologie entre l'aspect individuel de chaque robot et l'aspect collectif de l'essaim, le développeur doit dessiner une matrice d'allocation qui

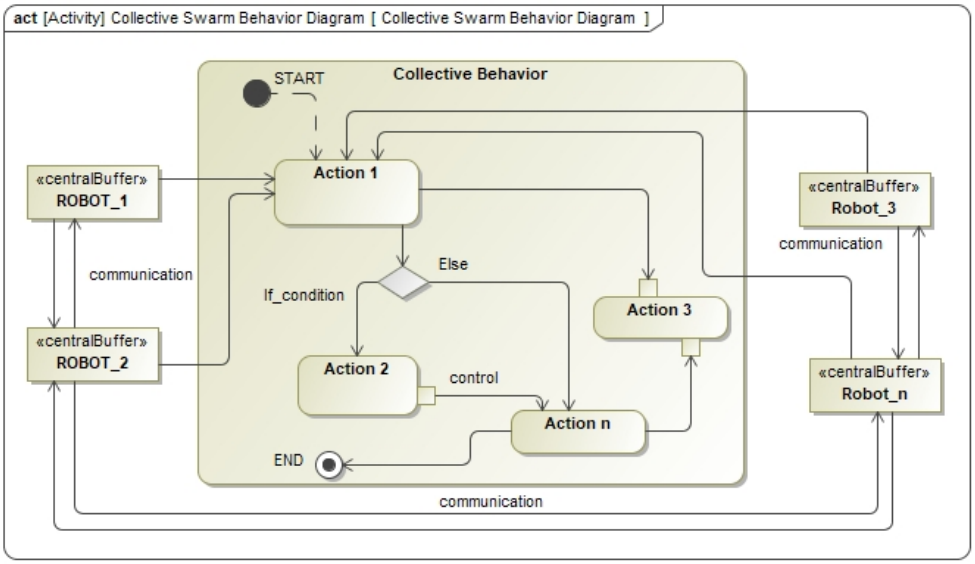


Figure 3.7 – Diagramme du comportement collectif de l'essaim CSBD

relie chaque comportement collectif de l'essaim aux fonctions individuelles réalisées par ses membres robots. La matrice présentée dans la Figure 3.8 résume la relation entre la plupart des comportements collectifs et les fonctions individuelles des robots de l'essaim.

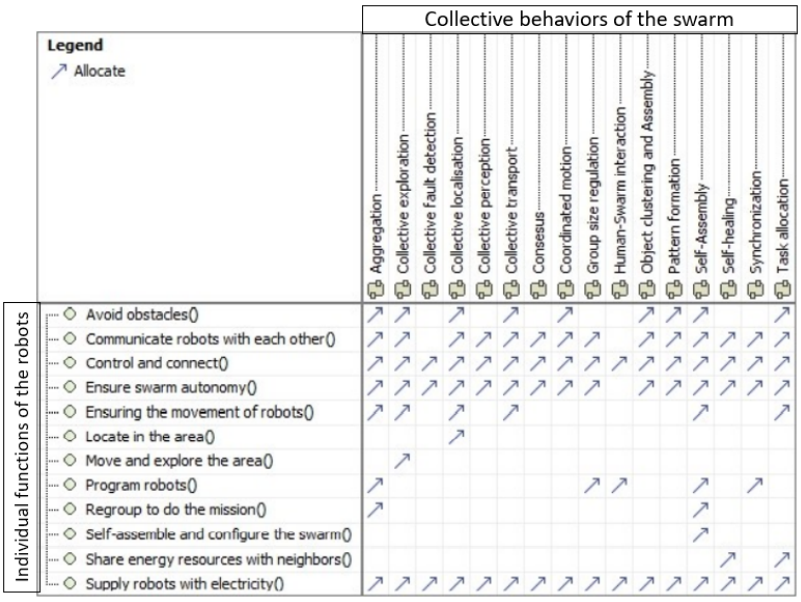


Figure 3.8 – Matrice d'allocation (Fonction-comportement)

- Modélisation de l'architecture de l'essaim :

Dans cette étape, le développeur doit décrire la mission du système de robot en essaim. Généralement, la mission est un ensemble de comportements collectifs réalisés par l'essaim pour assurer une application telle que le nettoyage ou la surveillance. Dans cette étape, il est difficile d'utiliser des diagrammes SysML standards car la définition d'une mission

d'essaimage sur ROS nécessite de nouvelles propriétés et attributs du système en essaim. Pour ces raisons, nous développons un nouveau diagramme que nous appelons "Swarm Mission Diagram SMD" en utilisant le langage dédié DSL qui décrit la mission à travers les concepts et attributs de ROS (nœuds, topics, services et actions). Le développeur de l'essaim peut utiliser le nouveau diagramme illustré à la Figure 3.9 pour décrire la mission de l'essaim et, avec cette représentation, l'implémentation sur ROS sera plus facile par la suite. Dans ce diagramme, nous choisissons la forme ellipse pour représenter le nœud et la forme rectangle pour représenter le topic.

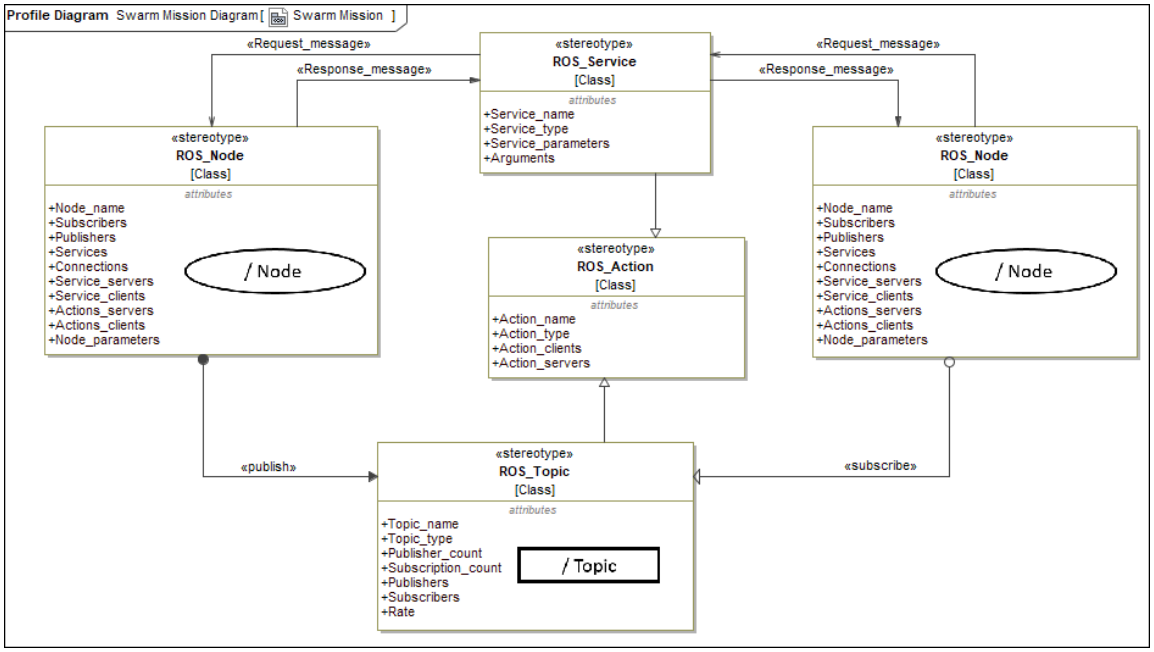


Figure 3.9 – Diagramme de mission d'essaim SMD en utilisant le langage dédié DSL

Ensuite, les spécifications des capacités matérielles des robots doivent être dérivées. La figure 3.10 représente l'architecture générale de l'essaim. Si l'essaim est homogène, alors ses membres sont homogènes. Dans le cas contraire, ses membres deviennent hétérogènes.

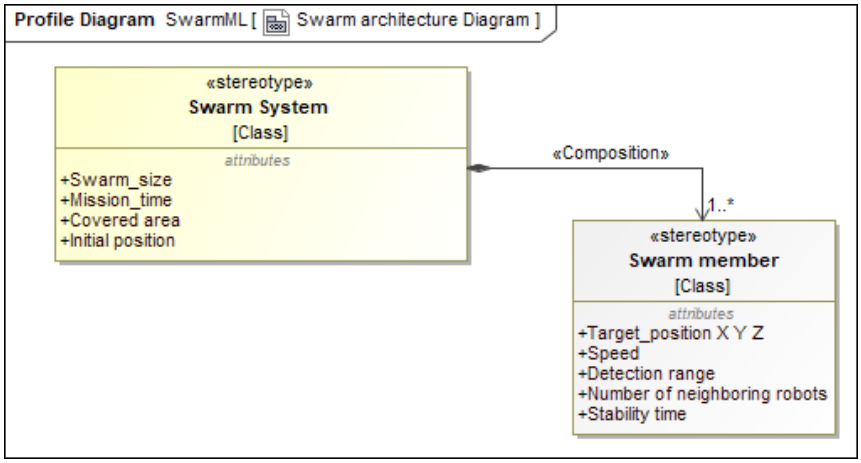


Figure 3.10 – Architecture générale de l'essaim utilisant le langage dédié DSL

• PHASE 2 : Implémentation à l'aide de ROS/ROS2

- Implémentation sur ROS :

Pour vérifier le développement effectué avec MBSE dans la phase précédente, les modèles développés du système en essaim doivent être implémentés dans l'environnement ROS. À cette fin, le concepteur crée un espace de travail sur ROS composé d'un ensemble de packages. Pour tenir compte du comportement de l'essaim, les robots individuels sont associés à des nœuds dans le système ROS, et un topic de communication est créé pour souscrire et afficher les informations échangées entre les robots individuels. Les packages ROS sont mis en œuvre sur la base des différents diagrammes SysML/SwarmML développés lors de la première phase de conception. En effet, le diagramme de comportement des robots individuels IRBD, le diagramme de comportement collectif de l'essaim CSBD et le diagramme de mission de l'essaim SMD sont utilisés pour créer les codes C++, python, ..., le diagramme de définition des blocs BDD et les diagrammes de blocs internes IBD sont utilisés pour créer le fichier URDF et le fichier WORLD dans ROS. Les codes décrivent le comportement des robots individuels et le comportement de l'essaim en général, le fichier URDF décrit la structure générale du système de l'essaim et le fichier WORLD décrit l'environnement de travail. En outre, le diagramme de comportement du robot individuel IRBD et le diagramme de comportement collectif de l'essaim CSBD sont également utilisés pour développer les algorithmes de contrôle du système en essaim dans ROS. La Figure 3.11 montre la méthodologie d'intégration de ROS.

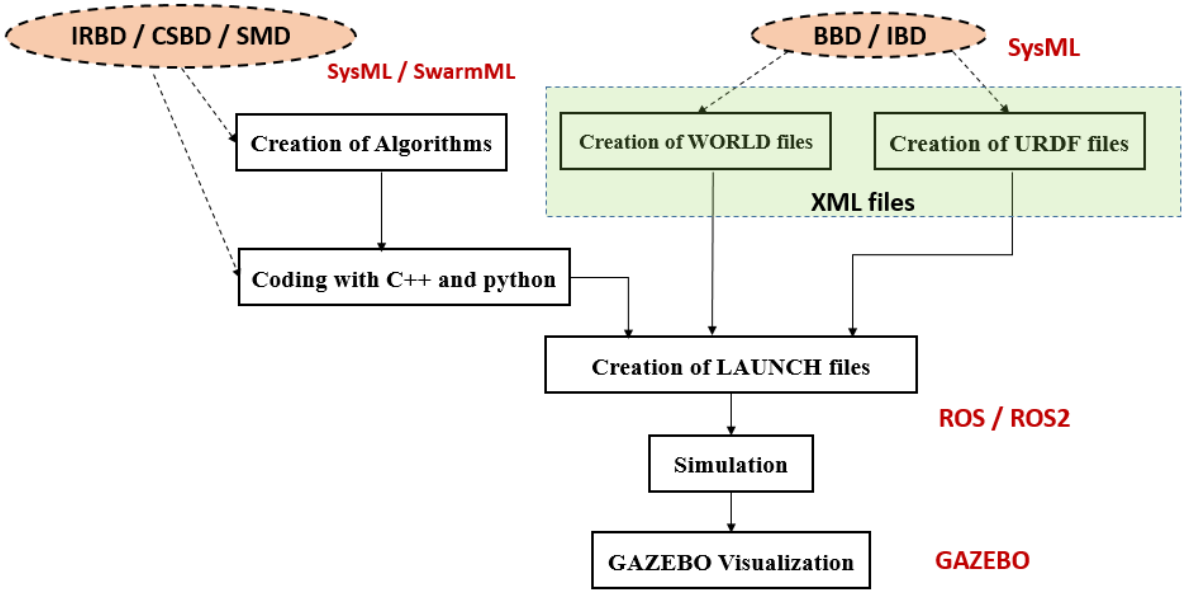


Figure 3.11 – Intégration de ROS à l'aide de SysML/ SwarmML

L'une des caractéristiques de ROS est la réutilisation du code développé par d'autres, qui peut être trouvé et partagé par les utilisateurs de ROS (sur le site GitHub, par exemple). Les utilisateurs peuvent trouver des versions écrites en Python et en C++ qui diffèrent en termes de restrictions, de performances et de légèreté du code. Les développeurs du système swarm peuvent les télécharger et les rendre utilisables en les compilant (à l'aide des processus de

construction catkin pour ROS et Colcon pour ROS2), et de nouvelles versions de ces codes seront disponibles sous forme d'autres packages basés sur ROS. Sur la base des descriptions SysML et SwarmML réalisées lors de la première étape de conception, les développeurs peuvent intégrer les codes les plus appropriés qui répondent aux exigences du système en essaim, de son architecture fonctionnelle et de son architecture physique. L'architecture logicielle sera alors plus facilement intégrée sur la base des packages ROS existants. La tâche du développeur consiste alors à adapter le code et à l'intégrer aux autres éléments de ROS.

Une fois les algorithmes et les codes développés sur ROS à travers les modèles SysML/SwarmML, le développeur exécute ces codes C++ et Python sur ROS. En effet, un espace de travail ROS composé de différents packages qui décrivent la structure de l'essaim et les fonctions fournies par chaque robot a été créé lors de cette étape. Lors de l'organisation des packages ROS, une bonne pratique consiste à les regrouper pour assurer la cohérence fonctionnelle. Cela peut être fait sous la forme d'un espace de travail. La Figure 3.12 montre l'architecture de cet espace de travail pour robot en essaim.

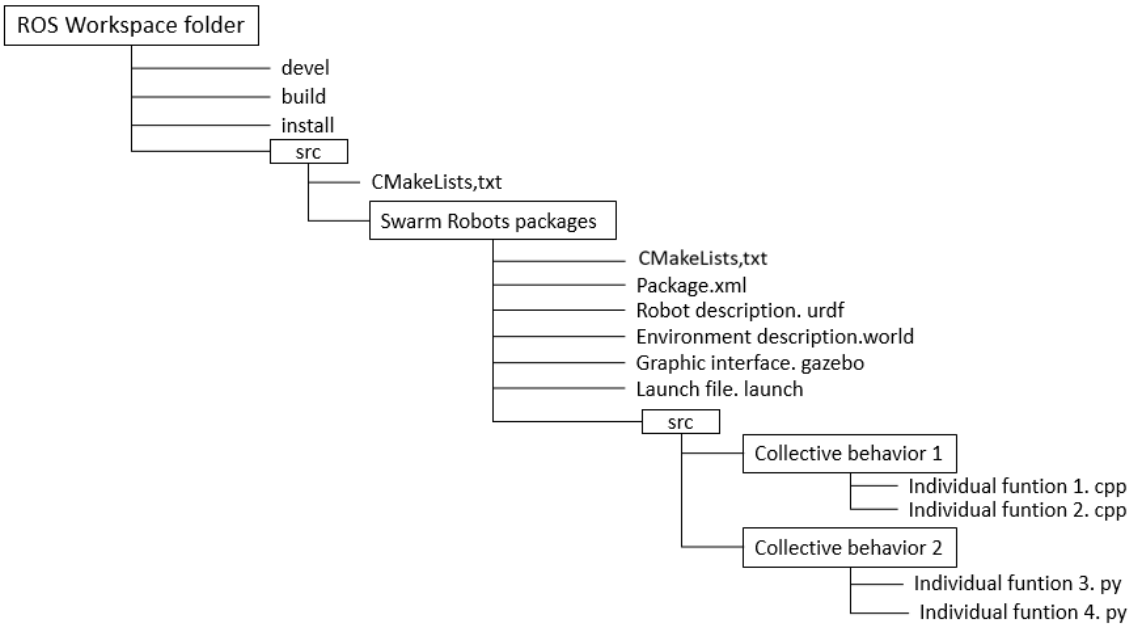


Figure 3.12 – Organisation de l'espace de travail d'un système de robot en essaim dans ROS

Dans cet espace de travail, un répertoire 'src' est créé dans lequel un répertoire de package contenant les fichiers d'un package peut être créé :

- *Le répertoire build* est utilisé pour créer les packages et contient donc tous les fichiers objets.

- *Le répertoire devel* contient les exécutables et les bibliothèques résultant de la compilation des packages et, par conséquent, spécifiés comme cible dans le fichier CMakeLists.txt.

- *Le répertoire install* ne contient que les fichiers qui sont explicitement installés par les directives d'installation spécifiées dans le fichier CMakeLists.txt des packages.

- *CMakeLists.txt* : Le fichier indiquant comment compiler et installer le package.

- *package.xml* : Le fichier ROS qui décrit l'identité du package et ses dépendances.

- *Un langage de description de la structure d'un essaim* : Unified Robot Description Format (URDF) est une spécification XML qui décrit à la fois la cinématique, les caractéristiques dynamiques, la géométrie et les capteurs d'un robot et décrit également la structure générale de l'essaim.

- *Un fichier WORLD* : ce fichier décrit l'environnement de travail. Il peut correspondre au diagramme de contexte (BDD) qui définit l'environnement SysML.

- *Un nœud* est un fichier exécutable dans un package ROS : Les nœuds ROS utilisent une bibliothèque client pour communiquer avec d'autres nœuds. Les nœuds peuvent s'abonner à des topics ou les publier. Les nœuds peuvent utiliser ou fournir un service. Les bibliothèques clientes sont rospy pour python et roscpp pour C++. Ces nœuds ROS contiennent les algorithmes qui traduisent les fonctions individuelles des robots.

- *Fichier de lancement* : Dans ce fichier, le développeur doit demander les autres fichiers (fichier urdf, fichier world, fonctions cpp, etc.) et initialiser les paramètres initiaux du système (taille de l'essaim, surface, etc.) pour décrire les comportements collectifs des robots de l'essaim.

- *L'interface graphique GAZEBO* représente la vue 3D finale du système en essaim.

- Vérification de l'essaim :

Dans cette étape, le programme doit être exécuté et l'essaim de robots doit fonctionner. Pour s'en assurer, le développeur peut extraire un graphe sur ROS qui s'appelle "rqt_graph". Rqt_graph est un plugin GUI de la suite d'outils Rqt. Avec rqt_graph, nous pouvons visualiser le graphique ROS de notre application swarm. Dans une seule fenêtre, nous pouvons voir tous nos nœuds en fonctionnement, ainsi que la communication entre eux. Les nœuds et les sujets seront affichés dans leurs namespaces. Mais, nous ne pouvons pas voir les services ROS dans le rqt_graph, seulement les sujets. Ceci est dû à la manière dont les services ont été implémentés. Comme nous l'avons vu lors du développement avec ROS, nous organisons généralement notre travail en packages et en nœuds. Au fur et à mesure que notre application en essaim se développe (plus d'actionneurs, plus de capteurs, plus de moyens de contrôler les robots, ...), il en va de même pour notre base de code. Par conséquent, nous aurons de plus en plus de nœuds, avec de plus en plus de communication entre eux (services, topics, actions). En outre, et c'est l'une des caractéristiques de ROS, nous utiliserons des nœuds ROS existants dans notre application. Par exemple, nous pouvons utiliser le nœud move_group (Moveit) pour la planification des mouvements. De cette façon, nous nous retrouvons avec de nombreux nœuds et topics, et cela peut devenir plus difficile à débugger. L'utilisation de rqt_graph nous aidera principalement avec ces deux choses : - Nous aurons une vue d'ensemble de notre système. Ceci est vraiment utile pour prendre de meilleures décisions pour les futures nouvelles parties de notre application. - Lorsque nous avons un bug quelque part dû à la communication entre les nœuds, nous serons en mesure de repérer facilement le problème. Peut-être qu'un nœud n'est pas correctement connecté à un autre, ou qu'il y a deux nœuds qui publient sur un sujet donné au lieu d'un seul, ce qui explique

pourquoi nous obtenons des valeurs étranges du côté des abonnés.

La Figure 3.13 représente rqt_graph généralisé qui montre le lien de communication entre un nœud éditeur et un nœud abonné à travers un topic. En fait, les noeuds sont représentés par des ellipses, les topics par des rectangles et la flèche indique le sens de propagation des données. Le noeud éditeur publie des informations sur l'interface de communication (topic) et le noeud abonné s'abonne à ce topic pour utiliser ces données publiées.

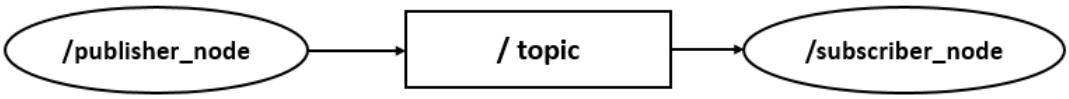


Figure 3.13 – rqt_graph généralisé dans ROS

Pour assurer la continuité et la cohérence de la méthodologie de conception, nous proposons dans notre méthodologie une comparaison entre les rqt_graphs extraits dans cette étape avec le diagramme de mission de l'essaim développé dans l'étape précédente. Cette comparaison a pour but de vérifier si les nœuds et les topics apparaissant dans les rqt_graphs sont compatibles avec les nœuds et les topics modélisés dans le diagramme de mission de l'essaim. Grâce à cette comparaison, nous pouvons juger si le système en essaim est bien développé et respecte le modèle développé ou non.

- Validation de l'essaim :

Dans cette étape, les outils de visualisation peuvent être utilisés pour valider le fonctionnement du système en essaim. Grâce à une simple visualisation du comportement du système, le concepteur peut vérifier si les fonctions individuelles de chaque robot et les comportements collectifs de l'essaim modélisés avec SysML/SwarmML apparaissent dans la version simulée du système en essaim. Pour vérifier les solutions avant de les mettre en œuvre dans des robots réels, les roboticiens et les développeurs ont besoin d'une plateforme expérimentale qui reproduit fidèlement l'environnement réel et les interactions physiques du système de robot en essaim avec cet environnement. Cette simulation leur permettra d'évaluer les performances du robot en matière de localisation, de planification des mouvements et de contrôle. Aujourd'hui, les logiciels de simulation deviennent un outil de plus en plus important pour tester les systèmes d'automatisation, tant dans l'industrie que dans la recherche. Dans le domaine de la robotique, il est utilisé pour évaluer les performances des robots. En combinant ROS, un middleware robotique populaire, et Unity, nous pouvons atteindre la précision et l'exactitude nécessaires pour nos simulations de systèmes d'essaims. En outre, Gazebo est un autre outil open source distribué sous la licence Apache 2.0, utilisé dans les domaines de la recherche en robotique et en intelligence artificielle. Enfin, RVIZ est une interface graphique ROS qui nous permet également de visualiser une grande quantité d'informations, en utilisant des plugins pour de nombreux types de sujets disponibles.

- Prototypage des essaims réels :

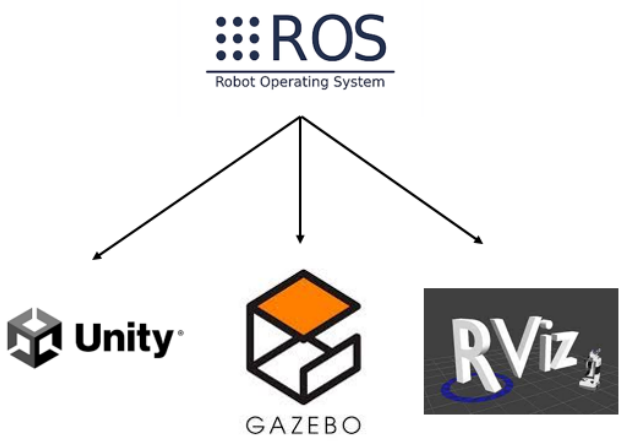


Figure 3.14 – Couplage de ROS/ROS2 avec des outils de visualisation

La dernière étape de notre méthodologie consiste à déployer le système sur des robots réels (prototypage). En fait, le concepteur doit choisir les spécifications matérielles nécessaires, soit des composants mécatroniques, soit des robots prêts à l'emploi, pour mettre en œuvre le système d'essaimage. De la même manière que pour la transition entre le modèle et la simulation, si l'implémentation sur des robots réels révèle que certaines hypothèses faites lors des phases précédentes ne tiennent pas, il peut être nécessaire de modifier la version simulée ou le modèle, afin de conserver la cohérence de tous les niveaux.

La dernière étape de notre méthodologie consiste à déployer le système sur des robots réels (prototypage). En fait, le concepteur doit choisir les spécifications matérielles nécessaires, soit des composants mécatroniques, soit des robots prêts à l'emploi, pour mettre en œuvre le système en essaim. De la même manière que pour la transition entre le modèle et la simulation, si l'implémentation sur des robots réels révèle que certaines hypothèses faites lors des phases précédentes ne tiennent pas, il peut être nécessaire de modifier la version simulée ou le modèle, afin de conserver la cohérence de tous les niveaux.

3.3 Conclusion

La robotique en essaim est un domaine multidisciplinaire qui facilite de nombreuses tâches et missions impossibles à réaliser par un seul robot. Cependant, ce domaine est encore compliqué et difficile. Du point de vue de la conception, il manque une méthodologie bien structurée qui puisse être suivie par le concepteur d'essaims pour construire un système robotique en essaim. Pour cette raison, nous avons proposé dans ce chapitre une méthodologie intégrée qui facilite la tâche du concepteur. Cette méthodologie est basée sur la modélisation avec la méthode MBSE pour spécifier les besoins et les comportements collectifs des essaims, puis sur la vérification des modèles développés et enfin sur la validation du système d'essaims par prototypage physique avec ROS et des robots réels. L'avantage de cette méthodologie est qu'elle permet d'assurer la continuité du processus de conception à chaque étape de la conception, c'est-à-dire que la transition entre la phase de spécification des exigences et la phase de modélisation comportementale et structurelle est vérifiée

soit par des matrices d'allocation, soit par une comparaison des données présentées sur les diagrammes SysML. Un autre avantage de cette méthodologie est qu'elle facilite l'intégration du système modélisé avec la méthode MBSE sur ROS puisque les modèles développés avec DSL sont prêts et compatibles avec les concepts ROS (Topics, nodes, services, ...). La validation du modèle simulé est très simple dans notre méthodologie ; soit par visualisation directe sur les outils de simulation (RVIZ, Gazebo, Unity, ...), soit par comparaison des données modélisées sur les diagrammes SysML/SwarmML et les données trouvées sur les diagrammes générés par ROS (rqt_graph).

Troisième partie

Validation de la méthodologie

Chapitre 4

Agrégation d'un système robotique en essaim

L'intelligence vise à simplifier ce qui est complexe, la vanité à complexifier ce qui devrait rester simple.

– Nicolas Plouvier

4.1 Introduction

L'agrégation est l'un des comportements fondamentaux des essaims dans la nature et est observée dans des organismes allant des organismes unicellulaires aux insectes sociaux et aux mammifères. L'agrégation aide les organismes de nombreuses façons, par exemple en évitant les prédateurs ou en résistant à des conditions environnementales défavorables. Certains comportements d'agrégation sont connus pour être facilités par des indices environnementaux ; par exemple, les mouches utilisent la lumière et la température pour s'agréger. Cependant, d'autres agrégations sont auto-organisées. L'agrégation des cafards et des bancs de poissons n'utilisent pas de tels indices mais sont plutôt le résultat de décisions coopératives émergentes.

Dans cette étude de cas, nous se concentrons sur les comportements d'agrégation auto-organisés pour les systèmes robotiques en essaim. Le comportement d'agrégation est essentiel pour ces systèmes, car pour la plupart des comportements de la robotique en essaim, les robots doivent être à proximité les uns des autres dans de nombreux cas. Le problème de l'agrégation, qui peut sembler plutôt trivial à première vue, est difficile puisque dans la plupart des systèmes robotiques en essaim, les individus doivent se fier à une perception plutôt grossière de leur monde. Par conséquent, il est difficile d'obtenir des comportements permettant de former de grandes agrégations qui sont au-delà de la portée de détection des individus. Ainsi, les agrégations de robots localement optimales doivent être transformées

en une agrégation globalement optimale uniquement par des règles locales et simples.

Dans ce chapitre, nous allons appliquer notre méthodologie de conception IDMSR sur un cas d'agrégation d'un essaim des robots. En particulier, ce cas d'agrégation est basé sur la méthode probabiliste et les chaînes de Markov temporelles déterministes (DTMC) car elles facilitent la modélisation en incluant les niveaux microscopique et macroscopique. Au niveau microscopique, le modèle représente le comportement d'un seul robot. Au niveau macroscopique, chaque état peut être utilisé pour compter le nombre de robots dans cet état particulier. A cette fin, ce chapitre sera organisé comme suit : dans la section suivante, nous présenterons les différentes approches proposées dans la littérature pour agréger un essaim de robots. Dans la troisième section, nous appliquerons notre méthodologie de conception IDMSR proposée dans le chapitre 3 sur un cas d'agrégation d'essaim de robots en utilisant le langage de modélisation de système SysML, l'outil de simulation multi-agent Anylogic et l'environnement de simulation ROS/Gazebo. Enfin, nous terminerons ce chapitre par une conclusion.

4.2 Les méthodes d'agrégation d'un système robotique en essaim

4.2.1 Définition

L'agrégation est une technique par laquelle des robots individuels se rassemblent pour accomplir des tâches, par exemple, un mouvement collectif ou un échange d'informations. Cette technique permet aux agents de l'essaim de se rapprocher spatialement les uns des autres dans une région particulière pour plus d'interaction. Le problème d'agrégation est étudié soit comme un problème indépendant, soit dans le cadre de tâches plus spécialisées impliquant le regroupement d'un certain nombre d'agents. Un groupe de robots auto-organisés est appelé un agrégat [73]. Un exemple d'agrégation de robots autonomes est illustré à la Figure 4.1.

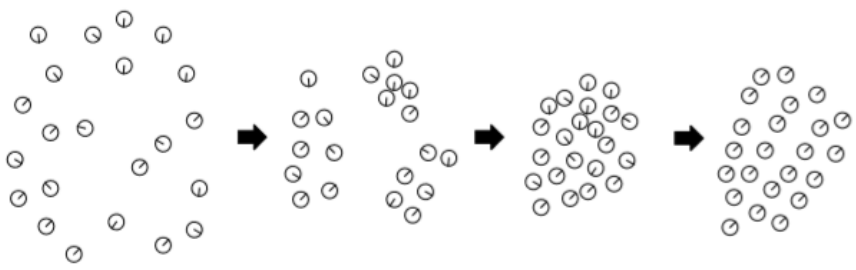


Figure 4.1 – Un exemple d'agrégation d'agents autonomes.

Parmi les approches de l'agrégation les plus efficaces d'un essaim de robots nous trouvons la méthode des forces virtuelles, la méthode évolutive, et la méthode probabiliste.

4.2.2 La méthode des forces virtuelles

Le comportement des robots autonomes est souvent modélisé à l'aide de la méthode des forces virtuelles. Elle est basée sur le calcul des forces qui déterminent le mouvement des robots les uns par rapport aux autres en tenant compte de l'emplacement des objets environnants. De nombreux groupes organisés d'animaux (par exemple, des essaims d'insectes, des volées d'oiseaux, des bancs de poissons) peuvent être modélisés à l'aide de forces attractives (grâce auxquelles les animaux voisins ont tendance à rester proches les uns des autres) et de forces répulsives (qui empêchent les collisions entre animaux) [74] . Chaque robot autonome se déplace en fonction de la force exercée sur lui par les robots voisins et en fonction de la distance qui les sépare. En général, les forces répulsives agissent à courte distance, tandis que les forces attractives agissent à des distances supérieures à une valeur prédéterminée.

La méthode des forces virtuelles est utilisée avec succès pour la description formelle de l'agrégation d'un essaim de robots [13, 133, 56]. Cependant, sa réalisation dans des systèmes avec des robots réels crée une série d'exigences strictes pour les capteurs de chaque robot, qui sont difficiles et coûteuses à mettre en œuvre. Les robots les plus simples dotés de capteurs autonomes sont caractérisés par un faible champ de visibilité, ce qui réduit considérablement leur capacité à distinguer les autres robots dans l'environnement. Des erreurs peuvent survenir lors de la détermination de la position relative des agents dans l'environnement, notamment lors de l'utilisation de capteurs infrarouges. De plus, les contraintes mécaniques créent ce qu'on appelle un effet de saturation dans les actionneurs du robot, ce qui limite l'amplitude des signaux d'entrée pour réguler le mouvement du robot [133]. Malgré ces limitations des capteurs, la méthode des forces virtuelles est largement utilisée pour contrôler les mouvements des robots dans les systèmes les plus simples.

4.2.3 Les méthodes évolutives

Dans le cas de la méthode de contrôle évolutionnaire, la dynamique d'agrégation est obtenue en utilisant des contrôleurs robotisés, dont les paramètres sont sélectionnés dans le processus d'évolution artificielle. Les réseaux neuronaux sont des exemples de contrôleurs utilisant cette méthode. Selon l'algorithme utilisé, les entrées des capteurs peuvent inclure des dispositifs capables de recevoir des informations sur l'environnement, et les sorties des actionneurs peuvent inclure des dispositifs permettant aux robots de communiquer entre eux. Des exemples d'algorithmes, utilisés pour la méthode d'évolution artificielle, sont l'algorithme génétique ou la sélection par tournoi [18, 12]. L'évolution artificielle applique le paradigme standard de l'évolution naturelle des populations dans la nature. Ce paradigme est basé sur le concept d'adaptation, qui détermine la capacité d'une population sélectionnée d'individus à s'adapter à la tâche.

Contrairement à la méthode d'évolution basée sur le concept d'adaptation, la méthode de recherche de la nouveauté implique une position privilégiée pour les robots dont le comportement diffère des modèles comportementaux des générations précédentes. Cette méthode permet d'éviter les conséquences négatives possibles de l'approche basée sur le concept d'adaptation, dans laquelle le maximum local de la fonction de fitness dans l'espace

des paramètres peut exclure la possibilité d'étudier toutes les autres parties de cet espace et donc limiter le processus d'évolution. Dans [54], l'algorithme de recherche de nouveauté est appliqué pour résoudre le problème de l'agrégation, et la caractéristique comportementale d'une population sélectionnée est basée sur les paramètres, tels que la distance moyenne de chaque robot au centre de masse de tous les robots ou le nombre total d'agrégats. Ces paramètres ont été mesurés plusieurs fois au cours du processus de simulation, et leurs valeurs (valeurs moyennes des différentes simulations effectuées) ont été ajoutées au vecteur de la caractéristique comportementale utilisé pour déterminer la similarité des différents modèles de comportement.

4.2.4 Les méthodes probabilistes

Dans l'approche probabiliste, le comportement de chaque robot a une composante aléatoire et est ajusté dans le processus d'interaction du robot avec l'environnement. Ce type de comportement est souvent observé dans la nature chez les insectes sociaux tels que les abeilles ou les cafards. Sur la base d'observations du comportement des insectes sociaux, des algorithmes probabilistes ont été créés pour contrôler les mouvements des robots. Ces algorithmes sont basés sur une machine à états finis (FSM) avec deux états principaux - "go" et "wait" [7] - qui correspondent aux algorithmes comportementaux. Dans certains cas, l'état "go" est divisé en deux états : lorsqu'un robot tente de se rapprocher d'autres robots ou, au contraire, de s'éloigner de ses voisins [8]. La décision de changer d'état peut être prise de manière totalement aléatoire, ou basée sur des signaux locaux (par exemple, la présence de robots dans les environs), ou encore sur des algorithmes et des mécanismes de signalisation plus complexes. Le concepteur d'un essaim choisit généralement les paramètres de la machine à états finis, tels que la probabilité de passer d'un état à l'autre ; cependant, des méthodes alternatives basées sur des techniques automatiques de détermination des paramètres ont récemment été développées [48].

Une caractéristique commune de tous les algorithmes d'agrégation probabiliste est la présence d'agrégats instables, dans lesquels les robots entrent et sortent constamment. La dynamique de l'agrégation se produit en raison des changements dans le comportement aléatoire du robot lors de la détection des robots adjacents. Alors que les robots désagrégés se déplacent généralement dans l'espace de manière aléatoire, la dynamique des robots agrégés est déterministe. Cependant, une composante aléatoire du comportement des robots agrégés est souvent nécessaire pour la formation de petites quantités d'agrégats plus grands afin d'éviter les situations dans lesquelles la présence de petits agrégats ne permet pas aux robots d'être attachés à des agrégats plus grands. Dans les études où les algorithmes basés sur les FSM ne sont pas utilisés, il n'y a généralement pas de distinction claire entre les robots agrégés et non agrégés. Cependant, la dynamique de l'essaim peut être déterminée par une métrique spéciale, telle que la distance moyenne entre les robots, et le caractère aléatoire du mouvement du robot peut être modifié sur une échelle continue [27].

Les chaînes de Markov temporelles déterministes (DTMC) sont souvent utilisés pour modéliser les systèmes robotique en essaim [81]. L'un des principaux avantages des DTMC est que, dans de nombreux cas, le modèle comprend à la fois les niveaux microscopique et macroscopique. Au niveau microscopique, le modèle représente le comportement d'un seul

robot. Au niveau macroscopique, chaque état peut être utilisé pour compter le nombre de robots dans cet état particulier. Par exemple, au niveau microscopique, on peut modéliser le comportement d'un seul robot avec un DTMC à trois états. Le même DTMC peut être augmenté en associant un compteur à chaque état. Chaque compteur garde la trace du nombre de robots dans l'état associé. Un autre avantage des DTMC est que leur utilisation peut faciliter la vérification des propriétés.

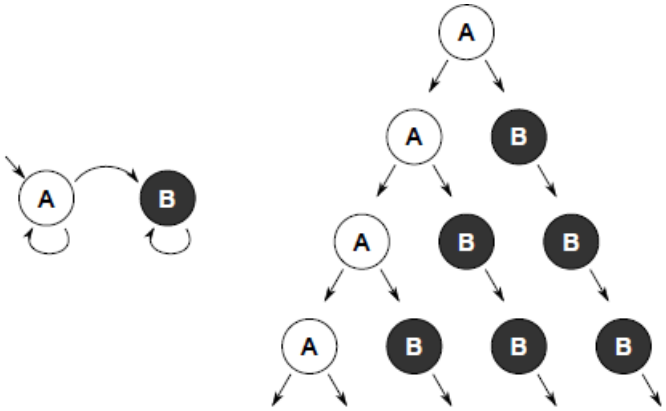


Figure 4.2 – Une chaîne de Markov simple (à gauche) et son arbre de calcul (à droite).

4.3 Application de la méthodologie IDMSR

4.3.1 Principe

Dans cette étude de cas d'agrégation, il existe une grande zone appelée C avec deux petites zones de la même taille appelées zone A et zone B, comme le montre la Figure 4.3. Chacune des deux zones A et B est suffisamment grande pour accueillir tous les robots. Plusieurs tailles d'essaim sont choisies afin de déterminer l'influence des interactions locales de l'essaim sur le comportement global émergent.

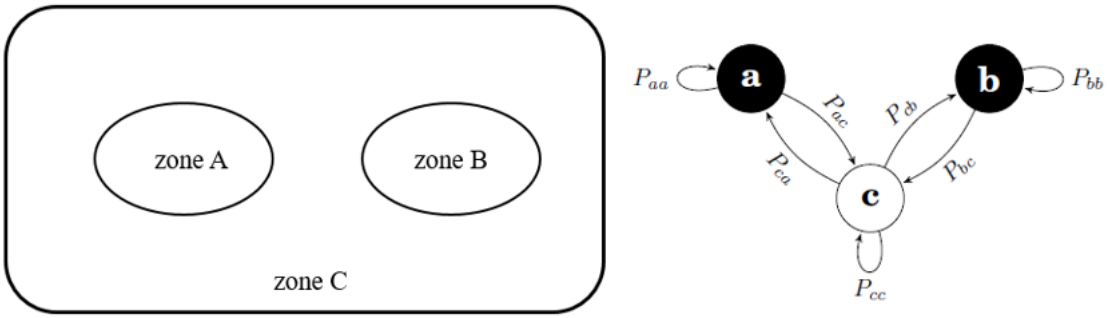


Figure 4.3 – Principe du cas d'agrégation.

4.3.2 Première phase : Modélisation avec la méthode MBSE

1) La spécification des exigences de conception de l'essaim :

Comme il a été noté dans la méthodologie de conception, les différents diagrammes SysML sont utilisés pour spécifier les exigences du système. Le diagramme des exigences illustré à la Figure 4.4 montre deux exigences nécessaires à la construction du système. Ces exigences représentent les propriétés de la mission. Les deux principales propriétés globales qui caractérisent la mission d'agrégation d'un essaim des robots sont les suivantes : Les robots doivent s'agréger dans l'une des deux zones A ou B pour former un agrégat avec un nombre bien déterminé de robots Nr et l'agrégat doit rester stable pendant un certain temps de stabilité globale Ts.

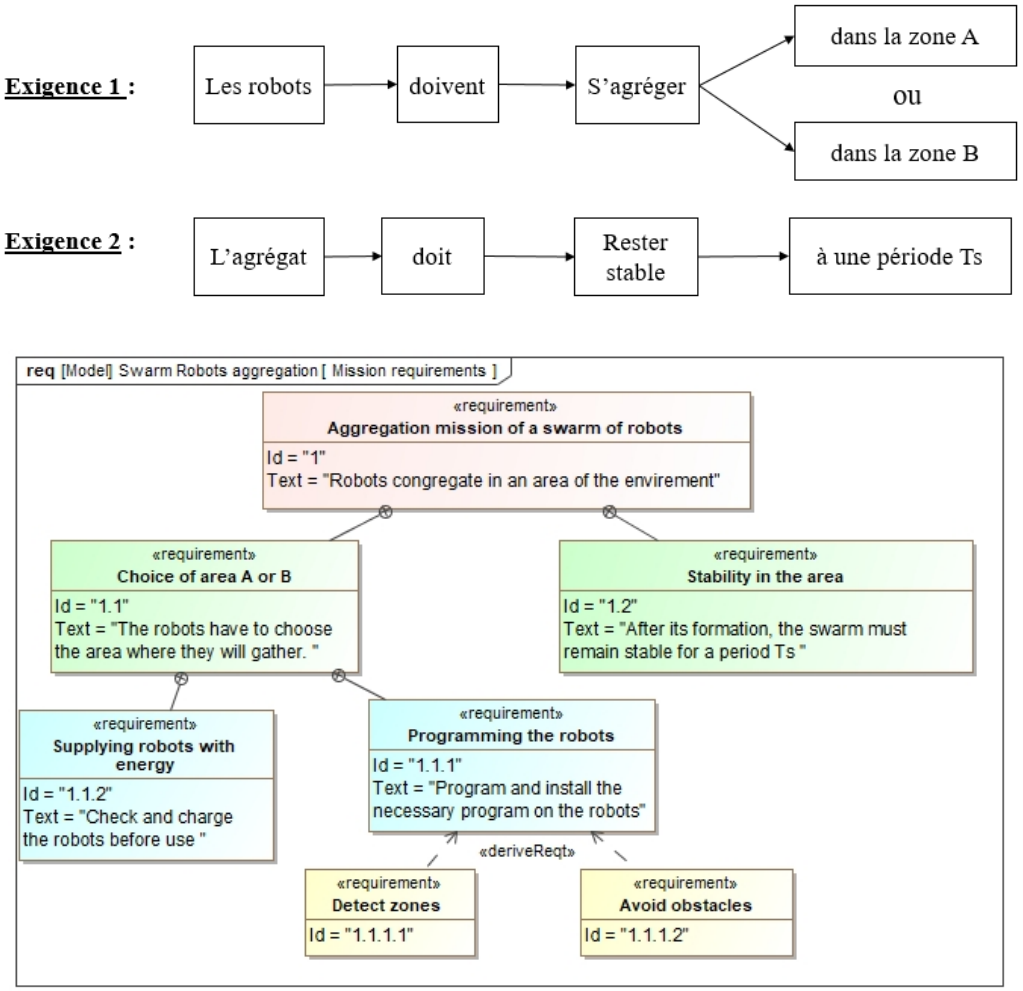


Figure 4.4 – Diagramme des exigences (propriétés de la mission).

2) Modélisation du comportement de l'essaim :

Dans cette étape, nous transformons le DTMC présenté dans la figure 4.3 en un diagramme d'état qui décrit l'état de notre système. En effet, nous choisissons le diagramme d'état car il décrit efficacement les états et les transitions du système introduit sur la DTMC. L'environnement est divisé en trois zones. Trois états sont définis : Sa, Sb et Sc. Un robot

dans la zone A ou B est dans l'état Sa ou Sb, respectivement. Le robot en dehors de la zone A ou B est dans l'état Sc.

Les scénarios possibles peuvent être décrits à l'aide des équations mathématiques suivantes : Un robot dans la zone C peut se déplacer soit dans la zone A, soit dans la zone B, soit rester dans la zone C. Cela signifie qu'un robot dans la zone C a une probabilité de passer de la zone C à la zone A égale à

$$\mathbb{P}_{ca} = \frac{A_A}{A_{arena}} \tag{4.1}$$

pour passer de la zone C à la zone B égale à

$$\mathbb{P}_{cb} = \frac{A_B}{A_{arena}} \tag{4.2}$$

et pour rester dans la zone C égale à

$$\mathbb{P}_{cc} = \frac{A_C}{A_{arena}} \tag{4.3}$$

Notez que Pca = Pcb, car les zones A et B ont la même taille et Aarena est la surface totale de l'arène.

La seule probabilité indépendante restante est Pac. En faisant varier la valeur de Pac, il est possible de trouver la meilleure valeur qui correspond à la formation de l'agrégat dans l'une des deux zones A ou B.

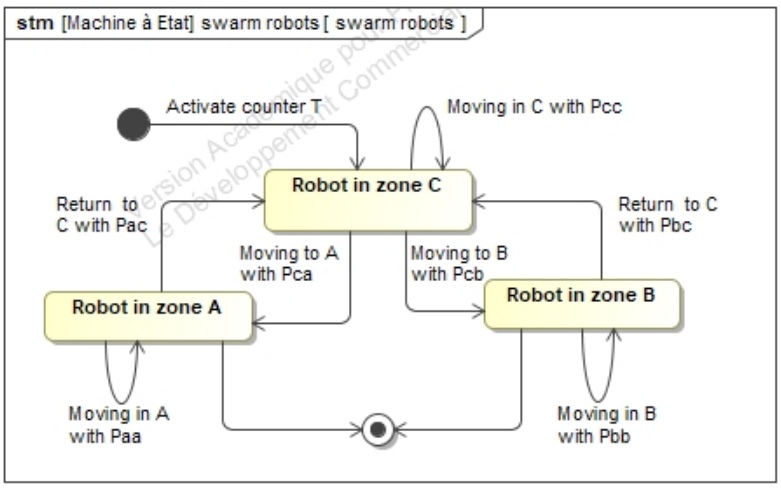
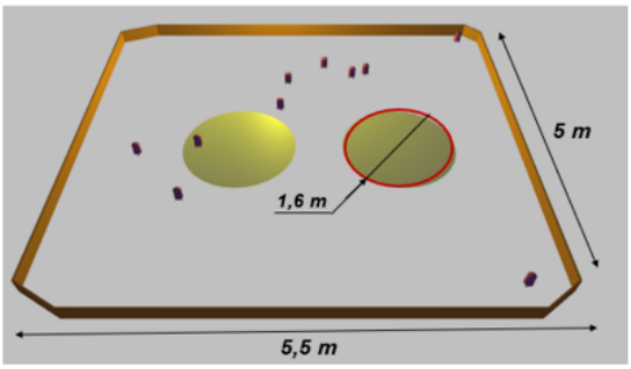


Figure 4.5 – Modélisation du comportement de chaque robot de l'essaim.

3) Simulation multi-agents avec Anylogic :

Dans cette étape, le modèle de simulation de l'essaim illustré à la Figure 4.6 est mis en œuvre sur l'outil de simulation multi-agents AnyLogic. Il s'agit d'un outil de simulation

développé par The AnyLogic Company. AnyLogic possède un langage de modélisation graphique et permet également d'étendre facilement le modèle de simulation avec du code Java. Les trois paramètres Pca, Pcb et Pcc ne dépendent que de la géométrie de l'arène.



$A_{arène} = 27.5\ m^2$

$A_{zone\ A} = A_{zone\ B} = 2\ m^2$

Swarm size: total number of robots

Waiting Time: Waiting time for each robot

P_{ca}	0,233
P_{cb}	0,233
P_{cc}	0,534

Figure 4.6 – Géométrie de l'arène

Pour un groupe de 20 robots et avec la variation de Pac, un agrégat est formé dans l'une des deux zones A ou B avec des périodes d'attente différentes. Voir la Figure 4.7.

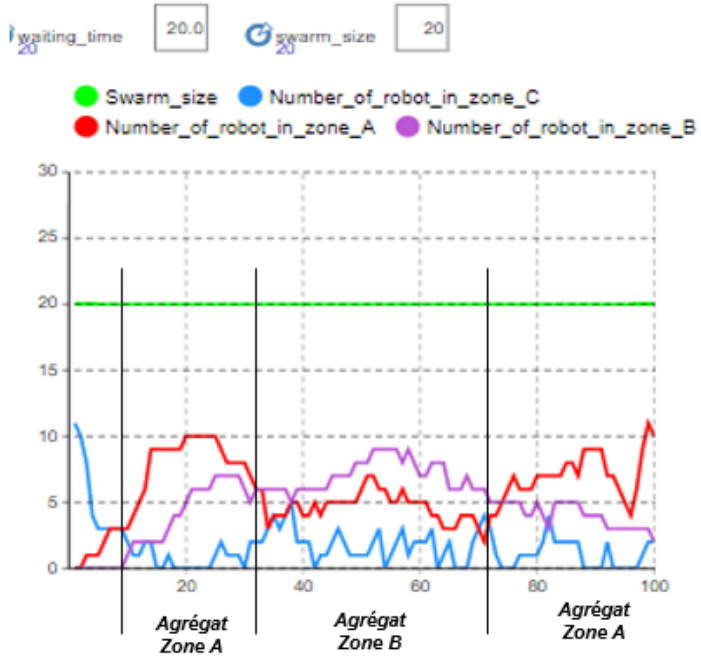


Figure 4.7 – Formation périodique d'un agrégat de robots dans les deux zones A et B.

Pour une valeur de Pac égale à 0.04, un agrégat de robots (entre 5 et 10 robots) s'est formé dans la zone A puis l'agrégat change de position dans la zone B avec un autre nombre de robots et un temps d'attente différent de celui de la zone A. Dans la suite de cette application, Pac est fixé à 0,04. Les deux indices de performance étudiés sont :

- *Le nombre de robots qui forment l'agrégat : Nr*

- *Temps de stabilité de l'agrégat : Ts*

a) Choix de la zone d'agrégation

L'utilisateur peut choisir la zone où se rassemble l'agrégat d'un essaim de robots. Il s'agit de modifier les probabilités suivantes : Pca et Pcb. Dans les exemples cités à la Figure 4.8, la taille de l'essaim est fixée à 30 robots et le temps d'attente de chaque robot est fixé à 20 secondes, puis les deux probabilités Pca et Pcb sont modifiées pour déterminer leur effet sur la formation de l'agrégat.

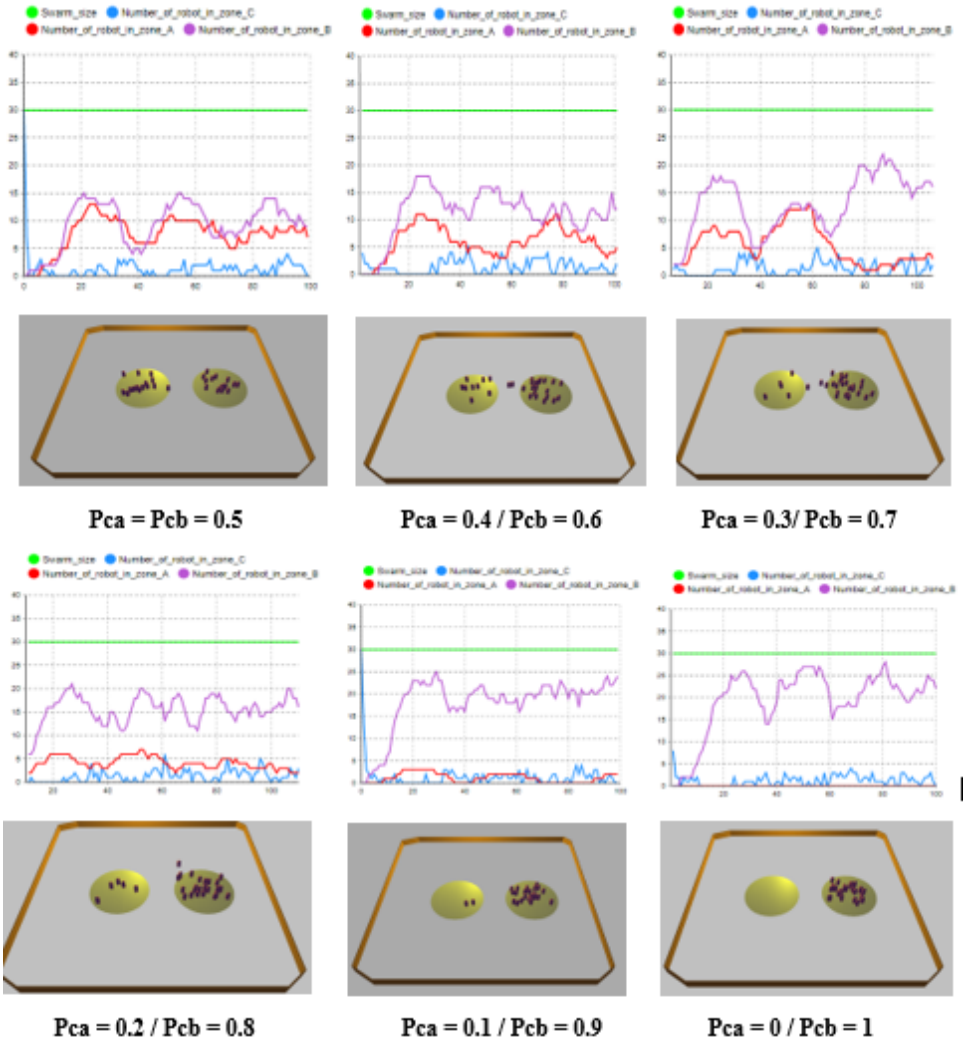


Figure 4.8 – Choix de la zone de formation de l'essaim

En fonction des deux paramètres Pca et Pcb, l'utilisateur peut orienter le système dans l'une des deux zones ; Pour Pca = Pcb = 0,5, il y a deux agrégats dans les deux zones avec une répartition égale des robots. Pour Pca <Pcb, tous les robots choisissent la zone B pour former leur agrégat. Enfin, pour Pca = 0 et Pcb = 1 : tous les robots sont regroupés dans la zone B.

Dans la suite de cette application, la formation d'agrégats est imposée dans la zone B. Le nombre total de robots est égal à 30 robots mais le nombre émergent de Nr robots qui forment cet agrégat est inconnu. De même, le temps d'attente pour chaque robot est égal

à 20 secondes (paramètre local) mais le temps de stabilité du groupe en général Ts reste inconnu. A l'aide de la Figure 4.9, il est possible de déterminer ces deux paramètres Nr et Ts.

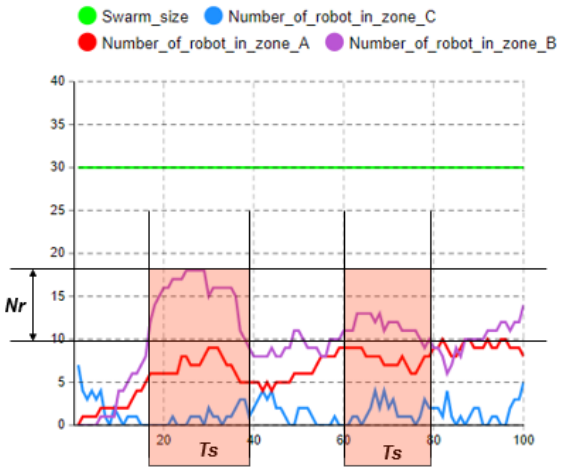


Figure 4.9 – Caractéristiques de l'agrégat Ts et Nr.

Le nombre de robots qui forment cet agrégat Nr est compris entre 10 et 19 robots. Le temps de stabilité de l'agrégat Ts varie lors de la formation de l'agrégat : pour la première agrégation le temps de stabilité est de 24 secondes alors que pour la deuxième agrégation il est de 20 secondes. Grâce aux paramètres locaux (temps d'attente, nombre de robots), l'utilisateur peut déterminer le nombre émergent de robots qui forment l'agrégat Nr et le temps de stabilité de l'agrégat Ts.

b) L'influence du temps d'attente local sur le temps de stabilité global

L'utilisateur peut modifier le paramètre "Waiting Time" local de chaque robot pour déterminer l'influence de ce paramètre sur le temps de stabilité global du système Ts. Pour un temps d'attente = 10 s. La Figure 4.10 montre que l'agrégat reste stable pendant les différentes périodes d'attente.

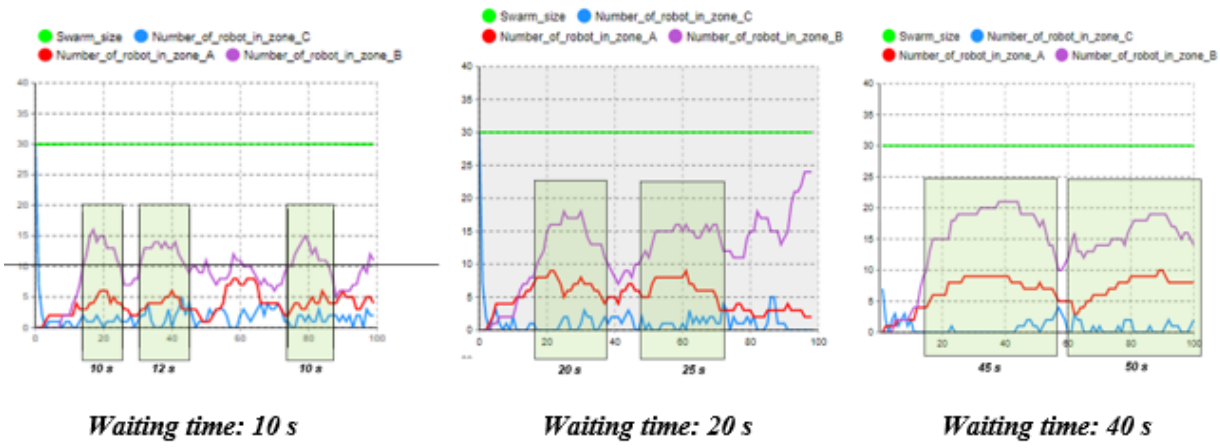


Figure 4.10 – Influence du temps d'attente local.

Le premier agrégat reste 10 secondes, le deuxième 12 secondes et le troisième 10 secondes. Le temps de stabilité émergent Ts est toujours plus proche de 10 secondes. En conclusion, le temps de stabilité Ts d'un agrégat dépend des paramètres locaux de chaque robot ; le temps de stabilité toujours émergent reste plus proche du temps d'attente de chaque robot.

c) L'influence du nombre de robots sur la formation de l'agrégat

L'utilisateur peut modifier le nombre de robots dans le système afin de déterminer le nombre de robots qui forment l'agrégat. Comme le montre la Figure 4.11, le nombre de robots qui forment l'agrégat dépend du nombre total de robots dans le système.

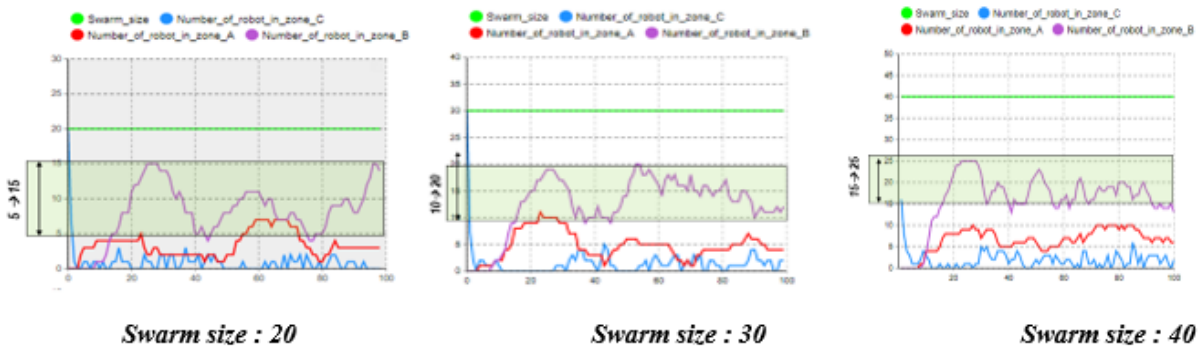


Figure 4.11 – Influence du nombre de robots sur la formation des agrégats.

Pour un système de 20 robots, le nombre émergent de robots qui forment l'agrégat est de 5 à 15 robots. Pour un système de 30 robots, le nombre de robots émergents qui forment l'agrégat est de 10 à 20 robots. Enfin, pour un système de 40 robots, le nombre de robots émergents qui forment l'agrégat est de de 15 à 25 robots.

Avec cette simulation multi-agents nous avons étudié les indicateurs de performance de notre système tels que le nombre de robots qui forment l'essaim Nr, le temps de stabilité de l'agrégat dans une des deux zones Ts. En fait, nous concluons de cette simulation sur Anylogic la robustesse et la flexibilité de notre système. En effet, la robustesse de notre système s'explique par l'adaptation aux différents changements de taille et de temps de stabilité de l'essaim. A ce stade, notre méthodologie de conception est bien appliquée et aboutit à un système qui répond aux exigences mentionnées au début du cycle de conception. Dans ce qui suit, nous allons modéliser la structure de notre système et l'implémenter sur ROS.

4) Modélisation fonctionelle de l'essaim :

Le diagramme d'activité SysML de la Figure 4.12 montre les différentes fonctions des robots individuels qui construisent le comportement collectif de l'essaim et les liens entre les robots individuels. En effet, chaque robot de l'essaim suit ce processus pour former un agrégat avec les autres robots dans l'une des deux zones de l'arène. Ce processus consiste à ce que les robots commencent à se déplacer vers la zone C. S'ils entrent en collision avec d'autres robots, ils changent de direction. Sinon, chaque robot décide de choisir la zone A ou B en fonction de la probabilité Pca et Pcb et reste un temps d'attente bien défini jusqu'à ce que les restes des robots se rejoignent pour former l'agrégat dans l'une des deux zones.

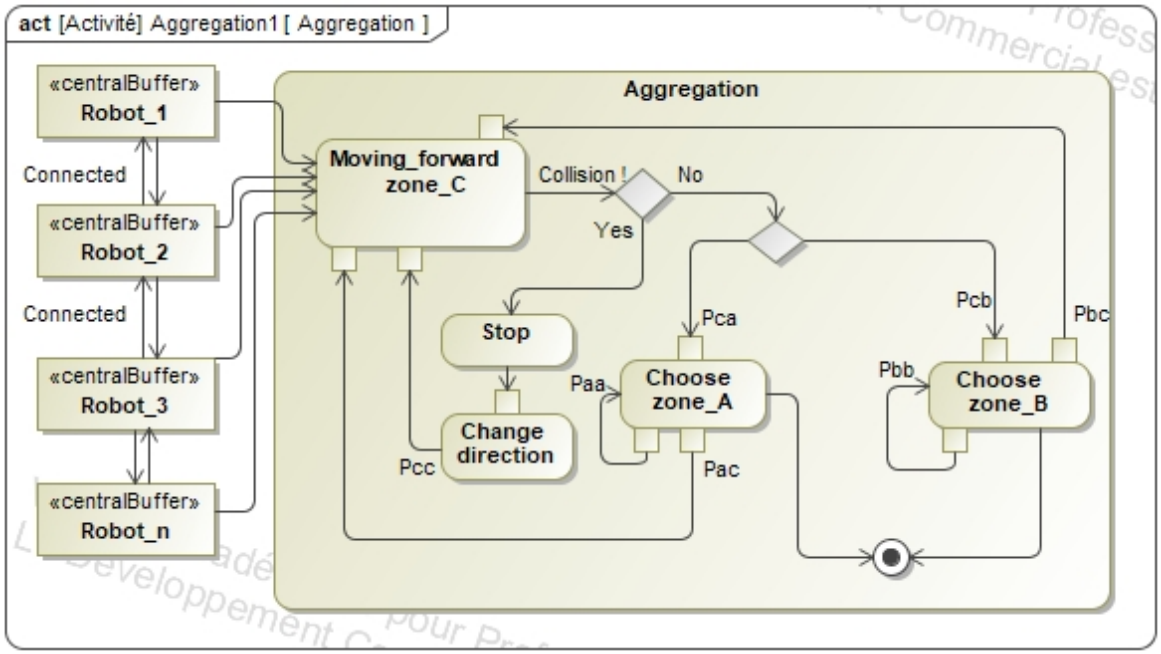


Figure 4.12 – Niveau fonctionnel de l'essaim.

5) Modélisation structurelle de l'essaim :

Dans cette étape, la structure générale de notre système doit être définie. Pour ce faire, la structure générale du système suit la description donnée dans un diagramme de définition des blocs, comme le montre la Figure 4.13.

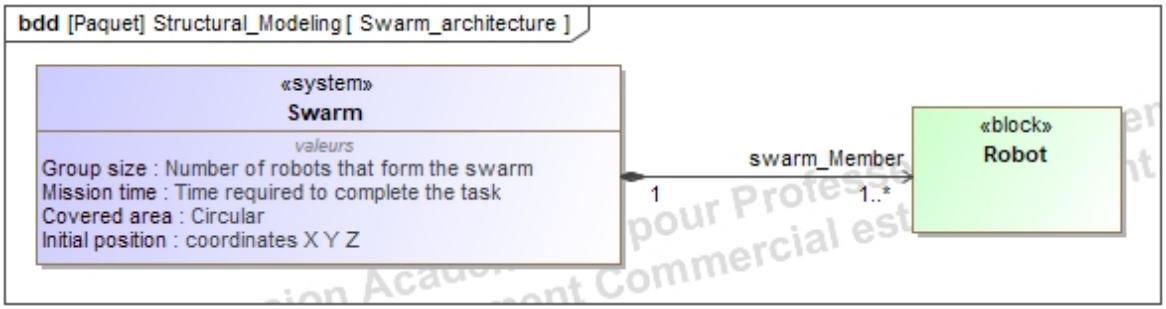


Figure 4.13 – L'architecture générale d'un système en essaim.

L'essaim est constitué d'un groupe de robots individuels. Le diagramme de définition de bloc illustré à la Figure 4.14 représente la structure des principaux sous-systèmes et composants de notre système. Ce système est composé : d'un système d'alimentation électrique (batteries et conducteurs) ; d'un système de mouvement (moteur à courant continu, roues et système de transmission du mouvement) ; d'un système de communication entre les robots ; d'un système de contrôle (compteurs horaires, capteurs de position et microcontrôleur).

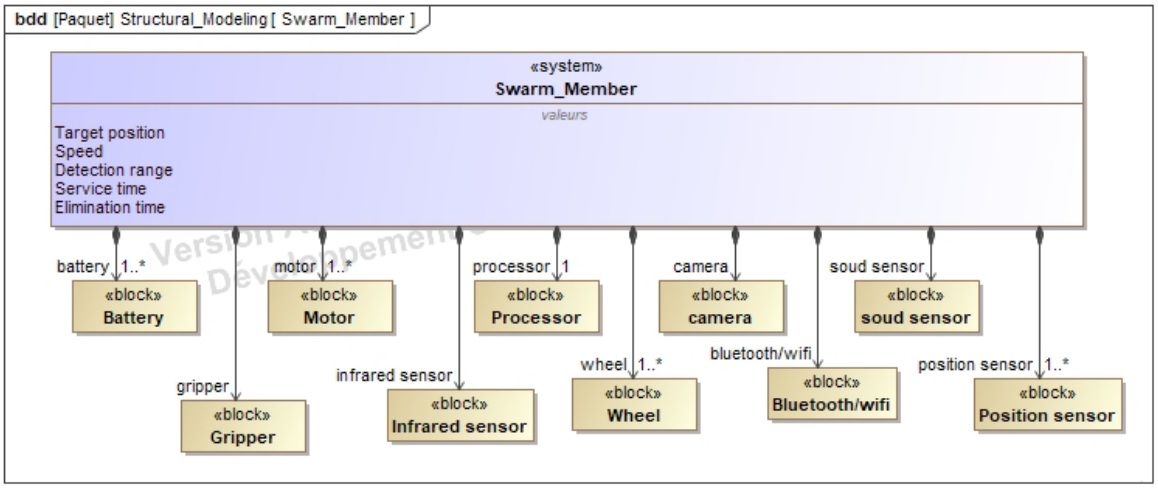
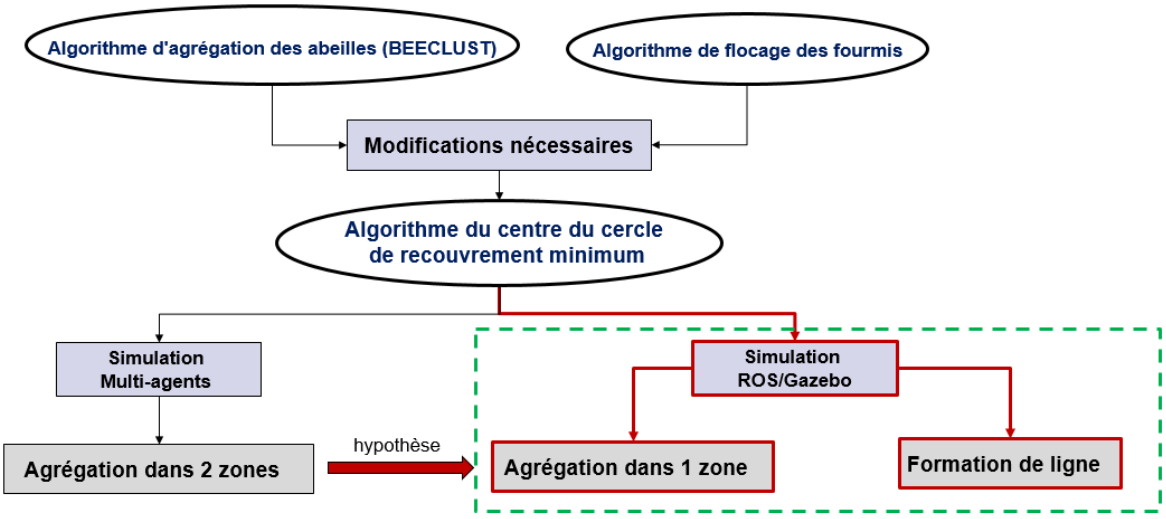


Figure 4.14 – Architecture d'un membre d'essaim en SysML.

4.3.3 Deuxième phase : Implémentation à l'aide de ROS/ROS2

1) La création de l'algorithme d'agrégation :

Pour implémenter notre application d'agrégation d'un système en essaim sur ROS, nous proposons une hypothèse afin de simplifier l'étude de cas. Cette hypothèse consiste en l'agrégation de l'essaim de robots dans une seule zone ou lieu des deux zones. Après une recherche approfondie dans les différents algorithmes développés sur ROS pour la mise en œuvre d'un cas d'agrégation d'un système en essaim, nous avons trouvé l'algorithme du centre du cercle de recouvrement minimum (center of minimum covering circle)[63, 64]. Cet algorithme est inspiré des comportements de deux animaux sociaux : l'agrégation des abeilles et la formation de ligne de fourmis.



a) Algorithme de flocage des fourmis

C'est un algorithme décentralisé, distribué et bio-inspiré des comportements de la colonie de termites et de fourmis (recherche de nourriture) pour construire le phénomène

d'agrégation [98]. Chaque robot de l'essaim émet périodiquement des impulsions IR. Les robots réagissent alors (se déplacer en ligne droite, tourner à gauche ou à droite) en fonction des informations fournies par leurs capteurs IR actifs et passifs. Ces capteurs sont interrogés périodiquement et les valeurs renvoyées sont ensuite comparées à des seuils prédéfinis (Figure 4.15.B) dans une architecture de subsomption simple (Figure 4.15.A).

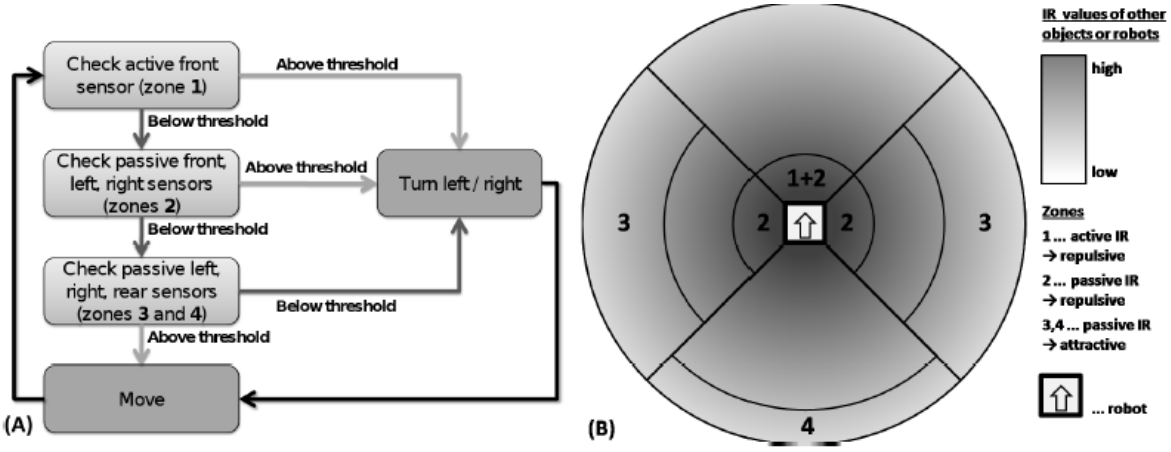


Figure 4.15 – A : Architecture de subsomption simple décrivant l'algorithme de flocage. B : Représentation simplifiée des valeurs IR détectées des autres objets [98].

Tout d'abord, la valeur de l'IR actif du capteur avant est interrogée pour savoir s'il y a un obstacle devant. Si la valeur de la lumière IR réfléchie est supérieure à un certain seuil, le robot se détourne dans une direction aléatoire. Il s'agit de la prévention de base des collisions de nos robots.

S'il n'y a pas d'objet sur son chemin, le robot vérifie les valeurs IR passives de tous les capteurs. Si le capteur avant, gauche ou droit est supérieur à un certain seuil, le robot se détourne de ce qui est probablement un autre robot trop proche. Cette règle est généralement appelée règle de séparation dans les algorithmes de flocage. S'il n'y a pas d'autre robot trop proche, le robot vérifie les valeurs IR passives de ses capteurs gauche, droit et arrière. Pour chaque capteur qui renvoie une valeur supérieure au seuil de lumière IR de l'environnement mais inférieure au seuil qui définit la distance maximale souhaitée par rapport à un autre robot dans ce secteur, le robot effectue une addition vectorielle de base et additionne tous les tours. Il décide ensuite de tourner dans une direction selon qu'il y a eu plus de virages à gauche ou plus de virages à droite. Les robots de la zone arrière déclenchent une réaction aléatoire de virage. Cette règle est généralement appelée règle de cohésion dans les algorithmes de flocage. La troisième règle des algorithmes de flocage est généralement la règle d'alignement qui génère la direction commune du mouvement dans un flocage.

b) Algorithme d'agrégation des abeilles (BEECLUST)

Le comportement d'agrégation des abeilles révèle que les agents suivent un algorithme simple basé sur les collisions, comme le montre la Figure 4.16.[9]

Les décisions sont prises après chaque collision. Cette collision ne se produit pas physiquement entre les châssis des robots. Ainsi, une collision inter-robots est définie pour détecter les radiations IR d'un autre robot. Les robots émettent des radiations IR en perma-

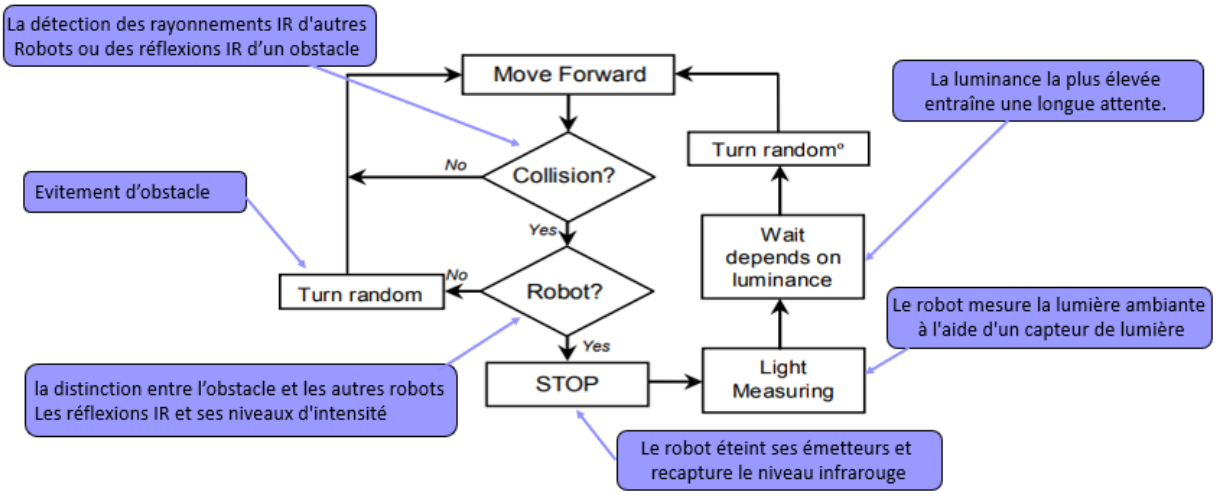


Figure 4.16 – Algorithme d'agrégation des abeilles (BEECLUST) [9].

nence. Les valeurs capturées par les récepteurs IR sont vérifiées par le processeur principal et celui-ci produit une interruption lorsque l'IR atteint un niveau seuil défini. Les réflexions IR et leurs niveaux d'intensité sont utilisés pour distinguer le mur des autres robots.

Lorsque le robot détecte le rayonnement IR, il éteint ses émetteurs et récupère le niveau IR. S'il y a un rayonnement IR, il s'agit d'un autre robot. Ce principe simple aide les robots à distinguer les obstacles des autres robots. Après avoir détecté un obstacle, le robot tourne et exécute une routine d'évitement des obstacles. À l'inverse, si le robot détecte un autre robot, il s'arrête et mesure la lumière ambiante à l'aide d'un capteur de lumière. Les robots sont équipés d'un capteur d'éclairage d'extension pour mesurer l'intensité de la lumière ambiante. Ce module est placé au sommet de la carte principale du robot.

c) Notre algorithme d'agrégation

Dans cet algorithme d'agrégation, le centre géométrique est remplacé par le centre du cercle de recouvrement minimum. Cet algorithme permet d'agréger un essaim de robots à deux roues par la méthode du centre géométrique. Chaque robot calcule le centre géométrique de tous ses voisins dans la zone de détection. Le premier vecteur de retour (feedback vector) part du robot vers le centre géométrique. Afin d'éviter les collisions, un modèle de ressort avec uniquement une force de propulsion (le ressort ne fait que repousser deux robots, il ne les entraîne pas ensemble) est utilisé pour tous les voisins dont la distance est inférieure à la longueur du ressort. Il s'agit du deuxième vecteur de rétroaction (feedback vector). Ensuite, les deux vecteurs de rétroaction seront fusionnés en un seul avec un rapport pondéré. En fait, notre contribution consiste à apporter les modifications nécessaires au niveau du type des robots [63, 64].

Pour comprendre le principe de fonctionnement de cet algorithme, nous avons illustré son pseudocode ci-dessous. La première étape consiste à initialiser les paramètres locaux et globaux du système, puis le processus d'agrégation a été développé selon une boucle pour jusqu'à une condition d'arrêt.

Début
Initialiser les paramètres locaux et globaux.
- *Taille de l'essaim N.*
- *Distance à maintenir lors de l'agrégation (SPRING_LENGTH)*
- *Plage de détection*
- *Temps de simulation*

Démarrer le temps du programme.
Pour *(i de 1 à N)* ***faire***
- *Calculez la distance entre deux robots.*
- *Trier la distance et enregistrer le changement d'index*
- *Trouver tous les voisins dans la plage de détection*
- *Trouver tous les voisins dans la plage de collision (SPRING_LENGTH .*
- *Calculer driving feedback vector basé sur les centre géométriques des robots voisins.*
- *Calculer collision feedback vector à partir des voisins de la plage de collision*
- *Fusionner le deux vecteurs (driving feedback vector and collision feedback vector)* ➔ *feedback vector.*
- *Calculer les vitesses des roues en fonction de feedback vector (les vitesses de roue sont calculées de sorte que le robot se déplace dans une courbe à rayon constant vers la destination définie par feedback vector).*
- *Gérer les vitesses des roues (Demande de service des vitesses de roue).*
- *Calculer le nombre de voisins dans la plage de collision*

Processus d'agrégation

Jusqu'à *(le nombre minimum de voisins dans la plage de collision est <3)*
Quitter le programme

La figure 4.17 représente l'organigramme de cette méthode d'agrégation bio-inspirée du comportement des fourmis et abeilles.

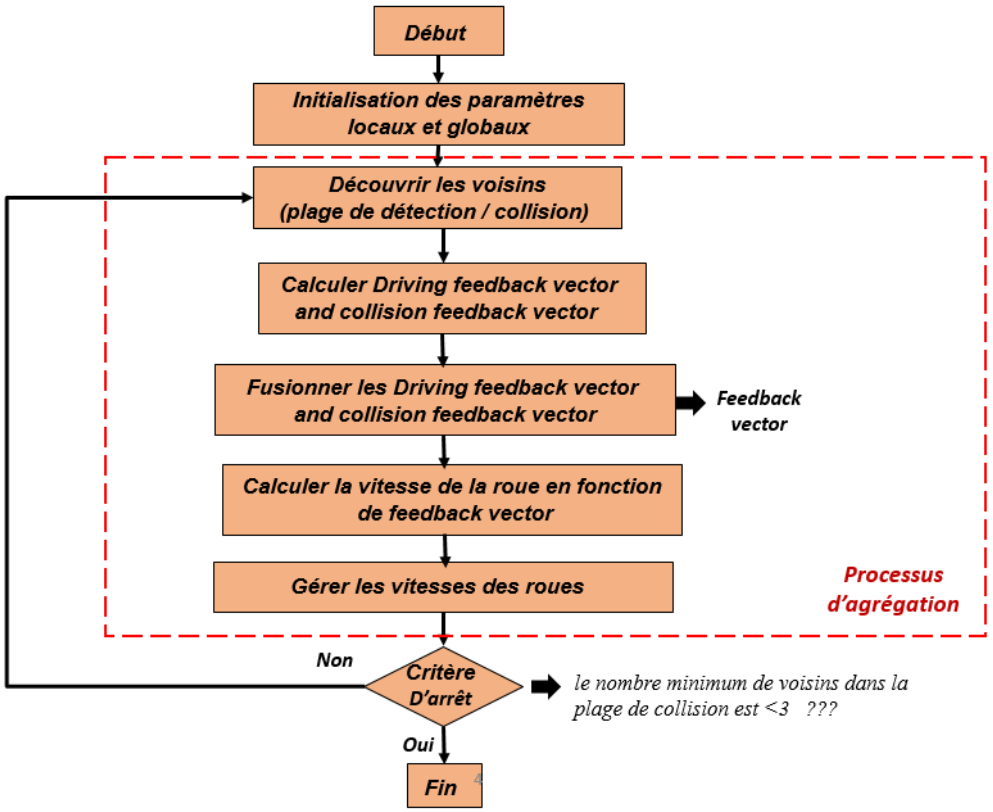


FIGURE 4.17 – Organigramme de l'algorithme d'agrégation .

Tout d'abord, nous avons utilisé des robots à deux roues pour obtenir les deux comportements collectifs (agrégation et formation de lignes), puis nous avons utilisé des robots Turtlebots pour obtenir les mêmes comportements et faciliter l'implémentation ultérieure de cet algorithme sur ce type de robots.

2) L'implémentation de l'algorithme d'agrégation sur ROS :

Dans cette étape, nous utilisons un package ROS gratuit existant sur le site Github [63]. Ce package utilise un robot à deux roues dans 2 comportements d'essaims : agrégation et formation de lignes. Les simulations fonctionnent très bien mais notre objectif était de remplacer le robot à deux roues, un robot simple, par un autre robot (le burger turtlebot) plus complexe, avec des capteurs qui lui permettent de se déplacer de manière autonome et d'éviter les obstacles.

Comme ce package a été conçu pour simuler les comportements d'essaims, il est très facile d'ajouter des robots et de les faire se déplacer de manière autonome en même temps. Afin d'expliquer notre travail, nous montons comment les différents fichiers sont appelés et exécutés pour le comportement d'un essaim. Le schéma 4.18 montre les fichiers nécessaires pour afficher les robots sur Gazebo.

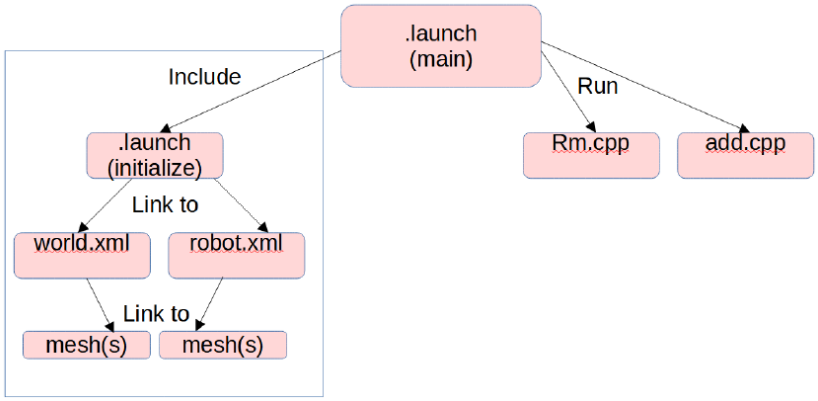


Figure 4.18 – Interaction entre les fichiers.

- *Robot manager (Rm)* : ce fichier gère le nombre et les indices des robots et publie des informations sur les robots à deux roues telles que : la position, l'indice, l'orientation et la vitesse.

- *Add* : ce fichier ajoute le nombre de robots à deux roues en communiquant avec le nœud du gestionnaire.

Notre objectif est de remplacer le robot two_wheels par le burger turtlebot. Et pour ce faire, nous avons modifié le fichier xml du robot précédent. Donc, nous avons proposé de garder la structure exacte du fichier xml, et de modifier le visuel des composants. Par exemple, nous avons remplacé les roues du robot à deux roues par les roues du burger turtlebot et ainsi de suite. Cette solution a fonctionné et maintenant nous pouvons avoir les comportements de l'essaim avec notre robot représenté dans la Figure 4.22. En effet, la Figure 4.20 montre le fichier xml du robot à deux roues avant la modification, tandis que la Figure 4.21 montre le fichier xml après la modification en remplaçant les composants du

robot à deux roues par les composants du robot burger Turtlebot.

Before

```xml
<?xml version="1.0"?>

<robot name="two_wheel_robot">


<link name="body_link">
  <inertial>
    <mass value="0.03" />
    <origin xyz="0 0 0.0127" />
    <inertia ixx="0.000032258" ixy="0.0" ixz="0.0"
      iyy="0.000032258" iyz="0.0"
      izz="0.000032258" />
  </inertial>

  <visual>
    <origin xyz="0 0 0.0127" />
    <geometry>
      <mesh filename="package://swarm_robot_description/urdf/mesh/twowheel_body.dae"/>
    </geometry>
  </visual>

  <collision>
    <origin xyz="0 0 0.0127" />
    <geometry>
      <!-- <box size="0.0254 0.0254 0.0254"/> -->
      <mesh filename="package://swarm_robot_description/urdf/mesh/twowheel_body.dae"/>
    </geometry>
  </collision>
</link>

<gazebo reference="body_link">
  <!-- tag name has to be mu1, though in gazebo they are mu and mu2 -->
  <mu1>0.1</mu1>
  <mu2>0.1</mu2>
</gazebo>
```

Figure 4.19 – Fichier xml du robot à deux roues.

After

```xml
<?xml version="1.0"?>

<robot name="two_wheel_robot" xmlns:xacro="http://ros.org/wiki/xacro">
        <xacro:include filename="$(find turtlebot3_description)/urdf/common_properties.xacro"/>
        <arg name="laser_visual" default="false"/>
        <arg name="imu_visual"   default="false"/>


<link name="body_link">
  <inertial>
      <origin xyz="0 0 0" rpy="0 0 0"/>
      <mass value="8.2573504e-01"/>
      <inertia ixx="2.2124416e-03" ixy="-1.2294101e-05" ixz="3.4938785e-05"
               iyy="2.1193702e-03" iyz="-5.0120904e-06"
               izz="2.0064271e-03" />
      </inertial>

        <visual>
      <origin xyz="-0.032 0 0.0" rpy="0 0 0"/>
      <geometry>
        <mesh filename="package://turtlebot3_description/meshes/bases/burger_base.stl"
scale="0.001 0.001 0.001"/>
      </geometry>
      <material name="light_black"/>
    </visual>

  <collision>
      <origin xyz="-0.032 0 0.070" rpy="0 0 0"/>
      <geometry>
        <box size="0.140 0.140 0.143"/>
      </geometry>
    </collision>
</link>
```

Figure 4.20 – Fichier xml du Turtlebot burger.

Cette figure présente une comparaison entre les deux comportements collectifs naturels des abeilles et des fourmis (agrégation et formation de lignes) et leurs simulations sur ROS/Gazebo avec les deux robots (robot à deux roues et Turtlebot burger).

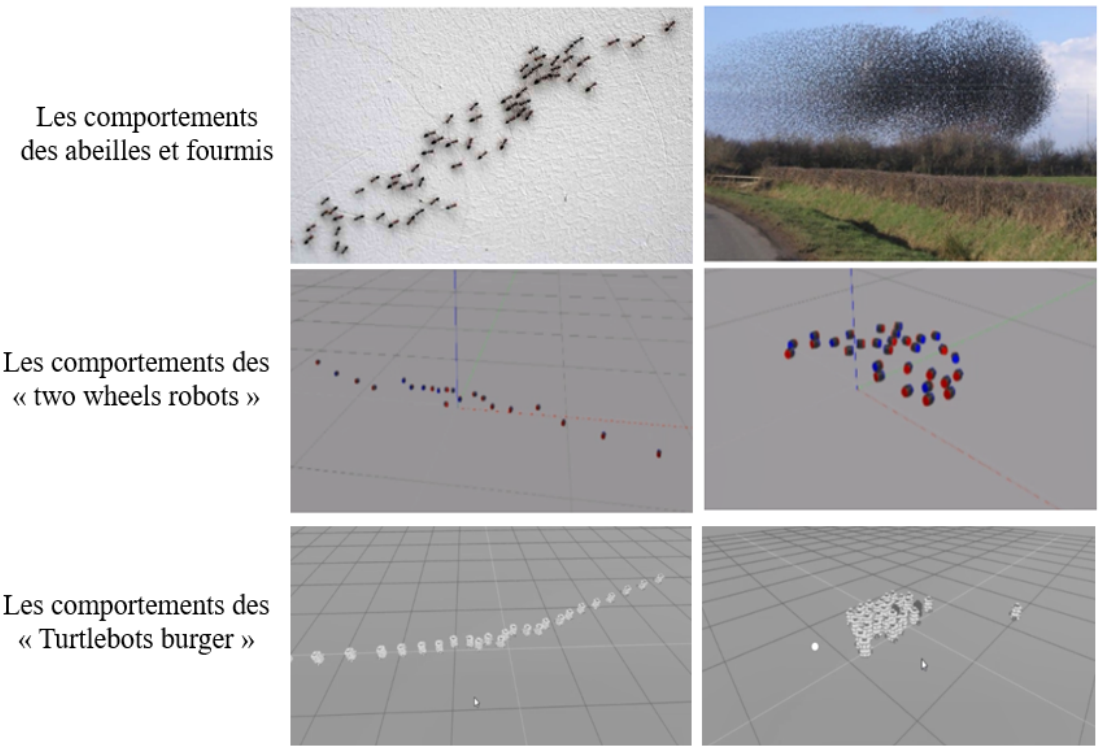


Figure 4.21 – Comparaison entre les comportements biologiques et leurs simulations sur ROS/Gazebo.

4.4 Conclusion

Dans ce chapitre, nous avons appliqué notre méthodologie de conception sur une étude de cas d'agrégation très appliquée dans le domaine de la robotique en essaim. Dans une première phase, nous nous sommes intéressés à la modélisation avec MBSE en utilisant les différents diagrammes SysML et SwarmML. Nous avons également simulé notre application d'agrégation sur un outil de simulation multi-agent (Anylogic) en utilisant la méthode probabiliste pour étudier l'influence des paramètres locaux (paramètres des individus) sur l'aspect global de l'essaim. Dans la deuxième phase, nous avons implémenté un algorithme d'agrégation de robots en essaim existant sur ROS/Gazebo mais nous avons modifié les robots à deux roues par des robots plus complexes (Turtlebots burger). En fait, le but de cette implémentation est d'abord de valider notre application d'agrégation et aussi de vérifier l'adaptabilité de cet algorithme avec n'importe quel type et nombre de robots.

Chapitre 5

Conception d'un système de véhicules à guidage automatique (AGVs)

La volonté trouve, la liberté choisit.
Trouver et choisir, c'est penser

– Victor Hugo

5.1 Introduction

Ces dernières années, les robots collaboratifs sont devenus l'un des principaux moteurs de l'industrie 4.0. Les progrès rapides des technologies de fabrication et des applications dans les industries augmentent la productivité. L'industrie d'aujourd'hui a besoin de la numérisation et de l'intelligence des processus de fabrication. L'industrie 4.0 représente la quatrième révolution industrielle qui se définit comme un nouveau niveau d'organisation et de contrôle de l'ensemble de la chaîne de valeur du cycle de vie du produit ; elle est orientée vers les besoins du client. L'industrie 4.0 est un concept réaliste qui comprend l'internet industriel, l'internet des objets et la fabrication intelligente. Par rapport aux robots industriels, les véhicules à guidage automatique (AGV) sont plus productifs, flexibles, polyvalents et plus sécurisés. Ils sont utilisés dans l'usine intelligente pour le transport des marchandises. Aujourd'hui, de nombreux producteurs et développeurs de robots industriels se sont lancés dans le secteur des véhicules à guidage automatique. Cependant, ils sont confrontés à plusieurs défis de conception des systèmes AGV, tels que la complexité et la discontinuité du processus de conception, ainsi que la difficulté de définir une décision décentralisée des systèmes.

L'introduction de tels systèmes robotiques dans les applications industrielles pose plusieurs problèmes qui ne peuvent être ignorés. Aujourd'hui, l'utilisation des AGV dans la logistique industrielle n'est pas encore très répandue dans les usines de fabrication. Le transport des matières premières et des produits finis est encore effectué par des chariots

élévateurs à fourche manuels. C'est pourquoi la logistique industrielle n'est pas encore bien intégrée dans les processus de fabrication modernes. L'utilisation de chariots élévateurs à fourche est peu efficace et entraîne une forte consommation d'énergie. En outre, l'utilisation des chariots élévateurs n'est pas sûre pour les travailleurs. C'est pourquoi de nombreux producteurs et développeurs de robots industriels se sont lancés dans le secteur des véhicules à guidage automatique afin de réduire ces problèmes, mais même l'utilisation de systèmes AGV présente des inconvénients en termes de productivité, de répétabilité, de précision et de sécurité du travail. En fait, l'intégration des systèmes AGV nécessite des coûts de maintenance élevés. En outre, l'utilisation des AGV est inadaptée aux tâches non répétitives et réduit la flexibilité des opérations.

De nombreux concepteurs s'inspirent des comportements collectifs des animaux sociaux, tels que le transport de nourriture par les fourmis ou les abeilles, pour concevoir des AGV. En effet, ces animaux suivent un chemin spécifique pour transporter la nourriture de la source à leur nid. Ce comportement peut être inspiré pour développer un algorithme de contrôle pour un système AGV. Malgré les possibilités offertes par les AGV, il reste de nombreux défis à surmonter pour assurer un développement efficace des AGV. Par exemple, la complexité et la discontinuité du processus de conception sont des problèmes majeurs rencontrés par les concepteurs et qui ralentissent l'intégration des AGV dans les usines intelligentes. En d'autres termes, il doit y avoir une continuité entre les données et les modèles utilisés tout au long du processus de conception pour éviter le risque d'erreurs et de reprises. En outre, le concepteur doit vérifier toutes les intégrations nécessaires (intégrations fonctionnelles, physiques et logicielles) pour garantir la cohérence de la conception. Par ailleurs, les méthodologies de conception des AGV doivent offrir la possibilité d'automatiser le processus de vérification des exigences de conception, en particulier les exigences liées aux possibilités de contrôle décentralisé permettant de prédire la décision collaborative du système d'AGV.

Afin de contribuer au développement des AGV et de faire face aux problèmes mentionnés ci-dessus, nous appliquons dans ce chapitre notre méthodologie de conception IDMSR sur un cas de conception d'un système des véhicules à guidage automatiques. En particulier, notre approche vise à réduire la discontinuité numérique et à améliorer la traçabilité entre les phases de spécification des exigences, de conception architecturale, d'intégration du code, de vérification de la conception et de validation par simulation. Pour ce faire, ce chapitre sera organisé comme suit : dans la prochaine section, nous présenterons les différentes méthodes proposées dans la littérature pour la conception de systèmes AGVs et nous détaillerons les différents problèmes de conception rencontrés tout au long du cycle de conception. Dans la troisième section, nous détaillerons l'implémentation de la méthode de conception proposée en utilisant le langage de modélisation de systèmes SysML et l'environnement de simulation ROS/Gazebo. Enfin, nous terminerons le chapitre par une conclusions.

5.2 Les méthodes de conception proposées pour la conception des systèmes AGVs

5.2.1 Processus de conception d'AGV existants

Aujourd'hui, les robots collaboratifs sont devenus un élément clé de l'industrie 4.0. Dans les applications industrielles, les AGV se sont révélés être une technologie très intéressante pour le transport de marchandises. Dans le domaine du développement des AGV, les chercheurs sont confrontés à plusieurs défis pour développer des AGV qui répondent à des besoins plus spécifiques et personnalisés en termes de charge à déplacer pour les applications de manutention, de localisation du véhicule dans son environnement, de planification de la trajectoire multi-robots, d'évitement des collisions, etc. Par exemple, Stouten et al. [127] ont décrit l'utilisation de systèmes AGV pour le transport coopératif de charges lourdes. L'étude de [85] s'est concentrée sur la conception d'un système de manutention basé sur des véhicules guidés automatisés pour un système de fabrication flexible. Ronzoni et al. [120] ont abordé le problème de la localisation du véhicule sans aucune information préalable sur son emplacement. Ils ont présenté une nouvelle méthode pour localiser un AGV doté d'un scanner laser et situé dans un environnement rempli de points de repère anonymes. Ils ont également proposé une méthode de correspondance de repères qui prend en compte les erreurs de mesure et les fausses détections, dues aux surfaces réfléchissantes présentes dans l'environnement. Leur stratégie a été validée par des expériences dans des environnements industriels réels et par une simulation sur des cartes d'usines réelles. Luna et al.[83] ont traité le problème de la planification de trajectoire multi-robots à travers un ensemble de nœuds de réseau qui guident les agents se déplaçant sur un graphe. Ils ont proposé des solutions partiellement décentralisées pour réduire la complexité du processus d'optimisation. En effet, les AGV sont généralement amenés à se déplacer dans des environnements encombrés, ils doivent donc être équipés d'un système de détection approprié, pour leur permettre d'identifier les obstacles. Dans un autre travail de recherche, Rodríguez-Seda et al. [119] ont proposé une stratégie décentralisée et coopérative d'évitement des collisions pour une paire d'agents en considérant des incertitudes de détection bornées et des contraintes d'accélération. Le contrôle de l'évitement n'est actif que lorsque le véhicule est proche d'un autre agent.

5.2.2 Méthodes de modélisation des systèmes AGVs

La modélisation du système représente un autre grand défi lié à la conception des systèmes AGV. En effet, la modélisation des AGV comprend plusieurs aspects tels que la cinématique non linéaire, le comportement dynamique du mouvement, le contrôle et la coordination du système pour évaluer les positions et les orientations des AGV. Par exemple, Sharma et al. [123] ont proposé de modéliser les AGV avec un modèle cinématique non linéaire qui décrit la position et la vitesse du robot, l'orientation et la vitesse angulaire de ses liens. Rajamani [117] utilise le modèle de bicyclette pour modéliser la dynamique latérale des AGV. Caruntu et al.[34] montrent que la dynamique longitudinale peut être modéli-

sée par un modèle de second ordre. Ces modèles décrivent la position et l'orientation du véhicule par rapport à un système de coordonnées. En outre, Oyelere [107] choisit des modèles non linéaires de type voiture et vélo pour décrire le mouvement dynamique des AGV. Caruntu et al. [33] illustrent le concept d'application de techniques de coordination et de contrôle bio-inspirées au développement de la fabrication et du transport de marchandises du futur. Ils présentent également une discussion sur les avantages et les inconvénients de plusieurs techniques pour leur utilisation dans des applications spécifiques.

Il est nécessaire de spécifier les exigences de conception d'un système robotique en essaim, de déterminer le comportement de chaque robot en fonction du comportement de l'essaim et de programmer ce comportement sur une plate-forme logicielle [5]. Des outils logiciels appropriés sont nécessaires pour maîtriser la complexité de la modélisation des systèmes en essaim [118]. Par exemple, l'ingénierie des systèmes à base de modèles (MBSE) est l'application formalisée de la modélisation pour soutenir les différentes étapes de l'évolution du système, de la phase de conception à toutes les phases du cycle de vie qui suivent. Ferreira et al.[47] développent un système de manutention intelligent basé sur un AGV en utilisant une approche MBSE. Ils conçoivent le noyau et le contrôleur de l'AGV dans l'environnement du langage de modélisation du système en utilisant le logiciel Visual Paradigm, puis ils mettent en œuvre le modèle dans le matériel. En conséquence, les tâches complexes de manutention, de navigation et de communication de l'AGV ont été accomplies et testées avec succès dans un environnement industriel réel. Ils ont également pris en compte les aspects ergonomiques et de sécurité dans la conception de l'AGV en utilisant un système de sécurité complet et en se conformant aux normes industrielles. Un certain nombre de langages de modélisation de systèmes sont utilisés dans l'industrie, tels que le langage de modélisation unifié (UML) et le langage de modélisation de systèmes (SysML). SysML est une extension de UML qui peut être utilisée pour modéliser des systèmes complets, y compris le matériel. Il existe plusieurs exemples d'applications de SysML pour la modélisation des systèmes de contrôle. Par exemple, Barth et al.[11] ont utilisé SysML pour modéliser et développer un nouveau contrôleur de pulvérisation thermique, du concept au système industriel entièrement fonctionnel. Selon Brecher [25], le principal avantage de l'utilisation de SysML est la possibilité de modéliser facilement des systèmes complexes, comme cela a été démontré dans le développement de la logique de commande d'un système de manipulation robotique. SysML est un langage de modélisation visuel qui peut être utilisé pour décrire la structure et le comportement d'un système en essaim. Les outils de modélisation peuvent être utilisés pour capturer la variété des diagrammes et maintenir la cohérence des éléments à travers les différentes représentations structurelles et comportementales du système.

5.2.3 Méthodes d'optimisation de la conception des systèmes AGVs

L'optimisation et l'ordonnancement des tâches pour les AGV est une autre question à prendre en compte dès les phases préliminaires de conception, car elle peut avoir un impact sur certaines variables et contraintes de conception telles que la durée de vie et le nombre d'AGV. Pour cette raison, plusieurs chercheurs travaillent sur des approches en essaim pour

développer des systèmes AGV. Par exemple, Mousavi et al. [103] ont développé un modèle mathématique intégré avec des algorithmes évolutionnaires (algorithme génétique (GA), optimisation par essaims de particules (PSO), et hybride GA-PSO) pour optimiser l'ordonnancement des tâches des AGV afin de minimiser la durée de vie et le nombre d'AGV. Jerald et al. [67] ont résolu le problème d'ordonnancement du système de fabrication flexible (FMS) et le problème de contrôle pendant le fonctionnement des véhicules à guidage automatique (AGV) en utilisant l'algorithme de l'essaim de particules (PSA). La mise en œuvre d'algorithmes inspirés de la nature pour le contrôle et la coordination de robots fait partie d'une approche de conception de robots appelée robotique en essaim. La robotique en essaim est donc une approche relativement nouvelle de la coordination d'un grand nombre de robots autonomes simples. Cette approche s'inspire du système des insectes sociaux qui présentent trois caractéristiques : robustesse, scalbilité et flexibilité. L'analyse et la conception de systèmes auto-organisés comme la robotique en essaim est une tâche difficile même si nous avons une connaissance complète de l'intérieur du robot. En outre, Lategahn et al [78] ont proposé une méthodologie intégrée basée sur les principes de la robotique en essaim pour concevoir un essaim de petits véhicules guidés automatisés (AGV). Cet essaim de robots est utilisé pour collecter des articles dans les rayons d'un magasin et les amener à un poste de préparation de commandes. Un défi de cette approche est de fournir des techniques efficaces de suivi et de localisation pour obtenir la position de l'AGV à tout moment.

5.2.4 Méthodes de simulation des systèmes AGVs

L'intégration, la vérification et la validation (IVV) sont des étapes décisives pour le développement de systèmes complexes. En ce qui concerne les AGV, il est nécessaire de disposer d'une plate-forme logicielle capable d'intégrer tous les codes logiciels liés aux composants matériels et de vérifier la coordination de tous les membres du système en essaim. Une telle plate-forme logicielle doit pouvoir vérifier par simulation les actions de gestion assurant un mouvement sans conflit pour mettre en œuvre un système de navigation qui répond aux exigences d'un système en essaim [55]. Pour ce faire, l'environnement de simulation doit être capable de localiser les membres de l'essaim, de définir l'environnement et de planifier des trajectoires optimales dans l'environnement. De plus, pour permettre la communication avec les capteurs et les actionneurs, une couche d'abstraction matérielle est nécessaire. Depuis 2014, Robot Operating System (ROS) propose un progiciel traitant des AGV. ROS est un middleware open-source pour les plateformes robotiques. Il fournit toutes les fonctionnalités nécessaires d'un système d'exploitation et permet le développement d'applications en C++ et Python. Cette plateforme utilise plusieurs concepts lors de son fonctionnement. Un nœud est une instance d'un exécutable. Il peut correspondre à un capteur, un moteur, un algorithme de traitement, un algorithme de surveillance, etc.

5.3 Application de la méthodologie IDMSR

Les véhicules à guidage automatique (AGV) sont des plateformes mobiles sans conducteur utilisées dans l'industrie pour le transport de matériaux. Dans cette étude de cas, nous examinons la conception d'un système AGV à usage général qui peut être adapté à plu-

sieurs environnements de travail pour transporter des matériaux.En effet, nous appliquons notre méthodologie de conception IDMSR developpé dans le chapitre 3. Dans ce qui suit, les différentes phases de la méthodologie IDMSR sont appliquées.

5.3.1 Première phase : Modélisation avec la méthode MBSE

- Exigences de conception de l'essaim :

Les AGV sont déployés dans de nombreux domaines d'application différents et les types de véhicules ont augmenté pour répondre aux besoins des clients. Ces types de véhicules sont utilisés dans les secteurs de la fabrication, de l'automobile, de l'entreposage et de l'alimentation. Cette variété d'applications spécifie les exigences du système polyvalent. Le premier diagramme SysML à traiter est le diagramme d'exigences pour le modèle AGV présenté à la Figure 5.1. Comme le montre cette figure, la relation de traçabilité "satisfy" est ajoutée pour assurer la traçabilité entre les différentes vues de conception. En effet, dans le cadre de la conception de systèmes complexes, cette traçabilité est essentielle car elle permet de vérifier que les exigences sont satisfaites pour chaque composant du système et pour le système dans son ensemble.

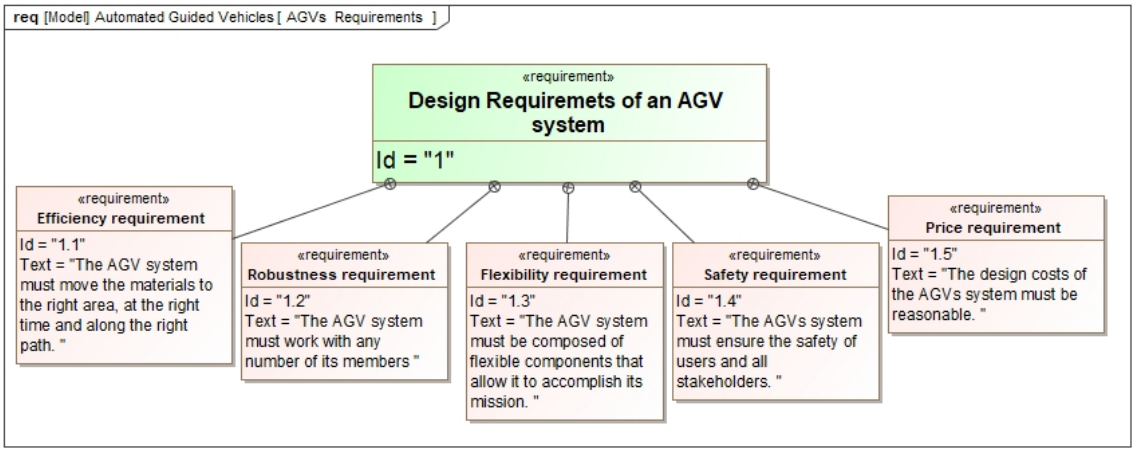


Figure 5.1 – Diagramme d'exigences de la conception de l'AGV en SysML

La conception des systèmes AGV est coûteuse en termes d'argent et de temps, car elle doit prendre en compte l'ensemble du cycle de vie du système, y compris la phase d'installation et la logistique. En effet, la définition des règles de circulation des AGV et de la carte des trajectoires est une opération qui prend du temps et qui peut augmenter les coûts d'installation et d'exploitation. En outre, l'efficacité est l'une des principales exigences de la conception des systèmes AGV. C'est la principale raison de l'adoption d'un système AGV pour la logistique industrielle. En outre, l'utilisation des AGV introduit de la flexibilité dans le système logistique. Enfin, les AGV doivent être intrinsèquement sûrs, pour ne jamais causer de blessures aux humains. Certaines des principales exigences de conception des AGV ont été énumérées ci-dessus, mais la liste pourrait être aussi longue que cela en fonction des besoins spécifiques. En outre, de nouvelles exigences de conception peuvent apparaître au cours du cycle de conception en fonction des décisions prises et des choix de solutions

et de technologies utilisées. A ce niveau, nous nous limitons aux exigences majeures qui nous permettent de commencer l'étude du comportement des AGV et en particulier de leur comportement en essaim.

- Modélisation du comportement de l'essaim :

Les véhicules à guidage automatique (AGV) sont utilisés pour la logistique industrielle automatisée, comme le transport de matières premières ou de produits finis. Ils sont utilisés pour la gestion du flux de ressources entre le point d'origine et le point de destination afin de répondre aux exigences de certaines parties prenantes, par exemple les entreprises ou les clients [121]. L'analyse du comportement des AGVs commence par la description des scénarios d'utilisation et des missions à accomplir par les AGVs. En langage SysML, les scénarios d'utilisation peuvent être modélisés à l'aide de diagrammes de cas d'utilisation SysML illustré à la Figure 5.2.

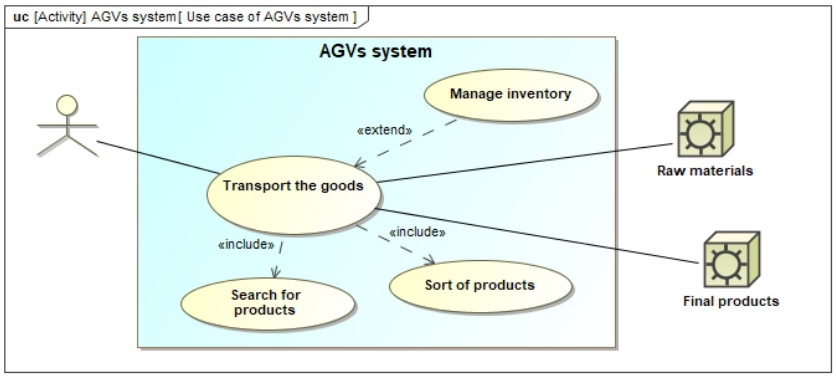


Figure 5.2 – Cas d'utilisation possible d'un système AGV.

- *Scénario AGV* : Un AGV transfère généralement une palette de marchandises depuis une ligne de production automatisée. La palette doit être amenée à la zone d'expédition. Parfois, les palettes doivent être stockées dans un entrepôt. La Figure 5.3 représente ce scénario AGV.

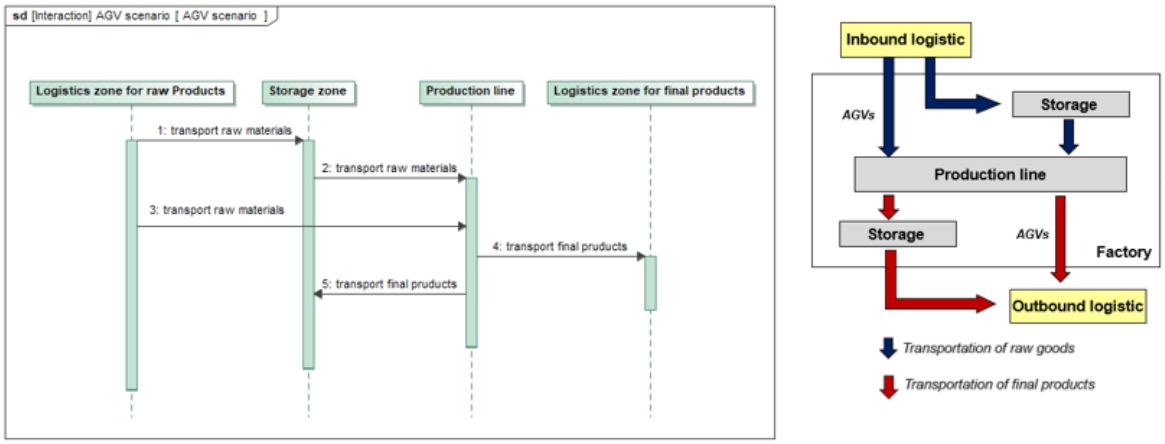


Figure 5.3 – Présentation descriptive d'un scénario AGV.

- *Mission d'une AGV* : Une mission doit être accomplie par un AGV. Elle commence par une tâche définie comme une séquence de segments de la carte de route à suivre par les AGV. Par la suite, chaque trajet est effectué par un AGV pour déplacer une palette de marchandises d'un endroit à un autre. En effet, les opérations de chargement et de déchargement sont effectuées au début et à la fin de chaque trajet. Cette mission peut être définie à l'aide du diagramme d'activités présenté à la Figure 5.4.

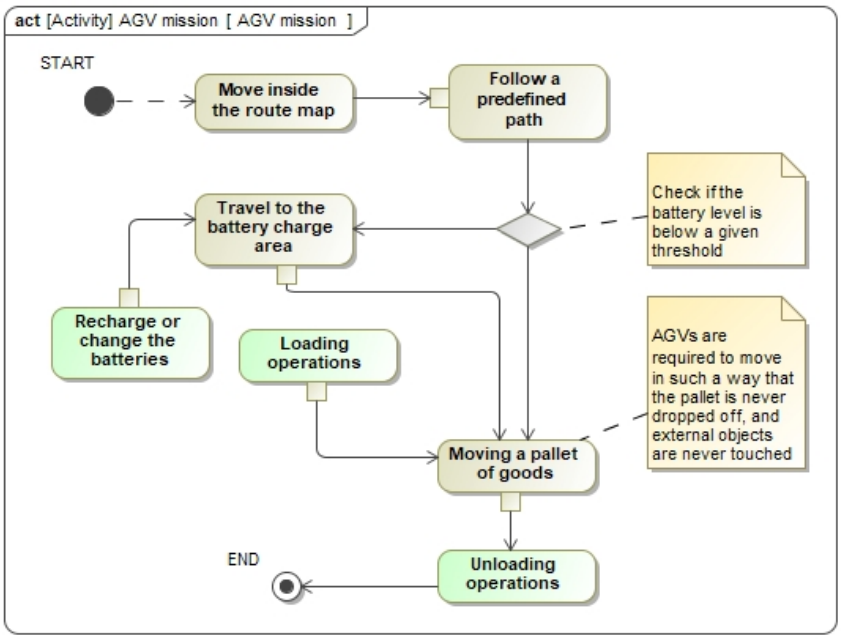


Figure 5.4 – Modélisation d'une mission d'une AGV sur SysML.

- *Mission de l'essaim d'AGVs* : Chaque AGV effectue la mission en parallèle avec les autres AGV (c'est-à-dire que tout au long de la mission globale, chaque AGV effectue sa mission). Cette mission d'essaim peut être définie à l'aide du diagramme d'activités à la Figure 5.5.

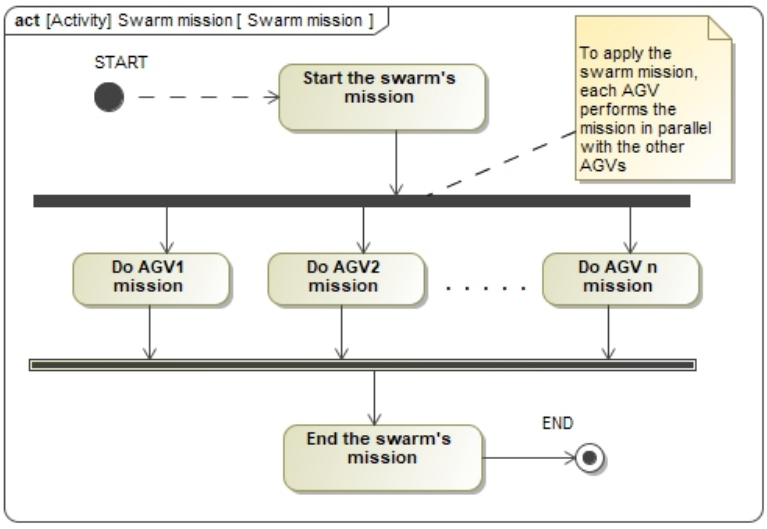


Figure 5.5 – Modélisation d'une mission en essaim d'AGVs en SysML.

- Modélisation de l'architecture de l'essaim :

L'essaim AGV est constitué d'un groupe d'individus AGV. Le diagramme de définition de bloc illustré à la Figure 5.6 représente la structure des principaux sous-systèmes et composants d'un AGV. Le sous-système de mouvement de l'AGV se compose de quatre roues et de quatre moteurs. Chaque roue est entraînée par un servomoteur avec une boîte de vitesses assemblée. Le système doit comporter quatre pilotes qui reçoivent des commandes et envoient un retour d'information à la carte contrôleur à 4 axes. En outre, le sous-système d'alimentation de chaque AGV est constitué d'une batterie autonome. Une caméra stéréo est ajoutée à l'avant du véhicule pour recueillir des images et des informations.

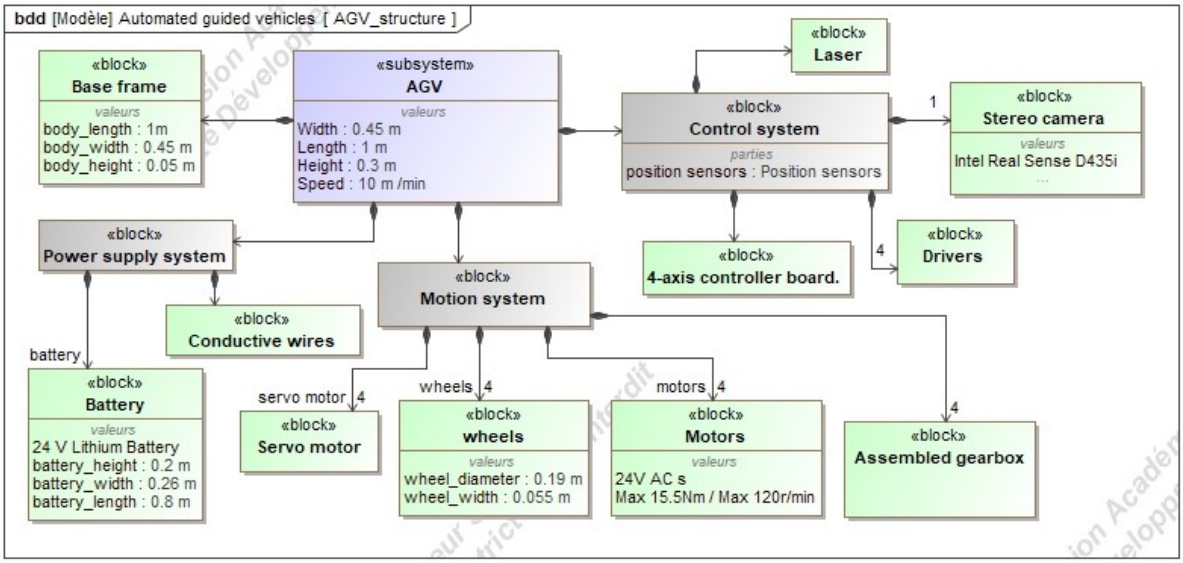


Figure 5.6 – L'architecture du système AGV en SysML.

Pour assurer la continuité de la conception, une matrice d'allocation illustré sur la Figure 5.7 est définie pour relier les composants matériels du système AGV aux fonctions individuelles fournies par chaque AGV.

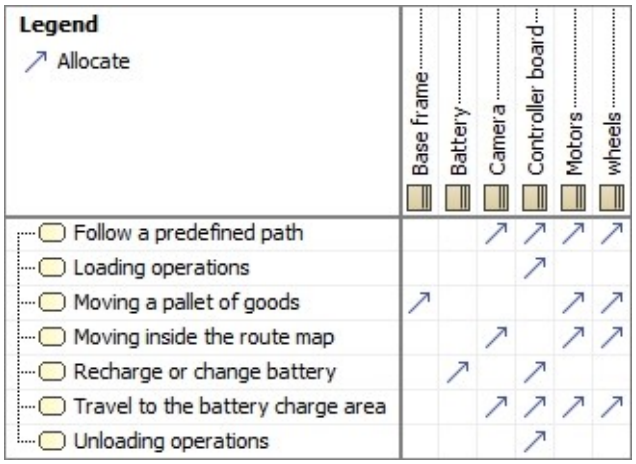


Figure 5.7 – Matrice d'allocation composants - fonctions.

- Simulation multi-agents :

Dans cette étape, un outil logiciel multi-agent appelé AnyLogic est utilisé pour mettre en œuvre le modèle de simulation AGV (AnyLogic Simulation Software 2016). Généralement, les AGV se déplacent sur les étages des installations sans être guidés par des chemins prédéfinis. En fait, notre modèle est composé de deux flottes d'AGV différentes. En utilisant les modèles développés avec la méthode MBSE ci-dessus, la simulation présentée dans la Figure 5.8 montre que les AGVs se déplacent entre les convoyeurs, les racks et les buffers des stations, ils trouvent automatiquement les chemins les plus courts et évitent les collisions avec les autres transporteurs et les obstacles.

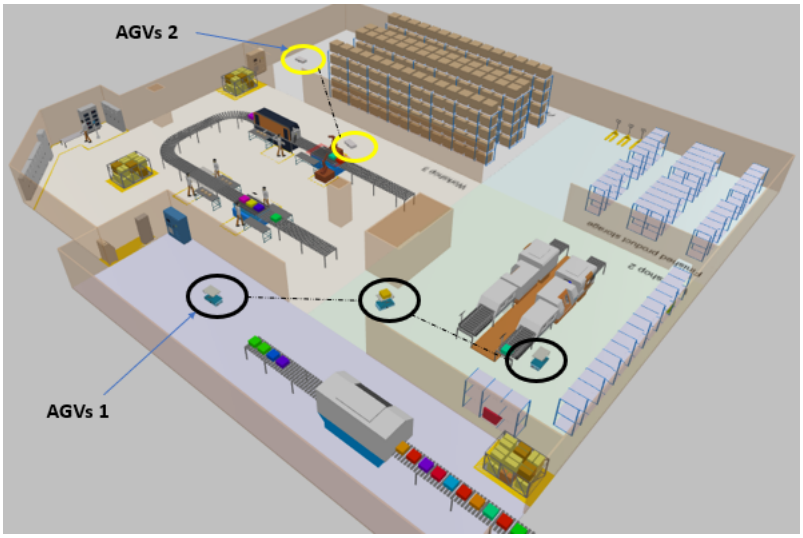


Figure 5.8 – Modèle simulé du système AGVs dans une usine intelligente.

5.3.2 Deuxième phase : Implémentation sur ROS

Pour vérifier le développement effectué dans la phase précédente, les modèles développés du système AGV doivent être mis en œuvre dans l'environnement ROS. Pour cela, le concepteur crée un espace de travail composé d'un ensemble de packages. Pour prendre en compte le comportement de l'essaim, les AGV individuels sont associés à des nœuds dans le système ROS, et un sujet de communication est créé pour souscrire et poster les informations échangées entre les AGV individuels. Les packages ROS sont implémentés sur la base des différents diagrammes SysML développés lors de la première phase de conception. En effet, le diagramme de définition des blocs BDD et les diagrammes de blocs internes IBD sont utilisés pour créer le fichier URDF et le fichier WORLD dans ROS. Le fichier URDF décrit la structure générale du système AGV et le fichier WORLD décrit l'environnement de travail (usine). Ensuite, les diagrammes des états et les diagrammes d'activité sont utilisés pour développer les algorithmes de contrôle du système AGV dans ROS. La Figure 5.9 montre la méthodologie d'intégration de ROS.

L'un des points forts de ROS est la réutilisation de codes développés par d'autres et qui peuvent être trouvés partagés par les utilisateurs de ROS (sur le site GitHub par exemple).

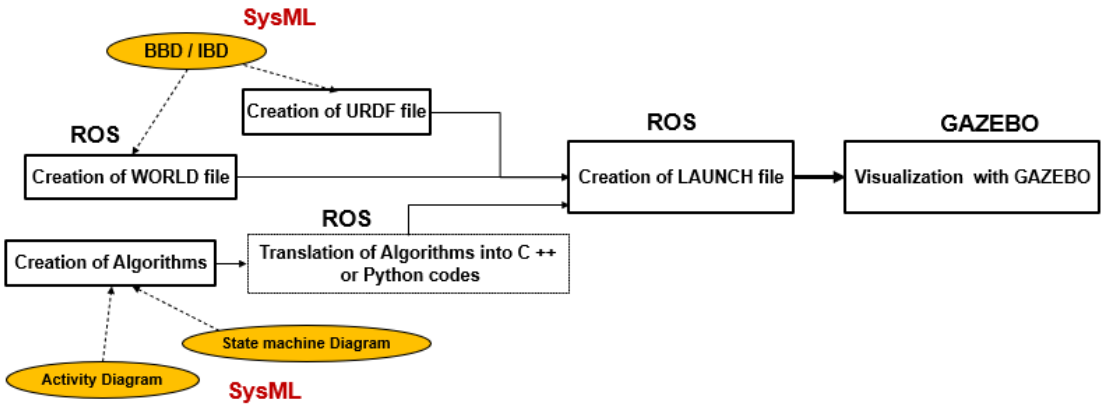


Figure 5.9 – Méthodologie d'intégration de ROS.

Les utilisateurs peuvent trouver des versions écrites en Python et en C ++ qui diffèrent par leurs restrictions, leurs performances et leur code allégé. Les développeurs de l'AGV peuvent les télécharger et les rendre utilisables en les compilant (en utilisant les processus de construction de catkin), et de nouvelles versions de ces codes seront disponibles en tant qu'autres packages basés sur ROS. Sur la base des descriptions SysML réalisées lors de la première étape de conception, les développeurs d'AGV peuvent intégrer les codes les plus appropriés qui respectent les exigences de l'AGV, l'architecture fonctionnelle et son architecture physique. L'architecture logicielle sera donc plus facilement intégrée sur la base des progiciels ROS existants. La tâche des développeurs consiste ensuite à adapter le code et à l'intégrer aux autres éléments ROS. Dans cette étude, nous choisissons d'adapter les packages "agvs" développés par la société "Robotnik Automation" [62] à notre cas d'étude pour valider notre méthodologie. En fait, nous suivons une approche ascendante guidée par un modèle d'information SysML pour sélectionner une solution de base d'un AGV afin de créer une nouvelle conception d'AGV qui répond aux exigences du système. Cela permet de gagner beaucoup de temps dans le processus de conception de l'AGV. Dans les sections suivantes, nous détaillerons comment implémenter un seul système AGV dans la plateforme ROS. Quelques remarques seront alors faites sur la simulation de l'essaim d'AGV.

- Création du fichier URDF :

La première étape consiste à créer le modèle descriptif URDF de l'AGV à des fins de simulation avec l'outil Gazebo (l'environnement 3D de ROS pour la simulation). Le fichier URDF peut être directement défini à l'aide des outils Gazebo sur la base de la description SysML de la structure de l'AGV. Cependant, pour une conception 3D complexe d'un AGV, il serait plus pratique de générer le fichier URDF à partir d'un outil de CAO 3D. Pour cela, certains outils de CAO, tels que SolidWorks et FreeCad, disposent de plugins permettant la génération automatique du fichier URDF en spécifiant les articulations et les liens du robot. Dans ce cas, la description SysML de la structure du robot peut être utilisée par le concepteur CAO pour définir le modèle CAO 3D de l'AGV. La Figure 5.10 montre la procédure de création d'un modèle multicorps pour l'environnement Gazebo à l'aide du plugin SolidWorks (SW2URDF). Pour chaque lien (composant) de l'AGV, il est nécessaire de spécifier le nom, le type de collision et le composant visuel. Il est également possible d'ajouter des caméras, des capteurs ou des moteurs aux liens. A ce stade, les valeurs des propriétés

physiques peuvent être définies pour chaque composant comme cela a été spécifié dans le modèle de structure SysML de l'AGV.

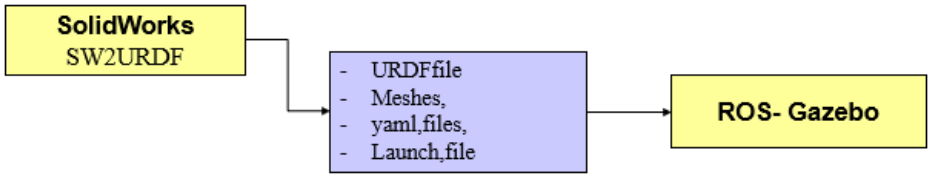


Figure 5.10 – Modélisation d'un système AGV de SolidWorks à Gazebo.

Dans un logiciel de CAO, tel que SolidWorks, le BDD SysML illustré à la Figure 5.6 aide le concepteur de CAO 3D à modéliser la structure de l'AGV. Ce diagramme définit les différents composants du système avec leurs propriétés physiques. D'autre part, le diagramme de blocs interne (IBD) illustré à la Figure 5.11 explique les interactions entre les composants qui aident à définir les différents liens et joints entre les éléments de l'AGV pendant la génération du fichier URDF.

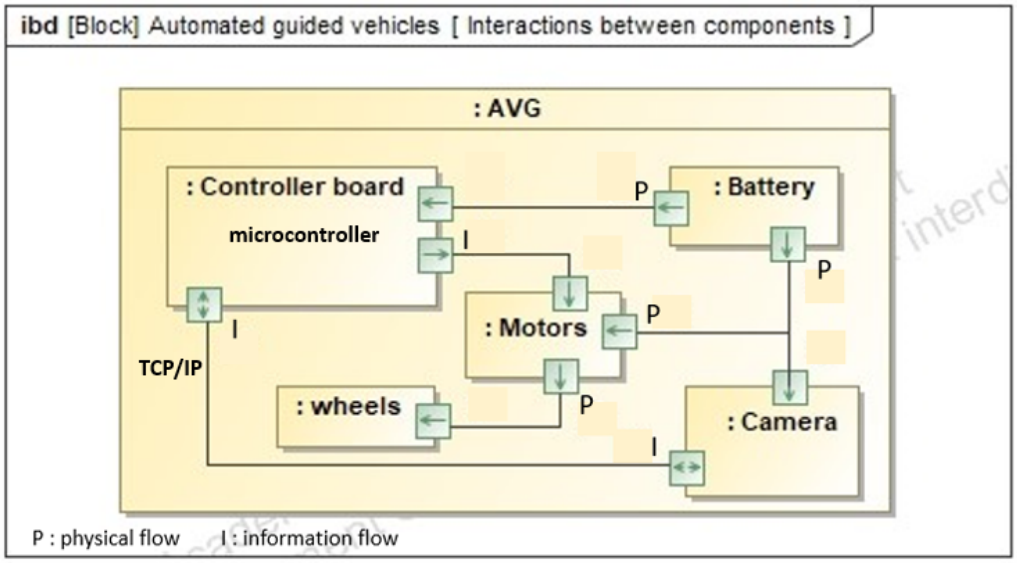


Figure 5.11 – Interaction entre les composants du système dans SysML.

La Figure 5.12 montre la structure 3D de l'AGV selon le modèle développé avec les diagrammes IBD et BDD. La largeur et la longueur sont respectivement de 0,45 m et 1 m et la hauteur de 0,3 m. Cette structure AGV est composée de quatre roues et de quatre moteurs. Chaque roue est entraînée par un servomoteur avec une boîte de vitesses assemblée et chaque AGV contient une batterie autonome.

- Création du fichier WORLD :

La deuxième étape consiste à créer un fichier WORLD qui décrit l'environnement de travail. Les utilisateurs de ROS peuvent réutiliser et adapter des fichiers World existants, ou les développer à l'aide de Gazebo ou d'un logiciel de CAO 3D. Dans ce cas d'étude, l'environnement du système AGV est une usine qui a été adoptée dans la bibliothèque de fichiers

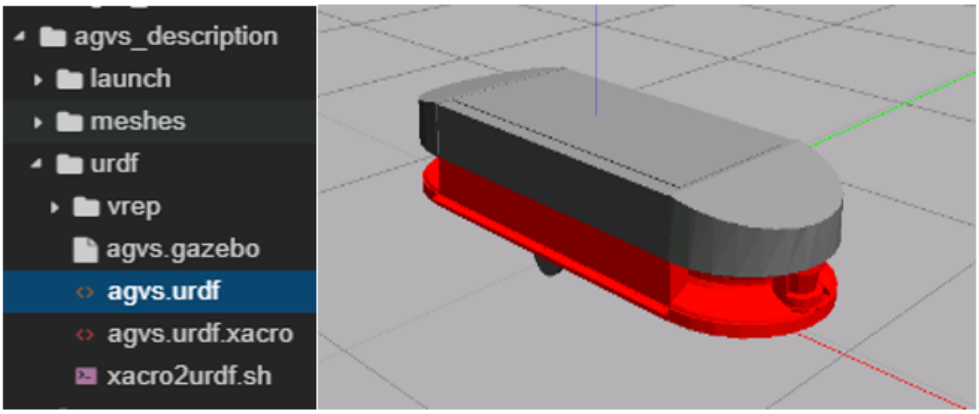


Figure 5.12 – Modèle 3D d'un AGV dans un Gazebo.

World de ROS/Gazebo. Le fichier "agvs_office.world" contient les différents éléments de l'usine présentés dans la Figure 5.13. Cette usine est une surface plane composée de murs, de grandes palettes, de palettes vides et de cadres de convoyeurs. Tous ces composants sont disponibles dans la bibliothèque ROS pour construire des environnements de travail personnalisables.

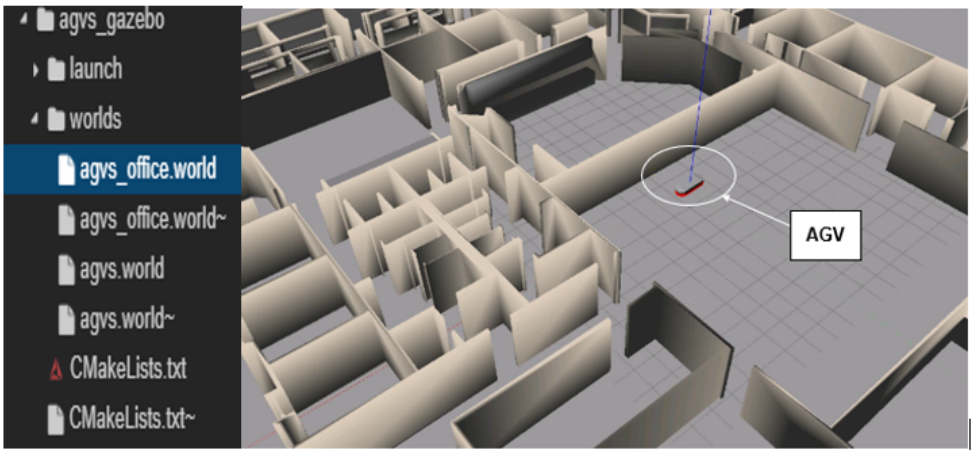


Figure 5.13 – Modèle 3D d'une usine dans Gazebo.

- Création des algorithmes :

La troisième étape consiste à créer les algorithmes qui gèrent le fonctionnement du système AGV dans l'usine. Notre méthodologie simplifie la création de ces algorithmes car les modèles développés avec les diagrammes des états en SysML sont utilisés pour développer ces algorithmes de fonctionnement. La Figure 5.14 montre le processus de création des algorithmes de fonctionnement des AGV à l'aide des diagrammes SysML.

La conversion de ces algorithmes en codes logiciels utilisables est l'étape la plus difficile qui requiert l'ingéniosité d'un développeur informatique. Cependant, avec le travail de préparation préalable (de diagramme des états aux algorithmes), la complexité de la tâche des développeurs est considérablement réduite grâce à un meilleur niveau de conformité entre

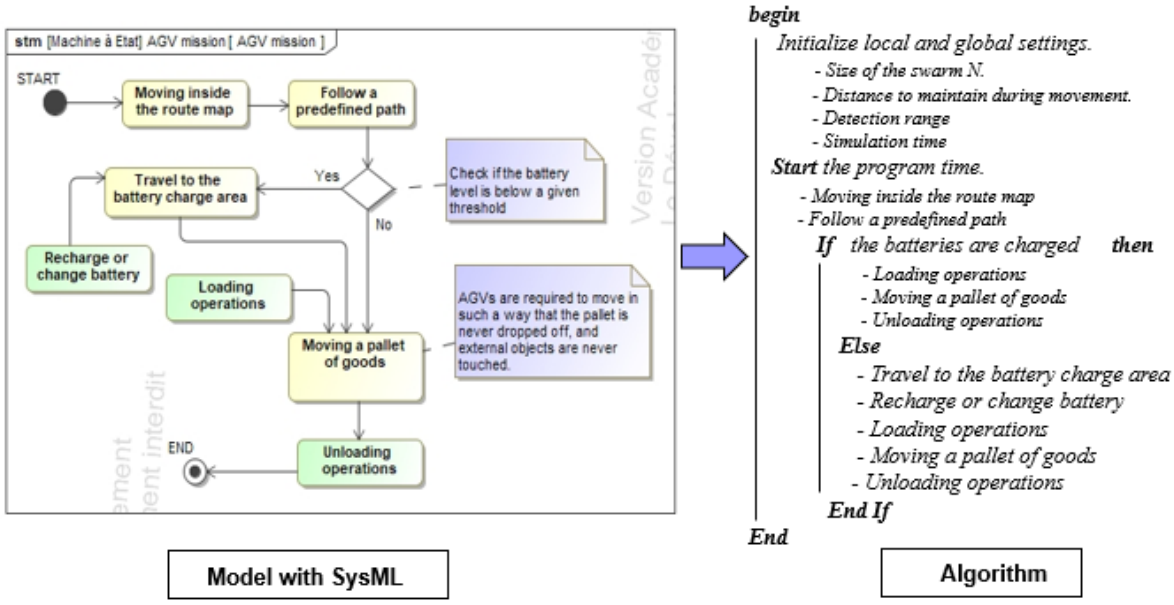


Figure 5.14 – Création d'algorithmes de fonctionnement des AGV à l'aide d'un diagramme SysML.

les différents codes. En fait, les développeurs informatiques traduisent les algorithmes en codes C ++ ou Python vers les nœuds ROS qui seront appelés ultérieurement par les fichiers d'exécution. Dans la plupart des cas, les développeurs informatiques peuvent trouver des codes partagés similaires à ceux qui doivent être mis en œuvre dans leur application, grâce au partage du code par la communauté ROS. Leur tâche consiste donc à adapter les codes existants et à ajouter les codes manquants, tout en restant guidés par le modèle SysML.

Comme exemple de code utilisé dans cette étude de cas, le code de positionnement et de cartographie simultanés (SLAM) du système AGV a été mis en œuvre pour dessiner une carte en estimant sa position actuelle dans un espace arbitraire. Le SLAM est défini comme le problème de la construction d'une carte en même temps que la localisation de l'AGV dans ce plan. En fait, l'AGV peut s'appuyer sur deux sources d'information : des informations qui lui sont propres et des informations recueillies dans son environnement. Lorsqu'il est en mouvement, l'AGV peut utiliser la navigation à l'estime et les informations renvoyées par ses capteurs (roues codeuses, consommation des moteurs, position d'un servomoteur, etc.) Cependant, ce type d'information n'est pas totalement fiable (glissement, jeu, friction, etc.). L'autre source d'information provient de capteurs et de systèmes dépendant de l'environnement et de sources externes (caméra et laser). Un autre environnement de visualisation ROS (Rviz) peut être utilisé pour voir comment la carte est créée lorsque le robot se déplace. Après avoir enregistré la carte, il est possible de déplacer l'AGV sur une trajectoire choisie. La Figure 5.15 représente la visualisation Rviz d'un AGV se déplaçant sur un chemin dans l'usine.

Enfin, Gazebo permet de visualiser le fonctionnement du système AGV en 3D. La Figure 5.16 montre le déplacement du système AGV dans l'environnement 3D de Gazebo. Le but de cette simulation était de vérifier si l'AGV est capable de parcourir un chemin mémorisé et de transporter des palettes de marchandises vers la zone de déchargement, de sorte qu'une

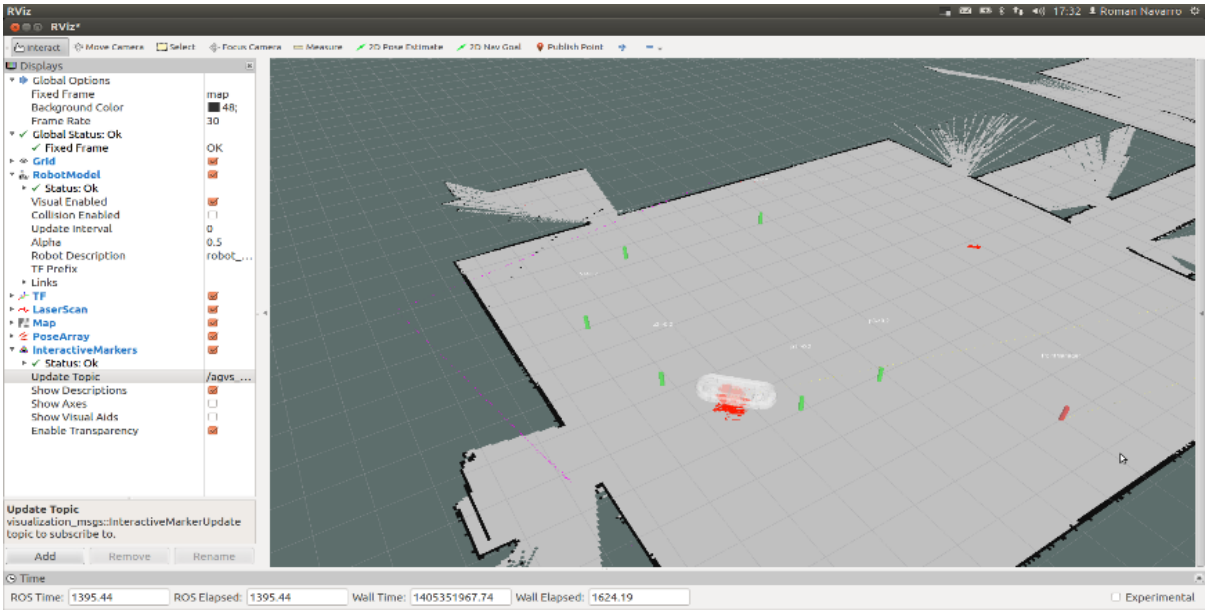


Figure 5.15 – Visualisation Rviz d'un AGV se déplaçant sur un chemin.

exigence principale de la conception soit vérifiée par la simulation.

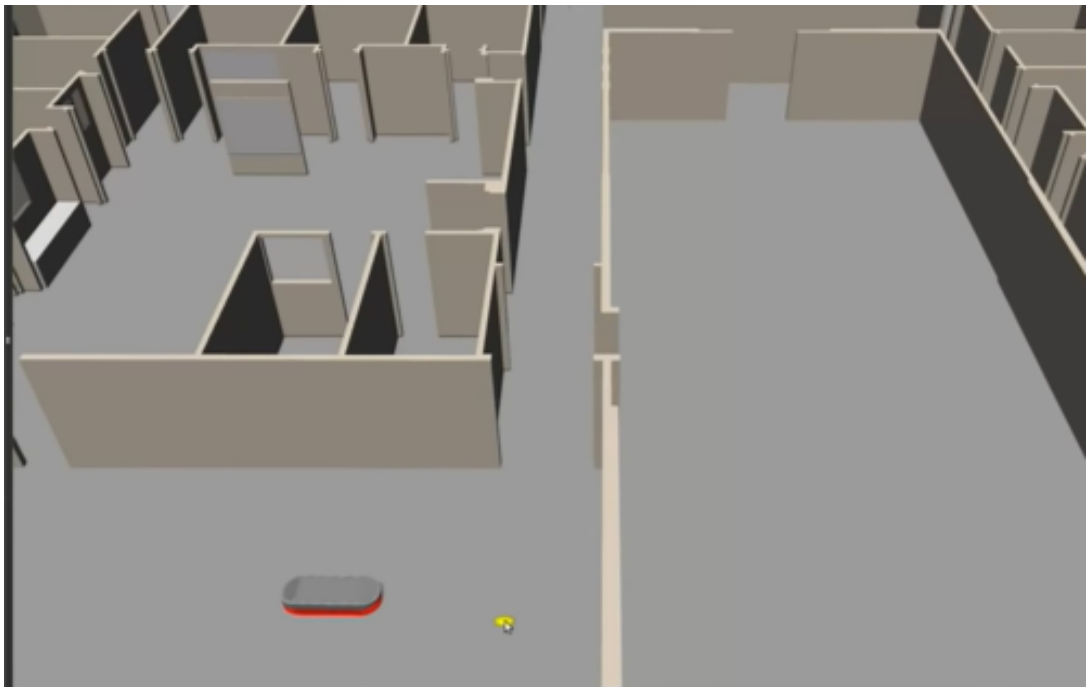


Figure 5.16 – Visualisation par GAZEBO d'un AGV se déplaçant sur une trajectoire.

L'étape finale de l'approche de conception proposée est la validation des exigences de conception à l'aide de la simulation ROS basée sur la description SysML. Le tableau 1 illustré sur la Figure 5.17 fournit des éléments de validation illustratifs de trois exigences de conception. Le tableau indique la conception, l'exigence, les actions de simulation effectuées dans l'environnement de simulation ROS et les informations SysML utilisées à cette

fin.

Requirement	*ROS Simulation*	*SysML Modeling*	
▪ The AGVs system must move materials to the correct area at the right time and path.	▪ Gazebo allows to view the AGVs of the system operating in 3D. the AGV will travel a memorized path and transport pallets of goods to the unloading area.	▪ The state machine diagram describes the mission accomplished by the AGV. Each trip is made by an AGV to move a pallet of goods from one location to another.	Satisfied
▪ An AGV system should consist of flexible components that allow it to get the mission done.	▪ A descriptive model of an AGV with Gazebo. ▪ Plugin SW2URDF in Solidworks ▪ Creation of urdf file that describes the structure of the AGV system	▪ The block definition diagram BDD represents the layout and main components of AGV developed. ▪ The IBD diagram which describes the interaction between the components of the system.	Satisfied
▪ The AGVs system must be operated in a well-defined environment	▪ Creation of WORLD file that describes the working environment. ▪ The AGVs system is worked in a factory.	▪ The IBD diagram which describes the interaction between the AGVs system and its environment.	Satisfied

Figure 5.17 – Satisfaction du système par rapport aux exigences initiales

5.4 Conclusion

Dans l'étude de cas précédente, nous avons illustré la méthodologie descendante basée sur les diagrammes SysML pour modéliser le système AGV. La modélisation du système complexe est simplifiée car la méthode regroupe les trois vues des représentations de l'AGV (vue comportementale, vue fonctionnelle et vue structurelle) au sein d'un même modèle. Ce type de modélisation garantit la cohérence des données car les règles du SysML donnent à chaque élément du modèle une définition unique, construite en rassemblant les informations de ses différentes représentations, et les empêchent de se contredire. De plus, avec la méthodologie ascendante basée sur l'environnement ROS, il nous a été facile d'intégrer les codes et les environnements de simulation existants pour le système AGV en fonction des spécificités du modèle SysML afin de les adapter à la conception spécifique de l'AGV. En effet, les informations contenues dans le modèle SysML aident le concepteur à mettre en œuvre le modèle de simulation 3D dans ROS, l'environnement de simulation, les attributs de l'AGV et les codes qui doivent respecter les algorithmes spécifiés dans SysML pour la description du comportement. Pour cela, les informations contenues dans les différents diagrammes SysML (BDD, IBD, diagramme des états, etc.) sont transformées en fichiers ROS (fichiers XML, fichiers URDF, codes, etc.). La validation de la conception du système AGV par simulation nécessite de définir les cas de test de simulation en fonction du diagramme d'exigences SysML.

Chapitre 6

La technologie multi-SLAM dans un environnement industriel

L'esprit n'a de frontières que celles qu'on lui impose.

– Rachel Gudet

6.1 Introduction

La localisation et la cartographie simultanées - en anglais Simultaneous localization and mapping (SLAM)- est l'un des problèmes les plus fondamentaux de la robotique collaborative, puisque l'estimation de l'ego-motion et la construction de cartes sont essentielles pour permettre la navigation autonome. Par conséquent, le SLAM collaboratif deviendra bientôt la clé du succès des futures applications robotiques. Dans ce chapitre, nous définissons dans la première partie, les concepts de base de la SLAM. Nous soulignons également les principaux défis et les limites du SLAM collaboratif appliqué aux robots mobiles autonomes. Dans la deuxième partie, nous appliquons notre méthodologie de conception IDMSR. En effet, notre application est la conception d'un système robotique composé de cinq Turtlebots3 (Burger) qui dessinent une carte en estimant la position actuelle dans une usine en utilisant la technologie SLAM. La carte a été créée avec les informations de distance obtenues à partir du capteur et les informations de pose de chaque robot lui-même. Ensuite, ces données cartographiques sont fusionnées à partir de chaque TurtleBot3 pour donner une carte globale de l'usine. Une fois la carte globale créée, un autre type de robot Turtlebot3 (Waffle) utilise cette carte pour se déplacer d'un endroit à un autre dans l'usine pour diverses opérations telles que la livraison de matériel, la surveillance ou le nettoyage. Nous concluons par une validation de notre méthodologie de conception.

6.2 La technologie Multi-SLAM dans un environnement industriel

6.2.1 Introduction

Dans cette étude de cas, nous appliquons notre méthodologie de conception IDMSR . Notre application est la conception d'un système robotique composé de cinq Turtlebots3 (Burger) qui dessinent une carte en estimant la position actuelle dans une usine en utilisant la technologie SLAM. La carte a été créée avec les informations de distance obtenues à partir du capteur et les informations de pose de chaque robot lui-même. Ensuite, ces données cartographiques sont fusionnées à partir de chaque TurtleBot3 pour donner une carte globale de l'usine. Une fois la carte globale créée, un autre type de robot Turtlebot3 (Waffle) utilise cette carte pour se déplacer d'un endroit à un autre dans l'usine pour diverses opérations telles que la livraison de matériel, la surveillance ou le nettoyage.

6.2.2 TurtleBot 3 : une plateforme mobile complète

TurtleBot3 est un petit robot mobile programmable, abordable, basé sur ROS, destiné à être utilisé dans l'enseignement, la recherche, les loisirs et le prototypage de produits. L'objectif du TurtleBot3 est de réduire considérablement la taille de la plateforme et d'en abaisser le prix sans avoir à sacrifier ses fonctionnalités et sa qualité, tout en offrant une extensibilité. Le TurtleBot3 peut être personnalisé de diverses manières, selon la façon dont on reconstruit les pièces mécaniques et dont on utilise les pièces optionnelles telles que l'ordinateur et le capteur. En outre, le TurtleBot3 est doté d'une carte mère SBC rentable et de petite taille, adaptée à un système embarqué robuste, d'un capteur de distance à 360 degrés et de la technologie d'impression 3D. Il existe deux version du Turtlebot3 : le Turtlebot3 burger et le Turtlebot3 Waffle [137].

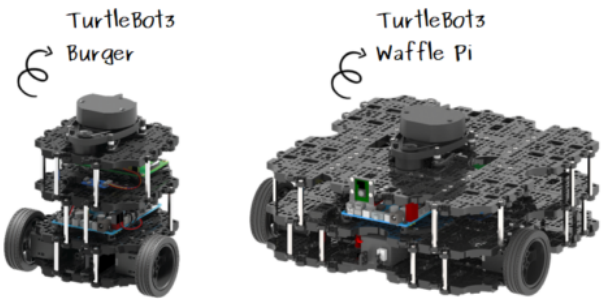


Figure 6.1 – Les deux versions de Turtlebot3 : Burger et Waffle [137]

Les spécifications de chaque version du Turtlebot3 sont listées dans ce tableau.

Items	Burger	Waffle
Maximum translational velocity	0.22 m/s	0.26 m/s
Maximum rotational velocity	2.84 rad/s (162.72 dcg/s)	1.82 rad/s (104.27 dcg/s)
Maximum payload	15kg	30kg
Size (L x W x H)	138mm x 178mm x 192mm	281mm x 306mm x 141mm
Weight (+ SBC + Battery + Sensors)	1kg	1.8kg
Threshold of climbing	10 mm or lower	
Expected operating time	2h 30m	2h

Figure 6.2 – Les spécifications de deux versions de Turtlebot3 [137]

6.2.3 Simultaneous localization and mapping (SLAM)

- Définition

Selon Lajoie et al [77], la Localisation et cartographie simultanées, en anglais Simultaneous localization and mapping (SLAM) est une estimation conjointe de l'état d'un robot et d'un modèle de son environnement, avec l'hypothèse clé qu'un robot en mouvement effectue la collecte de données de manière séquentielle. L'état du robot comprend d'une part sa pose (position et orientation) et éventuellement d'autres quantités telles que les paramètres d'étalonnage des capteurs, et d'autre part, le modèle d'environnement (c'est-à-dire la carte) est constitué de représentations de points de repère, construites à partir des données traitées par les capteurs extéroceptifs du robot, tels que les caméras ou les lidars.

- SLAM avec un seul robot

En général, les systèmes SLAM sont divisés en deux parties, le front-end et le back-end, chacune impliquant différents domaines de recherche. La partie frontale est responsable des tâches liées à la perception, telles que l'association de données et l'extraction de caractéristiques. Le back-end génère des estimations de l'état final en utilisant les résultats du front-end. Le back-end utilise des outils issus des domaines de l'optimisation, de la théorie des graphes et de la théorie des probabilités. En fait, le frontal traite les données des capteurs pour générer des mesures d'ego-motion, de fermeture de boucle et de points de repère, tandis que le back-end effectue une estimation conjointe de la carte et de l'état du robot. La figure 6.3 montre un aperçu d'une structure SLAM commune. En effet, la trajectoire du robot est représentée comme un graphe de poses à des moments consécutifs discrets (c'est-à-dire un graphe de poses) et la carte comme un ensemble de points de repère observés [28].

- SLAM collaboratif (C-SLAM)

Plusieurs chercheurs ont étudié la manière d'utiliser plusieurs agents pour effectuer le SLAM : on parle alors de SLAM collaboratif ou distribué. Le SLAM distribué augmente la

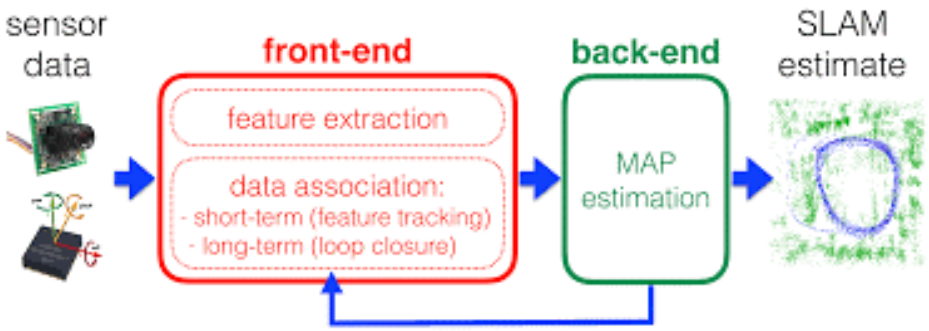


Figure 6.3 – Vue d'ensemble du SLAM mono-robot [29]

robustesse du processus SLAM, le rendant moins sensible aux défaillances catastrophiques. Les défis du SLAM distribué sont le calcul des chevauchements de cartes et le partage d'informations entre des agents dont la bande passante de communication est limitée [44]. L'utilisation de plusieurs robots au lieu d'un seul permet d'effectuer de nombreuses tâches plus rapidement et plus efficacement. C'est le cas du SLAM collaboratif (C-SLAM) avec des robots mobiles autonomes. En effet, il est avantageux et parfois nécessaire d'améliorer les solutions SLAM en algorithmes C-SLAM coordonnés plutôt que d'effectuer un SLAM à robot unique sur chaque robot [43]. Les algorithmes C-SLAM se concentrent sur la fusion des données recueillies sur chaque robot individuel en des estimations globalement cohérentes d'une carte commune et de l'état de chaque robot. Cette coordination permet à chaque robot de bénéficier de l'expérience de l'ensemble de l'équipe, ce qui conduit à une localisation et une cartographie plus précises que les multiples instances de SLAM à un seul robot. Cependant, cette coordination introduit de nombreuses nouvelles caractéristiques et de nouveaux défis inhérents aux systèmes multi-robots.

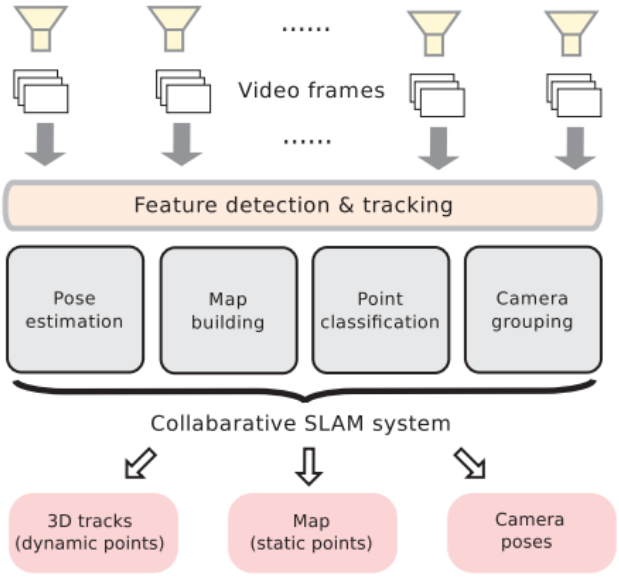


Figure 6.4 – Vue d'ensemble du SLAM collaboratif [43]

6.2.4 Application de la méthodologie IDMSR

Dans cette section, nous appliquons notre méthodologie de conception IDMSR developpé dans le chapitre 3. Dans ce qui suit, les différentes phases de la méthodologie IDMSR sont appliquées.

• PHASE 1 : Modélisation avec la méthode MBSE

- Exigences de conception de l'essaim :

Comme indiqué dans notre méthodologie, la spécification des exigences est la première étape, le diagramme d'exigences SysML présenté dans la Figure 6.5 a été utilisé pour décrire cette mission. La mission principale du système est l'utilisation de la technologie Multi-SLAM pour dessiner la carte globale de l'usine afin qu'un autre robot puisse ensuite l'utiliser pour naviguer dans l'usine. Cette mission nécessite deux sous-exigences principales : l'une est liée à l'exigence matérielle et l'autre est liée à l'ensemble des logiciels qui fournissent le Multi-SLAM et la navigation.

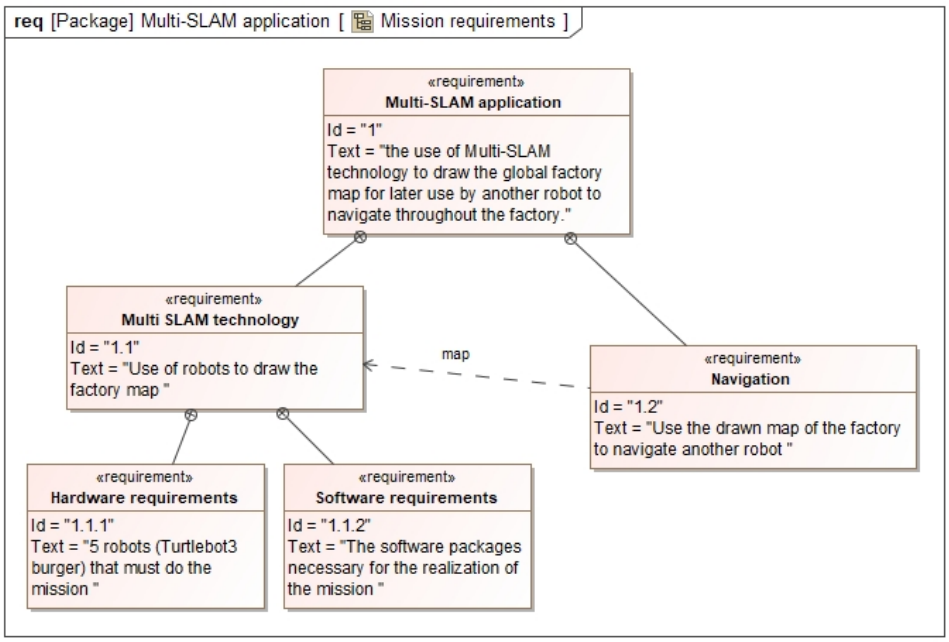


Figure 6.5 – La mission Multi-SLAM avec le diagramme des exigences SysML

- Modélisation du comportement de l'essaim :

Dans cette étape, le diagramme de comportement individuel du robot IRBD est utilisé pour décrire le comportement SLAM de chaque robot de l'essaim. Chaque robot commence à se déplacer dans une zone de l'industrie ; il a été contrôlé par l'utilisateur pour gérer sa vitesse et sa direction linéaire et angulaire. Une fois que le robot se déplace, il active ses capteurs et envoie simultanément des messages qui informent sur sa position, son orientation

et la distance mesurée avec le LDS. Une carte est immédiatement dessinée et enregistrée dans le maître ROS. La Figure 6.6 explique cette application SLAM avec un seul robot.

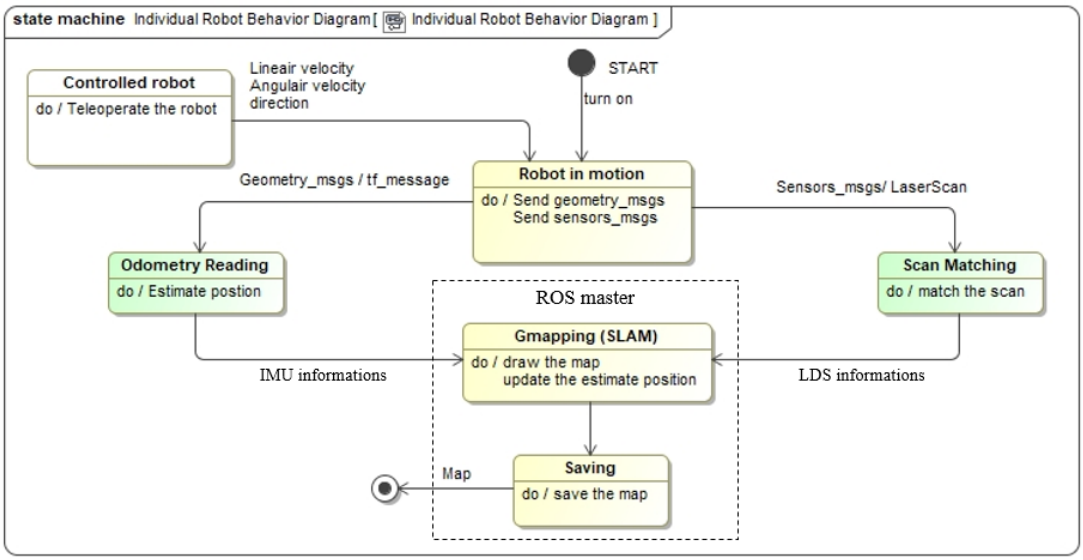


Figure 6.6 – Diagramme de comportement individuel du robot IRBD pour l'application SLAM

Dans cette étape, nous utilisons le diagramme de comportement des essaims collectifs CSBD pour modéliser le SLAM collaboratif (C-SLAM) de l'essaim de 5 Turtlebots. Au début, les 5 Turtlebots Burger sont chargés dans l'usine, chaque robot commence à faire la technique SLAM définie ci-dessus. Une fois que chaque robot a dessiné sa carte, le programme ROS fusionne ces cartes en une carte globale de l'usine et la sauvegarde. Ensuite, un autre robot Turtlebot Waffle utilise cette carte sauvegardée pour naviguer dans toute l'industrie en évitant tous les obstacles et les murs. La Figure 6.7 explique l'application de C-SLAM avec 5 Turtlebots burger et la navigation avec Waffle.

- Modélisation de l'architecture de l'essaim :

Notre système est composé de 5 Turtlebots Berger qui forment le C-SLAM et d'un Turtlebot Waffle qui effectue la navigation dans l'usine. Le diagramme de la figure 6.8 représente les composants de notre système et les positions des robots.

Pour faciliter l'implémentation du système en essaim sur l'environnement ROS, nous utilisons le diagramme de mission de l'essaim développé dans notre méthodologie pour définir les différents nœuds et sujets que nous utiliserons par la suite sur ROS. Nous utilisons pour chaque turtlebot de l'essaim des nœuds nommés /turtlebot3_teleop pour téléopérer le robot, ces nœuds publient les informations de vitesse, direction, position dans les topics /cmd_vel. Nous définissons également des noeuds nommés /turtlebot3_slam_gmapping qui sont responsables de la réalisation de la technologie SLAM, ces noeuds traitent toutes les données reçues par les capteurs "Laser_scan", et les données du robot "tf" (odom, base_footprint, base link, ...) et publient les données de la carte de chaque robot dans un topic noté /tb_map. Une fois que toutes les cartes dessinées par chaque robot sont prêtes, un noeud nommé /map_server fusionne toutes les cartes en une seule carte globale publiée dans le topic /map.

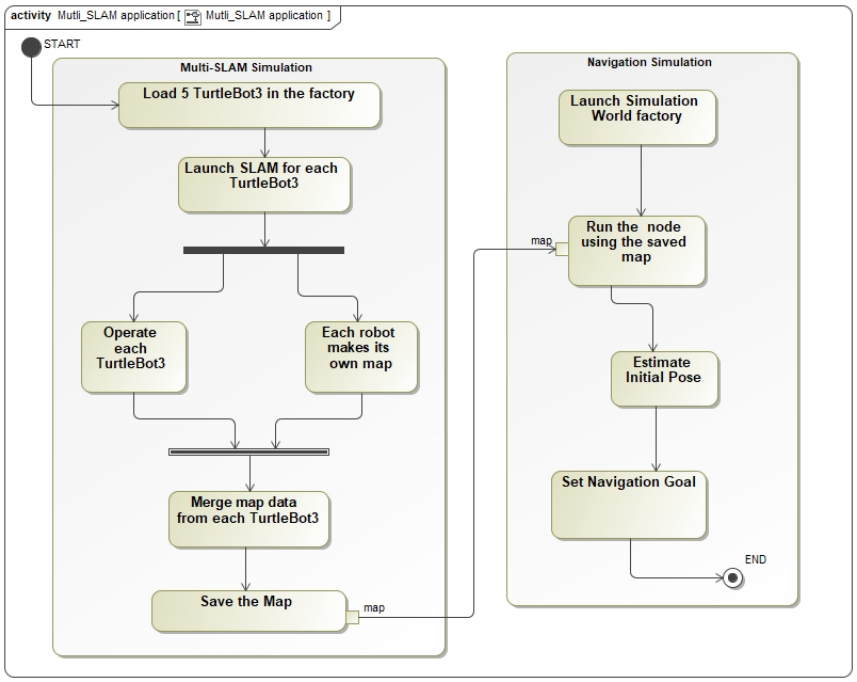


Figure 6.7 – Diagramme de comportement collectif de l'essaim CSBD pour C-SLAM

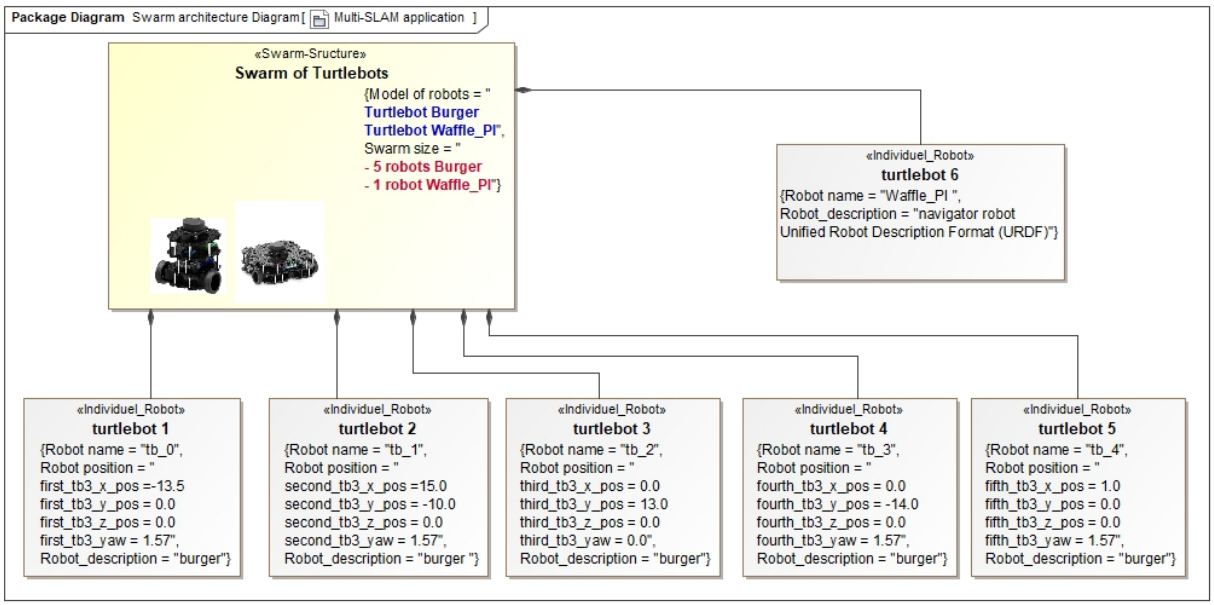


Figure 6.8 – Architecture générale de l'essaim utilisant le langage dédié DSL

Le robot Waffle s'abonne alors au topic /map pour utiliser la carte globale afin de naviguer dans l'usine en évitant tous les obstacles et en se dirigeant vers la destination proposée par l'utilisateur. Le noeud /move contient le code qui décrit la méthode de navigation du robot en utilisant la carte enregistrée. Et enfin, le nœud /gazebo est créé pour l'affichage sur l'interface graphique GAZEBO. La Figure 6.9 représente le diagramme de la mission Swarm qui décrit les différentes étapes de notre application.

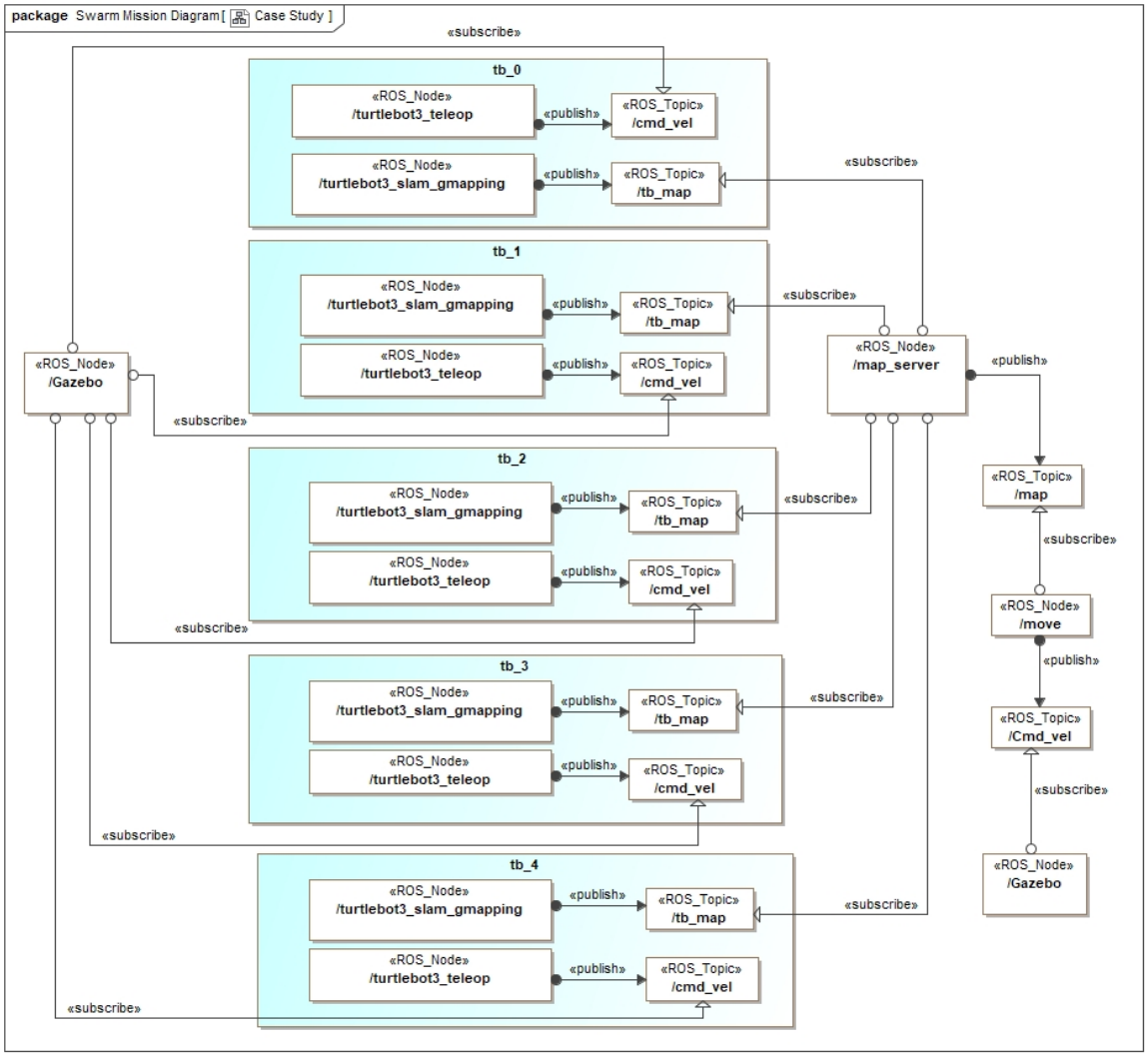


Figure 6.9 – Diagramme de mission de l'essaim SMD pour l'application C-SLAM

• **PHASE 2 : Implémentation à l'aide de ROS/ROS2**

- Implémentation sur ROS :

Dans cette phase, les modèles du système sont développés. C'est le rôle du développeur de logiciels maintenant de développer les codes nécessaires et d'adapter les codes existants à son application. Pour notre application, nous avons développé des codes python / C++, nous avons également utilisé des codes existants sur GitHub et des packages de Turtlebot3 sur le site "ROBOTIS e-Manual" développé par la société ROBOTIS pour implémenter l'application C-SLAM sur ROS. La première étape consiste à créer le modèle descriptif URDF des deux modèles Turtlebots (burger et waffle) à des fins de simulation avec l'outil Gazebo. Le fichier URDF peut être directement défini à l'aide des outils Gazebo sur la base de la description SysML de la structure du Turtlebot. Pour notre part, nous avons utilisé les fichiers URDF développés par la société ROBOTIS et nous les avons intégrés dans notre espace de travail ROS. La Figure 6.10 montre les deux modèles de Turtlebot simulés sur Gazebo.

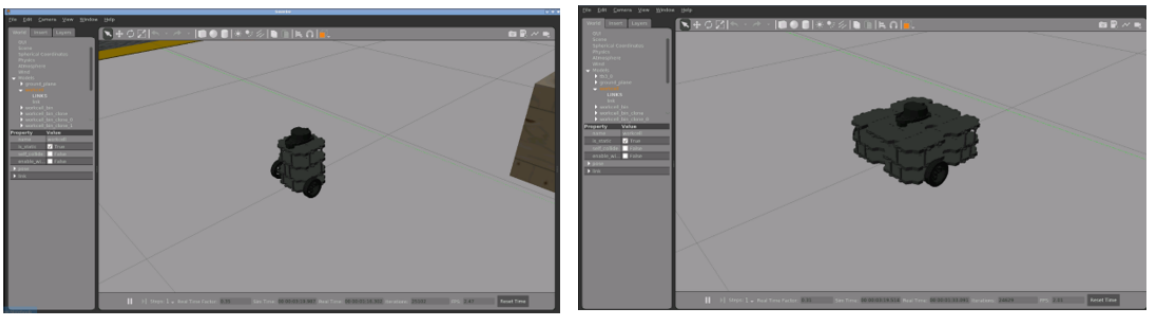


Figure 6.10 – Modèles Turtlebot simulés sur Gazebo

La deuxième étape consiste à créer un fichier World qui décrit l'environnement de travail. Les utilisateurs de ROS peuvent réutiliser et adapter des fichiers World existants, ou les développer à l'aide de Gazebo ou d'un logiciel de CAO 3D. Dans notre étude, l'environnement de travail est une usine qui a été adoptée dans la bibliothèque de fichiers World de ROS/Gazebo. Le fichier "factory" contient les différents éléments de l'usine, comme le montre la Figure 6.11.

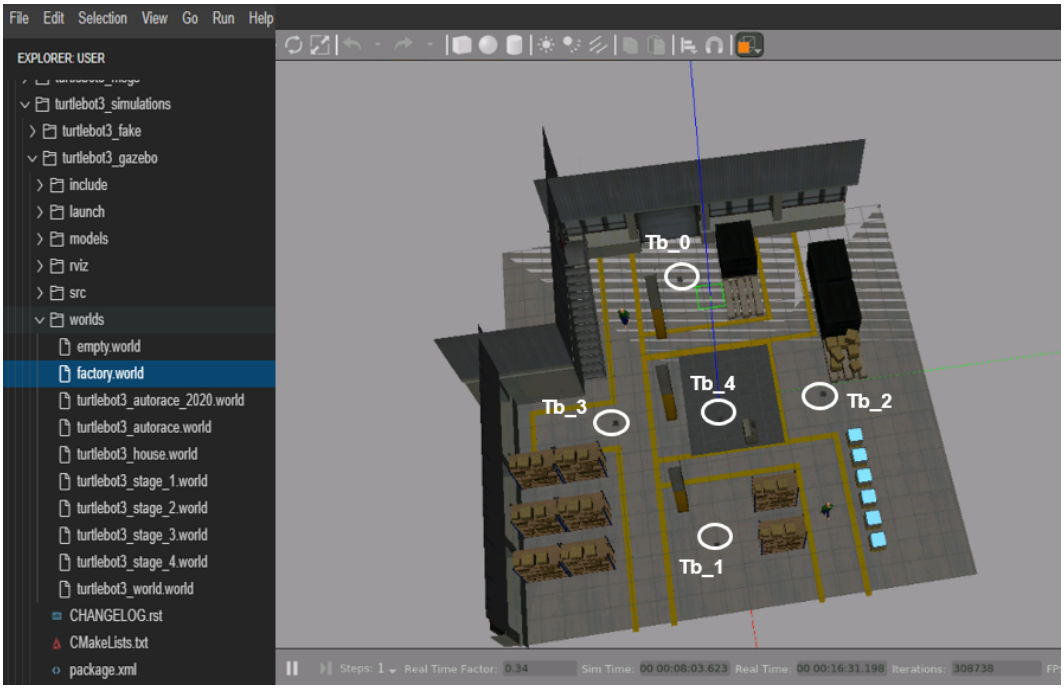


Figure 6.11 – Modèle 3D d'une usine dans Gazebo.

Une fois que les fichiers URDF qui définissent la structure des robots et le fichier world qui définit la fabrique du travail sont créés et implémentés sur ROS, nous démarrons notre application, chaque burger turtlebot de l'essaim est situé dans une zone bien définie, il commence à se déplacer et à créer sa carte locale simultanément. Lorsque les cartes locales sont prêtes, le nœud /map_server fusionne les cartes en une seule carte globale. La Figure 6.12 montre la carte globale fusionnée de l'usine.

La navigation consiste à déplacer le robot d'un endroit à une destination spécifiée dans

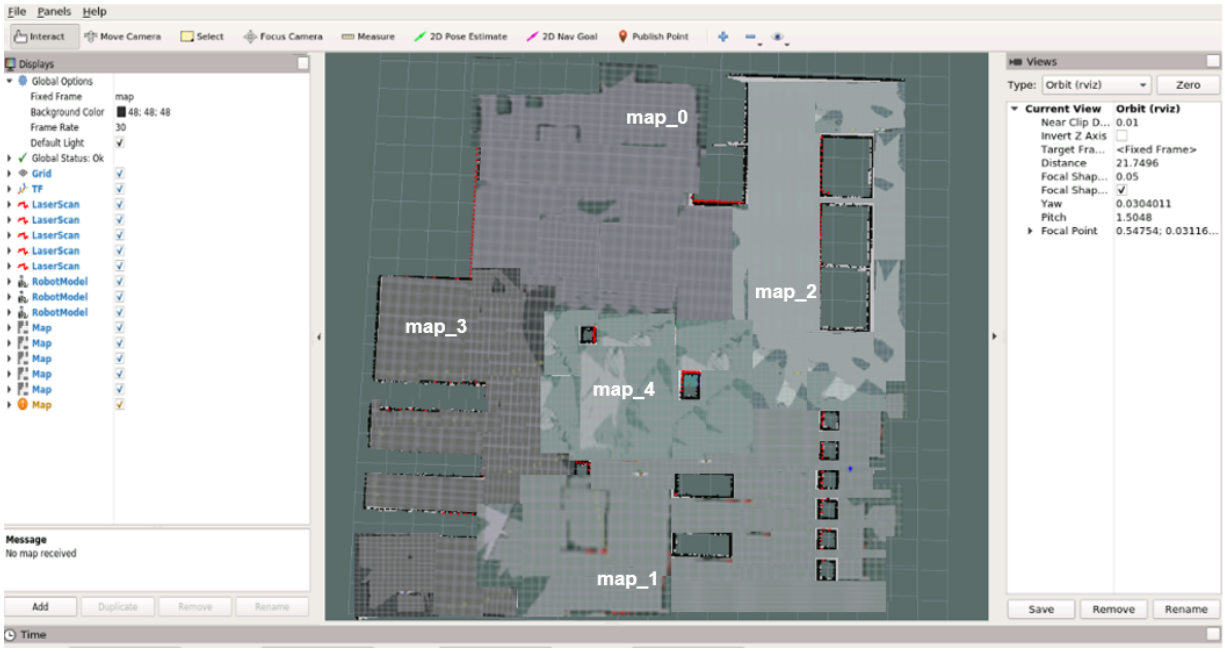


Figure 6.12 – Carte globale de l'usine sur RVIZ.

un environnement donné. Pour ce but, une carte qui contient les informations géométriques des objets Comme modélisé dans notre application, la carte a été créée avec les informations de distance obtenues par le capteur et les informations de pose du robot lui-même. Ainsi, la carte globale de l'usine a été créée et sauvegardée dans notre espace de travail sous forme de fichier pgm, le Waffle utilise maintenant cette carte pour naviguer dans l'usine. La Figure 6.13 montre la navigation du Waffle dans l'usine à l'aide de la carte globale sur RVIZ.

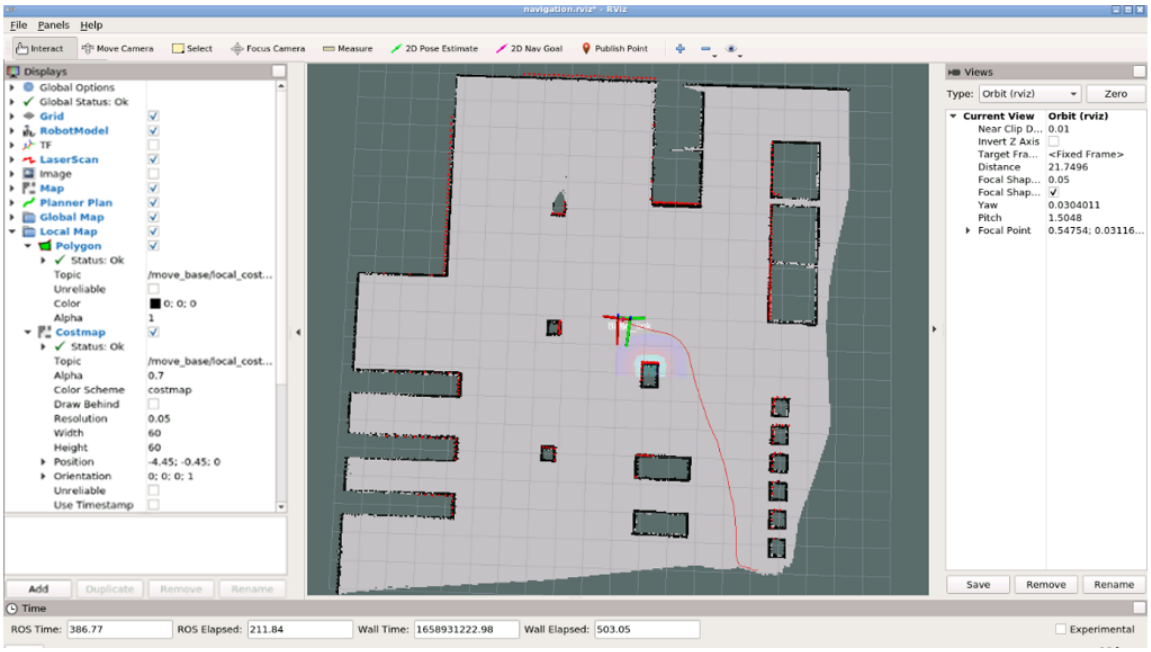


Figure 6.13 – Navigation du Waffle à l'aide de la carte globale sur RVIZ.

- Vérification / Validation de l'essaim :

Parmi les points forts de notre méthodologie de conception, on a la vérification visuelle de notre système en essaim avec les différents outils de visualisation (Gazebo, RVIZ), nous pouvons vérifier et valider la cohérence des modèles développés avec la version simulée obtenue en comparant le Swarm Mission Diagram de SwarmML illustré sur la Figure 6.9 avec les composants de rqt_graph générés par ROS illustrée sur la Figure 6.14.

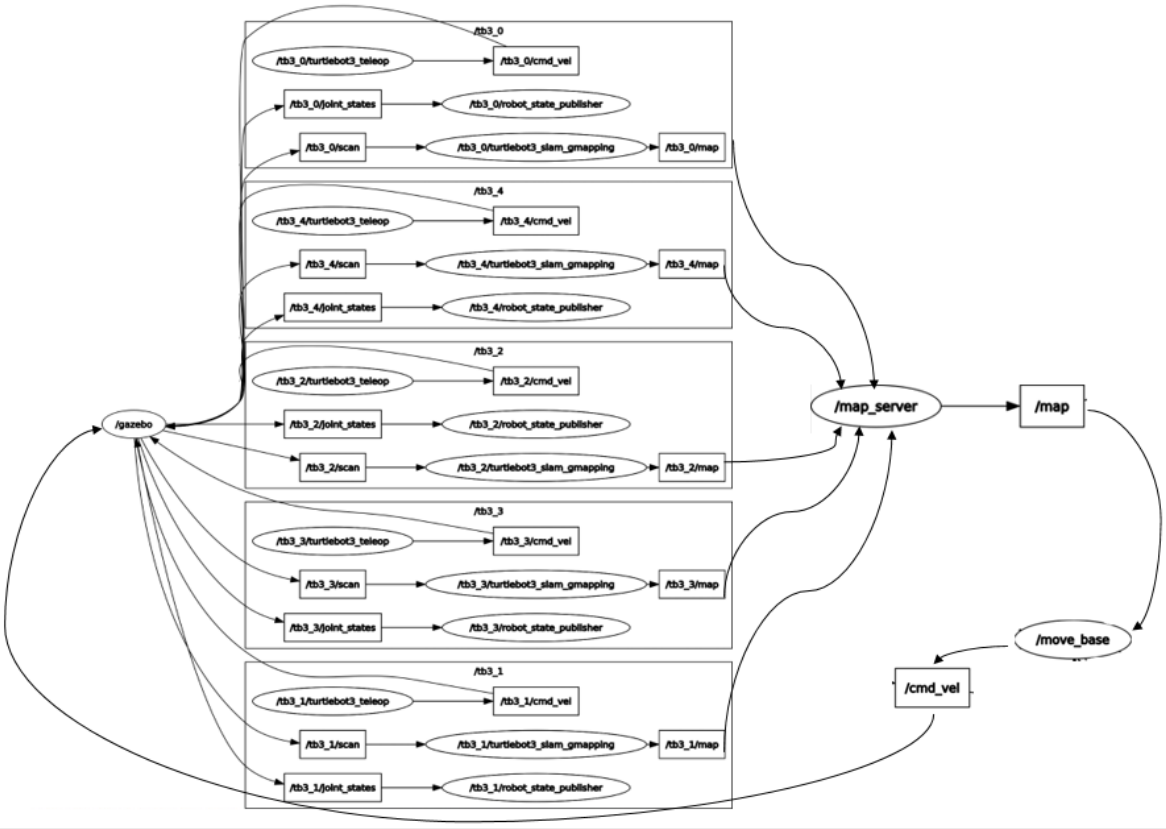


Figure 6.14 – rqt_graph généré par ROS.

A partir de cette figure, nous pouvons remarquer qu'il y a une cohérence entre les composants modélisés avec SMD (nœuds, topics) et les composants du rqt_graph généré par ROS ce qui prouve la réussite de notre version simulée du système.

- Prototypage des essaims réels :

C'est la dernière étape de notre application. Dans cette étude de cas industrielle, nous n'avons pas implémenté notre système dans une usine mais nous avons crée un prototype multi-SLAM avec 3 Burger Turtlebots dans notre laboratoire. Cette étape nécessite beaucoup de préparations matérielles comme l'assemblage des Turtlebots3 puisque les Turtlebots3 sont livrés en pièces détachées dans des boîtes, et des préparations logicielles comme la configuration du réseau pour assurer un protocole de communication entre les robots.

a - Assemblage de Turtlebot3 burger : Cette étape est très importante et un peu difficile, il faut suivre des instructions pour assembler le TurtleBot3. Pour cela, la société qui a construit

le Turtlebot3, appelée ROBOTICS, a créé un manuel d'assemblage pour faciliter cette tâche. Dans ce manuel[60], il y a différentes instructions détaillées, étape par étape, pour expliquer la méthode d'assemblage. La Figure 6.15 montre les étapes de l'assemblage du Turtlebot3 burger.

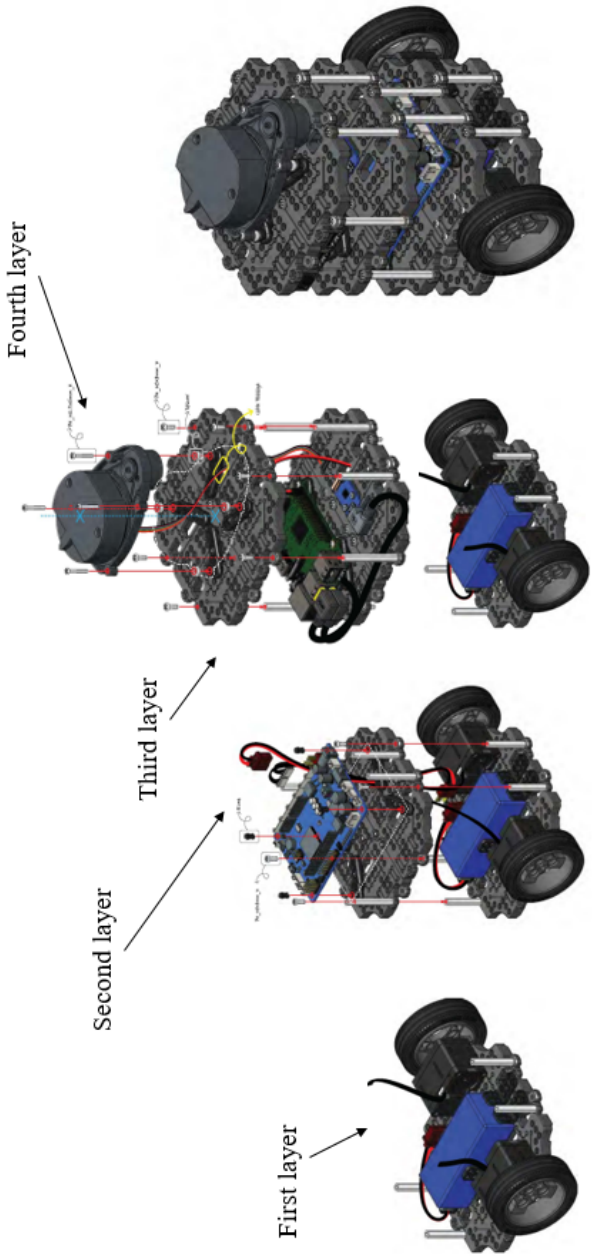


Figure 6.15 – Les étapes d'assemblage du Turtlebot3 burger[60]

b - Installation des logiciels et programmes : Dans cette étape, nous avons d'abord installé Ubunto20.04 sur notre machine pour faciliter la tâche d'installation de ROS puisque ROS n'est fonctionnel que sur le système d'exploitation Linux. Une fois qu'on a Ubunto sur notre machine, on a installé ROS Kinetic et toutes les packages de Turtlebots3 l'aide des instructions illustrées sur le site de ROBOTICS e-Manual [61].

c - Configuration du réseau : C'est l'étape la plus importante de notre application. Nous avons créé un réseau local pour que les robots communiquent entre eux et avec le ROS master grâce à leurs adresses IP. Il s'agit du protocole de communication TCP/IP. Ce protocole permet aux robots d'envoyer et de recevoir des données. L'acronyme TCP/IP signifie Transmission Control Protocol/Internet Protocol. Il permet aux appareils connectés à Internet de communiquer entre eux sur des réseaux. Pour connecter le Turtlebot à un réseau Wi-Fi pour la première fois, on a connecté à ce robot un écran en HMDI, un clavier et une souris en USB pour récupérer l'adresse IP de la Raspberry pi. Il suffit ensuite de la connecter au même réseau. Par la suite le robot se connectera automatiquement au réseau Wi-Fi qu'il a enregistré. La Figure 6.16 décrit le protocole de communication TCP/IP.

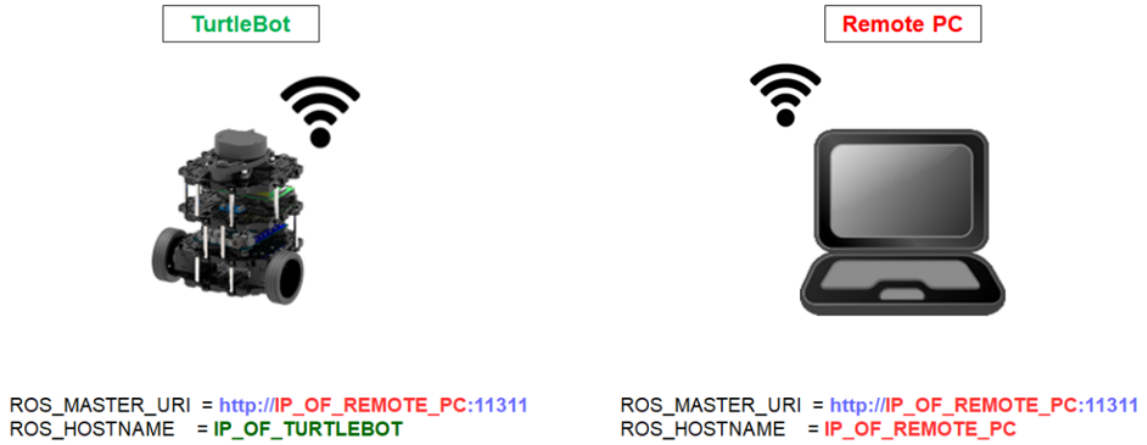


Figure 6.16 – Protocole de communication TCP/IP.

d- Application de SLAM avec un seul Turtlebot3 : Nous avons appliqué la technologie SLAM dans notre laboratoire qui consistait en deux pièces de 69 m2 de surface avec un seul robot turtlebot burger dont la vitesse moyenne est égale à 0.12m/s. Nous avons constaté que le temps nécessaire pour manipuler ce robot et créer la cartographie du laboratoire est de 20 minutes. La Figure 6.17 représente la carte crée du laboratoire.

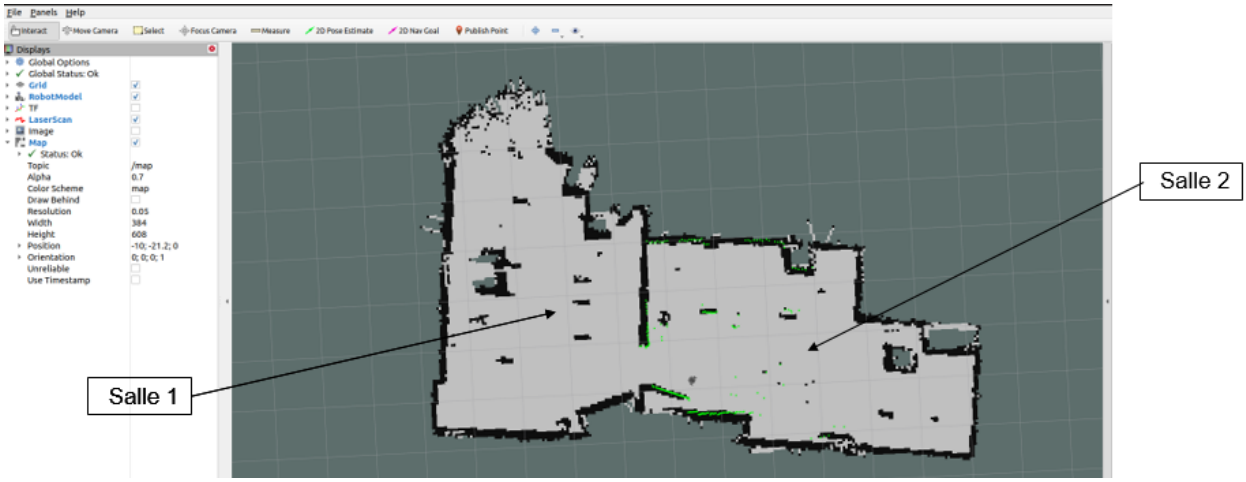


Figure 6.17 – Carte crée du laboratoire avec la technologie SLAM.

e- Application de Multi-SLAM avec un 3 Turtlebot burger : Dans cette étape, nous avons appliqué la technologie Multi-SLAM avec 3 robots Burger sur la même surface du laboratoire pour voir le temps nécessaire à la création de cette carte. Nous avons constaté que le temps est réduit à seulement 6 minutes. Ceci définit l'avantage d'utiliser le Multi-SLAM avec un essaim de robots Turtlebots au lieu du simple SLAM avec un seul robot. La Figure 6.18 représente cette carte crée.

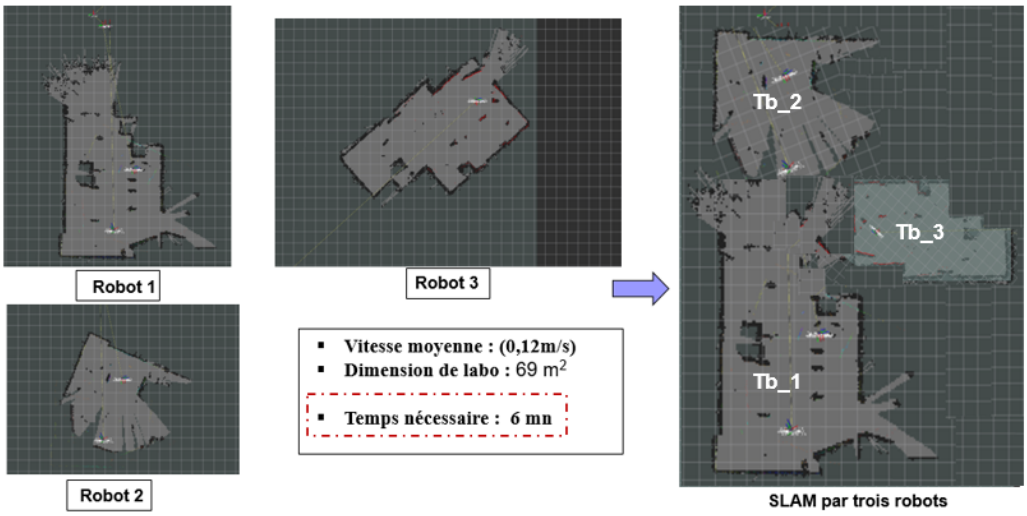


Figure 6.18 – Carte crée du laboratoire avec Multi-SLAM

Après plusieurs essais, le 3ème robot perd sa connexion au réseau local pendant la mission, mais la mission est toujours effectuée en ignorant le robot défectueux ou hors réseau, ce qui confirme la redondance de notre technologie Multi-SLAM. Les résultats sont illustrées sur la Figure 6.19.

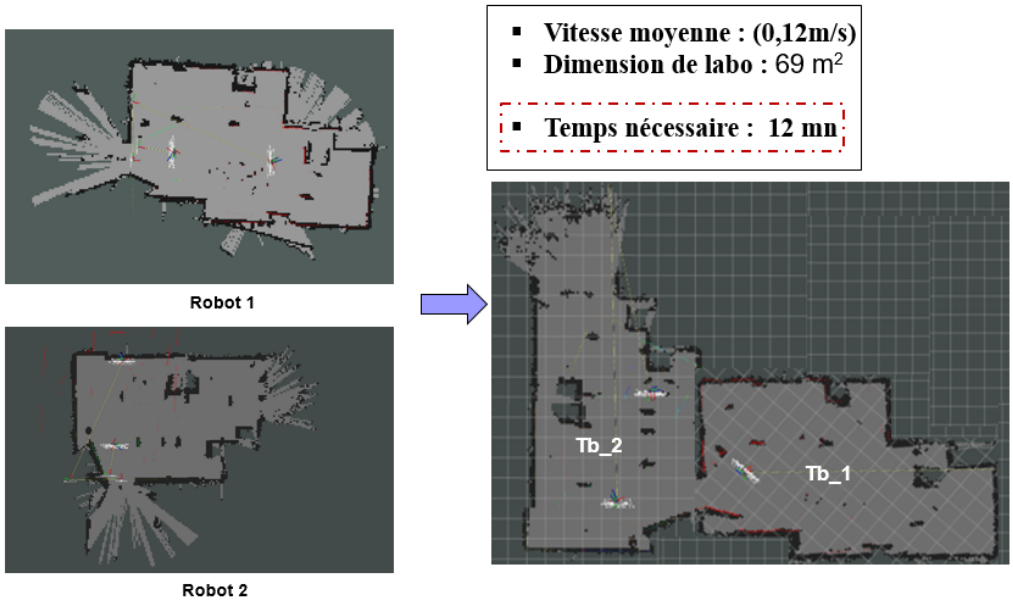


Figure 6.19 – Carte crée du laboratoire par 2 robots

Dans ce tableau, nous avons résumé le temps nécessaire pour chaque mission.

	Surface à parcourir	Vitesse moyenne de robot	Temps nécessaire de la mission
SLAM avec 1 robot	69 m²	0,12m/s	**20 min**
Multi-SLAM avec 3 robot	69 m²	0,12m/s	**6 min**
Multi-SLAM avec 2 robot	69 m²	0,12m/s	**12 min**

Figure 6.20 – Tableau récapitulatif des résultats obtenus

Nous avons constaté que le temps de mission pour un seul robot est supérieur à celui de 3 robots et de 2 robots, ce qui est logique puisque le SLAM avec un seul robot nécessite plus de temps qu'une mission avec plusieurs robots dans laquelle chaque robot dessine une carte de la zone où se situer. C'est la première caractéristique du Multi-SLAM : l'évolutivité. En effet, la modification de la taille de l'essaim ne nécessite pas de reprogrammer les robots individuels et n'a pas d'impact majeur sur le comportement collectif qualitatif. Cela permet aux essaims de robots d'atteindre l'évolutivité, c'est-à-dire de conserver leurs performances lorsque de nouveaux agents rejoignent le système, car ils peuvent s'adapter à n'importe quelle taille d'environnement, dans une plage raisonnablement large. De plus, la méthode Multi-SLAM peut atteindre la tolérance aux pannes car l'essaim peut faire face à la perte ou à la défaillance de certains robots, ce qui explique la redondance de la technologie Multi-SLAM, comme dans notre cas, lorsque le 3ème robot perd sa connexion au réseau local et que la mission se poursuit malgré tout en ignorant le robot défaillant ou hors réseau.

6.3 Conclusion

Dans ce chapitre, nous avons traité une étude de cas de C-SLAM avec des robots Turtlebots dans un environnement industriel. Dans cette étude de cas, nous avons illustré la méthode descendante basée sur les diagrammes SysML/SwarmML pour modéliser le système en essaim. La modélisation du système complexe est simplifiée car la méthode regroupe les trois vues des représentations du système (vue comportementale, vue fonctionnelle et vue structurelle) au sein d'un même modèle. De plus, avec la méthodologie ascendante basée sur l'environnement ROS/ROS2, il nous a été facile d'intégrer les codes et les environnements de simulation existants pour le système selon les spécifications faites dans le modèle SysML/SwarmML pour les adapter à la conception spécifique du système en essaim. En effet, les informations contenues dans le modèle SysML/SwarmML aident le concepteur à mettre en œuvre le modèle de simulation 3D dans ROS/ROS2, l'environnement de simulation, les attributs du système et les codes qui doivent respecter les algorithmes spécifiés dans SysML pour la description du comportement. Pour ce faire, les informations contenues dans les différents diagrammes développés (IRBD, CSBD, SMD, etc.) sont transformées en fichiers ROS/ROS2 (fichiers XML, fichiers URDF, codes, etc.). La validation de la conception

du système en essaim par simulation nécessite de définir les cas de test de simulation en fonction du diagramme d'exigences SysML.

Dans ce chapitre, nous avons également réalisé des expériences pratiques de la technologie Multi-SLAM en utilisant les robots Turtlebot3 burger et le Midelwere ROS. Grâce à ces expériences, nous avons testé les performances de notre prototype réalisé, telles que l'évolutivité et la redondance de notre système.

Conclusion générale et perspectives

Le concept de robots en essaim est une exigence très importante aujourd'hui de l'industrie 4.0 en raison de sa robustesse, de sa flexibilité et de son évolutivité. Les robots en essaim ont de nombreux comportements collectifs de base pour faire face aux applications complexes du monde réel, comme la recherche de nourriture ou la construction. En effet, nous avons classé ces comportements en quatre catégories : comportements de navigation, comportements d'organisation spatiale, prise de décision collective et autres comportements collectifs. Pendant longtemps, aucune approche structurée n'a été développée pour concevoir le système de robot en essaim.

Notre thèse aborde le problème scientifique de l'absence d'une approche bien fondée pour concevoir un système de robot en essaim. Une nouvelle méthodologie pour la conception de robots en essaim a été développée. Cette méthodologie est basée sur la spécification des exigences du système en essaim et la modélisation des fonctions et des comportements à l'aide de différents diagrammes SysML. Ensuite, le développeur simule les modèles développés sur un outil multi-agent pour les valider et enfin implémente les modèles simulés sur des robots réels. Nous avons exploité la diversité des outils de modélisation (MBSE, SysML, DSL, ...) et de simulation (multi-agent, ROS / ROS2 ...) pour créer cette méthodologie structurée qui facilite et approche les processus de conception et de fabrication du système robotique en essaim.

Tout au long de ce rapport, nous avons utilisé les diagrammes SysML/ SwarmML pour faciliter la modélisation de systèmes complexes tels que les systèmes de robots en essaim. En effet, le langage SysML avec personnalisation DSL simplifie la modélisation de l'essaim de robots car il regroupe les trois vues des représentations du système (vue fonctionnelle, vue structurelle et vue comportementale) au sein d'un même modèle. Cela garantit la cohérence des données car les règles de SysML donnent à chaque élément du modèle une définition unique, construite en rassemblant les informations de ses différentes représentations, et les empêchent de se contredire. Cela réduit à la fois le risque d'erreurs et le temps passé à vérifier les données par rapport aux approches documentaires qui ne disposent pas d'une telle protection. De plus, la modélisation avec SysML facilite l'utilisation intensive de la simulation car un modèle SysML ou un modèle SwarmML peut rassembler toutes les informations pour modéliser le système, simuler son comportement et comparer les résultats avec les exigences pour valider (ou non) les solutions. Cependant, la modélisation avec SysML présente également certaines limites. Par exemple, lorsqu'un développeur crée un diagramme SysML pendant le développement d'un programme, le diagramme peut devenir confus ou très complexe. En outre, le passage du niveau du modèle graphique au niveau de l'implémentation du logiciel reste une tâche difficile. Il dépend de l'expérience et de la créativité du développeur.

Dans les études des cas traitées, nous avons illustré la méthode descendante basée sur les diagrammes SysML/SwarmML pour modéliser le système en essaim. La modélisation du système complexe est simplifiée car la méthode regroupe les trois vues des représentations du système (vue comportementale, vue fonctionnelle et vue structurelle) au sein d'un même modèle. De plus, avec la méthodologie ascendante basée sur l'environnement ROS/ROS2,

il nous a été facile d'intégrer les codes et les environnements de simulation existants pour le système selon les spécifications faites dans le modèle SysML/SwarmML pour les adapter à la conception spécifique du système en essaim. En effet, les informations contenues dans le modèle SysML/SwarmML aident le concepteur à mettre en œuvre le modèle de simulation 3D dans ROS/ROS2, l'environnement de simulation, les attributs du système et les codes qui doivent respecter les algorithmes spécifiés dans SysML pour la description du comportement. Pour ce faire, les informations contenues dans les différents diagrammes développés (IRBD, CSBD, SMD, etc.) sont transformées en fichiers ROS/ROS2 (fichiers XML, fichiers URDF, codes, etc.). La validation de la conception du système en essaim par simulation nécessite de définir les cas de test de simulation en fonction du diagramme d'exigences SysML.

Dans cette thèse, nous avons également réalisé des expériences pratiques de la technologie Multi-SLAM en utilisant les robots Turtlebot3 burger et le Midelwere ROS. Grâce à ces expériences, nous avons testé les performances de notre prototype réalisé, telles que l'évolutivité et la redondance de notre système.

Comme perspectives et travaux futurs, nous proposerons d'autres applications de cette méthodologie dans d'autres environnements avec d'autres types de robots pour s'assurer de l'efficacité de la méthodologie. Nous réfléchirons également à d'autres solutions pour assurer la continuité de la méthodologie comme l'utilisation d'outils de modélisation et de simulation qui génèrent des données directement exploitables d'une étape à l'autre tels que MATLAB Simulink et ROS2.

LISTE DES PUBLICATIONS LIÉES À LA THÈSE

- Articles de revue :

- Aloui Khalil, Guizani Amir, Hammadi Moncef, Soriano Thierry, Haddar Mohamed. (2021). Integrated design methodology of automated guided vehicles based on swarm robotics. Applied Sciences, 11(13), 6187.

- Aloui Khalil, Guizani Amir, Hammadi Moncef, Soriano Thierry, Haddar Mohamed. (2022) An integrated design methodology for swarm robotic systems development using Model-Based Systems Engineering and Robot Operating System, Research in Engineering Design (en cours de révision).

- Aloui Khalil, Guizani Amir, Hammadi Moncef, Haddar Mohamed, Soriano Thierry .(2022) A New SysML Profile for Autonomous Mobile Robots Development : ROS2ML, Journal of Mechanical Engineering Science. (en cours de révision)

- Articles des conférences :

- Aloui Khalil, Hammadi Moncef, Guizani Amir, Soriano Thierry, Haddar Mohamed. A new SysML Model for UAV Swarm Modeling : UavSwarmML. In : 2022 IEEE International Systems Conference (SysCon). IEEE, 2022. p. 1-8.

- Aloui Khalil, Guizani Amir, Hammadi Moncef, Haddar Mohamed, Soriano Thierry. A Top Down Approach to Ensure the Continuity of the Different Design Levels of Swarm Robots. In : 2021 18th International Multi-Conference on Systems, Signals and Devices (SSD). IEEE, 2021. p. 1438-1445.

- Aloui Khalil, Guizani Amir, Hammadi Moncef, Soriano Thierry, Haddar Mohamed. System level specification and multi-agent simulation of manufacturing systems. In : International Conference on advances in Materials, Mechanics and Manufacturing. Springer, Cham, 2021. p. 30-40.

- Aloui Khalil, Hammadi Moncef, Guizani Amir, Soriano Thierry, Haddar Mohamed. Development of an AGV System Using MBSE Method and Multi-agents' Technology. In : International Conference Design and Modeling of Mechanical Systems. Springer, Cham, 2023. p. 103-114.

- Aloui Khalil, Hammadi Moncef, Guizani Amir, Soriano Thierry, Haddar Mohamed. Modeling and Simulation of A Swarm Robot Application Using MBSE Method and Multi-Agent Technology : Monitoring Oil Spills. In : International Workshop on Modelling and Simulation of Complex Systems for Sustainable Energy Efficiency. Springer, Cham, 2021. p. 96-105.

- Aloui Khalil, Hammadi Moncef, Guizani Amir, Soriano Thierry, Haddar Mohamed. On the continuity of the swarm robot design using MBSE method and simulation. In : 13ème CONFERENCE INTERNATIONALE DE MODELISATION, OPTIMISATION ET SIMULATION (MOSIM2020), 12-14 Nov 2020, AGADIR, Maroc. 2020.

Bibliographie

[1] Mehdi Abedi et Farhad Soleimanian Gharehchopogh. “An improved opposition based learning firefly algorithm with dragonfly algorithm for solving continuous optimization problems”. In : *Intelligent Data Analysis* 24.2 (2020), p. 309-338.

[2] Bryan Adams et al. “Humanoid robots: A new kind of tool”. In : *IEEE Intelligent Systems and Their Applications* 15.4 (2000), p. 25-31.

[3] Gerkey B. et al. “Stage/2D multiple-robot simulator...", URL: http://playerstage.sourceforge.net”. In.

[4] Javier Alonso-Mora et al. “Multi-robot system for artistic pattern formation”. In : *2011 IEEE international conference on robotics and automation*. IEEE. 2011, p. 4512-4517.

[5] Khalil Aloui et al. “Integrated design methodology of automated guided vehicles based on swarm robotics”. In : *Applied Sciences* 11.13 (2021), p. 6187.

[6] Brian DO Anderson et al. “Rigid graph control architectures for autonomous formations”. In : t. 28. 6. IEEE, 2008, p. 48-63.

[7] Tamio Arai, Enrico Pagello, Lynne E Parker et al. “Advances in multi-robot systems”. In : *IEEE Transactions on robotics and automation* 18.5 (2002), p. 655-661.

[8] Ronald C Arkin, Ronald C Arkin et al. *Behavior-based robotics*. MIT press, 1998.

[9] Farshad Arvin et al. “Imitation of honeybee aggregation with collective behavior of swarm robots”. In : *International Journal of Computational Intelligence Systems* 4.4 (2011), p. 739-748.

[10] Ferdinando Auricchio. “A continuous model for the simulation of manufacturing swarm robotics”. In : *Computational Mechanics* (2022), p. 1-8.

[11] D Barth, IA Gorlach et G Gruhler. “Development of a novel controller for a HVAF thermal spray process”. In : *Proceedings of the International Conference on Competitive Manufacturing (Coma’10)*. 2010.

[12] DI Batishchev et S Isayev. “Optimization of multi functions using genetic algorithms”. In : *Interuniversity collection of scientific papers" High technologies in engineering, medicine and education".(Voronezh: VGTU) pp* (1997), p. 4-17.

[13] Levent Bayindir. “A probabilistic geometric model of self-organized aggregation in swarm robotic systems”. In : (2012).

[14] Levent Bayindir et Erol Şahin. “A review of studies in swarm robotics”. In : *Turkish Journal of Electrical Engineering and Computer Sciences* 15.2 (2007), p. 115-147.

[15] Levent Bayindir et Erol Şahin. “A review of studies in swarm robotics”. In : *Turkish Journal of Electrical Engineering and Computer Sciences* 15.2 (2007), p. 115-147.

[16] Gerardo Beni. “From swarm intelligence to swarm robotics”. In : *International Workshop on Swarm Robotics*. Springer. 2004, p. 1-9.

[17] Wang J. Beni G. “Swarm intelligence in cellular robotic systems.” In : *In Robots and biological systems : towards a new bionics*. 1993, p. 703-712.

[18] Tobias Blickle et Lothar Thiele. “A comparison of selection schemes used in genetic algorithms. TIK-Report 11, TIK Institut fur Technische Informatik und Kommunikationsnetze”. In : *Computer Engineering and Networks Laboratory, ETH, Swiss Federal Institute of Technology, Gloriastrasse* 35.8092 (1995), p. 279-284.

[19] Eric Bonabeau et al. “Swarm intelligence: from natural to artificial systems”. In : *Oxford university press*. 1. 1999.

[20] Darko Bozhinoski et Mauro Birattari. “Towards an integrated automatic design process for robot swarms”. In : *Open Research Europe* 1.112 (2021), p. 112.

[21] Manuele Brambilla et al. “Property-driven design for robot swarms: A design method based on prescriptive modeling and model checking”. In : *ACM Transactions on Autonomous and Adaptive Systems (TAAS)* 9.4 (2014), p. 1-28.

[22] Manuele Brambilla et al. “Property-driven design for swarm robotics”. In : *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*. 2012, p. 139-146.

[23] Manuele Brambilla et al. “Swarm robotics: a review from the swarm engineering perspective”. In : *Swarm Intelligence* 7.1 (2013), p. 1-41.

[24] Manuele Brambilla et al. “Swarm robotics: a review from the swarm engineering perspective”. In : *Swarm Intelligence* 7.1 (2013), p. 1-41.

[25] Christian Brecher, Johannes A Nittinger et Andreas Karlberger. “Model-based control of a handling system with SysML”. In : *Procedia Computer Science* 16 (2013), p. 197-205.

[26] Dennis M Buede et William D Miller. “The engineering design of systems: models and methods”. In : (2016).

[27] Martin Burger, Jan Haškovec et Marie-Therese Wolfram. “Individual based and mean-field modeling of direct aggregation”. In : *Physica D: Nonlinear Phenomena* 260 (2013), p. 145-158.

[28] Cesar Cadena et al. “Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age”. In : *IEEE Transactions on robotics* 32.6 (2016), p. 1309-1332.

[29] Cesar Cadena et al. “Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age”. In : *IEEE Transactions on robotics* 32.6 (2016), p. 1309-1332.

[30] Cindy Calderón-Arce, Juan Carlos Brenes-Torres et Rebeca Solis-Ortega. “Swarm Robotics: Simulators, Platforms and Applications Review”. In : *Computation* 10.6 (2022), p. 80.

[31] Scott Camazine et al. “Self-organization in biological systems”. In : *Self-Organization in Biological Systems*. Princeton university press, 2020.

[32] Gilles Caprari et Roland Siegwart. “Mobile micro-robots ready to use: Alice”. In : *2005 IEEE/RSJ international conference on intelligent robots and systems*. IEEE. 2005, p. 3295-3300.

[33] Constantin F Caruntu et al. “Bio-inspired Coordination and Control of Autonomous Vehicles in Future Manufacturing and Goods Transportation”. In : *IFAC-PapersOnLine* 53.2 (2020), p. 10861-10866.

[34] Constantin F Caruntu et al. “Connected cooperative control for multiple-lane automated vehicle flocking on highway scenarios”. In : *2019 23rd International Conference on System Theory, Control and Computing (ICSTCC)*. IEEE. 2019, p. 791-796.

[35] Rodolphe Charrier. “L’intelligence en essaim sous l’angle des systèmes complexes: étude d’un système multi-agent réactif à base d’itérations logistiques couplées”. In : 2009.

[36] Jianing Chen, Melvin Gauci et Roderich Gross. “A strategy for transporting tall objects with a swarm of miniature mobile robots”. In : *2013 IEEE International Conference on Robotics and Automation*. IEEE. 2013, p. 863-869.

[37] Anders Lyhne Christensen, Rehan OGrady et Marco Dorigo. “From fireflies to fault-tolerant swarms of robots”. In : *IEEE Transactions on Evolutionary Computation* 13.4 (2009), p. 754-766.

[38] Maia R. D. De Castro L. N. Cruz D. P. F. “A critical discussion into the core of swarm intelligence algorithms.” In : *Evolutionary Intelligence, 12(2)*. 2019, p. 189-200.

[39] “Cyberbotics ltd,"Webots User Guide ...", URL : https://cyberbotics.com/doc/guide/introduction-towebots”. In.

[40] Swagatam Das et al. “Bacterial foraging optimization algorithm: theoretical foundations, analysis, and applications”. In : *Foundations of computational intelligence volume 3*. Springer, 2009, p. 23-55.

[41] Prithviraj Dasgupta. “A Dynamic-bid Auction Algorithm for Cooperative, Distributed Multi-Robot Task Allocation”. In : *UNO Technical Report (cst-2009-2)* (2009).

[42] Marco Dorigo et Thomas Stützle. “Ant colony optimization: overview and recent advances”. In : *Handbook of metaheuristics* (2019), p. 311-351.

[43] Gregory Dudek et al. “A taxonomy for swarm robots”. In : *Proceedings of 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS’93)*. T. 1. IEEE. 1993, p. 441-447.

[44] Ruwan Egodagamage et Mihran Tuceryan. “Distributed monocular visual SLAM as a basis for a collaborative augmented reality framework”. In : *Computers & Graphics* 71 (2018), p. 113-123.

[45] Tom Erez, Yuval Tassa et Emanuel Todorov. “Simulation tools for model-based robotics: Comparison of bullet, havok, mujoco, ode and physx”. In : *2015 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2015, p. 4397-4404.

[46] Jeff A Estefan et al. “Survey of model-based systems engineering (MBSE) methodologies”. In : *Incose MBSE Focus Group* 25.8 (2007), p. 1-12.

[47] Tremaine Ferreira et IA Gorlach. “Development of an automated guided vehicle controller using a model-based systems engineering approach”. In : *South African journal of industrial engineering* 27.2 (2016), p. 206-217.

[48] Gianpiero Francesca et al. “AutoMoDe: A novel approach to the automatic design of control software for robot swarms”. In : *Swarm Intelligence* 8.2 (2014), p. 89-112.

[49] Lorenzo Garattoni et Mauro Birattari. “Swarm robotics”. In : *Wiley Encyclopedia of Electrical and Electronics Engineering. Hoboken: John Wiley & Sons* (2016), p. 1-19.

[50] Simon Garnier. “Décisions collectives dans des systèmes d’intelligence en essaim”. Thèse de doct. Toulouse 3, 2008.

[51] Simon Garnier, Jacques Gautrais et Guy Theraulaz. “The biological principles of swarm intelligence”. In : *Swarm intelligence* 1.1 (2007), p. 3-31.

[52] Simon Garnier, Jacques Gautrais et Guy Theraulaz. “The biological principles of swarm intelligence”. In : *Swarm intelligence* 1.1 (2007), p. 3-31.

[53] “Gazebo, URL: http://playerstage.sourceforge.net/gazebo/gazebo.html.” In.

[54] Jorge Gomes, Paulo Urbano et Anders Lyhne Christensen. “Evolution of swarm robotics systems with novelty search”. In : *Swarm Intelligence* 7.2 (2013), p. 115-144.

[55] Amir Guizani et al. “Multi-agent approach based on a design process for the optimization of mechatronic systems”. In : *Mechanics & Industry* 18.5 (2017), p. 507.

[56] Emily J Hackett-Jones, Kerry A Landman et Klemens Fellner. “Aggregation patterns from nonlocal interactions: Discrete stochastic and continuum modeling”. In : *Physical Review E* 85.4 (2012), p. 041912.

[57] Matthew Hause et al. “The SysML modelling language”. In : *Fifteenth European Systems Engineering Conference*. T. 9. 2006, p. 1-12.

[58] Adam T Hayes et Parsa Dormiani-Tabatabaei. “Self-organized flocking with agent failure: Off-line optimization and demonstration with real robots”. In : *Proceedings 2002 IEEE international conference on robotics and automation (cat. no. 02ch37292)*. T. 4. IEEE. 2002, p. 3900-3905.

[59] Owen Holland et Chris Melhuish. “Stigmergy, self-organization, and sorting in collective robotics”. In : *Artificial life* 5.2 (1999), p. 173-202.

[60] “https://emanual.robotis.com/docs/en/platform/turtlebot3/hardware_setup/#hardware-assembly”. In : ().

[61] “https://emanual.robotis.com/docs/en/platform/turtlebot3/quick-start/”. In : ().

[62] "https://github.com/RobotnikAutomation/agvs". In.

[63] "https://github.com/yangliu28/swarm$_robot_os_sim$". In.

[64] "https:https://www.youtube.com/watch?v=BKXuokwg3Xgab$_channel = YangLiu$". In.

[65] Xinge Huang et al. "Exploration in extreme environments with swarm robotic system". In : *2019 IEEE international conference on mechatronics (ICM)*. T. 1. IEEE. 2019, p. 193-198.

[66] Andrew Ilachinski. *Cellular automata: a discrete universe*. World Scientific Publishing Company, 2001.

[67] J Jerald et al. "Scheduling optimisation of flexible manufacturing systems using particle swarm optimisation algorithm". In : *The International Journal of Advanced Manufacturing Technology* 25.9 (2005), p. 964-971.

[68] Shifali Kalra et Sankalap Arora. "Firefly algorithm hybridized with flower pollination algorithm for multimodal functions". In : *Proceedings of the international congress on information and communication technology*. Springer. 2016, p. 207-219.

[69] Simranpreet Kaur et Shivani Sharma. "Performance evaluation of artificial bee colony and compressive sensing based energy efficient protocol for WSNs". In : *2017 13th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)*. IEEE. 2017, p. 91-96.

[70] Sanza Kazadi, John R Lee et Julie Lee. "Artificial physics, swarm engineering, and the hamiltonian method". In : *World congress on engineering and computer science*. 2007, p. 623-632.

[71] Sanza T Kazadi. *Swarm engineering*. California Institute of Technology, 2000.

[72] Miquel Kegeleirs, David Garzón Ramos et Mauro Birattari. "Random walk exploration for swarm mapping". In : *Annual conference towards autonomous robotic systems*. Springer. 2019, p. 211-222.

[73] Serge Kernbach et al. "Re-embodiment of honeybee aggregation behavior in an artificial micro-robotic system". In : *Adaptive Behavior* 17.3 (2009), p. 237-259.

[74] Serge Kernbach et al. "Re-embodiment of honeybee aggregation behavior in an artificial micro-robotic system". In : *Adaptive Behavior* 17.3 (2009), p. 237-259.

[75] Sergey Kornienko, Olga Kornienko et Paul Levi. "Minimalistic approach towards communication and perception in microrobotic swarms". In : *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2005, p. 2228-2234.

[76] C Ronald Kube et Eric Bonabeau. "Cooperative transport by ants and robots". In : *Robotics and autonomous systems* 30.1-2 (2000), p. 85-101.

[77] Pierre-Yves Lajoie et al. "Towards collaborative simultaneous localization and mapping: a survey of the current research landscape". In : *arXiv preprint arXiv:2108.08325* (2021).

[78] Julian Lategahn, Marcel Müller et Christof Röhrig. “Global localization of automated guided vehicles in wireless networks”. In : *2012 IEEE 1st International Symposium on Wireless Systems (IDAACS-SWS)*. IEEE. 2012, p. 7-12.

[79] Kristina Lerman et Aram Galstyan. “Mathematical model of foraging in a group of robots: Effect of interference”. In : *Autonomous robots* 13.2 (2002), p. 127-141.

[80] Kristina Lerman, Alcherio Martinoli et Aram Galstyan. “A review of probabilistic macroscopic models for swarm robotic systems”. In : *International workshop on swarm robotics*. Springer. 2004, p. 143-152.

[81] Kristina Lerman, Alcherio Martinoli et Aram Galstyan. “A review of probabilistic macroscopic models for swarm robotic systems”. In : *International workshop on swarm robotics*. Springer. 2004, p. 143-152.

[82] Xiaodong Li et Maurice Clerc. “Swarm intelligence”. In : *Handbook of Metaheuristics*. Springer, 2019, p. 353-384.

[83] Ryan Luna et Kostas E Bekris. “Network-guided multi-robot path planning in discrete representations”. In : *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2010, p. 4596-4602.

[84] Steven Macenski et al. “Robot Operating System 2: Design, architecture, and uses in the wild”. In : *Science Robotics* 7.66 (2022), eabm6074.

[85] B Mahadevan et TT Narendran. “Design of an automated guided vehicle-based material handling system for a flexible manufacturing system”. In : *The International Journal of Production Research* 28.9 (1990), p. 1611-1622.

[86] NOR EL HOUDA MAIZI. “Coordination bio-inspirée d’un essaim de robots: application au problème de recherche et de sauvetage.” In : (2020).

[87] Marco Mamei, Matteo Vasirani et Franco Zambonelli. “Experiments of morphogenesis in swarms of simple mobile robots”. In : *Applied Artificial Intelligence* 18.9-10 (2004), p. 903-919.

[88] James N Martin. *Systems engineering guidebook: A process for developing systems and products*. CRC press, 2020.

[89] Sonia Martinez, Jorge Cortes et Francesco Bullo. “Motion coordination with distributed information”. In : t. 27. 4. IEEE, 2007, p. 75-88.

[90] Alcherio Martinoli, Kjerstin Easton et William Agassounon. “Modeling swarm robotic systems: A case study in collaborative distributed manipulation”. In : *The International Journal of Robotics Research* 23.4-5 (2004), p. 415-436.

[91] Alcherio Martinoli, Kjerstin Easton et William Agassounon. “Modeling swarm robotic systems: A case study in collaborative distributed manipulation”. In : *The International Journal of Robotics Research* 23.4-5 (2004), p. 415-436.

[92] Ivan Maza et al. “Classification of multi-UAV architectures”. In : *Handbook of unmanned aerial vehicles* (2015), p. 953-975.

[93] Ryan McCune et al. “Investigations of dddas for command and control of uav swarms with agent-based modeling”. In : *2013 Winter Simulations Conference (WSC)*. IEEE. 2013, p. 1467-1478.

[94] James Dwight McLurkin. “Stupid robot tricks: A behavior-based distributed algorithm library for programming swarms of robots”. Thèse de doct. Massachusetts Institute of Technology, 2004.

[95] Hans Meinhardt. “Biological pattern formation: new observations provide support for theoretical predictions”. In : *Bioessays* 16.9 (1994), p. 627-632.

[96] Kevin L Mills. “A brief survey of self-organization in wireless sensor networks”. In : t. 7. 7. Wiley Online Library, 2007, p. 823-834.

[97] Enrico Mingo Hoffman et al. “Yarp based plugins for gazebo simulator”. In : *International Workshop on Modelling and Simulation for Autonomous Systems*. Springer. 2014, p. 333-346.

[98] Christoph Moeslinger, Thomas Schmickl et Karl Crailsheim. “A minimalist flocking algorithm for swarm robots”. In : *European Conference on Artificial Life*. Springer. 2009, p. 375-382.

[99] Francesco Mondada, Edoardo Franzi et Andre Guignard. “The development of khepera”. In : *Experiments with the Mini-Robot Khepera, Proceedings of the First International Khepera Workshop*. CONF. 1999, p. 7-14.

[100] Francesco Mondada et al. “SWARM-BOT: A new distributed robotic concept”. In : *Autonomous robots* 17.2 (2004), p. 193-221.

[101] Francesco Mondada et al. “The cooperation of swarm-bots: Physical interactions in collective robotics”. In : *IEEE Robotics & Automation Magazine* 12.2 (2005), p. 21-28.

[102] Francesco Mondada et al. “The e-puck, a robot designed for education in engineering”. In : *Proceedings of the 9th conference on autonomous robot systems and competitions*. T. 1. CONF. IPCB: Instituto Politécnico de Castelo Branco. 2009, p. 59-65.

[103] Maryam Mousavi et al. “Multi-objective AGV scheduling in an FMS using a hybrid of genetic algorithm and particle swarm optimization”. In : *PloS one* 12.3 (2017), e0169817.

[104] Mauricio F Munoz. *Agent-based simulation and analysis of a defensive UAV swarm against an enemy UAV swarm*. Rapp. tech. Naval Postgraduate School Monterey CA, 2011.

[105] Manasa Nadipally. “Optimization of methods for image-texture segmentation using ant colony optimization”. In : *Intelligent data analysis for biomedical applications*. Elsevier, 2019, p. 21-47.

[106] Iñaki Navarro et Fernando Matía. “An introduction to swarm robotics”. In : *International Scholarly Research Notices* 2013 (2013).

[107] Solomon Sunday Oyelere. “The application of model predictive control (MPC) to fast systems such as autonomous ground vehicles (AGV)”. In : *IOSR Journal of Computer Engineering* 3.3 (2014), p. 27-37.

[108] James O’Keeffe et al. “Towards fault diagnosis in robot swarms: An online behaviour characterisation approach”. In : *Annual Conference Towards Autonomous Robotic Systems*. Springer. 2017, p. 393-407.

[109] Lynne E Parker. “Designing control laws for cooperative agent teams”. In : *[1993] Proceedings IEEE International Conference on Robotics and Automation*. IEEE. 1993, p. 582-587.

[110] Srikanta Patnaik, Xin-She Yang et Kazumi Nakamatsu. *Nature-inspired computing and optimization*. T. 10. Springer, 2017.

[111] Carlo Pinciroli et al. “ARGoS: a modular, multi-engine simulator for heterogeneous swarm robotics”. In : *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2011, p. 5027-5034.

[112] Carlo Pinciroli et al. “ARGoS: a modular, parallel, multi-engine simulator for multi-robot systems”. In : *Swarm intelligence* 6.4 (2012), p. 271-295.

[113] Carlo Pinciroli et al. “Self-organised recruitment in a heteregeneous swarm”. In : *2009 International Conference on Advanced Robotics*. IEEE. 2009, p. 1-8.

[114] Jim Pugh et Alcherio Martinoli. “Distributed adaptation in multi-robot search using particle swarm optimization”. In : *International Conference on Simulation of Adaptive Behavior*. Springer. 2008, p. 393-402.

[115] Jim Pugh et al. “A fast onboard relative positioning module for multirobot systems”. In : *IEEE/ASME Transactions on Mechatronics* 14.2 (2009), p. 151-162.

[116] Morgan Quigley, Brian Gerkey et William D Smart. *Programming Robots with ROS: a practical introduction to the Robot Operating System*. " O’Reilly Media, Inc.", 2015.

[117] Rajesh Rajamani. *Vehicle dynamics and control*. Springer Science & Business Media, 2011.

[118] Ana Luísa Ramos, José Vasconcelos Ferreira et Jaume Barceló. “Model-based systems engineering: An emerging approach for modern systems”. In : *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 42.1 (2011), p. 101-111.

[119] Erick J Rodríguez-Seda, Dušan M Stipanović et Mark W Spong. “Collision avoidance control with sensing uncertainties”. In : *Proceedings of the 2011 American Control Conference*. IEEE. 2011, p. 3363-3368.

[120] Davide Ronzoni et al. “AGV global localization using indistinguishable artificial landmarks”. In : *2011 IEEE International Conference on Robotics and Automation*. IEEE. 2011, p. 287-292.

[121] Lorenzo Sabattini et al. “Technological roadmap to boost the introduction of AGVs in industrial applications”. In : *2013 IEEE 9th International Conference on Intelligent Computer Communication and Processing (ICCP)*. IEEE. 2013, p. 203-208.

[122] Erol Şahin. “Swarm robotics: From sources of inspiration to domains of application”. In : *International workshop on swarm robotics*. Springer. 2004, p. 10-20.

[123] Bibhya Sharma, Jito Vanualailai et Avinesh Prasad. “A-Strategy: Facilitating Dual-Formation Control of a Virtually Connected Team”. In : *Journal of Advanced Transportation* 2017 (2017).

[124] Hillary Sillitto et al. “Systems engineering and system definitions”. In : *INCOSE*. 2019.

[125] Avtar Singh et al. “Swarm Robotics: A Review from Mechanical Engineering Perspective”. In : ().

[126] Susan Stepney, Fiona AC Polack et Heather R Turner. “Engineering emergence”. In : *11th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS’06)*. IEEE. 2006, 9-pp.

[127] Bart Stouten et Aart-Jan de Graaf. “Cooperative transportation of a large object-development of an industrial application”. In : *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA’04. 2004*. T. 3. IEEE. 2004, p. 2450-2455.

[128] Velamuri Suresh et al. “Demand response-integrated economic dispatch incorporating renewable energy sources using ameliorated dragonfly algorithm”. In : *Electrical Engineering* 101.2 (2019), p. 421-442.

[129] Ekaterina Tolstaya et al. “Learning decentralized controllers for robot swarms with graph neural networks”. In : *Conference on robot learning*. PMLR. 2020, p. 671-682.

[130] Vito Trianni et Stefano Nolfi. *Self-Organising Sync in a Robotic Swarm*. Citeseer, 2008.

[131] Ali E Turgut et al. “Kobot: A mobile robot designed specifically for swarm robotics research”. In : *Middle East Technical University, Ankara, Turkey, METU-CENG-TR Tech. Rep* 5.2007 (2007).

[132] Pietro Valdastri et al. “Micromanipulation, communication and swarm intelligence issues in a swarm microrobotic platform”. In : *Robotics and Autonomous Systems* 54.10 (2006), p. 789-804.

[133] Jito Vanualailai et Bibhya Sharma. “A Lagrangian-based swarming behavior in the absence of obstacles”. In : *Workshop on Mathematical Control Theory, Kobe University*. 2010, p. 8-10.

[134] Richard Vaughan. “Massively multi-robot simulation in stage”. In : *Swarm intelligence* 2.2 (2008), p. 189-208.

[135] Frank Weiskopf et al. “Control of cooperative, autonomous unmanned aerial vehicles”. In : *1st UAV Conference*. 2002, p. 3444.

[136] Xin-She Yang. “Cuckoo search and firefly algorithm: overview and analysis”. In : *Cuckoo search and firefly algorithm* (2014), p. 1-26.

[137] PYO Yoonseok et al. “ROS robot programming”. In : *Seoul, Republic of Korea: ROBOTIS Co., Ltd* (2017).

[138] Maria Zemzami et al. “Applying a new parallelized version of PSO algorithm for electrical power transmission”. In : *IOP Conference Series: Materials Science and Engineering*. T. 205. 1. IOP Publishing. 2017, p. 012032.