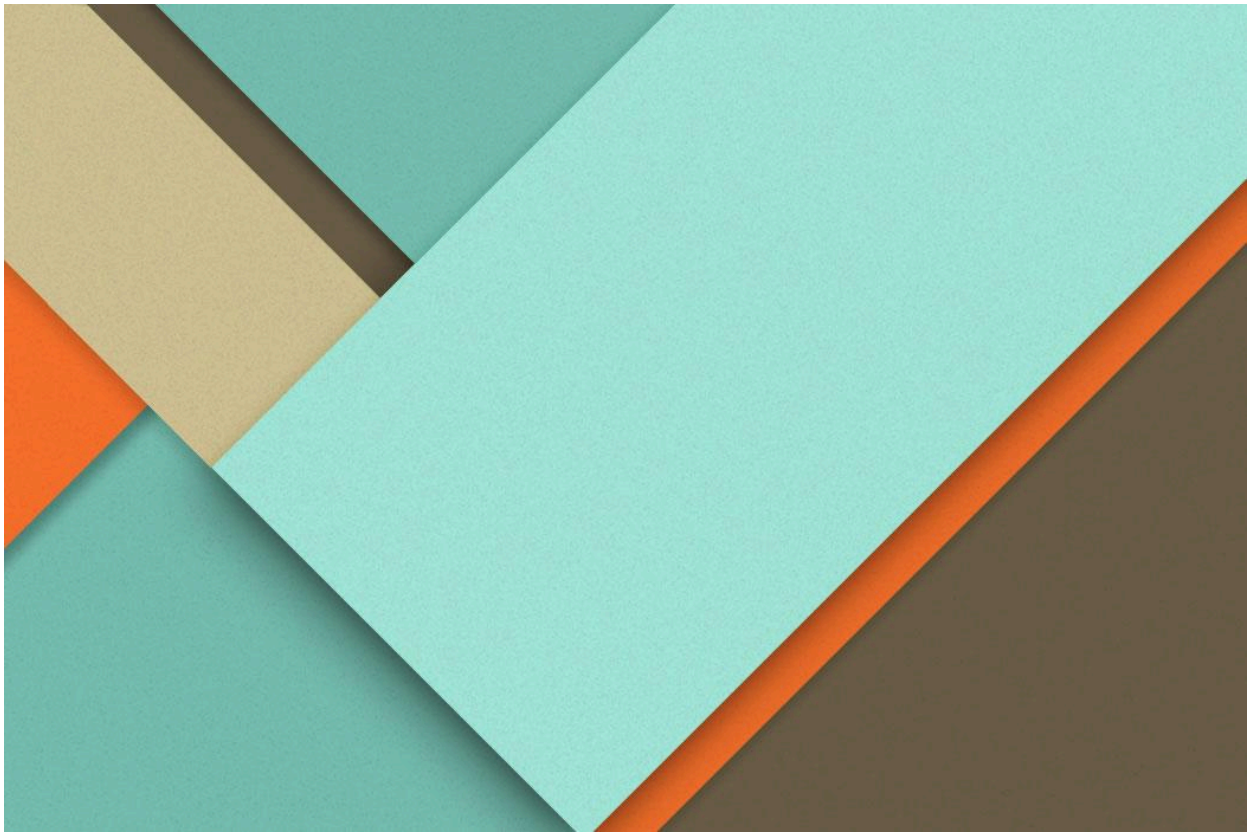


# Projet final DAAR 2025 - Modèle d'Acteur dans les Systèmes Multi-Agents

25.01.2026

—

MARSSO Denn 28608482
M2 STL

# I. Introduction et définition du problème

## 1. Contexte

Les Systèmes Multi-Agents (SMA) représentent un paradigme de calcul distribué où plusieurs entités autonomes (agents) interagissent pour résoudre des problèmes complexes. Le modèle d'acteur (Actor Model), introduit par Carl Hewitt en 1973, fournit un cadre formel pour la conception de ces systèmes concurrents.

Dans ce projet, nous implémentons des stratégies de combat robotique dans le simulateur Simovies, ou deux équipes de 5 robots s'affrontent. Chaque robot est un acteur autonome qui :

- Perçoit son environnement via des capteurs (radar 360 degrés, détection frontale)
- Prend des décisions locales basées sur des machines à états finis
- Communique avec ses coéquipiers via broadcast asynchrone
- Agit sur l'environnement (déplacement, tir)

## 2. Définition formelle du problème

Soit A = {a1, ..., an} un ensemble d'agents et E l'environnement partage.

Chaque agent IA est défini par le tuple (Si, Pi, Ai, delta_i, Ci) ou :

- Si : ensemble fini d'états internes
- Pi : E -> Oi : fonction de perception
- Ai : ensemble d'actions possibles (move, fire, turn, broadcast)
- delta_i : Si x Oi x Mi -> Si x Ai : fonction de transition
- Ci : canal de communication (messages reçus)

L'objectif est de maximiser une fonction d'utilité collective U(A) représentant le score de l'équipe (ennemis éliminés, survie des robots).

### 3. Structure de données et FSM finale

**Machine à états finis (FSM)** : la version finale de notre stratégie B utilise des FSM simples, robustes, et fortement locales.

- Main Bot B (Défenseur) : états {ADVANCING, HOLDING, DODGING}
- Secondary Bot B (Kamikaze) : états {SEARCHING, ATTACKING}

Cette réduction volontaire d'états limite les comportements instables (blocage, oscillations) et conserve une forte réactivité temps réel.

**Communication inter-agents (version finale)** :

- La version finale ne repose pas sur un protocole de synchronisation complexe entre Main Bots.
- Nous avons testé une variante avec broadcast de position du kamikaze (tracking) et regroupement, mais elle a été abandonnée (voir Section 5) car elle dégraderait les performances.

## II. Analyse théorique des algorithmes

### 1. Modèle d'acteur et propriétés

Le modèle d'acteur garantit plusieurs propriétés essentielles (Hewitt, Bishop, Steiger, 1973) :

- **Encapsulation** : chaque robot maintient un état privé inaccessible directement
- **Communication asynchrone** : les messages sont non bloquants
- **Comportement réactif** : réponse aux stimuli (perception + messages)

Notre implémentation correspond au schéma suivant (simplifié) :

```
public void step() {
    // Perception locale (radar, front sensor)
    // Decision par etat (FSM)
    switch (state) {
        case ADVANCING: stepAdvancing(); break;
        case HOLDING:   stepHolding();   break;
        case DODGING:   stepDodging();   break;
    }
}
```

## 2. Choix de conception : simplicité et verticalité

Dans un environnement temps réel, ajouter des etats de coordination (regroupement, chasse, consensus) peut augmenter :

- le nombre de transitions,
- les cas limites (messages obsolètes),
- les risques de blocage collectif.

Nous avons donc privilégié une stratégie verticale et simple :

- un robot qui voit une cible passe en engagement,
- un robot qui ne voit rien continue la pression vers l'Ouest,
- l'évitement reste local et court.

Ce choix favorise un comportement stable et reproductible.

## 3. Variante testée : tracking kamikaze + regroupement (non retenue)

Nous avons testé une variante dans laquelle :

- le kamikaze broadcastait régulièrement une position estimée (x, y),
- les Main Bots tentaient de suivre cette position via un etat de chasse / regroupement.
- le tout devait permettre une chasse groupée donc plus puissante

En pratique, cette variante a augmenté :

- les blocages (forteresse coincee contre des obstacles, car lorsque un est bloqué ils le sont tous),
- les mauvais angles de tir (rotation excessive, tirs alliés),
- la complexité (et donc les bugs).

La version finale supprime ce mécanisme et obtient de meilleurs résultats, cependant l'ancienne version est disponible dans les classes OldTeamB...

# III. Présentation des algorithmes implémentés

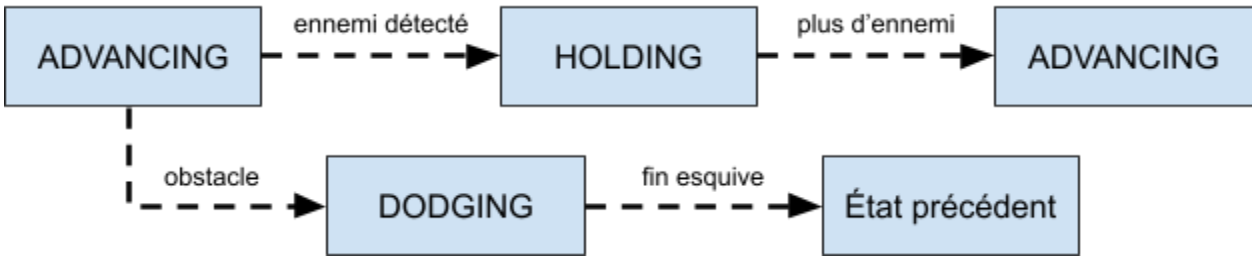
## 1. Stratégie B - "Fantom Danger" (version finale)

### *a. Main Bot B - Défenseur vertical autonome*

**Principe** :

- Avance vers WEST par segments.
- Si un obstacle est détecté en frontal (non allié), déclenche un DODGING court (NORTH ou SOUTH), puis reprend la direction WEST.

**FSM** :




**Code clé (idée) :**

```
// Scan rapide + tir cadence
scanAndShoot();

// Etat ADVANCING : avance vers l'Ouest, esquive si bloque
// Etat HOLDING   : vise + tire sur l'ennemi le plus proche
// Etat DODGING   : mouvement court vertical, puis retour
```

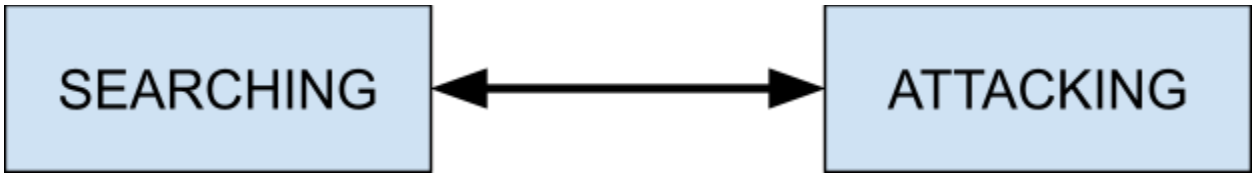
### *b. Secondary Bot B - Kamikaze agressif simple*

**Principe** :

- Cherche en permanence via radar.
- Dès qu'un ennemi est détecté : ATTACKING (fonce pour le bloquer).

- Si l'ennemi disparaît du radar (éliminée ou hors champ) : retour immédiat en SEARCHING.

**FSM** :




Ce comportement simple évite la poursuite "a vide" et maintient une pression constante, cela permet de bloquer les robots adverses et provoquer des bugs.

### 2. Stratégie A - "KD Runners" (comparaison)

**Principe** :

- Les secondary sont des éclaireurs, ils cherchent les ennemis envoient la position puis passe en mode esquive (mouvements aléatoires larges pour esquiver les balles et garder l'attention de l'ennemi)
- les main sont des chasseurs, dès qu'ils reçoivent la position ennemis ils accélèrent pour les chasser

La stratégie A (equipe A) a finalement servi de comparaison interne, elle perdait en continu contre tout les bots car trop sensibles aux blocages et bugs :

- comportements plus offensifs et plus mobiles,
- coordination plus faible,
- performances moins stables face à des adversaires agressifs.

## IV. Méthodologie de test

### 1. Protocole expérimental

**Configuration** :

- Arene : 3000 x 2000 mm
- Equipe A : spawn côté gauche (X ~ 200-500)

- Equipe B : spawn côté droit bas (X ~ 2500-2800)

**Métriques observées** :

- Taux de victoire (parties gagnées)
- Nombre de robots survivants en fin de partie (qualitatif)
- Stabilité du comportement (blocages / oscillations)

### 2. Jeux de donnees et resultats

Tests effectués sur **10 parties** par adversaire :

| Adversaire | Victoires | Défaites | Taux |
|---|---|---|---|
| BootingBerzerk | 9 | 1 | 90% |
| CampFire | 10 | 0 | 100% |
| RandomFire | 10 | 0 | 100% |

**Observations** :

- Contre **BootingBerzerk**, la stratégie B finale gagne 9/10, la défaite survenant typiquement lors d'un enchaînement défavorable (angles + collisions).
- Contre **CampFire**, victoire systématique, mais de nombreux robots restent vivants longtemps (style défensif proche).
- Contre **RandomFire**, victoire systématique grâce à la régularité et la pression continue.

## V. Discussion et analyse critique

### 1. Forces de l'approche finale

1. **Simplicité robuste** : peu d'etats, peu de transitions, moins de cas limites
2. **Verticalité efficace** : angles de tir plus propres, moins de rotations inutiles
3. **Réactivité** : bascule rapide en engagement dès qu'une cible est visible

4. **Tolérance aux défaillances** : la perte d'un robot n'interrompt pas le comportement des autres

## 2. Limites identifiées

1. **Pas de stratégie de formation explicite** : la cohésion de groupe est émergente, pas contrôlée
2. **Tir non prédictif** : vise la direction actuelle, sans anticipation de trajectoire
3. **Évitement local** : le DODGING peut parfois conduire à des détours sous-optimaux
4. **Peu agressif** : le manque d'agressivité peut conduire à un état ou les kamikaz meurent et tout les autres survivent car ils ne se croisent pas (donc défaite aux points)

## 3. Retour d'expérience : pourquoi la coordination par tracking a dégradé les performances ?

La variante "kamikaze + tracking + forteresse qui suit" a été testée, mais :

- la forteresse se bloquait plus souvent,
- les Main Bots se retrouvaient dans de mauvais angles pour tirer,
- La complexité augmente les comportements incohérents.

Conclusion : la **verticalité** et la **simplicité** apportent un meilleur compromis que la coordination explicite. Garantir une coordination de groupe et éviter au maximum de séparer le groupe ou le bloquer permet de garder une puissance de feu considérable et de souvent débuter les combats a 2 ou 3 contre 1. Cependant, contre des stratégies peu agressives, cela peut se retourner contre nous car il y a un manque de détection.

## 4. Améliorations possibles

1. Tir semi-prédictif (estimation simple de mouvement ennemi)
2. Règles de spacing légères (éviter la collision entre alliés sans regroupement forcé)
3. Ajuster dynamiquement la cadence de tir selon la distance (proche = plus rapide)
4. Détecter les ennemis et les chasser

# VI. Conclusion et perspectives

## 1. Bilan

Ce projet illustre l'applicabilité du modèle d'acteur aux systèmes multi-agents robotiques :

- acteurs autonomes (robots) avec état privé,
- perception locale,
- décisions par FSM,
- communication éventuelle (variante testée).

Les résultats expérimentaux montrent que la version finale de la stratégie B, plus simple et plus verticale, est la plus efficace sur notre suite de tests, avec des taux de victoire élevés et une meilleure stabilité.

## 2. Perspectives

Des extensions envisageables incluent :

- un tir plus intelligent (anticipation),
- une formation dynamique légère (sans regroupement rigide),
- des mécanismes d'exploration plus variés lorsque l'ennemi est absent.