Couplage dynamique et passage à l'échelle de modèles socio-environnementaux complexes

Arthur Brugiere

▶ **To cite this version:**

Arthur Brugiere. Couplage dynamique et passage à l'échelle de modèles socio-environnementaux complexes. Modélisation et simulation. Sorbonne Université, 2025. Français. ⟨NNT : 2025SORUS566⟩. ⟨tel-05560358⟩

HAL Id: tel-05560358

https://theses.hal.science/tel-05560358v1

Submitted on 20 Mar 2026

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

HAL Authorization

THÈSE DE DOCTORAT
DE L'UNIVERSITÉ SORBONNE UNIVERSITÉ

Spécialité : Sciences et technologies de l'information et de la communication
École doctorale : EDITE De Paris

Réalisée au

UMI 209 UMMISCO / LMI ACROSS

présentée par

Arthur Brugière

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ SORBONNE UNIVERSITÉ

Sujet de la thèse :

Couplage dynamique et passage à l'échelle de modèles socio-environnementaux complexes

---

Soutenue le 9 Décembre 2025 devant le jury composé de :

M.	Rémy Courdier	Rapporteur
Mme	Julie Dugdale	Rapporteuse
M.	Raphael Duboz	Examinateur
M.	Vinh Ho Tuong	Président du jury - Examinateur
M.	Doanh Ngoc Nguyen	Co-Directeur de thèse
M.	Alexis Drogoul	Directeur de thèse

Comprendre, c'est avant tout unifier.
— Albert Camus, *Noces* (1939)

Remerciements

Ces travaux de thèse n'auraient pu voir le jour sans le concours de nombreuses personnes, que je tiens à remercier chaleureusement ici.

Mes premiers remerciements vont naturellement aux membres du jury qui ont accepté d'évaluer ce travail. Je remercie Rémy Courdier et Julie Dugdale pour avoir endossé le rôle de rapporteurs et pour la qualité de leurs retours, ainsi que Raphael Duboz et Vinh Ho Tuong pour leur participation en tant qu'examinateurs. Leur lecture attentive et leurs retours ont contribué à enrichir ce travail.

Toute ma gratitude va à Alexis Drogoul, mon directeur de thèse, pour sa confiance depuis le premier jour, son encadrement bienveillant, la vision scientifique et sa capacité à toujours remettre les choses en perspective au bon moment. Merci également à Doanh Ngoc Nguyen, mon co-directeur, pour son soutien tout au long de ces années. Travailler à vos côtés a été une chance et un plaisir.

Une grande partie de cette thèse s'est construite au contact de la communauté GAMA, qui m'a accueilli bien avant le début de ces travaux et dont l'énergie collective a nourri chacune de mes réflexions. Merci, sans ordre particulier, à Patrick, Baptiste, Lucas, Benoît, Kévin, Nghi, Arnaud, Tri et tous les autres, pour les discussions, le soutien et les moments partagés. Vous avez tous, à un moment ou à un autre, contribué à faire avancer ce travail.

Au laboratoire ACROSS, merci à tous les collègues qui ont partagé ces années avec moi. Une pensée particulière pour Léo, Jeanne et Diep Anh, qui ont été présents dans les bons moments comme dans les moins bons.

Enfin, les mots les plus importants sont pour ma famille et mes proches, qui ont su être là dans les moments de doute et de joie. Et surtout, à ma femme :

Em là nguồn động lực lớn nhất của anh.

Et merci, enfin, à tous ceux que cette page oublie maladroitement : vous savez qui vous êtes, et merci.

Dynamic coupling and scaling of complex socio-environmental models

Abstract

Socio-environmental systems exhibit multi-scale complexity requiring modeling approaches capable of simultaneously capturing phenomena operating at different levels of spatial and temporal organization. While multi-level agent-based models (ML-ABM) offer an appropriate paradigm, their practical implementation remains confronted with methodological challenges concerning semantic coordination between heterogeneous levels, spatio-temporal synchronization, and dynamic management of transitions between scales.

Building from the study of three representative models (ESCAPE for urban risk management, COMOKIT for epidemiological modeling, and SIMPLE for environmental education) and a state-of-the-art review of socio-environmental systems, this thesis develops a unified methodological framework for multi-level modeling within the GAMA platform, transcending the limitations of existing *comodeling* and *capture/release* approaches. The originality lies in the reconciliation of tight and loose coupling paradigms through a unified language that covers the continuum of multi-level design patterns and facilitates fluid transitions between these patterns. The use of COMOKIT as a guiding thread model concretely illustrates the application of this unifying approach through the dynamic integration of micro, meso, and macro levels within a coherent epidemiological simulation.

The contribution of this thesis articulates around two components: a framework of formalized semantic constraints ensuring conceptual coherence of inter-level interactions, and an automated spatio-temporal coordination system. The implementation takes the form of a GAML language extension offering an intuitive declarative syntax that masks technical complexity while preserving necessary expressiveness.

Couplage dynamique et passage à l'échelle de modèles socio-environnementaux complexes

Résumé

Les systèmes socio-environnementaux présentent une complexité multi-échelle nécessitant des approches de modélisation capables de capturer simultanément des phénomènes opérant à différents niveaux d'organisation. Bien que les modèles à base d'agents multi-niveaux (ML-ABM) offrent un paradigme adapté, leur mise en œuvre pratique demeure confrontée à des défis méthodologiques concernant la coordination sémantique entre niveaux hétérogènes, la synchronisation spatio-temporelle, et la gestion dynamique des transitions entre échelles.

Partant de l'étude de trois modèles représentatifs (ESCAPE pour la gestion des risques urbains, COMOKIT pour la modélisation épidémiologique, et SIMPLE pour l'éducation environnementale) et d'un état de l'art des systèmes socio-environnementaux, cette thèse développe un cadre méthodologique unifié pour la modélisation multi-niveaux au sein de la plateforme GAMA, transcendant les limitations des approches existantes de *comodeling* et de *capture/release*. L'originalité réside dans la réconciliation des paradigmes de couplage fort et faible via un langage unifié permettant de couvrir le continuum des modèles de conception multi-niveaux et de faciliter les transitions fluides entre ces patterns. L'usage de COMOKIT comme modèle fil rouge illustre concrètement l'application de cette approche unificatrice à travers l'intégration dynamique des niveaux micro, méso et macro au sein d'une simulation épidémiologique cohérente.

La contribution de cette thèse s'articule autour de deux composantes : un cadre de contraintes sémantiques formalisé assurant la cohérence conceptuelle des interactions inter-niveaux, et un système de coordination spatio-temporelle automatisée. L'implémentation prend la forme d'une extension du langage GAML offrant une syntaxe déclarative intuitive qui masque la complexité technique tout en préservant l'expressivité nécessaire.

Table des matières

1
Introduction

1.1 Modélisation intégrée des systèmes socio-environnementaux

1.1.1 Vue d'ensemble

Nous vivons depuis plusieurs année dans ce que l'on appelle l'Anthropocène (Crutzen and Brauch, 2016), caractérisé par une expansion planétaire de l'influence humaine sur l'environnement. Cette situation justifie la mise en place d'approches holistiques et intégrées visant à comprendre les systèmes humains et naturels couplés, incluant leur surveillance, leur analyse et leur modélisation.

Le concept de systèmes socio-environnementaux (en anglais *Socio-Environmental Systems*, abrégé *SES*) s'inscrit dans cette perspective. Il désigne un système dynamique complexe réunissant des acteurs humains et des composantes naturelles, en évolution constante sous l'influence de pressions internes ou externes (Giupponi et al., 2022). Ces pressions peuvent être attribuées à des facteurs sociaux (e.g., changements démographiques) et écologiques (e.g., fluctuations climatiques), ou provenir d'une combinaison de ces deux catégories (e.g., le changement climatique d'origine anthropique).

Malgré son intérêt pour la description des systèmes complexes homme-nature, ce concept de SES dépasse largement les capacités actuelles de modélisation. D'une part, les SES ne se distinguent pas fondamentalement des *"systèmes complexes ordinaires"*, c'est-à-dire de systèmes dont la complexité apparente peut être imputée à un ensemble de caractéristiques telles qu'un grand nombre d'éléments potentiellement hétérogènes, des relations potentiellement non linéaires et discontinues, ainsi qu'une dynamique d'émergence de formes (phénomènes, structures, agrégats, organismes ou problèmes) à divers niveaux d'abstraction ou d'échelles, auxquels les éléments s'adaptent (Berkes and Folke, 1998).

Par exemple, dans les dynamiques de populations, une variation infime des conditions environnementales, telle qu'une hausse de 1–2 degrés celsius de la température moyenne, peut entraîner un effondrement brutal et non linéaire des stocks

halieutiques (Anh et al., 2024; Bae et al., 2016; Thinh et al., 2017). Ce type de bascule illustre la difficulté de prédire les réponses des SES à de légères perturbations.

La notion d'*émergence*, ainsi que celles de *niveaux* et d'*échelles*, occupe une place centrale dans les diverses théories de la complexité (Anderson, 1999; Gil-Quijano et al., 2010). La formation spontanée de structures ou de patterns à l'échelle macroscopique, à partir d'interactions microscopiques indépendantes, constitue en effet une signature qui distingue les systèmes simples des systèmes complexes.

Ainsi, dans de nombreux SES agricoles d'Asie du Sud-Est, des réseaux informels de coopération pour la gestion de l'eau émergent des décisions individuelles de fermiers, sans coordination centrale préalable (Diepart et al., 2019; Plews-Ogan et al., 2017). La compréhension de ces dynamiques d'auto-organisation constitue d'ailleurs un enjeu majeur pour le développement d'outils participatifs d'aide à la décision, comme l'illustrent Biré et al. (2025) dans leur approche de modélisation collaborative appliquée aux systèmes d'irrigation vietnamiens.

En général, les *échelles* mobilisées dans l'étude des systèmes complexes relèvent de trois dimensions quantifiables : l'espace, le temps et la taille (Gil-Quijano et al., 2010). Ces dimensions permettent de définir et de situer les *niveaux* d'observation, de description ou d'analyse d'un système. Il est ainsi courant de recourir à une distinction micro, méso et macro, inspirée de la vision hiérarchique des systèmes et sous-systèmes popularisée par Herbert Simon dans son article fondateur « The Architecture of Complexity » (Simon, 1991). Suivant cette perspective, la micro-échelle est constituée par un grand nombre d'éléments fondamentaux indivisibles, engagés dans des interactions locales à court terme. À mesure que l'on s'élève dans l'échelle d'observation, les dimensions spatiales et temporelles s'élargissent tandis que le nombre d'éléments diminue. Chaque niveau, à l'exception du niveau le plus large, s'insère dans un niveau supérieur qui le contraint spatialement et temporellement.

Pour rendre cette distinction plus parlante dans le cas des SES, prenons l'exemple d'une gestion forestière tropicale : la micro-échelle représente les arbres individuels, la méso-échelle une parcelle forestière (quelques hectares) et la macro-échelle l'ensemble du massif forestier (Brockelman et al., 2011; Khiewbanyang et al., 2017; Sangsupan et al., 2021; Tiansawat et al., 2022). Cette hiérarchie spatiale permet de formuler des questions qui relient la dynamique des individus (croissance, mortalité des arbres) aux patrons de succession forestière (méso), puis aux services écosystémiques régionaux (macro).

Ces niveaux peuvent être explicites dans la description du système, soit parce qu'ils ont une existence reconnue dans la réalité (e.g., un organisme, une structure physique), soit parce qu'ils correspondent à des échelles auxquelles les observateurs souhaitent décrire ou comprendre le système. De même, le passage entre ces niveaux peut être décrit explicitement dans la description du système (e.g., sous la forme d'une fonction d'agrégation ou de désagrégation entre les niveaux), ou il peut être considéré comme émergent. Ici, l'*"émergence"* fait référence à la manière dont des propriétés structurelles ou fonctionnelles peuvent spontanément émerger, de manière non planifiée, à un niveau donné à partir de l'auto-organisation des éléments identifiés à un niveau plus fin (Serugendo et al., 2006), cette notion d'auto-organisation se référant à un processus ascendant (bottom-up) dans lequel un système modifie son organisation interne pour s'adapter aux changements, tout à la fois, de ses objectifs et de son environnement, sans intervention externe ou induit par un comportement descendant (top-down).

Toutes ces caractéristiques se traduisent par une réelle difficulté à produire une description analytique et déterministe des comportements des systèmes complexes, car les modèles de ces systèmes impliquent de pouvoir représenter des comportements et des dynamiques non linéaires, mal décrites ou chaotiques opérant simultanément à plusieurs niveaux d'abstraction.

1.1.2 Objectifs : aide à la décision, compréhension, sensibilisation

La modélisation des systèmes socio-environnementaux (SES) vise à répondre à des enjeux complexes où les interactions entre dynamiques humaines et naturelles produisent des effets émergents difficilement prévisibles. Dans ce contexte, les modèles jouent un rôle fondamental en fournissant des outils permettant d'analyser ces dynamiques sous différents angles. Trois objectifs peuvent être identifiés : l'aide à la décision, l'amélioration de la compréhension des systèmes, et la sensibilisation des acteurs concernés. Ces objectifs s'inscrivent dans le cadre des huit défis de la modélisation des SES identifiés précédemment.

L'un des rôles majeurs des modèles SES est de soutenir la prise de décision, par un public expert ou non, dans des contextes où l'incertitude est forte et où de multiples parties prenantes sont impliquées. En simulant divers scénarios et en évaluant les conséquences de différentes stratégies, les modèles permettent d'anticiper les effets possibles des politiques publiques, des interventions environnementales ou des dynamiques socio-économiques. Par exemple, dans la gestion des catastrophes naturelles, des approches variées telles que la dynamique des systèmes (Forrester, 1997), les réseaux bayésiens (Aguilera et al., 2011) ou des modèles hybrides combinant simulation et optimisation (Kelly et al., 2013) peuvent être mobilisées pour analyser les comportements des individus et des infrastructures. Ainsi, ces modèles permettent d'évaluer l'impact de différentes politiques publiques ou la résilience des écosystèmes.

Au-delà de son rôle opérationnel, la modélisation des SES contribue à l'avancement des connaissances scientifiques en permettant d'explorer des mécanismes sous-jacents aux dynamiques observées. Les modèles offrent un cadre expérimental *in-silico* dans lequel il est possible de tester des hypothèses sur les interactions entre acteurs, les rétroactions systémiques ou l'émergence de comportements collectifs. Par exemple, dans l'étude des épidémies, l'utilisation de modèles compartimentaux tels que *Susceptible-Infected-Recovered (SIR)* (Kermack and McKendrick, 1927) et leurs extensions (Keeling and Rohani, 2008) permet d'analyser la propagation des maladies et d'évaluer l'efficacité des mesures de contrôle. De même, en modélisation urbaine, l'analyse des interactions spatiales permet de comprendre les dynamiques d'urbanisation et leur impact sur les systèmes socio-économiques (Batty, 2007) et environnementaux (Brugière et al., 2019).

Enfin, la modélisation des SES joue un rôle essentiel dans la sensibilisation des acteurs aux enjeux de durabilité et dans la communication des résultats scientifiques auprès du grand public. En proposant des outils interactifs et des visualisations dynamiques, elle permet de rendre accessibles des concepts complexes et d'encourager une appropriation des connaissances par un large éventail d'usagers, allant des décideurs aux citoyens. Par exemple, l'utilisation de jeux sérieux (Barreteau et al., 2013; Biré et al., 2025; De la Torre et al., 2021) et de plateformes interactives (Brugière et al., 2019) permet d'explorer les impacts des décisions individuelles et collectives dans le cadre du changement climatique ou de la gestion des ressources. En

impliquant les parties prenantes dans l'exploration des scénarios, ces modèles facilitent la compréhension des systèmes complexes dans lesquels nous vivons et permet une compréhension simplifiée des interdépendances et des effets de long terme sur ces SES (Janssen and Ostrom, 2006).

1.1.3 Illustration dans différents projets

Pour illustrer ce point, nous présentons ci-dessous trois exemples concrets de modèles à base d'agents (en anglais *Agent-Based Modeling*, abrégé *ABM*) appliqués à des contextes variés, mettant en évidence leur pertinence pour l'analyse des interactions complexes au sein des systèmes socio-environnementaux.

1.1.3.a ESCAPE

Le projet *ESCAPE* (Daudé et al., 2019) permet de simuler des catastrophes naturelles en milieu urbain, principalement en France et au Vietnam, afin d'évaluer et d'améliorer les stratégies d'évacuation et la planification des réponses aux risques naturels et technologiques.

ESCAPE permet de simuler et d'optimiser des stratégies d'évacuation dans un cadre expérimental sans nécessiter de tests en conditions réelles. Il modélise de manière réaliste les évacuations urbaines en intégrant la dynamique des déplacements (piétons, véhicules, transports en commun), les perceptions des individus en situation de crise, les interactions entre populations et services de gestion de crise, ainsi que les effets systémiques tels que les embouteillages ou les mouvements de panique. Par exemple, ESCAPE a été appliqué à l'évacuation de Rouen (France) face à un risque technologique, et à celle de Saumur lors d'une crue soudaine, démontrant sa flexibilité pour s'adapter à différents contextes de risques et configurations urbaines.

Dans une étude portant sur l'évacuation d'un quartier de Hanoï (Vietnam) menacé par la rupture du barrage de Hoa Binh, ESCAPE a été couplé avec le modèle hydraulique HEC-RAS (Bruner, 2008) pour simuler la propagation des inondations et leurs impacts sur les déplacements de la population (Chapuis et al., 2019a) illustré dans la figure 1.1. Ce couplage affine les résultats en intégrant la dynamique réaliste de la montée des eaux et en ajustant les stratégies d'évacuation en fonction de l'évolution du phénomène. Il améliore ainsi la prise en compte des contraintes physiques et la réactivité des individus face aux événements.

Cependant, la combinaison de ces modèles a soulevé plusieurs défis opérationnels, dont, notamment, la nécessité d'assurer une compatibilité entre les formats de données et les échelles temporelles des simulations. Ces difficultés rencontrées dans ESCAPE illustrent directement les enjeux de l'intégration de modèles hétérogènes avec, tout á la fois, la synchronisation entre différents niveaux d'abstraction et la gestion des échelles spatiales et temporelles, comme analysé dans la section 1.1.2. En effet, le modèle hydraulique HEC-RAS et la simulation d'évacuation d'ESCAPE fonctionnent à des résolutions spatiales et temporelles différentes, ce qui implique des ajustements pour garantir une intégration exacte des résultats. De plus, la modélisation des comportements humains en situation de crise doit être synchronisée avec la dynamique physique de la montée des eaux, ce qui ajoute une complexité supplémentaire à l'articulation des décisions d'évacuation avec l'évolution du phénomène.

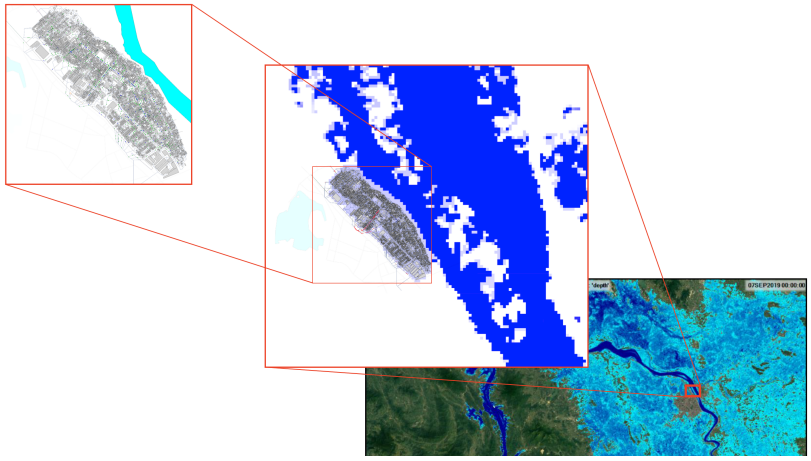

FIGURE 1.1 : La simulation ESCAPE (sur la gauche), la simulation HEC-RAS (sur la droite), et le modèle intégré (au centre).

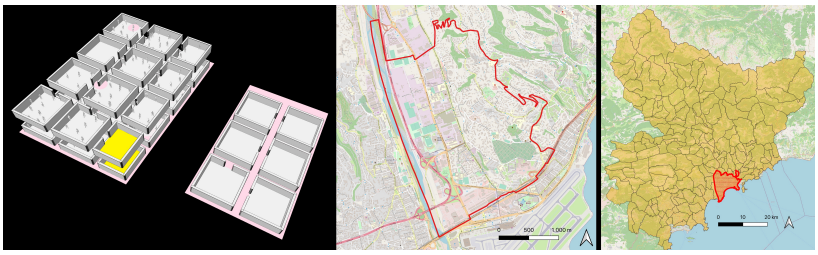

FIGURE 1.2 : COMOKIT Micro (sur la gauche), COMOKIT Meso (au centre), COMOKIT Macro (sur la droite). Ces trois modèles sont développés en utilisant le framework COMOKIT v2, mais ne communiquent pas entre eux.

1.1.3.b COMOKIT

Le projet *COMOKIT* (COVID-19 Modeling Kit) (Gaudou et al., 2020; Taillandier et al., 2024b) est un environnement logiciel de modélisation à base d'agents, conçu pour analyser et comparer l'impact des politiques de santé publique face à la pandémie de COVID-19. Développé initialement pour répondre aux besoins du gouvernement vietnamien, il vise à assister la prise de décision en simulant diverses interventions à l'échelle d'une petite ville.

COMOKIT repose sur une architecture intégrant quatre sous-modèles qui interagissent de façon dynamique : (1) un modèle de transmission de la maladie de personne à personne et via l'environnement, (2) un modèle d'évolution du statut épidémiologique individuel, (3) un modèle de mobilité individuelle et quotidienne à l'échelle de l'heure et du bâtiment et (4) un modèle permettant de représenter les impacts de la combinaison de plusieurs politiques d'intervention (Chapuis et al., 2021a). Cette combinaison permet d'examiner l'effet combiné des comportements individuels et des politiques sanitaires sur la propagation de l'épidémie, et d'évaluer des scénarios alternatifs pour en optimiser l'impact.

Une évolution majeure introduite dans la seconde version de COMOKIT (Taillandier et al., 2024b) est l'approche multi-échelle, permettant de modéliser et simuler les politiques de contrôle épidémique à différentes échelles : au niveau *micro*, à l'échelle d'un bâtiment, au niveau *meso*, à l'échelle d'un village, et au niveau *macro*, à l'échelle d'une région (représenté dans la figure 1.2).

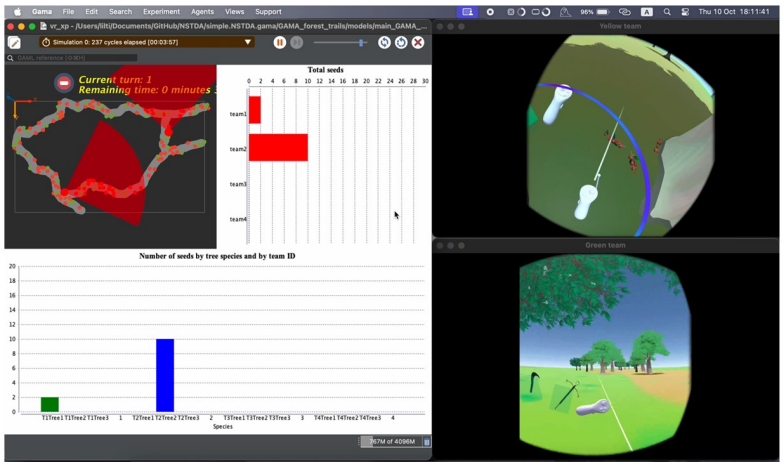

FIGURE 1.3 : Exemple d'implémentation du projet SIMPLE dans le cas d'étude BiodeVRestorer. Modèle scientifique (sur la gauche), usage en réalité virtuelle (sur la droite).

Cependant, bien que cette nouvelle version du *cadre logiciel* (traduit en *framework* dans la suite de ce texte) permette son usage à différentes échelles, celles-ci ne sont pas intégrées dans un même modèle et nécessitent d'implémenter différents modèles pour différentes échelles. Cette nécessité d'implémenter différents modèles pour différentes échelles met en lumière le défi du couplage multi-niveaux et la complexité de capturer les interactions entre ces échelles. Ce constat illustre une limite commune aux SES, où la modélisation des dynamiques à l'échelle micro (individus/bâtiments), méso (village) et macro (région) exige une coordination difficile entre méthodologies hétérogènes (Simon, 1991). Par exemple, les modèles microscopiques de mobilité individuelle (comme COMOKIT Micro) nécessitent des données spatiales et temporelles précises, tandis que les modèles macroscopiques (COMOKIT Macro) peuvent opérer sur des agrégats statistiques, ce qui rend leur synchronisation complexe.

1.1.3.c SIMPLE

Le projet SIMPLE (Simulation and Immersive Learning for Participatory Environmental education) (Drogoul et al., 2025; Taillandier et al., 2024a) permet de créer des outils interactifs et immersifs dédiés à l'éducation au développement durable en couplant des modèles scientifiques et la réalité virtuelle (VR) (Drogoul et al., 2025). Il vise à intégrer des simulations basées sur des modèles scientifiques dans des environnements de réalité virtuelle afin de sensibiliser les enfants aux enjeux environnementaux et sociaux tout en dépassant les limitations des approches traditionnelles en rendant l'apprentissage plus engageant et accessible.

Il permet, par exemple, d'intégrer divers sous-modèles adaptés aux problématiques socio-environnementales locales. Ceux-ci peuvent concerner la modélisation écosystémique, qui représente les interactions entre les espèces et leur habitat, ou encore la gestion des ressources naturelles, qui étudie les stratégies d'adaptation des populations face aux crises écologiques. En figure 1.3, un cas d'application illustre cette approche : un jeu immersif dédié à la préservation de la biodiversité forestière, dans lequel les joueurs collectent des graines et restaurent une forêt virtuelle selon des principes écologiques réalistes (Aumpuchin et al., 2024).

Cependant, cette intégration entre modèle et VR pose plusieurs défis techniques et méthodologiques. Notamment, la nécessité d'assurer une compatibilité entre les

modèles scientifiques et les environnements immersifs, et la limitation des modèles explorables à un unique niveau (généralement, à un niveau *micro* : à échelle humaine). Cette limitation souligne l'importance de développer des approches permettant de naviguer dans et d'intégrer différents niveaux d'abstraction, un des enjeux centraux de la modélisation des SES. Par exemple, dans le cas d'étude BiodeVRestorer (figure 1.3), si le modèle actuel se concentre sur l'action individuelle (collecte de graines, restauration de parcelles), une approche multi-niveaux permettrait de :

— Lier le micro au macro : montrer comment les actions humaines individuelles (ex. plantation d'une plante) influencent l'équilibre écologique à l'échelle du bassin versant (niveau *macro*), via des indicateurs comme la biodiversité ou la productivité de l'écosystème.
— Inclure des dynamiques méso : intégrer des facteurs sociaux (ex. coopération entre joueurs) ou environnementaux (ex. variations climatiques saisonnières) qui agissent à l'échelle d'une communauté ou d'un territoire (niveau *méso*).
— Permettre un zoom hiérarchique : ainsi, un joueur pourrait visualiser en temps réel comment sa décision (micro) s'inscrit dans un scénario à long terme (macro), renforçant la compréhension des rétroactions systémiques et l'exploration des dynamiques multi-niveaux.

Cette navigation entre niveaux serait essentielle pour capturer la complexité des SES, où les dynamiques individuelles, collectives et environnementales interagissent de manière non linéaire, et où la réalité virtuelle pourrait servir de pont entre abstraction théorique et expérience sensorielle.

1.2 Enjeux de recherche

Les défis de l'intégration de modèles hétérogènes et de la gestion des échelles spatiale et temporelle prennent une dimension particulière dans la modélisation des Systèmes Socio-Environnementaux (SES), en raison de la complexité inhérente à la représentation des interactions homme-nature et des différents niveaux d'abstraction. Ces défis émergent directement des caractéristiques propres aux SES, où les dynamiques humaines (sociales, économiques) et naturelles (biophysiques, écologiques) interagissent de manière non linéaire et émergent des rétroactions à multiples échelles (Elsawah et al., 2020; Giupponi et al., 2022).

Il existe des caractéristiques spécifiques aux systèmes socio-environnementaux (SES), liées, entre autres, à l'étude des problèmes complexes qui émergent des interactions entre les systèmes humains (c'est-à-dire sociaux, économiques) et naturels (c'est-à-dire biophysiques, écologiques, environnementaux). Bien qu'il existe de nombreuses façons de décrire les SES (notamment dans la littérature anthropologique ou sociologique), les modèles dynamiques apparaissent comme des outils indispensables pour les comprendre et les gérer, car ils permettent d'étudier le comportement de ces systèmes non seulement dans des conditions passées et présentes, mais aussi futures, grâce à l'analyse de scénarios ou à l'exploration virtuelle de chemins possibles. Avec l'avènement de nouvelles techniques et une puissance de calcul nouvelle d'une part, et les défis croissants en matière de durabilité d'autre part, la modélisation des SES est utilisée pour atteindre plusieurs objectifs, tels que l'aide à la recherche et à la prise de décision, ou encore la sensibilisation et la promotion de l'éducation et de la communication sur des questions sensibles (Elsawah et al., 2020). Cependant, pour

être utiles, ces modèles doivent pouvoir gérer, de manière intégrée, la complexité de ces SES, souvent caractérisés par des boucles de rétroaction complexes dans les interactions humain-nature et humain-société (Drogoul et al., 2016; Gain et al., 2021), à différents niveaux d'abstraction et avec différents niveaux de détails (Lippe et al., 2019). Plus généralement, ils doivent répondre aux huit principaux défis de la modélisation des SES énumérés par Elsawah et al. (2020) :

1. Faire le lien entre les épistémologies des différentes disciplines ;
2. Traiter de manière intégrée l'incertitude des modèles ;
3. Combiner des méthodes et des sources de données qualitatives et quantitatives ;
4. Gérer les échelles et les changements d'échelle ;
5. Capturer les changements systémiques dans les SES ;
6. Intégrer la dimension humaine ;
7. Favoriser l'adoption des modèles de SES ;
8. Exploiter de nouveaux types et sources de données ;

En lisant l'article et les arguments derrière chacun de ces *défis*, il est évident que ces défis ne sont pas unique aux SES, mais qu'ils y sont plus exacerbés dans ce domaine que dans d'autres : par exemple, l'incertitude structurelle (défi 2) s'amplifie lorsque les données écologiques et sociales sont lacunaires, tandis que l'adoption des modèles par les décideurs (défi 7) se heurte à la difficulté de communiquer des processus mêlant comportements humains et dynamiques biophysiques. Néanmoins, parmi ces huit défis, trois apparaissent comme le cœur des problématiques propres aux SES : la gestion des échelles et des changements d'échelle (défi 4), la capture des transformations systémiques (défi 5) et l'intégration explicite de la dimension humaine (défi 6). Ce triptyque vise à relier des mécanismes locaux à des dynamiques globales, à représenter des rétroactions non linéaires et à traduire les comportements individuels en effets collectifs observables, conditions indispensables pour appréhender la complexité multi-niveaux des interactions homme-nature.

1.2.1 Intégration de modèles hétérogènes

L'intégration de modèles hétérogènes (Duboz, 2004) représente un enjeu majeur dans la modélisation des systèmes socio-environnementaux (SES) et s'inscrit dans ces huit défis identifiés plus tôt, notamment la nécessité de combiner des méthodes et des sources de données qualitatives et quantitatives, ainsi que la capture des changements systémiques au sein des SES (points 3 et 5). La diversité des approches et des niveaux de modélisation (aussi bien des niveaux spatio-temporels que de niveaux de représentations plus ou moins abstraits) complexifie l'interopérabilité des modèles, rendant essentielle l'utilisation de cadres méthodologiques rigoureux.

Parmi les méthodologies proposées pour structurer et documenter ces modèles complexes, le protocole *Overview, Design concepts, and Details (ODD)* de Grimm et al. (2010) s'est imposé comme une référence incontournable. Initialement conçu pour la modélisation à base d'agents (*ABM*), il a évolué pour mieux prendre en compte les défis posés par l'intégration de modèles hétérogènes et intégrés. La mise à jour la plus récente de ce protocole insiste sur l'importance d'une documentation claire des modèles multi-paradigmes et des méthodes permettant d'assurer leur interopérabilité (Grimm et al., 2020). Cette évolution témoigne de la reconnaissance croissante du besoin d'intégrer des approches complémentaires pour mieux représenter la complexité

des systèmes socio-environnementaux. Ce développement s'inscrit également dans la mouvance *Keep It Descriptive Stupid (KIDS)* (Edmonds and Moss, 2004), qui cherche à décrire les SES de la manière la plus détaillée possible pour avoir une représentation du système la plus précise possible.

L'intérêt principal de l'intégration de modèles hétérogènes est, donc, de combiner différentes perspectives et méthodes de simulation afin d'obtenir une représentation plus complète et réaliste des dynamiques étudiées. Par exemple, les modèles basés sur les agents sont particulièrement adaptés pour simuler des interactions locales et des prises de décision individuelles, tandis que des modèles statistiques offrent une vision plus globale et agrégée des dynamiques systémiques. Intégrer ces deux visions permet d'avoir une vision et une compréhension plus précise et complète du système modélisé.

Le projet ESCAPE (présenté en 1.1.3.a) illustre bien ces défis en combinant un modèle de simulation multi-agents pour l'évacuation urbaine avec des modèles hydrodynamiques pour la propagation des inondations. La combinaison de ces modèles est essentielle pour correctement modéliser et prévoir les réactions individuelles des populations face aux catastrophes naturelles. De manière similaire, le projet COMOKIT (présenté en 1.1.3.b) intègre des modèles épidémiologiques à compartiment et des simulations multi-agents afin de correctement représenter l'évolution d'épidémies et l'analyse de l'impact des politiques de santé publique sur cette population.

1.2.2 Gestion des échelles spatiale et temporelle

Un autre enjeu de la modélisation des systèmes socio-environnementaux (SES), en continuité avec l'intégration des modèles hétérogènes, vu précédemment, concerne la gestion des échelles spatiales et temporelles. Les processus dynamiques entre les différents sous-modèles qui structurent ces systèmes opèrent, généralement, à des pas de temps (cycle d'exécution du modèle) et des espaces variés, nécessitant une articulation rigoureuse pour garantir la cohérence des simulations et assurer une représentation fidèle des interactions multi-échelles (Brunsdon and Singleton, 2015; Grimm et al., 2020; Liu et al., 2007).

La granularité des données influence directement la fidélité des interactions modélisées. Une résolution trop fine accroît la charge computationnelle et peut nuire à l'interprétation des résultats, tandis qu'une résolution trop agrégée peut masquer des dynamiques locales cruciales. De même, les échelles temporelles varient entre des processus se déroulant sur des décennies, comme l'évolution des écosystèmes, et des événements brefs tels que la transmission d'un virus d'un individu à un autre (Karsakov et al., 2016; Wu, 2004).

Le projet ESCAPE (présenté en 1.1.3.a) illustre ces enjeux en intégrant des approches multi-échelles. En effet, il associe un modèle hydrodynamique simulant la propagation des inondations à grande échelle (à une échelle macro) avec un modèle à base d'agents capturant les décisions individuelles d'évacuation (à une échelle micro). Ce couplage permet d'évaluer l'efficacité des stratégies en prenant en compte la dynamique rapide des flux d'eau et la réponse plus progressive des populations. SIMPLE (présenté en 1.1.3.c) adopte, également, une approche similaire en calculant la simulation des impacts environnementaux sur de vastes territoires avec, comme données d'entrées, des simulations et des interactions locales immersives en réalité virtuelle.

1.2.3 Importance du couplage multi-niveaux

Découlant des points précédents, un défi majeur réside dans le développement d'une méthode opérationnelle permettant d'assurer une intégration cohérente entre ces modèles. Cette approche est essentielle pour connecter des sous-modèles hétérogènes et assurer une gestion cohérente des échelles spatiales et temporelles. En intégrant ces dimensions, il devient possible de capturer les rétroactions complexes entre dynamiques locales et globales tout en maintenant une continuité des interactions au sein du système modélisé (Bousquet and Le Page, 2004; Gil-Quijano et al., 2010).

Le couplage de modèles hétérogènes présente, en soi, plusieurs défis. D'une part, ces modèles reposent souvent sur des formalismes différents (équations différentielles, modèles multi-agents, approches statistiques), rendant leur intégration complexe (Nguyen, 2010). La conversion des variables d'un modèle à l'autre nécessite des mécanismes adaptés pour éviter la perte d'information ou l'altération des dynamiques simulées. De plus, la diversité des structures de données et des hypothèses sous-jacentes peut engendrer des incohérences, rendant indispensable une description de couplage rigoureuse pour assurer une compatibilité effective (Mercure et al., 2016).

D'autre part, ces modèles fonctionnent souvent à des échelles temporelles différentes, nécessitant un travail de synchronisation entre ceux-ci. Certains processus doivent être mis à jour en continu, tandis que d'autres évoluent à des intervalles plus espacés. Une mauvaise gestion de ces temporalités peut entraîner des décalages dans la simulation, compromettant la validité des résultats. Il est crucial d'adopter une description de couplage précise des temporalités de chaque sous-modèle, garantissant ainsi une synchronisation pertinente entre ces échelles et permettant l'exploration de dynamiques locales, globales et mixtes (Lagadeuc and Chenorkian, 2009).

L'élaboration d'un cadre de couplage clair et précis est indispensable pour structurer ces interactions. Un format bien défini permet d'éviter toute ambiguïté quant aux modalités d'échange entre modèles et facilite leur interopérabilité (Verburg et al., 2013). Une documentation rigoureuse, combinée à des interfaces de communication standardisées, contribue également à améliorer la reproductibilité des résultats et l'adoption des solutions de couplage par la communauté scientifique.

Le projet ESCAPE illustre bien ces enjeux. ESCAPE associe un modèle hydrodynamique simulant la propagation des inondations à grande échelle avec un modèle multi-agents représentant les décisions individuelles d'évacuation. Ce couplage a nécessité une description précise des espaces (le modèle agent étant un sous-espace du modèle hydrologique) et des temporalités (le modèle agent se calculant chaque seconde et le modèle hydrologique chaque minute). Également, un travail a dû être réalisé pour transmettre les données du modèle hydrologique vers le modèle agent, qui n'offrait aucun moyen de communication par défaut.

1.3 Objectifs et positionnement de la thèse

1.3.1 Problématique générale et objectifs

L'analyse des enjeux de recherche présentés précédemment, ainsi que l'étude des projets ESCAPE, COMOKIT et SIMPLE, révèlent une problématique centrale : bien que la modélisation multi-niveaux représente une direction de recherche prometteuse pour appréhender la complexité des systèmes socio-environnementaux, sa mise en

œuvre pratique demeure confrontée à des défis méthodologiques et techniques considérables. Ces défis concernent notamment la coordination sémantique entre niveaux d'abstraction hétérogènes, la synchronisation temporelle et spatiale des processus multi-échelles, et la gestion dynamique des transitions entre échelles d'observation.

L'objectif général de cette thèse est de faciliter la conception, la coordination et l'évolution de modèles multi-niveaux au sein de la plateforme GAMA (Taillandier et al., 2019). Cette recherche s'inscrit dans une volonté de rendre ces modèles plus accessibles, plus souples et plus cohérents, tout en maintenant une expressivité suffisante pour représenter des dynamiques complexes à différentes échelles.

Plus concrètement, cette recherche vise à développer un langage de coordination sémantique permettant de : (1) décrire de manière unifiée les relations de couplage entre niveaux hétérogènes, en tenant compte de leurs dimensions spatiales, temporelles et de leurs objectifs ; (2) simplifier l'intégration et la transition entre différents patterns de modélisation multi-niveaux (Brugière et al., 2022) ; et (3) permettre une évolution fluide de l'architecture multi-niveaux d'un modèle au cours de la simulation, sans recourir à des reconstructions lourdes ou des réécritures profondes du code.

1.3.2 Positionnement par rapport aux travaux existants

Les travaux présentés dans cette thèse s'inscrivent dans la continuité des approches de modélisation multi-niveaux déjà mises en œuvre dans la plateforme GAMA, notamment à travers les approches de *comodeling* (Huynh, 2016) et de *capture/release* (Vo, 2012). À l'image de ces contributions antérieures, les travaux de cette thèse prendront la forme d'une extension du langage GAML, directement intégrée à la plateforme.

Cependant, cette approche se distingue fondamentalement des solutions existantes par sa vocation unificatrice. Là où le *comodeling* et le *capture/release* constituent des outils efficaces mais cloisonnés, correspondant respectivement aux *design patterns* (ou *patron de conception* en français, mais l'anglicisme sera préféré dans ce texte) de *Collaboration* et de *Russian Dolls*, cette thèse propose de les réinterpréter non plus comme des mécanismes séparés, mais comme des fondations complémentaires d'un cadre conceptuel unifié.

L'originalité de cette proposition réside dans sa capacité à réconcilier les paradigmes apparemment contradictoires du couplage fort et du couplage faible, en proposant un langage unifié dont la sémantique opérationnelle permet de couvrir l'ensemble du continuum des design patterns multi-niveaux et de faciliter les transitions fluides entre ces patterns. Contrairement aux approches actuelles qui obligent les modélisateurs à choisir à priori un paradigme de couplage, ce cadre permettra d'adapter dynamiquement l'architecture multi-niveaux aux besoins évolant avec la construction du modèle intégré.

1.3.3 Approche adoptée

Pour réaliser ces objectifs, les travaux s'appuieront sur une approche méthodologique structurée en plusieurs phases complémentaires.

Dans un premier temps, nous développerons les fondements théoriques du cadre de coordination proposé, en nous appuyant sur des paradigmes pré-existants dans la

plateforme GAMA tout en les enrichissant d'une couche sémantique formelle inspirée des standards de couplage interopérable, tels que le *High-Level Architecture* (Kuhl et al., 1999). Cette phase théorique permettra de formaliser les contraintes sémantiques nécessaires à la coordination cohérente de modèles et de définir les mécanismes de transition dynamique entre échelles.

Dans un second temps, nous procéderons à l'implémentation technique du cadre proposé sous la forme d'une extension du langage GAML. Cette implémentation exploitera l'architecture modulaire de GAMA pour intégrer de manière transparente les nouveaux mécanismes de coordination, garantissant ainsi la compatibilité avec les modèles existants et la facilité d'adoption par la communauté des modélisateurs.

Enfin, nous validerons empiriquement ce nouvel outil à travers l'application à des cas d'étude représentatifs des défis rencontrés dans la modélisation des systèmes socio-environnementaux tel que le projet COMOKIT (Taillandier et al., 2024b). Cette validation portera à la fois sur les aspects fonctionnels (capacité à représenter fidèlement des dynamiques multi-niveaux complexes) et non-fonctionnels (accessibilité pour les modélisateurs, maintenabilité du code).

L'ambition de cette thèse est donc double : proposer un cadre théorique robuste pour la coordination multi-niveaux, et offrir un outillage pratique facilitant l'usage de ces approches dans des contextes appliqués, notamment dans les domaines socio-environnementaux où la prise en compte simultanée de multiples échelles d'observation constitue un enjeu méthodologique central.

2

État de l'art

2.1 Introduction

2.1.1 Modèles à base d'agent (ABM)

Comme identifié dans la Section 1.2, la modélisation des systèmes socio-écologiques (SES) présente de grands défis en raison, par exemple, du besoin d'intégrer des informations interdisciplinaires ou des sources de données variées. Néanmoins, les SES possèdent des caractéristiques spécifiques, notamment l'inclusion d'éléments humains et la représentation des systèmes à différents niveaux d'abstraction, qui rendent ces systèmes particulièrement difficiles à modéliser avec les techniques traditionnelles. Pour surmonter ces obstacles, diverses approches ont été développées (Kelly et al., 2013), parmi lesquelles la modélisation à base d'agents (en anglais *Agent-Based Modeling*, abrégé *ABM*) s'est distinguée par sa capacité à représenter fidèlement les comportements humains individuels et sociaux (Adam et al., 2017; Railsback and Grimm, 2019).

Les modèles à base d'agents offrent, en effet, des opportunités claires pour la modélisation des SES et aident à répondre à certaines des principales faiblesses des diverses catégories de modèles décrites dans Courdier (2003); Giupponi et al. (2022); Schulze et al. (2017), en particulier en ce qui concerne la simulation du comportement individuel et social et leur capacité à fournir une représentation descriptive et générative du système simulé selon les quatre dimensions brièvement décrites ci-dessous (basées sur le travail de Giupponi et al. (2022)).

La première dimension est *l'hétérogénéité* . En général, les ABM consistent en des simulations dynamiques détaillées dans lesquelles de nombreux agents humains, naturels et hétérogènes interagissent : cela évite une représentation grossière, moyenne et donc irréaliste ou incorrecte des composantes du système. Les agents humains peuvent varier dans leurs caractéristiques démographiques, leur localisation, leurs dotations, leurs capacités individuelles, leur vision du monde, leurs attitudes et leur comportement. Les agents naturels peuvent également varier dans leurs attributs spatiaux et temporels.

La deuxième dimension est la *complexité individuelle*. Comparés aux agents naturels, les agents humains sont plus complexes à simuler, car ils effectuent des processus délibératifs et prennent des décisions individuelles autonomes (Groeneveld et al., 2017; Schwarz et al., 2020). La complexité comportementale découle des modèles mentaux des agents (Schlüter et al., 2017; Ta et al., 2017b; Taillandier et al., 2021) ou plus communément de leurs *"architectures"*, qui incluent leurs capacités cognitives, de raisonnement et d'apprentissage, basées sur l'abondance de théories des sciences sociales sur la manière dont les agents humains se comportent dans divers contextes. L'ABM a le potentiel de permettre l'exploration de cet ensemble de théories de la prise de décision, y compris la capacité des agents à apprendre de leurs expériences passées (Vo, 2012), ce qui est extrêmement important pour les simulations à long terme de l'évolution des SES.

La troisième dimension porte sur les *interactions*, et en particulier **les *interactions sociales***. Non seulement les agents humains sont délibératifs, mais ils sont aussi sociaux : ils communiquent avec d'autres agents et leur comportement découle d'interactions dans de multiples contextes avec d'autres agents humains et non humains et l'environnement (Gotts et al., 2019). Cet aspect est fondamental pour capturer des dynamiques telles que le regroupement, l'imitation, l'apprentissage et les processus de diffusion, comme les phénomènes de contagion émotionnelle (Ta et al., 2017a). C'est une caractéristique cruciale pour la modélisation des SES dans la mesure où les interactions des agents, et en particulier les relations informelles et les dynamiques d'opinion, peuvent façonner des schémas collectifs de comportement.

La quatrième dimension traite de la *représentation* des organisations et de l'*émergence* de structures organisationnelles. Les agents humains sont délibératifs et sociaux, mais ils sont aussi organisationnels et peuvent se structurer en diverses formes, hiérarchiques ou non. En même temps, les normes et institutions, qu'elles soient fixes ou émergentes, peuvent amener les individus à agir différemment de leurs choix individuels, ce qui peut être crucial pour comprendre l'apparition ou la disparition de certaines dynamiques (Wall, 2016).

Le potentiel de l'ABM est donc clair pour représenter les dimensions sociales, d'hétérogénéité et d'interaction des SES, et est, en effet, l'approche privilégiée par une majorité des chercheurs confrontés à ce besoin (Giupponi et al., 2022), mais des défis méthodologiques considérables subsistent, en particulier, liés à la représentation explicite des échelles et des niveaux d'abstraction, absents du méta-modèle classique de la modélisation basée sur les agents (Vo et al., 2012a).

2.1.2 ABM multi-niveaux (ML-ABM)

Nous avons vu dans les sections 1.2 et 2.1.1 que l'ABM a le potentiel d'aborder certains des défis soulevés par la modélisation des SES, en particulier les trois points énumérés ci-dessous (extrait de Giupponi et al. (2022)) :

4. Gérer les échelles et les changements d'échelle ;
5. Capturer les changements systémiques dans les SES ;
6. Intégrer la dimension humaine ;

La manière dont ces points sont abordés dans la littérature sur l'ABM dépend fortement des besoins des modélisateurs et des objectifs des modèles eux-mêmes ; toutes les questions de modélisation ne nécessitent pas de représenter simultanément

plusieurs niveaux ou l'émergence de structures et de fonctions, ce qui explique l'existence de différentes architectures pour implémenter des ABM *multi-niveaux* (en anglais *Multi-Level Agent-Based Model*, abrégé *ML-ABM*). Cette multiplicité d'offres est la raison pour laquelle le vocabulaire n'est pas complètement fixé, rendant parfois difficile de trouver des correspondances directes entre les concepts utilisés en science des systèmes complexes et ceux proposés dans les différentes approches ML-ABM. Par exemple, au lieu d'utiliser des concepts d'échelles temporelles et spatiales, les concepteurs d'ABM doivent gérer des algorithmes de planification et des encapsulations pour traduire les contraintes temporelles et spatiales et les transferts entre agents représentant des niveaux ; de même, les *changements systémiques* peuvent être représentés par différents aspects : instanciation dynamique d'agents, injection de nouveau code, etc.

L'implémentation de ces niveaux et de leurs interactions pour représenter les échelles et leurs liens dépendent alors de deux aspects : d'une part, les choix des modélisateurs concernant la complexité résultante des modèles, qui dépend en fin de compte de leur utilisation : doivent-ils être suffisamment simples pour être compris par tous ? Doivent-ils présenter des propriétés émergentes uniquement trouvées dans les systèmes complexes *"réels"* ? Doivent-ils permettre différents niveaux d'exploration et d'explication ? ; d'autre part, les limitations techniques et computationnelles des langages et systèmes informatiques disponibles pour implémenter les architectures ABM utilisées. Ces deux aspects ont évolué au fil du temps. On observe à la fois une complexification des questions posées aux modèles et une augmentation considérable des capacités de calcul dédiées aux simulations. Cependant, cela ne signifie pas que les approches dites *"simples"* aient perdu leur pertinence ; elles demeurent adaptées dans la plupart des cas, ce qui explique la diversité des propositions présentées dans la suite de ce chapitre (Brugière et al., 2022). Morvan (2012) a examiné ces points et démontré qu'ils ont été appliqués dans de nombreux domaines de recherche différents tels que la recherche biomédicale, le flux humain, les sciences sociales, l'écologie, etc.

Certains travaux antérieurs ont établi des catégories de ML-ABM, tentant de fournir une grille de lecture sur les questions de modélisation précédentes. Notamment Mathieu et al. (2018), qui a extrait quatre catégories basées sur les choix de couplage des modélisateurs (suivant un arbre de décision) entre deux niveaux. Cette approche représente un outil précieux pour les modélisateurs construisant des ML-ABM, mais ne donne pas plus de détails sur le mécanisme interne global du modèle. Ce chapitre fournit une analyse complémentaire et vise à examiner les architectures, cadres et approches ML-ABM existants sous l'angle du contrôle des niveaux dans un modèle intégré.

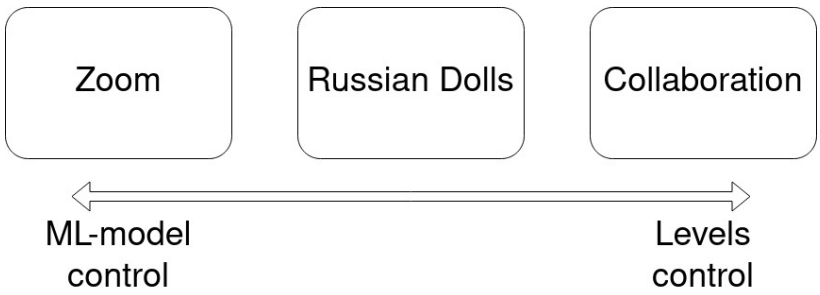

FIGURE 2.1 : Continuum du contrôle ML-ABM allant du contrôle totalement délégué au modèle intégré dans l'approche *Zoom*, ou maintenant l'autonomie des niveaux dans l'approche *Collaboration*.

Comme illustré dans la figure 2.1, l'évolution du contrôle des niveaux se place le long d'un continuum. Ce continuum va d'une rigidité complète (Zoom) sans aucune autonomie des niveaux à des niveaux partiellement autonomes, mais fonctionnant sous un contrôle rigide (Russian Dolls) à des niveaux autonomes avec un contrôle relativement flexible (Collaboration). Évidemment, ces trois jalons ne sont pas imperméables, et il existe des modèles et des travaux se plaçant tout au long de ce continuum.

Ces différences de contrôle des niveaux peuvent provenir de la conceptualisation, de la création et de l'organisation des niveaux dans une architecture multi-niveaux, qui est gérée différemment entre les différentes approches présentées ici. L'architecture *Zoom* (revue dans la partie 2.2.1), qui coordonne les niveaux en tant que modèles distincts, chacun construit avec une seule échelle temporelle et spatiale, utilise des fonctions pour permettre de passer d'un niveau à un autre. D'autres approches visent à avoir une composition plus discrète entre les niveaux et incluent chaque niveau dans un autre selon un ordre hiérarchique spatio-temporel. Cette extension peut conserver le concept de planification de l'ABM, comme avec la catégorie des *Russian Dolls* (revue dans la partie 2.2.2) qui permet de construire un modèle intégré au coût de perdre le contrôle interne des niveaux, ou, sans cette perte de contrôle, comme avec la *Collaboration* (revue dans la partie 2.2.3) qui permet un couplage faible des niveaux.

Ces différences de contrôle des niveaux créent une complexité supplémentaire dans l'exploitation de ces modèles (point 5 de Elsawah et al. (2020)). Lorsqu'un modèle commence à avoir différents niveaux (indépendamment de leur niveau de contrôle), il devient plus compliqué de capturer et d'analyser les motifs émergents du système global ou entre les niveaux décrits. Cet aspect analytique peut être nécessaire pour les modélisateurs qui doivent conserver des informations détaillées de chaque niveau et assurer l'interaction entre eux. Les modèles des *Russian Dolls* et de *Collaboration* offrent différentes approches pour ce faire. Le premier utilise des outils d'analyse similaires à ceux des ABM classiques, car cette architecture multi-niveaux reste très proche de celle de l'ABM, alors que le second utilise des outils plus proches du génie logiciel.

Cependant, certains modèles multi-niveaux n'ont pas besoin de cela pour analyser et capturer ces changements, comme dans le premier modèle multi-niveaux présenté dans Gil-Quijano et al. (2010) sur la croissance d'une tumeur cancéreuse. Au début de la simulation, le micro-niveau (à l'échelle cellulaire) est important car il permet l'émergence d'un groupe de cellules qui peut être observé et identifié comme la tumeur. Cependant, ces cellules peuvent, ensuite, être agrégées en un agent plus grand à une échelle supérieure et décider de remplacer le niveau plus détaillé (micro) par un niveau plus large (macro) pour voir l'évolution de cette tumeur sur les organes. Le micro-niveau était utile pour simuler précisément la création de la tumeur ; une fois celle-ci créée, ce niveau n'est plus utile dans le modèle et peut donc être supprimé selon la volonté du modélisateur.

Par conséquent, le choix d'un modèle dans le continuum pour construire un ML-ABM devra être fait par les modélisateurs en fonction, d'une part, de certaines contraintes de modélisation ou de limitations plus générales de développement logiciel ; et, d'autre part, comme vu précédemment, en suivant des choix de modélisation pour représenter le système. Dans le premier cas, comme dans l'exemple précédent portant sur une tumeur, les modélisateurs peuvent choisir une construction plus simple des niveaux, car l'émergence entre les niveaux n'est pas l'intérêt principal. Concernant les limitations logicielles, celles-ci peuvent, par exemple, provenir du fait qu'un modèle

est composé de sous-modèles de nature différente (ABM, EBM, etc.), qui peuvent être plus compliqués à intégrer dans une architecture de type *Russian Dolls* ; dans ce cas, il est plus simple d'utiliser l'une des autres architectures. Pareillement, le choix d'un modèle ML-ABM peut être une combinaison des deux choix de modélisation et de logiciel.

2.2 Typologie des architectures ML-ABM

Afin de mettre en avant les avantages de chaque approche ainsi que d'illustrer quelles questions peuvent être répondues par quelles typologies de couplage multi-niveaux, un modèle simple à base d'agents d'évacuation de piétons (composé d'agents `People` marchant de gauche à droite) sera utilisé tout au long de cette section pour illustrer chaque architecture.

2.2.1 Zoom

2.2.1.a Problématique rencontrée dans les SES

À la première extrémité du continuum (cf. Figure 2.1), le contrôle le plus rigide est représenté par le *design pattern* appelé "*Zoom*" : chaque niveau est un modèle indépendant, représentant un niveau d'abstraction différent du système avec différents types de modèles (pouvant être basés sur des agents, des équations, ou autres) et à n'importe quelle échelle. Dans ce design pattern, un seul niveau (c'est-à-dire un seul modèle) est traité à la fois, et les modélisateurs décrivent comment le modèle passe d'un niveau à un autre à l'aide d'une fonction de transition permettant d'agréger ou de désagréger des éléments entre les niveaux.

En d'autres termes, lors d'un *zoom avant* (ou *zoom in*), le modèle (1) appelle une fonction de transition (définie par les modélisateurs) sur le niveau actuel, (2) utilise ce résultat pour initialiser le nouveau niveau, puis (3) détruit le niveau précédent, désormais inutilisé. La cohérence du système entre les niveaux est assurée par les modélisateurs à travers la fonction de transition choisie.

Cependant, comme les niveaux ne sont pas persistants dans l'exécution du modèle, le second problème des SES, à savoir capturer les changements systémiques, est très limité (voire impossible) avec ce pattern. Cette contrainte limite son utilisation pour répondre au besoin de capture des dynamiques entre les échelles du modèle.

Dans de nombreux SES, l'approche Zoom sert à *alterner* entre des vues macro, pour saisir des tendances globales (densités de population, changements d'occupation du sol), et des vues micro, pour représenter la prise de décision individuelle ou la dynamique fine des interactions.

2.2.1.b Définition

Pour le décrire de manière plus formelle : le design pattern *Zoom* correspond à une représentation où les niveaux sont explicitement décrits (généralement sous forme d'agents) dans un modèle avec une intégration destructrice. Les comportements des niveaux, leurs échelles, les fonctions de transfert, etc., sont également explicitement décrits. Cependant, un seul niveau est actif à la fois dans les simulations. Cela garantit qu'il n'y ait pas de compétition ou de conflit entre les niveaux de représentation

pendant l'exécution du modèle. Chaque niveau dispose d'une fonction de transfert d'agrégation et de désagrégation explicite, qui décrit comment passer d'un niveau à l'autre. Ainsi, l'utilisation de cette fonction engendre la destruction du niveau source actuellement utilisé pour créer le niveau suivant.

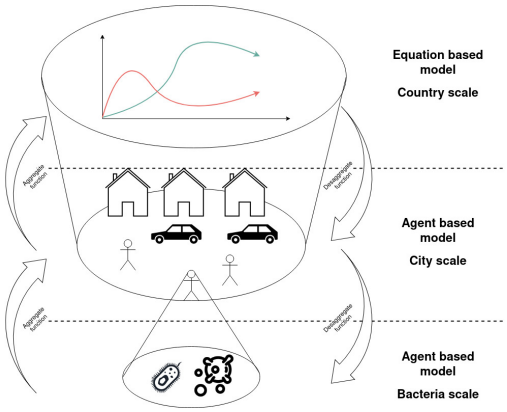

FIGURE 2.2 : Représentation simplifiée de l'architecture *Zoom*

La figure 2.2 illustre l'architecture globale de ce design pattern. Dans cette illustration, le modèle comporte trois niveaux : un modèle basé sur des équations et deux modèles basés sur des agents. Comme montré, chaque niveau représente une échelle différente et est spatialement et temporellement limité par le niveau supérieur. Enfin, il est possible de passer d'un niveau à l'autre en suivant des fonctions de transfert (agrégation, désagrégation, etc.) prédéfinies dans le modèle intègre. Cette construction garde chaque niveau indépendant, mais seulement un niveau peut être exécuté à la fois.

Si cette typologie est appliquée à l'exemple d'un modèle d'évacuation, il permet de modéliser et de simuler la foule soit sous forme d'un modèle basé sur des agents (c'est-à-dire le modèle initial), soit comme une équation de mécanique des fluides (Anh et al., 2012; Henderson, 1974; Xiong et al., 2013). *Zoom* n'autorisant pas leur traitement simultané ; les modélisateurs devraient donc définir des fonctions de transition pour passer d'un niveau à un autre. Dans cet exemple, l'approximation champ moyen des agents (Parr et al., 2020) peut être utilisée pour passer du niveau *agent* au niveau *équation*, comme illustré dans la figure 2.3.

Dans un exemple plus concret, une évolution du projet ESCAPE (présenté en 1.1.3.a, un modèle d'évacuation urbaine) permettrait de représenter la foule simulée alternativement (i) par un modèle multi-agent à un niveau micro décrivant les trajectoires individuelles dans les ruelles de la ville, ou (ii) par un modèle continu inspiré des équations de mécanique des fluides pour capturer la densité des flux piétons à l'échelle d'un quartier entier. Le transfert entre ces échelles utiliserait une fonction de transfert par champ moyen pondéré permettant d'agréger les positions individuelles en un champ de densité avant de passer au niveau équationnel, puis de désagréger au besoin pour revenir à un niveau agent tout en ne maintenant qu'un seul niveau actif, conformément au pattern *Zoom*.

2.2.1.c Revue de la littérature

Contrairement aux autres design patterns présentés plus tard dans ce chapitre, l'implémentation du pattern *Zoom* est simple et ne nécessite pas un cadre technique ou

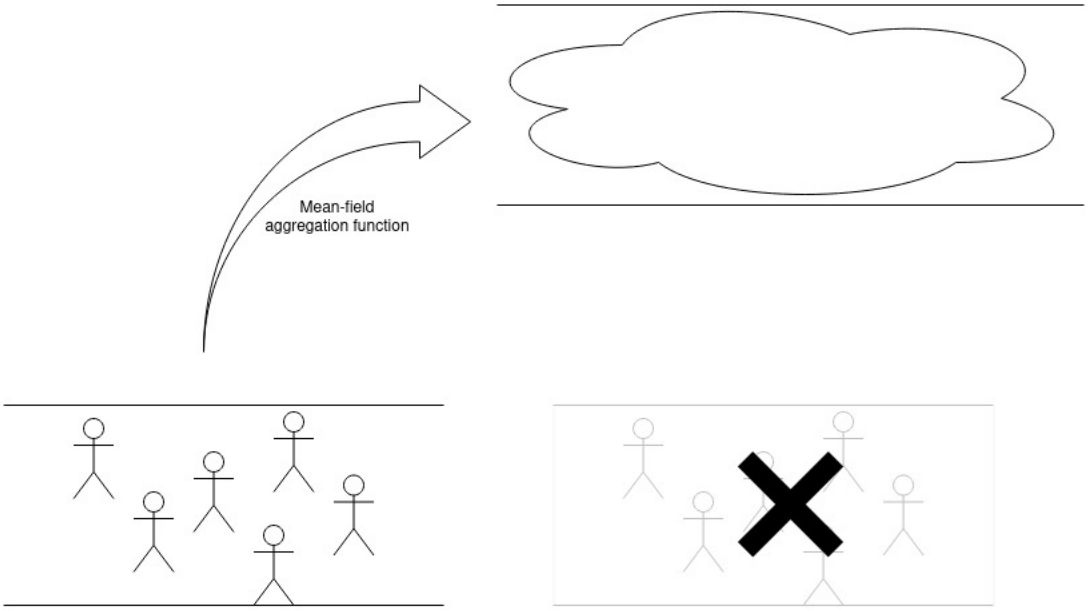

FIGURE 2.3 : Représentation simplifiée du modèle d'exemple dans le design pattern *Zoom*, avec agrégation des agents du micro-niveau à une équation au macro-niveau. Cette opération est destructrice ; après la transition, le premier niveau est supprimé et le modèle fonctionne uniquement avec le macro-niveau.

méthodologique pour créer ce type de modèle multi-niveau basé sur des agents (ML-ABM). La plupart des implémentations sont *ad hoc* (c'est-à-dire propre au modèle pour lequel elles ont été utilisées, et non réutilisable dans d'autres modèles). Ainsi, plutôt que de les lister, cette revue se focalise sur les fonctions mathématiques utilisées pour permettre le passage d'un niveau à un autre.

Il est néanmoins pertinent de noter que ce design pattern est souvent préféré dans le domaine de la modélisation du trafic routier (Abouaissa et al., 2006) et de la simulation de foules (Crociani et al., 2016; Kiselev et al., 2016).

Parmi les fonctions de transfert les plus couramment utilisées figurent des approches d'agrégation, notamment la théorie du champ moyen (Kadanoff, 2009; Parr et al., 2020) et les méthodes d'agrégation de variables (Auger et al., 2008). Dans ces approches, les dynamiques complexes à l'échelle micro, impliquant de nombreux agents interagissants entre eux et avec leur environnement, sont approximées par une dynamique moyenne unique du système à l'échelle macro. Ces approches ont été appliquées à de nombreux domaines, notamment la physique, l'intelligence artificielle, l'épidémiologie, l'écologie, la biologie, la théorie des jeux, etc. Cependant, elles ont l'inconvénient de ne pas conserver une représentation détaillée du système et, en particulier, n'offre pas de flexibilité dans le transfert d'informations détaillées entre les niveaux, entre autres, en raison de leur nature mathématique.

Une manière de surmonter cette limitation consiste à utiliser une représentation en graphe comme niveau intermédiaire, permettant un transfert d'informations plus flexible entre les différents niveaux d'un système complexe, soit sous forme de flux d'informations ascendants (*bottom-up*), soit descendants (*top-down*) à travers les niveaux. Cette idée a été introduite par Nguyen (2010), ainsi que dans des travaux connexes et étendus (Nguyen et al., 2008, 2010, 2012). Cette méthodologie a d'abord été testée sur des systèmes théoriques en écologie/population et a rapidement été

appliquée à des systèmes épidémiologiques (Thuy, 2018), pédologiques (Lèye et al., 2015), de gestion des déchets (Nguyen-Trong et al., 2017) et autre.

2.2.1.d Atouts et contraintes

La plus grande force de ce design pattern réside dans sa simplicité d'utilisation et d'implémentation. Cette simplicité se trouve dans l'indépendance des niveaux entre eux. En effet, ils ne sont reliés que par des fonctions de transition pour passer d'un modèle à un autre. Cette indépendance laisse notamment aux modélisateurs une totale liberté quant à la nature des modèles pouvant être utilisés comme niveaux (basés sur des agents, basés sur des équations, etc.).

De plus, grâce à cette autonomie des niveaux, le développement des couches d'un modèle multi-niveau peut être réalisé en parallèle en permettant à une équipe multidisciplinaire de travailler sur différents modèles (en termes de nature, d'échelles temporelles et spatiales, etc.), qui peuvent facilement être intégrés dans un pattern *Zoom*. Par ailleurs, le développement et l'extension d'un seul niveau peuvent être effectués indépendamment de l'imbrication globale de ce niveau dans le reste du modèle.

Deux autres forces découlent de l'indépendance des niveaux : premièrement, ils sont, par nature, réutilisables dans d'autres modèles et, deuxièmement, ils peuvent être arrangés à la discrétion des modélisateurs. Cette ré-utilisabilité permet de récupérer des modèles hautement spécialisés (ABM, EBM, etc.) provenant d'une autre équipe de modélisateurs pour améliorer et enrichir le modèle en cours de développement. Ces modèles peuvent être des modèles indépendants ou provenir d'un autre ML-ABM de type *Zoom*.

Ensuite, le choix dans l'arrangement des niveaux entre eux permet de suivre la vision et l'architecture que les modélisateurs souhaitent appliquer. Cela renforce aussi l'indépendance entre les niveaux, car ils n'ont pas nécessairement besoin d'être liés selon des échelles ou autres contraintes (sauf pour la faisabilité de la fonction de transition, qui est quasiment toujours réalisable).

Une dernière force de ce design pattern réside dans l'économie de ressources qu'il permet. En effet, en ne calculant qu'un seul niveau à la fois, l'ajout de nouveaux niveaux n'affecte pas (ou que marginalement) les ressources consommées par l'exécution de ce modèle multi-niveau.

De ces forces découlent néanmoins une limitation majeure dans l'utilisation de ce design pattern. La destruction des niveaux inutilisés (positive pour réduire la consommation des ressources du modèle) entraîne inévitablement une perte d'information en provenance de ces niveaux et, de surcroît, empêche l'apparition et l'exploration de phénomènes émergents à plusieurs échelles et entre ces niveaux.

2.2.2 Russian Dolls

2.2.2.a Problématique rencontrée dans les SES

Pour rappel, les SES utilisent le ML-ABM pour tenter de traiter la représentation abstraite à différentes échelles (spatiales et temporelles) et pour capturer les changements systémiques dans l'ensemble du modèle et entre les niveaux.

Le design pattern des *Russian Dolls* aborde ces deux problèmes en offrant un contrôle moins rigide sur les niveaux : ils disposent de la possibilité, lorsque cela est nécessaire, d'agir de manière autonome, mais s'inscrivent dans une structure et une coordination très hiérarchique. Chaque niveau est développé à une échelle et dans un espace spatio-temporel donnés, puis ils sont coordonnés tous ensemble du point de vue du modèle par rapport à chaque échelle. En d'autres termes, pour l'échelle temporelle, cela signifie qu'un niveau exécuté à l'échelle de la minute sera exécuté 60 fois entre les étapes d'un niveau fonctionnant à une échelle horaire ; quant à l'échelle spatiale, chaque niveau est limité et délimité dans un sous-espace du niveau supérieur, comme une maison (sous-niveau) est un espace compris dans celui d'une ville (niveau supérieur).

En conséquence, ce modèle force les niveaux à être utilisés dans une architecture hiérarchique strict, chaque niveau étant plus petit ou égal dans son échelle temporelle et spatiale au niveau supérieur. Il est également important de noter que les niveaux ne sont plus indépendants, rendant ce ML-ABM très comparable aux modèles intégrés. Ainsi, par rapport au modèle *Zoom* précédent, aucun niveau (et donc aucune information) n'est perdu ou détruit, ce qui permet de capturer l'émergence et les changements systémiques de chaque niveau du modèle et entre les niveaux exécutés.

Cette architecture est basée sur un concept de génie logiciel appelé *couplage fort* (ou "*tight coupling*" en anglais) (Rijpma, 1997). Un composant (dans le ML-ABM, il s'agit du niveau) dans ce concept est généralement très dépendant des autres et doit connaître beaucoup du fonctionnement des autres composants avec lesquels il fonctionne. Modifier la logique interne d'un composant dans une application fortement couplé nécessite souvent des modifications dans plusieurs autres composants. Mais cela offre l'avantage d'un système global très stable et d'une forte intégration d'un composant avec les autres.

2.2.2.b Définition

Ce modèle, ainsi que ses implémentations correspondantes dans différentes plateformes, offre un support pour représenter plusieurs niveaux simultanément. Cependant, cette typologie doit respecter la contrainte d'un contrôle strict sur le comportement des niveaux et l'échange d'informations. Cela peut être mis en œuvre en utilisant des schémas de planification similaires à ceux du formalisme holonique (Tchappi et al., 2018).

Dans cette approche "holonique", les niveaux sont globalement re-orchestrés au sein du modèle intégré pour exécuter le modèle complet par niveau (c'est-à-dire, un seul niveau est exécuté à la fois, et l'ensemble du modèle est considéré comme un modèle ABM simple à un seul niveau). Cela nécessite de mélanger processus et structures (qui sont centraux dans la pensée systémique). L'organisation en plusieurs niveaux influence, nécessairement, les aspects structurels et fonctionnels du modèle : sur le plan structurel, les éléments (ici les agents, niveaux, etc.) appartiennent ou sont contenus dans d'autres éléments, et sur le plan fonctionnel, les éléments dépendent des dynamiques des autres. Cela se traduit généralement par une implémentation particulière des concepts d'échelles spatiales et temporelles (jouant un rôle central dans les mécanismes multi-niveaux), ainsi qu'à l'extension ou à la modification de certaines propriétés des ABM : les agents peuvent désormais "appartenir" à d'autres ou être contenus dans d'autres, par conséquent, ces agents sont exécutés par d'autres et perdent, donc, leur propre contrôle.

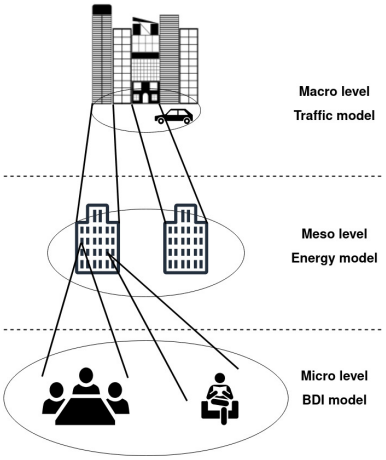

FIGURE 2.4 : Représentation simplifiée de l'architecture des *Russian Dolls* où les niveaux sont (spatialement et temporellement) des sous-ensembles du niveau supérieur

Dans l'exemple illustré par la figure 2.4, le ML-ABM réorchestré selon le design pattern de la *Russian Dolls* sera exécuté comme suit : le macro-niveau (c'est-à-dire à l'échelle de la ville) est exécuté en premier, où les véhicule se déplacent selon un modèle de trafic. Ensuite, chaque bâtiment sera sous-exécuté au méso-niveau, en calculant sa consommation énergétique. Enfin, pour chaque pièce du bâtiment, le micro-niveau est programmé et exécute chaque agent selon un autre modèle. Une fois que tous les niveaux ont été exécutés, le modèle commence un nouveau cycle en suivant ce même processus.

De plus, chaque niveau peut avoir un effet réciproque au sein et avec d'autres niveaux, conduisant à certains motifs émergents dans le modèle multi-niveaux. Par exemple, le niveau méso pourrait être influencé par des actions effectuées au niveau micro : si une pièce est utilisée pour cuisiner, faire une visioconférence ou lire un livre, cela influencera différemment l'énergie consommée au niveau méso. De plus, si nous devions ajouter un modèle de pollution à un niveau supérieur au niveau macro actuel, il pourrait être alimenté par notre niveau macro (avec la pollution des véhicules) et notre niveau méso (avec la pollution due à la génération d'électricité).

La planification globale expliquée dans l'exemple peut être adaptée selon les choix des modélisateurs en fonction de leurs besoins. Elle peut commencer par le niveau le plus petit jusqu'au plus grand (une exécution dite *Bottom-Up*), ou l'inverse (*Top-Down*). Ainsi, tous les niveaux sont synchronisés et calculés ensemble, conservent les détails de leurs agents et peuvent dynamiquement dessiner des propriétés émergentes entre les niveaux.

Cette implémentation hiérarchique est très similaire et comparable à ce qui est appelé un *couplage fort* (ou *tight coupling* en anglais) en génie logiciel. Il s'agit d'un type de couplage qui décrit un système dans lequel les parties logicielles ne sont pas seulement liées entre elles, mais sont également très dépendantes les unes des autres.

Dans cette comparaison, les composants d'un ML-ABM suivant le pattern *Russian Dolls* sont ses niveaux et partagent de nombreuses caractéristiques de développement avec les composants des systèmes à couplage fort, qui sont souvent vus comme des inconvénients (McConnell, 2004; Voinov and Shugart, 2013) :

— Une modification dans un module (niveau) entraîne généralement une cascade de modifications dans les autres.

— La composition de l'architecture logicielle des modules (niveaux) peut nécessiter plus d'efforts et de temps en raison de l'interdépendance accrue entre les modules.
— Un module particulier (niveau) peut être plus difficile à réutiliser et à tester, car les parties dépendantes doivent être incluses.

En revenant à l'exemple du modèle de piétons, l'utilisation de l'architecture en *Russian Dolls* permet, par exemple, d'intégrer ce modèle, qui simule les mouvements des piétons sur une seule route, dans un modèle à plus grande échelle simulant le trafic dans l'ensemble d'une ville. Ce couplage permettrait au niveau supérieur de bénéficier de simulations plus fines sur des points critiques, comme une simulation plus détaillée des flux de personnes évacuant une zone donnée. Cependant, cette intégration nécessiterait une modification des deux modèles et perdrait la possibilité d'exécuter ou de considérer chaque niveau de manière autonome. Ces niveaux sont intégrés et à considérer comme un tout dans un modèle multi-niveau. De cette manière, en considérant que la ville (c'est-à-dire le niveau à grande échelle) a un pas de temps de 1 heure, et que la foule évacuant (c'est-à-dire le niveau à petite échelle) à un pas de temps de 30 minutes, le modèle calculerait la foule une fois de plus, entre chaque exécution du niveau ville, comme illustré dans la figure 2.5.

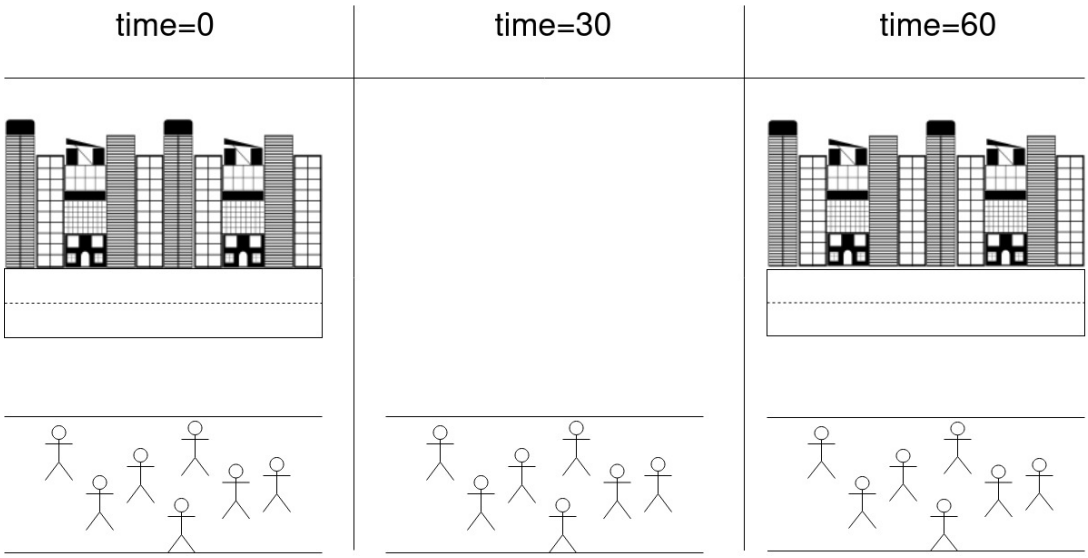

FIGURE 2.5 : Représentation simplifiée du modèle d'exemple dans le pattern des *Russian Dolls* replanifiant l'exécution des deux modèles. Le micro-modèle est exécuté toutes les 30 minutes et le macro-modèle toutes les heures, laissant le micro-modèle seul exécuté au *temps=30*.

En appliquant ce pattern *Russian Dolls* au projet *ESCAPE* (présenté en 1.1.3.a), on pourrait l'imbriquer dans une architecture en trois niveaux :

— Niveau macro – Propagation de la crue : Un modèle hydraulique HEC-RAS calcule, toutes les 60 secondes, l'extension spatiale et la hauteur d'eau dans toute la ville.
— Niveau méso – Gestion de crise par quartier : Pour chaque quartier urbain, ESCAPE réévalue toutes les 10 s la répartition des moyens (pompes, voies fermées) et l'ouverture des centres d'hébergement, à partir des hauteurs d'eau fournies par HEC-RAS.

— Niveau micro – Décisions des ménages et circulation multi-modale : Les agents (personnes évacuants) choisissent chaque seconde d'évacuer, d'attendre ou de protéger leurs biens ; et le calcul de leurs mobilité (piétons, motorisés ou en transports en commun).

Ainsi, 60 cycles micro sont exécutés à l'intérieur de 6 cycles méso, eux-mêmes contenus dans un cycle macro, assurant la cohérence temporelle entre montée des eaux et réactions de la population. Une implémentation similaire a été faite par Chapuis et al. (2019a) avec uniquement le niveau macro et micro.

2.2.2.c Revue de la littérature

Différents outils et *cadres logiciels* (traduit en *framework* dans la suite de ce texte) ont été développés au fil des ans pour faciliter la construction et l'usage de ce type de ML-ABM par les modélisateurs. Certains d'entre eux ont été créés et utilisés dans le domaine du trafic urbain comme **Multi-Level Mesa** (Pike, 2019) qui fournit des méthodes pour aider à gérer les interactions complexes des agents et des modules d'agents (par exemple, des groupes) à travers plusieurs hiérarchies (c'est-à-dire niveaux).

Mais, certains frameworks plus génériques ont également été créés pour les modélisateurs, soit pour appliquer un certain formalisme, soit pour étendre des plateformes ABM déjà existantes, comme :

— **GEAMAS** (GEneric Architecture for MultiAgent Simulation) (Marcenac and Giroux, 1998) est un framework **pionnier de ML-ABM** intégrant trois niveaux de description (micro, méso, macro) pour l'intégration de modèles hétérogènes (Duboz, 2004). Les niveaux micro et macro représentent respectivement les points de vue des agents et du système, tandis que le niveau méso représente une agrégation d'agents dans un contexte spécifique. Les niveaux communiquent entre eux de manière asynchrone. Il a été appliqué à divers domaines tels que des modèles thermodynamiques (Combes et al., 2012).

— **ML-DEVS** (Multi Level DEVS) (Steiniger et al., 2012) est une **extension du formalisme DEVS** (Concepcion and Zeigler, 1988) qui permet la simulation de modèles multi-échelles (et non seulement de modèles couplés dont le comportement est déterminé par les comportements de leurs sous-modèles). Il existe deux types de relations entre les niveaux : la propagation d'informations et l'activation d'événements. Cependant, ML-DEVS se concentre sur une modélisation multi-échelles verticale et ne supporte, donc, que des structures hiérarchiques strictes de modèles. Ce formalisme étendu a été implémenté, entre autre, dans le framework James II (Himmelspach and Uhrmacher, 2009) pour développer des modèles multi-niveaux.

— **CRIO** (Capacity Role Interaction Organization) (Cossentino et al., 2007) est un méta-modèle organisationnel dédié au ML-ABM basé sur le **concept de holon**. Il a été spécialement développé et utilisé pour les simulations multi-échelles de flux de piétons. Il a été utilisé pour modéliser les activités humaines (Lin et al., 2011) ainsi que comme base pour construire un méta-modèle holonique normatif pour les systèmes multi-agents (Missaoui et al., 2017).

— **LevelSpace** Hjorth et al. (2020) est une **extension pour NetLogo** (Tisue and Wilensky, 2004) qui permet aux modélisateurs de créer dynamiquement des

modèles à l'intérieur d'autres modèles (de la même manière que les agents sont créés à l'intérieur des modèles), ce qui implique une structure hiérarchique forte.

— **Capture Release** (Vo, 2012) est une **extension de la plateforme GAMA** (Taillandier et al., 2019) qui permet d'agréger des agents dans des agents de niveau supérieur. Ce processus modifie le comportement des agents capturés qui seront contrôlés par l'agent plus grand. Il a été utilisé, par exemple, dans un modèle d'évacuation (Vo et al., 2012a) et plusieurs modèles jouets affichant la formation de groupes de boids (Reynolds, 1987) en tant que bancs (Vo et al., 2012b).

Enfin, certaines implémentations *ad hoc* ont également été utilisées dans différents projets et domaines de recherche avec des approches similaires, comme en médecine avec certains modèles de tumeurs (Lepagnot and Hutzler, 2009), dans le trafic urbain (Bosmans et al., 2022; Grignard et al., 2018) ou encore dans des projets de modélisation hydrologique (Servat et al., 1998).

2.2.2.d Atouts et contraintes

Il est possible d'extraire certaines forces et faiblesses communes de l'utilisation du pattern *Russian Dolls* par les modélisateurs.

La persistance des niveaux est l'une des forces de ce pattern par rapport au cinquième défi des SES de Elsawah et al. (2020). Il permet aux modélisateurs d'analyser et de capturer l'émergence entre les niveaux et d'explorer les problématiques de modélisation sur une représentation multi-niveaux du système.

Une autre force de cette architecture est son architecture hiérarchique naturelle. L'imbrication spatio-temporelle des niveaux est ce qui est perçu de notre réalité et rend la conceptualisation de ces modèles multi-niveaux simple à manipuler.

Cependant, cette force constitue aussi l'une de ses principales limites : les niveaux ne sont pas indépendants, ce qui engendre plusieurs problèmes déjà évoqués, similaires à ceux du *couplage fort* en génie logiciel. Comme les niveaux font partie intégrante du modèle global, leur extension ou leur débogage s'en trouvent d'autant plus complexes.

Dans son ensemble, cette typologie est très proche des ABM classiques, car elle n'apporte aucun nouveau concept ou méthode pour construire ou utiliser des ML-ABM, mais en étend certains (comme l'orchestration des pas de temps de l'ABM) à des fins explicites multi-niveaux. Par conséquent, il a été très simple pour la communauté de modélisation de développer ce type de modèle et de créer une grande variété d'outils simples et puissants pour construire des *Russian Dolls*.

Mais, cette force crée, également, une limitation dans la nature des niveaux utilisables. Comme le modèle complet sera proche d'un simple ABM, il est difficile d'ajouter des niveaux qui ne sont pas basés sur les agents (mais pas impossible comme dans Chapuis et al. (2021b)). De plus, il sera impossible d'utiliser autre chose qu'un ABM pour un niveau intermédiaire (comme dans une architecture de modèle multi-niveaux avec un ABM au niveau macro, basé sur les équations au niveau méso et un ABM au niveau micro). Ce type de structure est possible avec le dernier design pattern.

2.2.3 Collaboration

2.2.3.a Problématique rencontrée dans les SES

Dans les SES, la question n'est pas seulement de représenter plusieurs *échelles*, mais aussi d'intégrer des composants (modèles, etc) de natures différentes dans un même modèle : une dynamique écologique basée sur des équations différentielles peut devoir intéragir avec un modèle socio-économique à base d'agents, un réseau énergétique optimisé par programmation linéaire, ou encore un modèle climatique discrétisé sur grille. Le design pattern *Collaboration* permet précisément cette hétérogénéité : il permet de coupler, de façon souple, des modèles qui décrivent des entités, des processus et des formalismes mathématiques radicalement distincts.

À l'autre extrémité du continuum présenté dans la figure 2.1, le design pattern *Collaboration* représente donc un contrôle *lâche* entre des niveaux autonomes. Chaque niveau reste un modèle indépendant (qu'il soit à base d'agents, d'équations, de règles ou d'apprentissage automatique) et conserve son propre pas de temps, sa grille spatiale et son dispositif de calibration.

Cependant, pour garantir la capture des motifs émergents au sein et entre les niveaux, ils ne sont pas créés et détruits pendant l'exécution de la simulation du modèle. De manière similaire au *Russian Dolls*, ils collaborent tous et sont exécutés en même temps.

Cette architecture découle des concepts de génie logiciel appelés *couplage faible* (ou *weak coupling* en anglais) (Kaye, 2003). L'idée derrière ce concept repose sur deux aspects principaux : (1) les composants (pour le ML-ABM, il s'agit des niveaux) sont faiblement associés (c'est-à-dire qu'ils ont une relation rompable) les uns avec les autres, ce qui fait que les changements dans un composant affectent le moins possible (voire pas du tout) l'existence ou les performances d'un autre composant ; et (2) chaque composant a peu (ou pas) de connaissances des définitions des autres composants séparés. Ces points protègent l'autonomie et l'indépendance des niveaux, en particulier le second, qui empêche les niveaux de perdre leur propre contrôle.

L'inconvénient de cette flexibilité est la complexité d'utilisation d'un tel modèle. Bien que ce type de couplage soit utilisé depuis des années et largement adopté par le domaine du développement logiciel, il nécessite un travail technique logiciel complexe pour connecter chaque niveau ensemble avec des concepts inhabituels pour les modélisateurs.

2.2.3.b Définition

Comme pour le pattern *Russian Dolls*, le pattern de *Collaboration* répond également à certains besoins de modélisation qui ne sont pas satisfaits par le pattern *Zoom*. Ainsi, il conserve les informations des niveaux sans les perdre. Cependant, pour garantir l'indépendance et l'autonomie de chaque niveau couplé, ce modèle applique un couplage faible. Par conséquent, la principale différence entre *Russian Dolls* et *Collaboration* réside dans la manière dont ils résolvent tous deux le problème de couplage entre ces niveaux.

La typologie *Collaboration* permet de se débarrasser de la contrainte de réorchestration globale du modèle précédent. Ici, l'architecture couple des modèles indépendants en tant que niveaux et permet un échange réciproque d'informations entre eux. Comme

cette interaction est bidirectionnelle, les niveaux ont une influence directe les uns sur les autres.

Dans la conception des systèmes informatiques, un système est dit faiblement couplé lorsque l'un de ses composants entretient une relation dissociable avec les autres, et lorsque ces composants ont peu, voire aucune connaissance, des définitions des composantes distinctes au sein du système. Dans l'approche *Collaboration*, les composantes correspondent à des niveaux du modèle, qui communiquent uniquement par messages logiciels tout en considérant chaque niveau comme une boîte noire. De plus, étant donné que les niveaux sont faiblement couplés et considérés comme des boîtes noires, chaque niveau peut être remplacé par un autre modèle sans nécessiter de travail particulier. Il est également possible d'étendre l'un de ces modèles en y intégrant un modèle préexistant, comme illustré dans Chapuis et al. (2019b).

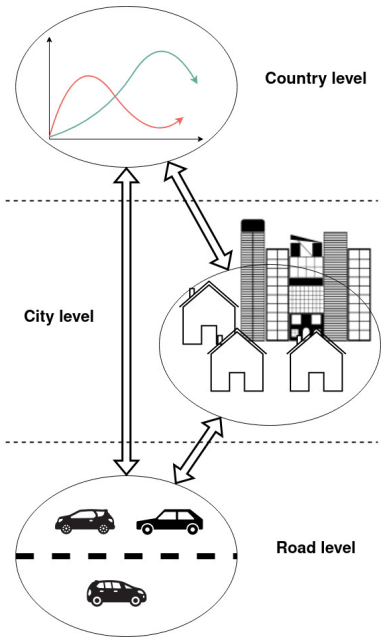

FIGURE 2.6 : Représentation simplifiée de l'architecture de *Collaboration*. Chaque niveau fonctionne de manière autonome et peut échanger des informations entre tous les autres niveaux du modèle intégré.

Dans la figure 2.6, chaque niveau correspond à un modèle indépendant qui envoie des informations (comme le résultat de sortie de chaque cycle traité) à d'autres niveaux (comme valeur d'entrée pour le prochain cycle à traiter) dans des directions choisies par les modélisateurs. Ces messages ne sont pas limités par l'ordre hiérarchique des niveaux dans le modèle, par exemple, le niveau micro peut interagir directement avec le niveau macro, et inversement.

Par conséquent, en modifiant la manière dont les niveaux sont couplés, ce pattern n'exige plus que les modélisateurs se concentrent ou travaillent sur l'orchestration des niveaux. Puisque chaque niveau est considéré comme une boîte noire, ils sont tous exécutés de manière autonome, chacun gardant le contrôle de l'évolution de son propre niveau. Cette distribution du contrôle entre les niveaux est particulièrement intéressante dans la représentation de systèmes où les fonctions sont, de ce fait, également réparties.

Cependant, cette distribution se fait au prix d'une plus grande complexité dans les modèles et peut soulever des problèmes de cohérence entre les niveaux, car les

interactions entre eux sont moins structurées et doivent être anticipées.

Dans l'exemple de modèle d'évacuation, il est possible d'appliquer ce pattern afin d'évacuer la foule par deux points de sortie tout en respectant un modèle basé sur la physique des fluides (Van Toll et al., 2020). Ainsi, le modélisateur créera une interface logicielle permettant d'envoyer des données de sortie (par exemple, un agent `Person` entrant dans cette zone), qui seront prises comme données d'entrées par le second modèle. Ce second modèle effectuera sa simulation de fluide, puis renverra des informations sur le chemin suivi par les agents (voir Figure 2.7). De cette manière, au lieu de représenter la foule comme des agents prenants individuellement des décisions, le modélisateur peut la considérer comme un fluide, ce qui forcera les agents à emprunter une sortie en se basant sur une dynamique de foule à une échelle plus macroscopique.

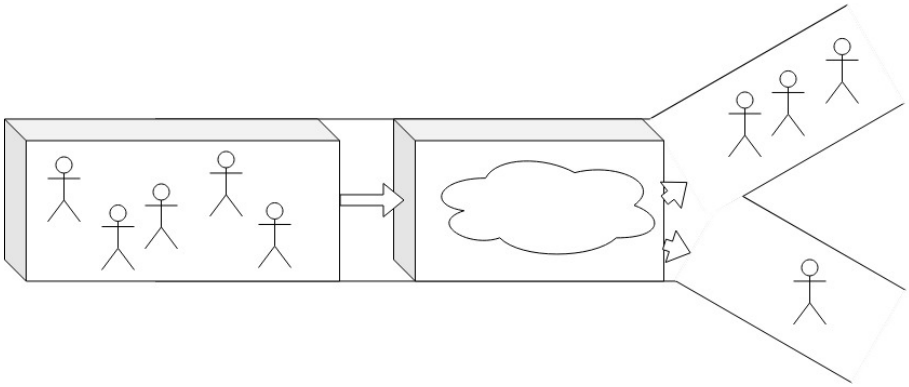

FIGURE 2.7 : Exemple du modèle de *Collaboration* permettant de simuler le modèle d'évacuation basé sur les agents puis de s'appuyer sur un modèle physique simulant la foule comme un fluide pour traiter le nombre d'agents se dirigeant vers une sortie ou une autre. Ces modèles sont traités indépendamment et sont considérés par les autres comme des boîtes noires.

2.2.3.c Revue de la littérature

Différents outils et cadres ont été développés au fil des ans pour faciliter ce type de ML-ABM pour les modélisateurs. Certains d'entre eux ont été spécifiquement créés et utilisés dans le domaine du trafic urbain comme :

— **SimMobility** (Lu et al., 2015) est structuré en trois composants (long, moyen et court terme) et suit une approche multi-niveaux basée sur l'aspect temporel. Les aspects de modélisation sont distribués entre les trois composants et réunis dans une seule base de données.

— **TraSMAPI** (Traffic Simulation Manager Application Programming Interface) (Soares et al., 2013) est une architecture de système intégrant (avec un couplage faible) les frameworks SUMO (Krajzewicz et al., 2002) et JADE (Bellifemine et al., 2005) pour construire des ML-ABM. Les agents JADE représentent les conducteurs (gérant les micro-comportements tels que le changement de voie) qui sont liés aux véhicules dans SUMO (gérant les modes de déplacement méso).

Certains outils plus génériques ont également été créés pour les modélisateurs, soit pour appliquer un certain formalisme, soit pour étendre des plateformes ABM déjà existantes, comme :

— **GEAMAS-NG** (David et al., 2011) est une version mise à jour de GEAMAS des *Russian Dolls*, où le framework fournit des outils pour détecter et concrétiser les phénomènes émergents entre différents types de modèles (comme les MAS et CBR (Corchado et al., 2009)). Cette version a été, par exemple, utilisé pour modéliser l'évolution des zones urbaines.
— **Co-modeling** (Huynh, 2016) est une **extension de la plateforme GAMA** modifiant les concepts de base du formalisme ABM en permettant aux agents d'être des modèles eux-mêmes. Avec cela, le ML-ABM consiste en une collection de modèles qui ne sont que faiblement couplés. Il a été utilisé pour coupler un ABM avec un modèle hydrologique dans un modèle d'évacuation (Chapuis et al., 2019b).

Enfin, comme pour les autres modèles, certains ML-ABM ont été développés en utilisant ce modèle de conception dans une implémentation *ad hoc*. C'est le cas dans des domaines tels que les systèmes biologiques, par exemple, dans de nombreux travaux sur la modélisation multi-niveaux du développement des tumeurs (Butner et al., 2017; Wang and Maini, 2017), le trafic urbain (Alqurashi and Altman, 2019) ou les modèles de gestion des ressources (Rahman et al., 2022). Une utilisation intéressante est faite par Yang et al. (2021) qui utilise la flexibilité de ce modèle pour facilement changer le niveau micro et explorer son impact sans avoir à modifier le reste du modèle.

2.2.3.d Atouts et contraintes

Cette typologie offre des qualités communes aux deux autres patterns. Comme dans le pattern *Zoom*, elle permet d'utiliser et de réutiliser n'importe quel type de modèles indépendants comme niveau dans le modèle multi-niveaux construit, et, comme avec *Russian Dolls*, il conserve les niveaux pendant toute l'exécution du modèle, permettant d'explorer l'émergence dans un système multi-niveaux. De plus, grâce à son couplage faible, il est possible pour les modélisateurs de conceptualiser et de coordonner les niveaux dans un ordre hiérarchique de leur choix.

Cependant, ce pattern est également le plus compliqué à utiliser pour les scientifiques et modélisateurs *non-développeurs*. Par exemple, dans le cadre de *comodeling* (Huynh, 2016), il nécessite trois fichiers techniques supplémentaires pour assurer la communication et la cohérence entre deux niveaux (soit un total de cinq fichiers). Cela crée une grande complexité lors du développement des modèles.

De plus, la structure globale du système oblige les modélisateurs à expliciter soigneusement la cohérence entre les niveaux lors de leur exécution, ce qui complique encore le processus de développement et peut, dans certains cas, conduire à un modèle multi-niveaux incohérent.

En raison de cette complexité cumulative, peu d'outils sont aujourd'hui capables d'en atténuer la difficulté de conception et de permettre à un large public de modélisateurs d'utiliser et d'explorer ce type de modèle.

2.3 ML-ABM dans GAMA

2.3.1 Présentation de la plateforme GAMA

La plateforme *GAMA* (*GIS Agent-based Modeling Architecture*) est un environnement open-source de modélisation et de simulation à base d'agents, conçu pour faciliter la création de simulations spatialement explicites (Taillandier et al., 2019). Développée principalement par un consortium international, dont l'unité de recherche UMMISCO de l'IRD, GAMA offre une architecture modulaire permettant d'intégrer diverses approches de modélisation au sein d'un cadre unifié. Cette plateforme, reconnue pour sa modularité et sa polyvalence, s'appuie sur des outils innovants pour représenter des systèmes socio-environnementaux complexes (Lesquoy et al., 2024; Taillandier et al., 2010).

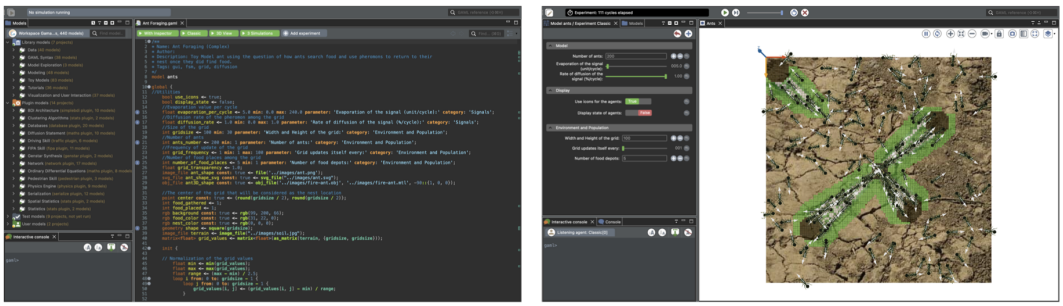

FIGURE 2.8 : Interface de la plateforme GAMA.
Sur la gauche, la vue *Modeling* qui permet de coder le modèle. Sur la droite, la vue *Simulation* qui affiche l'exécution d'un modèle.

Au cœur de GAMA se trouve le langage de modélisation dédié, *GAML* (*GAma Modeling Language*), spécialement conçu pour être accessible aux modélisateurs et utilisateurs sans compétences avancées en programmation. En tant que langage déclaratif et orienté objet, GAML propose des constructions syntaxiques intuitives spécifiques à la modélisation à base d'agents, mettant l'accent sur la lisibilité et la modularité du code (Taillandier et al., 2019, 2012). Cette conception facilite la définition précise des entités du modèle, de leurs attributs et de leurs comportements, simplifiant ainsi l'écriture, la maintenance et l'extension des modèles.

Un atout majeur de GAMA réside dans son intégration native des données *GIS* (Systèmes d'Information Géographique), permettant la manipulation transparente d'informations géospatiales complexes. La plateforme autorise l'importation et la gestion de formats couramment utilisés, tels que les shapefiles et les rasters, simplifiant ainsi la configuration des environnements de simulation à partir de données réelles (Taillandier and Drogoul, 2010; Taillandier et al., 2012). Cette intégration directe est particulièrement bénéfique pour les études nécessitant une géolocalisation précise ou une représentation dynamique des environnements.

De plus, la plateforme GAMA se distingue, également, par son architecture ouverte, reposant sur le framework OSGi (Open Services Gateway initiative) (Hall et al., 2011), qui facilite le développement et l'intégration de plugins pour étendre ses fonctionnalités. Cette modularité permet aux développeurs d'ajouter de nouvelles capacités à la plateforme sans perturber son noyau central, assurant ainsi une flexibilité et une adaptabilité accrues. Par exemple, l'extension SIMPLE (Drogoul

et al., 2025) permet de coupler les simulations basées sur les agents de GAMA avec des environnements de réalité virtuelle, offrant ainsi des visualisations immersives et interactives des simulations.

Enfin, GAMA bénéficie d'une communauté de recherche active qui enrichit continuellement la plateforme avec de nouvelles extensions, documentations et études de cas. Cette dynamique collaborative contribue à la diversité des domaines d'application couverts, allant de l'étude de la pollution urbaine (Kafando et al., 2019) aux simulations épidémiologiques (Gaudou et al., 2020), en passant par la gestion des ressources hydriques (Grignard et al., 2020), faisant de GAMA un écosystème robuste et évolutif pour la conception de simulations à base d'agents.

2.3.2 Des approches de couplage modulaires

La modélisation multi-niveaux peut être conceptualisée comme une forme de couplage modulaire, comme défini dans le design pattern de *Collaboration* (section 2.2.3). Cette approche consiste en une méthodologie structurée visant à décomposer un modèle complexe en sous-modèles distincts, chacun limité et défini par une échelle spatio-temporelle, et en utilisant un *couplage faible* entre ces modèles intégrés. Dans ce contexte de couplage faible, la plateforme GAMA propose notamment le *comodeling* (Huynh, 2016), une approche qui s'inscrit directement dans le design pattern de *Collaboration*. Cette approche modulaire, contrairement aux modèles monolithiques, permet de gérer efficacement la complexité des SES simulés en facilitant leur évolutivité et en améliorant leur lisibilité globale. La cohérence globale est assurée par la définition explicite d'interfaces claires : formats de données standardisés, protocoles de communication prédéfinis et normes de synchronisation temporelle (Morvan, 2012). La modularité favorise ainsi une modélisation progressive, où chaque sous-système peut être conçu, testé et optimisé indépendamment avant son intégration finale, limitant les risques d'erreurs systémiques.

D'un point de vue méthodologique, la modularité simplifie les tâches de validation et de calibration, permettant des tests isolés des sous-systèmes avant intégration. Cela réduit significativement les coûts associés à la gestion des interactions complexes entre les différentes échelles (Brugière et al., 2022). En termes de réutilisabilité, des sous-modèles bien documentés (c'est-a-dire avec une description technique claire permettant l'intégration d'un modèle dans d'autres travaux) peuvent être aisément adaptés à divers contextes d'application, optimisant ainsi la productivité scientifique (Morvan, 2012). Par ailleurs, la modularité facilite la coopération entre spécialistes de disciplines variées (économistes, géographes, écologues, etc.), chacun contribuant à un sous-modèle expert clairement identifié avant l'intégration globale (Hjorth et al., 2020; Lepagnot and Hutzler, 2009).

Cependant, ces bénéfices méthodologiques s'accompagnent de défis techniques. Principalement, la gestion des interfaces logicielles entre sous-modèles pose des enjeux majeurs, nécessitant des stratégies robustes telles que l'utilisation de standards de couplage (comme HLA (Dahmann and Morse, 1998)) pour assurer la cohérence spatio-temporelle des modèles et leur interopérabilité (Kang and Aldstadt, 2019).

Ces défis soulignent l'intérêt du développement de mécanismes spécifiques permettant ces travaux. La plateforme GAMA y propose plusieurs solutions techniques (Drogoul et al., 2013) : d'une part, le *comodeling* (Huynh, 2016), et, d'autre part, le *capture/release* (Vo, 2012), détaillés dans les sections suivantes, qui apportent tous deux des solutions différentes à ces enjeux méthodologiques et techniques. Le

comodeling offre un cadre technique de couplage et d'intégration des modèles ; et le *capture/release* permet, quant à lui, une gestion dynamique des niveaux d'abstraction des agents décrit par le modélisateur, assurant ainsi un fonctionnement unifié des niveaux.

2.3.3 Comodeling

2.3.3.a Introduction

Le *comodeling*, introduit par Huynh (2016), constitue une approche originale focalisée principalement sur le couplage dynamique de modèles afin de simuler efficacement des systèmes complexes. Cette méthodologie répond spécifiquement aux exigences clés des modélisateurs en termes de réutilisabilité, d'évolutivité (scalabilité), d'expressivité et de flexibilité. Elle propose un cadre logiciel fondé sur une méthodologie conceptuelle qui fournit un langage dédié à la description précise des couplages spécifiques entre modèles. Ce cadre facilite également l'encapsulation de modèles existants tout en assurant la traduction efficace des multiples échelles spatiales et temporelles, via de simples extensions à un langage de modélisation existant.

La particularité centrale du *comodeling* réside dans la représentation des modèles couplés, appelés *co-modèles*, sous la forme de modèles à base d'agents où certains agents encapsulent un ou plusieurs modèles, définissant ainsi une description récursive similaire aux approches MADKIT ou DEVS (Duboz et al., 2006; Gutknecht and Ferber, 2000; Zeigler et al., 1997). Dans ce cadre, les interactions entre ces *micro-modèles* se décrivent naturellement comme celles observées traditionnellement entre agents dans les systèmes multi-agents classiques.

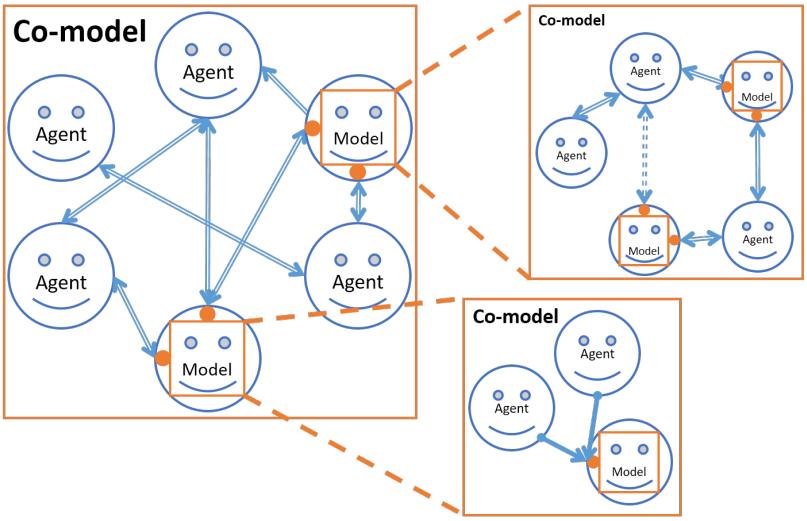

FIGURE 2.9 : Illustration de la composition de micro-modèles comme agents au sein d'un modèle intégré utilisant le *comodeling* (Huynh, 2016).

Chaque micro-modèle encapsulé au sein d'un agent est alors considéré simultanément comme une *encapsulation*, exposant ses attributs et comportements, et comme une représentation directe du modèle qu'il instancie dans le co-modèle, illustré par la Figure 2.9. Cette structure récursive permet aux micro-modèles eux-mêmes d'être également des co-modèles, offrant ainsi une grande souplesse de composition et de recomposition dynamique des modèles en simulation.

Le couplage proposé par le *comodeling* repose fondamentalement sur un *couplage faible* entre les modèles intégrés, permettant ainsi aux sous-modèles de rester autonomes et de communiquer via des interfaces logicielles définies par le modélisateur intégrant ces modèles. Ce couplage faible est essentiel pour garantir une flexibilité maximale et minimiser les dépendances (pouvant nécessiter des modifications internes) entre les modèles couplés, facilitant ainsi leur réutilisation et leur interchangeabilité. Grâce à cette caractéristique de couplage faible et à la considération de boîte noire des agents, il devient particulièrement simple d'intégrer de nouveaux modèles, qu'ils aient été initialement conçus ou non comme des sous-modèles.

Cette approche s'appuie, donc, sur les principes de la modélisation multi-agents et de l'ingénierie logicielle orientée agents en offrant une solution flexible et dynamique pour orchestrer les interactions entre modèles distincts.

Cet outil se catégorise dans le *design pattern* de *Collaboration* (présenté en 2.2.3). Pour rappel, ce pattern préserve l'autonomie des modèles intégrés et facilite la structuration claire des interactions multi-niveaux, renforçant la modularité, la réutilisabilité et la lisibilité des compositions dynamiques de modèles. L'accent porté sur ce couplage structuré et systématique permet non seulement une meilleure intégration de modèles hétérogènes, mais garantit également une cohérence globale au sein des simulations résultantes.

2.3.3.b Mise en œuvre

La mise en œuvre pratique du *comodeling* dans la plateforme GAMA repose sur une démarche structurée et rigoureuse. Cette section détaille explicitement les différentes étapes nécessaires à la construction d'un modèle intégré utilisant cette méthode, illustrée ici par un exemple concret du couplage d'un modèle Proies-Prédateurs, et visualisé par la figure 2.10 :

0. **Modélisation** : Chaque micro-modèle doit être conçu et développé indépendamment avant de travailler à l'intégration de ceux-ci. Ici, un modèle simple est développé avec deux types d'agents `Predator` et `Prey` intéragissant par le premier chassant le dernier
1. **Importation** : La première étape d'intégration consiste à importer les micro-modèles à coupler. Chaque micro-modèle importé doit être associé à un identifiant unique qui sert d'alias dans le co-modèle. Dans GAMA, l'importation se fait par l'utilisation explicite de la commande `import`, suivie du chemin vers le micro-modèle et de son identifiant associé.

   ```
   model Coupled
   import "animal_resource.gaml" as Preys
   import "predator_prey.gaml" as Predators
   ```

2. **Instantiation** : Une fois les micro-modèles importés, leur instanciation se réalise par la commande `create`, propre au langage GAML. Cette commande permet de créer des agents à instancier dans la simulation et, éventuellement, d'autres attributs, comme le nombre d'agents, leurs conditions initiales ou paramètres spécifiques. Dans la logique du comodeling, le modélisateur doit également préciser quelle `experiment`(une `experiment` dans GAMA définit les paramètres, les sorties observées et le déroulement d'une simulation au sein d'un modèle) du micro-modèle sera utilisé pour son exécution au sein du co-modèle.

```
create Preys.simple;
create Predators.simple;
```

3. **Intégration** : Une étape essentielle consiste à définir explicitement le mécanisme de couplage, c'est-à-dire comment les micro-modèles interagissent et échangent des données entre eux. Dans notre exemple, le couplage est réalisé par substitution des proies du second modèle (`Predators`) par les animaux du premier modèle (`Preys`) qui seront chassés.

```
reflex coupling_method {
    ask (Predators.simple) {
        // Remove prey from base model
        ask simulation.prey {do die;}
    }

    // Add prey from the first model to be known by the
   second
    Predators.simple.simulation.available_preys <-
    list(Preys.simple.simulation.animal);
}
```

4. **Exécution** : L'exécution d'un micro-modèle au sein du co-modèle s'effectue par une gestion explicite ordonnant à l'agent représentant le micro-modèle d'effectuer ses étapes de simulation une par une. De part la considération des micro-modèles comme des agents, les modélisateurs peuvent accéder directement leurs attributs et comportements, ainsi qu'échanger des données entre ces derniers. Parmi les attributs accessibles figurent notamment les attributs spatiaux (`shape`) et temporels (`step`, `starting_date`), facilitant l'intégration et la synchronisation explicite des micro-modèles par les modélisateurs.

```
reflex coupling_method {
    // [...]
    ask [Preys.simple, Predators.simple] { do step; }
}
```

2.3.4 Capture/Release

2.3.4.a Introduction

L'approche *capture/release*, introduite par Vo (2012), constitue un cadre opérationnel spécifiquement conçu pour gérer les représentations multi-niveaux au sein d'un même modèle ABM intégré à la plateforme GAMA. Cette approche se distingue nettement des méthodes traditionnelles, comme le *comodeling* de Huynh (2016), qui reposent principalement sur du couplage logiciel de modèles distincts pour obtenir une représentation multi-niveaux. Au contraire, *capture/release* offre une architecture opérationnelle permettant la coexistence et la gestion dynamique de plusieurs niveaux de représentations directement intégrés dans un même modèle, sans nécessiter de couplage externe.

Cette méthode se concentre sur la flexibilité des représentations en permettant des transitions fluides entre niveaux d'abstraction (micro, méso, macro) pendant

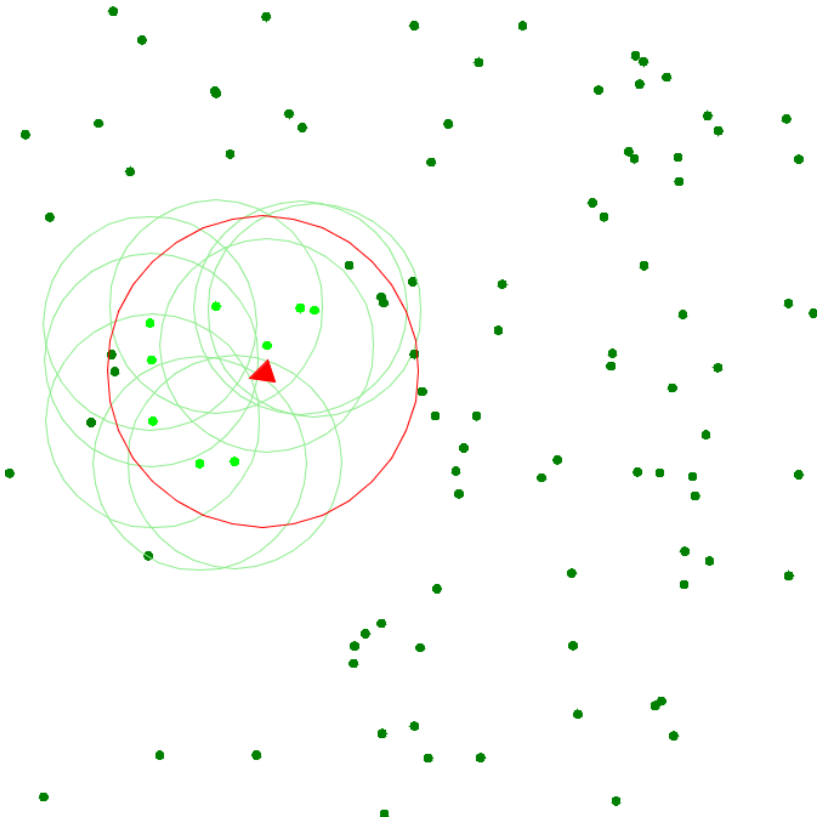

FIGURE 2.10 : Illustration du modèle intégré Proies-Prédateurs utilisant le *comodeling* (Huynh, 2016).

l'exécution de la simulation. Elle repose sur deux opérations fondamentales : (1) la *Capture*, consistant à agréger dynamiquement plusieurs agents micro en une représentation agrégée, simplifiant le niveau de détail tout en préservant la cohérence comportementale ; et (2) la *Release*, qui rétablit une représentation détaillée des agents précédemment agrégés lorsque les conditions du modèle le nécessitent. Cette capacité à naviguer entre différents niveaux d'abstraction au sein d'un unique modèle répond directement aux contraintes computationnelles et sémantiques inhérentes aux simulations multi-niveaux.

L'intérêt majeur de *capture/release* réside dans son architecture logicielle opérationnelle intégrée à la plateforme GAMA et à son langage GAML, permettant une gestion cohérente, performante et naturelle (pour les modélisateurs déjà familier avec la plateforme GAMA) des changements d'échelle au sein d'un unique modèle à base d'agent.

2.3.4.b Mise en œuvre

La mise en œuvre pratique de l'approche *capture/release* dans la plateforme GAMA repose sur une démarche structurée et rigoureuse. Cette section détaille explicitement les différentes étapes nécessaires à la construction d'un modèle intégré utilisant cette méthode, illustrée ici par une variante du modèle Proies-Prédateurs, et visualisé par la figure 2.11 :

Dans cette version, les agents `Prey` peuvent fuir vers des abris protecteurs s'ils sont proches d'agents `Predator`. L'utilisation de *capture/release* offre une gestion

dynamique d'abstraction des agents `Prey` lorsqu'ils fuient vers des abris (`Shelter`) et leur *release* ultérieur sous un état temporairement invisible afin d'éviter une prédation immédiate. Cette spécificité permet d'expérimenter directement l'impact dynamique des transitions multi-niveaux sur les comportements collectifs et individuels du modèle.

0. **Modélisation** : Implémentation classique du modèle avec ses trois type d'agents.

```
species prey { /* ... */ }
species predator { /* ... */ }
species shelter { /* ... */ }
```

1. **Définition des niveaux d'abstraction** : Identifier clairement les niveaux d'abstraction nécessaires (micro, méso, macro) selon les objectifs spécifiques de la simulation.

```
species prey { /* ... */ } // micro niveau

species shelter {
    // [...]
    species prey_in_shelter parent: prey {} //(méso niveau,
   état agrégé)
}
```

2. **Conception des critères de transition** : Établir les critères précis (temporels, spatiaux ou comportementaux) déclenchant les opérations de `capture` et de `release`.

```
species shelter {
    // Récupére la liste des Prey rentrant dans le Shelter
    list<prey> chased_preys update: (prey) where ( (each.
   shape intersects shape) and (each.state = 'flee_predator
   ') );

    // Critère de capture : Prey fuyant les Predator et
   entrant dans le Shelter
    reflex capture_chased_preys when: !(empty (chased_preys)
   ) { /* ... */ }

    // Critère de release : temps écoulé dans le Shelter sup
   -érieur à `prey_in_shelter_max_time`
    reflex release_member_preys when: (time - each.
   in_shelter_time) > prey_in_shelter_max_time { /* ... */ }
}
```

3. **Implémentation des mécanismes d'agrégation et de désagrégation** : Implémenter les fonctions de transition lors des opérations de *Capture* (agrégation) et celles de restauration avec l'opération de *Release* (désagrégation).

```
species shelter {
    reflex capture_chased_preys when: !(empty (chased_preys)
   ){
        // Capture: Aggrégation des agents Prey en
   Prey_in_shelter
        capture chased_preys as: prey_in_shelter {
            state <- 'in_shelter'; // Change l'état de l'
   agent agrégé
```

```
                shape <- ( triangle (4.0) ) at_location location
    ;
            }
        }

        reflex release_member_preys when: (time - each.
    in_shelter_time) > prey_in_shelter_max_time {
            // Release: Désaggrégation des agents
    Prey_in_shelter en Prey
            release to_be_released in: world as: prey {
                state <- 'invisible'; // Change l'état de l'
    agent désagrégé
                shape <- at_location (square(prey_size), self.
    location);
            }
        }
}
```

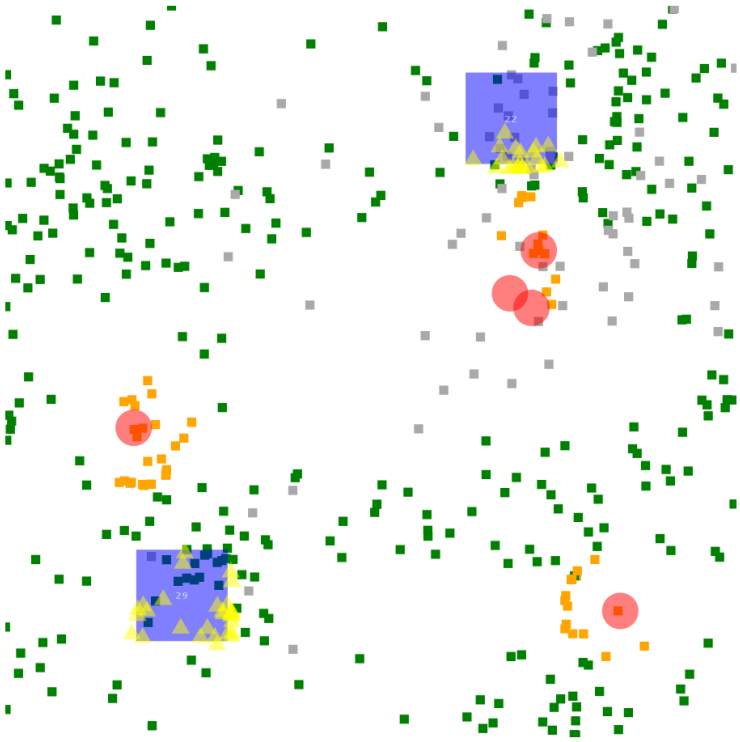

FIGURE 2.11 : Illustration du modèle intégré Proies-Prédateurs utilisant *capture/release* (Vo, 2012). Les `Prey` sont en vert, les `Predator` en rouge. Lorsque les `Prey` sont chassées, ils changent de couleur (orange) et se dirigent vers les `Shelter` (en bleu). Une fois dans le `Shelter`, ils sont agrégés (*capture*) et changent de représentation (triangle jaune), puis sont renvoyés aléatoirement sur la carte lorsqu'ils sont désagrégés (*release*) (en gris).

2.4 Discussion

La revue des architectures ML-ABM présentée dans les sections précédentes révèle un paradoxe fondamental : bien que ces approches offrent des solutions

conceptuelles élégantes pour représenter la complexité multi-échelles des systèmes socio-environnementaux, leur mise en œuvre pratique demeure confrontée à des défis méthodologiques et techniques considérables. Cette discussion analyse d'abord les limitations intrinsèques des approches ML-ABM générales, puis examine comment la plateforme GAMA a tenté de les surmonter, avant d'identifier de nouvelles contraintes liées aux solutions proposées.

2.4.1 Limitations des approches ML-ABM

L'analyse comparative des trois design patterns ML-ABM (*Zoom*, *Russian Dolls*, *Collaboration*) révèle un ensemble de limitations structurelles qui transcendent les spécificités de chaque approche. Ces contraintes, identifiées à travers l'examen des forces et faiblesses de chaque pattern, constituent des verrous méthodologiques majeurs pour l'adoption généralisée des techniques ML-ABM dans la modélisation des systèmes socio-environnementaux.

2.4.1.a Limitations conceptuelles transversales

La première catégorie de limitations concerne les défis conceptuels communs à l'ensemble des approches ML-ABM dans laquelle la **rigidité architecturale** constitue un problème récurrent : chaque design pattern impose une vision particulière de l'organisation multi-niveaux qui contraint fortement les choix de modélisation. Le pattern *Zoom*, de par sa nature destructive, interdit toute coexistence de niveaux, limitant l'exploration des interactions inter-échelles. De leur côté, les *Russian Dolls* imposent une hiérarchie spatiale et temporelle stricte qui ne correspond pas nécessairement aux réalités des systèmes modélisés. Enfin, la *Collaboration*, bien que plus flexible, nécessite une expertise technique considérable pour gérer les interactions complexes entre composants autonomes.

Cette rigidité se traduit par une **faible adaptabilité évolutive** : une fois qu'un modèle est conçu selon un pattern particulier, le passage vers un autre pattern nécessite une refonte quasi-complète de l'architecture. Cette contrainte est particulièrement problématique dans le contexte des systèmes socio-environnementaux, où l'évolution des questions de recherche peut nécessiter des changements d'échelle d'observation ou des modifications des niveaux de détail (Coates et al., 2024; Verburg et al., 2013).

La **complexité de validation** représente un autre défi transversal. Chaque pattern génère des artefacts comportementaux spécifiques : les transitions destructives du *Zoom* peuvent masquer des dynamiques importantes, les contraintes hiérarchiques des *Russian Dolls* peuvent introduire des biais structurels, et la coordination décentralisée de la *Collaboration* peut générer des comportements émergents difficiles à tracer et calibrer. Cette diversité de comportements complique l'établissement de protocoles de validation standardisés pour les ML-ABM.

2.4.1.b Limitations techniques et opérationnelles

Au niveau technique, la **gestion des échelles spatio-temporelles** constitue un verrou majeur partagé par l'ensemble des approches. Bien que chaque pattern propose des mécanismes de coordination, aucun de ces patterns (et peu des frameworks associés) ne fournit un cadre systématique pour gérer automatiquement les disparités d'échelles. Les modélisateurs doivent manuellement spécifier les correspondances

spatiales, définir les ratios temporels et gérer les transformations d'attributs lors des transitions inter-niveaux dans des implémentations *ad-hoc* à chaque modèle. Cette charge technique constitue un frein important à l'adoption, particulièrement pour les modélisateurs non spécialistes du génie logiciel ou de l'informatique plus généralement (Taillandier et al., 2019).

La **complexité d'implémentation** varie selon les patterns mais demeure systématiquement élevée. Le pattern *Zoom* nécessite la conception de fonctions de transition mathématiquement robustes et une gestion du chargement de la destruction des modèles, les *Russian Dolls* exigent une planification fine des orchestrations hiérarchiques, et la *Collaboration* impose la maîtrise de concepts avancés de couplage logiciel. Cette complexité technique crée une barrière à l'entrée considérable qui limite l'accessibilité des ML-ABM aux équipes disposant d'une expertise informatique avancée.

La **charge computationnelle** représente également un défi transversal, bien qu'elle se manifeste différemment selon les approches. Le *Zoom*, malgré son efficacité en termes de ressources, génère des surcoûts lors de possibles transitions fréquentes. Les *Russian Dolls* souffrent de la synchronisation continue de tous les niveaux, même lorsque certains évoluent lentement ou qu'il n'y a que peu d'intéractions entre les niveaux. La *Collaboration* peut engendrer des surcoûts de communication importants entre composants faiblement couplés. Ces contraintes computationnelles complexifient l'applicabilité des ML-ABM à des systèmes de grande taille ou nécessitant une résolution temporelle fine, alors que l'un des usages principal de ces architectures intégrées est justement ce passage à l'échelle.

2.4.1.c Limitations méthodologiques

Sur le plan méthodologique, l'**absence de directives standardisées** pour le choix du pattern approprié constitue une limitation majeure. Les modélisateurs doivent naviguer entre les trois approches sans disposer de critères objectifs pour évaluer leur adéquation à un contexte particulier. Cette absence de cadre décisionnel génère des choix souvent arbitraires ou basés sur la familiarité technique plutôt que sur l'adéquation conceptuelle (Brugière et al., 2022; Grimm et al., 2020; Morvan, 2012).

Également, la **limitation des outils de développement** disponibles constitue un frein pratique important. La majorité des plateformes de simulation ne proposent qu'un support partiel ou expérimental (Hjorth et al., 2020) pour les ML-ABM, obligeant souvent les modélisateurs à développer des solutions *ad hoc* spécifiques à leur contexte. Cette dispersion des efforts limite l'accumulation d'expertise, la réutilisabilité des solutions développées ainsi que le développement d'une méthodologie ou d'une structure globale et solidifiée par une expertise globale.

2.4.2 Apports et nouvelles limitations de GAMA

Face aux limitations identifiées précédemment, la plateforme GAMA a développé des réponses méthodologiques et technologiques au travers des approches de *comodeling* (Huynh, 2016) et de *capture/release* (Vo, 2012). Ces contributions représentent des avancées significatives dans le domaine des ML-ABM, mais génèrent simultanément de nouvelles contraintes qui révèlent la complexité intrinsèque du problème de la modélisation multi-niveaux.

2.4.2.a Contributions aux ML-ABM dans GAMA

La principale contribution de GAMA réside dans la **concrétisation opérationnelle** des concepts ML-ABM au sein d'une plateforme unifiée. Contrairement aux approches théoriques ou aux implémentations *ad hoc*, GAMA offre des mécanismes intégrés, documentés et validés technologiquement pour la mise en œuvre de modèles multi-niveaux. Cette institutionnalisation facilite l'adoption et la diffusion des techniques ML-ABM au-delà des cercles spécialisés en informatique.

Le *comodeling*, d'une part, apporte une réponse directe aux limitations de la *Collaboration* en fournissant un cadre structuré pour le couplage faible de modèles hétérogènes. Sa principale innovation réside dans l'encapsulation récursive des modèles comme agents, permettant une intégration conceptuellement cohérente avec le paradigme ABM. Cette approche résout élégamment le problème de l'hétérogénéité des formalismes en offrant une interface logicielle unifiée pour des modèles de nature diverse (modèle à base d'agents, d'équations, etc.).

Le mécanisme de *capture/release*, d'autre part, constitue une innovation majeure pour la gestion dynamique des niveaux d'abstraction. Contrairement aux *Russian Dolls* traditionnelles qui imposent une hiérarchie statique, cette approche permet des transitions fluides entre différents niveaux d'abstrations des agents simulés au cours de la simulation. Cette capacité d'adaptation dynamique répond directement aux besoins d'exploration flexible des systèmes socio-environnementaux, où les échelles d'intérêt peuvent évoluer selon le contexte simulé.

De plus, l'intégration native de ces outils dans le langage de programmation GAML représente un avantage considérable par rapport aux solutions externes. Les modélisateurs bénéficient d'une syntaxe multi-niveau cohérente avec le reste du code du modèle intégré, réduisant significativement la courbe d'apprentissage. Cette intégration facilite également la maintenance et l'évolution des modèles multi-niveaux, contrairement aux solutions de couplage externe souvent fragiles.

2.4.2.b Contraintes d'usage du *comodeling*

Malgré ses apports conceptuels, le *comodeling* génère de nouvelles contraintes qui limitent son adoption pratique. La **complexité d'usage** constitue le principal frein a l'adoption de cet outil : bien que le mécanisme simplifie conceptuellement le couplage de modèles, sa mise en œuvre pratique nécessite une forte maîtrise technique. Les modélisateurs doivent gérer explicitement l'importation des micro-modèles, leur encapsulation en agents, leur instantiation via des expérimentations dédiées, et la définition manuelle des mécanismes de couplage. Cette charge technique reste largement inaccessible aux modélisateurs ne possédant pas d'expertise avancée en programmation et en génie logiciel.

Parmi ces limites, l'**absence d'une structure grammaticale explicite** permettant de gérer clairement les contraintes sémantiques associées au couplage de modèles hétérogènes, notamment concernant les dimensions spatiales et temporelles, constitue un verrou majeur (Huynh, 2016). Bien que l'approche actuelle permette l'accès aux attributs spatiaux et temporels (tels que `step`, `starting_date` et `shape`), elle ne propose pas une grammaire organisée et standardisée pour définir et gérer ces contraintes sémantiques de façon claire. Cette lacune oblige les modélisateurs à gérer manuellement les correspondances d'échelles, les transformations d'attributs et la synchronisation temporelle, générant des risques d'incohérence et des difficultés

de validation. Par exemple, différents micro-modèles peuvent représenter les échelles temporelles (journalière, mensuelle, annuelle) ou les unités spatiales (pixels, cellules de grille, bâtiments, villes) de manière incohérente, rendant ainsi leur alignement et leur intégration plus complexes. De telles incohérences peuvent entraîner des erreurs dans les résultats de simulation, nécessitant une vérification manuelle et des ajustements par les utilisateurs.

Un exemple direct de cette complexité se constate à travers la **rigidité des interfaces de communication** entre micro-modèles, qui constitue un verrou technique récurrent. Les échanges de données s'appuient sur des conventions ad hoc (noms d'attributs, types, pas de temps implicites) et sur des points d'intégration dispersés dans le code des agents. En l'absence d'un contrat sémantique explicite, toute modification locale d'un micro-modèle (p. ex. changement d'unité, de résolution spatiale, ou de calendrier) se propage sous la forme d'effets de bord difficiles à diagnostiquer. Il en résulte la nécessité de développer des couches d'adaptation spécifiques (mappage d'unités, agrégation/désagrégation, interpolation/extrapolation), qui accroissent la dette technique et limitent la réutilisabilité.

Un autre inconvénient repose dans la **complexité technique de la mise en œuvre**. D'une part, l'intégration de chaque micro-modèle dans une structure d'agent est compliquée et lourde à implémenter. Les utilisateurs doivent précisément instancier les micro-modèles via des expérimentations dédiées, gérer manuellement leurs exécutions et sous-exécutions, orchestrer explicitement les échanges de données, et assurer la cohérence des cycles de vie. D'autre part, cette méthodologie reste largement inaccessible pour les modélisateurs ne possédant pas de compétences avancées en programmation. En effet, bien que le *comodeling* facilite le couplage de modèles hétérogènes dans GAMA, il ne masque pas la complexité logicielle de ce couplage et ne fait aucune hypothèse implicite pour en faciliter la mise en œuvre.

Enfin, la **gestion des dépendances croisées** entre micro-modèles pose des défis conceptuels et techniques majeurs. Lorsque des modèles interdépendants évoluent selon des logiques bidirectionnelles complexes, le *comodeling* peut générer des situations de couplage fort (rétroactions instantanées, boucles algébriques, synchronisations fines) qui contredisent sa philosophie initiale d'encapsulation modulaire. Cette dérive vers le couplage fort réduit la modularité, rigidifie les interfaces et complique la maintenance ainsi que l'évolution incrémentale des modèles intégrés.

Pris ensemble, ces défauts font du *comodeling* un *outil expert* peu utilisé malgré sa grande flexibilité d'usage, principalement réservé aux utilisateurs possédant une expertise avancée en couplage logiciel et une bonne maîtrise de la plateforme GAMA. Ils mettent en évidence le besoin d'un *cadre sémantique explicite* et d'*abstractions de couplage* plus élevées, capables de rendre déclaratives les contraintes d'alignement spatio-temporel, de réduire la charge d'orchestration et de sécuriser l'interopérabilité au moyen de contrats vérifiables.

2.4.2.c Contraintes d'usage du *capture/release*

Le mécanisme de *capture/release*, bien qu'innovant (Vo, 2012), génère ses propres contraintes opérationnelles. Une première limite tient à la **gestion des transitions dynamiques** : le passage répété des agents entre différents niveaux d'abstraction engendre des changements d'état qui peuvent provoquer des conflits difficiles à anticiper ou à résoudre automatiquement. La nécessité d'assurer une cohérence comportementale et sémantique rigoureuse au moment précis des opérations de

capture ou de *release* impose un travail d'anticipation particulièrement complexe des règles de transition. La conception et le paramétrage des modèles doivent expliciter les conditions d'entrée et de sortie, la préservation des invariants (conservation de masse, d'énergie, de cardinalité), ainsi que les politiques de transfert d'attributs et d'historique, sous peine d'introduire des artefacts de simulation.

Une limite majeure, **intrinsèque à la nature intégrée** de *capture/release*, réside dans son ancrage au sein d'un modèle unique. Ce couplage fort, en comparaison de l'approche modulaire du *comodeling*, rend particulièrement délicate l'évolution ou la réutilisation partielle des composants. Toute modification substantielle d'une composante peut exiger des réajustements importants des mécanismes de *capture* et de *release*, ce qui réduit la modularité, la flexibilité et la maintenabilité à long terme. Cette contrainte architecturale freine l'application de *capture/release* dans des contextes où les modèles doivent évoluer fréquemment, être configurés différemment selon les cas d'usage, ou être partagés entre équipes.

Par ailleurs, la **hiérarchie stricte** imposée par *capture/release* (chaque agent ne pouvant appartenir qu'à un seul hôte simultanément) limite la représentation de scénarios où les agents participent à plusieurs contextes en parallèle. Les situations de multi-appartenance (p. ex. individus agrégés dans des véhicules, eux-mêmes agrégés dans des villes, tout en appartenant simultanément à des groupes sociaux transversaux) sont difficilement exprimables sans recourir à des contournements complexes. Cette contrainte se répercute sur la capacité à modéliser des interactions multi-contextes, la gestion des priorités d'influence, et la résolution cohérente des effets concurrentiels.

Enfin, la **gestion des priorités et des conflits** lors de transitions simultanées pose des défis algorithmiques non résolus. Lorsque plusieurs mécanismes de capture concurrents ciblent les mêmes agents, ou lorsque des conditions de *release* contradictoires sont satisfaites au même instant, il n'existe pas de règles d'arbitrage standardisées. Les comportements résultants peuvent alors dépendre d'effets de bord liés à l'ordonnancement (*race condition*), compromettant la reproductibilité et la validité des résultats. L'absence d'un formalisme déclaratif pour exprimer des politiques d'arbitrage (priorités, préemption, exclusion mutuelle, file d'attente) et d'un système de vérification statique pour détecter les conflits de règles accroît le risque d'ambiguïtés et complexifie la validation.

En somme, si *capture/release* apporte un support natif aux transitions d'échelle au sein d'un modèle monolithique, ses contraintes (complexité des règles de transition, couplage fort, hiérarchie exclusive et absence d'arbitrage déclaratif) appellent des mécanismes complémentaires pour sécuriser et assouplir l'orchestration des changements d'état multi-niveaux.

2.4.2.d Limitations transversales des solutions dans GAMA

Les solutions proposées dans la plateforme GAMA présentent des limitations transversales qui limitent leur adoption et leur efficacité, au-delà des contraintes spécifiques à chaque paradigme. Ces limitations sont notamment amplifiées par la **dichotomie conceptuelle** entre les approches *comodeling* (reposant sur un couplage faible) et *capture/release* (un couplage fort). Ces paradigmes s'appuient sur des logiques contradictoires qui imposent aux modélisateurs un choix exclusif dès les phases initiales de conception. Cette absence de fluidité et de combinaison adaptative

limite leur capacité à concevoir des modèles évolutifs, capables de répondre à des besoins méthodologiques changeants.

Cette rigidité méthodologique est renforcée par l'**impossibilité de migration architecturale** entre ces deux paradigmes. Ainsi, un modèle développé selon la logique du *comodeling* ne peut être efficacement adapté à la logique du *capture/release* sans nécessiter une refonte complète, et inversement dans une certaine mesure. Cette limitation structurelle restreint considérablement la flexibilité requise pour l'exploration itérative et adaptative des modèles, tout en contraignant les modélisateurs à des choix figés dès les premières étapes de la modélisation.

De surcroît, l'**absence d'unification sémantique** entre ces paradigmes constitue une barrière considérable à la communication et à la collaboration scientifique. Les concepts spécifiques au *capture/release* tels que « micro-modèles », « agents contenants », « capture » et « release » ne s'alignent pas nécessairement sur les concepts et terminologies utilisées dans le *comodeling* ni d'autres cadres méthodologiques des ML-ABM. Cette disparité sémantique nuit à l'interopérabilité des approches tout en limitant le transfert de connaissances entre disciplines et communautés scientifiques.

Enfin, la **courbe d'apprentissage élevée** qu'exigent chacun de ces paradigmes constitue un frein supplémentaire à leur adoption. Chaque approche mobilise un cadre conceptuel distinct, accompagné d'un vocabulaire propre et de bonnes pratiques spécifiques, nécessitant un investissement formateur considérable. Cette charge nécessaire pour maitriser ces outils multi-niveaux représente un obstacle particulièrement important pour les équipes interdisciplinaires disposant de ressources limitées en expertise informatique, réduisant ainsi l'accessibilité de ces solutions GAMA dans de nombreux projets qui pourraient en bénificier.

2.4.2.e Validation empirique des limitations auprès de la communauté GAMA

L'analyse des données quantitatives et qualitatives collectées lors de la conférence *GAMA Days 2024* auprès de la communauté scientifique d'utilisateur (Brugière and Nguyen-Ngoc, 2024) confirme empiriquement les limitations (tant conceptuelles que pratiques) des outils ML-ABM de la plateforme GAMA identifiées précédemment. De manière globale, cette analyse révèle un paradoxe : les utilisateurs démontrent un intérêt pour les approches multi-niveaux, mais celui-ci coexiste avec des obstacles pratiques substantiels à leur adoption.

La complexité excessive des outils existants, notamment l'approche *comodeling*, constitue l'obstacle principal pour 47% des répondants, validant l'analyse de la rigidité architecturale entre couplage fort et faible. L'expression d'un participant illustre cette problématique : *"comodeling is too syntactically complex to be used on a daily basis"*. Cette observation confirme les conséquences pratiques de l'absence d'unification sémantique entre paradigmes.

Seulement 30% des participants ont effectivement utilisé des modèles multi-niveaux, malgré un intérêt exprimé. À noter que ce sous-ensemble est très majoritairement composé de développeurs de la plateforme GAMA, donc des profils très experts de celle-ci et avec de fortes compétences informatiques. Cette faible adoption suggère que les contraintes architecturales induites par ces outils découragent les modélisateurs. De plus, la difficulté d'intégration des modèles ABM avec d'autres frameworks, mentionnée par 21% des répondants, confirme les limitations d'interopérabilité

identifiées. À l'autre extrémité du spectre se trouvent 32% des répondants qui ignorent les fonctionnalités multi-niveaux disponibles dans GAMA et confirment les obstacles d'appropriation conceptuelle.

Enfin, malgré le faible nombre d'utilisateurs, l'analyse des résultats révèle qu'approximativement 75% des participants expriment un intérêt pour des outils ML-ABM simplifiés et unifiés au sein de la plateforme GAMA.

La correspondance entre l'analyse critique théorique et l'étude de Brugière and Nguyen-Ngoc (2024) valide l'ensemble des limites identifiées des outils de la plateforme GAMA. Chaque limitation structurelle trouve son écho dans les retours d'expériences, démontrant que les obstacles conceptuels se traduisent par des freins pratiques mesurables. Cette validation empirique constitue un fondement solide pour justifier le développement d'un cadre unifié résolvant les limitations confirmées par l'expérience utilisateur.

2.4.3 Vers un cadre méthodologique unifié

L'analyse des limitations des approches ML-ABM générales et des solutions partielles apportées par GAMA révèle la nécessité d'une approche méthodologique fondamentalement différente. Plutôt que de proposer de nouveaux mécanismes spécialisés, cette thèse développe un cadre conceptuel unifié qui transcende les dichotomies actuelles pour offrir une vision intégrée de la modélisation multi-niveaux au sein de la plateforme GAMA.

2.4.3.a Principe d'unification sémantique

La contribution centrale de cette thèse réside dans le développement d'un *langage unifié* dont la sémantique opérationnelle réconcilie les paradigmes apparemment contradictoires du couplage fort et du couplage faible. Cette unification ne vise pas à remplacer les mécanismes existants, mais à les réinterpréter comme des manifestations particulières d'un cadre conceptuel plus général. Le *comodeling* et le *capture/release* deviennent ainsi des patterns spécialisés d'un continuum plus large de solutions multi-niveaux.

Cette approche unificatrice permet de résoudre plusieurs limitations identifiées précédemment. D'abord, elle autorise la **transition dynamique entre ces paradigmes** au cours d'une même simulation, permettant aux modélisateurs d'adapter l'architecture multi-niveaux aux besoins évolutifs de leur exploration. Ensuite, elle établit un **vocabulaire conceptuel commun** qui facilite la communication interdisciplinaire et l'accumulation d'expertise transversale tout en masquant la complexité de chacun de ces outils.

2.4.3.b Architecture adaptative et évolutive

Le cadre proposé introduit le concept d'**architecture adaptative** qui permet l'évolution fluide de l'organisation multi-niveaux selon les contextes de simulation. Contrairement aux approches actuelles qui imposent des choix architecturaux définitifs, ce cadre autorise des reconfigurations dynamiques guidées par des contraintes

sémantiques formalisées. Cette flexibilité répond directement aux besoins d'exploration itérative et d'évolution empirique caractéristiques de la recherche en systèmes socio-environnementaux.

Le coeur de cette architecture réside dans la séparation claire entre la **spécification sémantique** des relations inter-niveaux (ce qui doit être préservé lors des transitions) et leur **implémentation technique** (comment ces préservations sont réalisées). Cette distinction permet aux modélisateurs de se concentrer sur les aspects conceptuels tout en bénéficiant d'une flexibilité technique.

2.4.3.c Accessibilité et adoption

Le cadre méthodologique développé privilégie délibérément l'**accessibilité aux modélisateurs non-informaticiens**. L'interface proposée adopte une syntaxe déclarative qui masque la complexité technique tout en préservant l'expressivité nécessaire pour des modélisations sophistiquées. Cette approche répond directement au verrou de l'adoption identifié précédemment.

L'intégration au sein de la plateforme GAMA garantit une **compatibilité ascendante** avec les modèles existants tout en offrant des fonctionnalités avancées aux utilisateurs qui souhaitent explorer des architectures plus complexes. Cette approche graduée permet une adoption progressive sans disruption des pratiques établies.

Le développement d'un cadre méthodologique unifié pour les ML-ABM représente ainsi une réponse systémique aux limitations identifiées dans l'état de l'art. Plutôt que d'ajouter de nouveaux mécanismes spécialisés, cette approche propose une vision intégrée qui valorise et étend les solutions existantes tout en ouvrant de nouvelles perspectives pour la modélisation des systèmes socio-environnementaux complexes.

Cette transition vers un cadre unifié constitue la motivation principale des développements méthodologiques présentés dans le chapitre suivant, où seront formalisés les mécanismes de contraintes sémantiques et de coordination spatio-temporelle qui fondent l'approche proposée.

3
Proposition méthodologique

Comme décrit dans le chapitre précédent, la modélisation multi-niveaux de modèles à base d'agents (ML-ABM) représente aujourd'hui une approche intéréssante pour comprendre et simuler la complexité des systèmes socio-environnementaux (Zhanli Sun et al., 2016). Cependant, malgré des avancées théoriques et technologiques significatives, la mise en œuvre pratique de modèles véritablement multi-niveaux demeure confrontée à des défis méthodologiques considérables concernant la coordination sémantique entre niveaux d'abstraction hétérogènes, la synchronisation temporelle et spatiale des processus multi-échelles, et la gestion dynamique des transitions entre échelles d'observation (Brugière et al., 2022; Lara and Vangheluwe, 2002; Morvan, 2012). Les approches actuelles de *comodeling* et de *capture/release* (Huynh, 2016; Vo, 2012), bien qu'efficaces dans leurs domaines respectifs, souffrent d'un manque d'unification conceptuelle et d'une rigidité structurelle qui limitent leur adaptabilité aux besoins évolutifs des modélisateurs (Chapuis et al., 2019b; Drogoul et al., 2013), constituant un obstacle majeur à l'adoption généralisée des techniques ML-ABM, particulièrement pour les modélisateurs non spécialistes du génie logiciel (Brugière et al., 2022; Taillandier et al., 2019).

Face à ces constats, cette proposition méthodologique vise à développer un cadre unifié et intégré qui transcende les limitations identifiées en proposant une approche holistique de la modélisation multi-niveaux (Uhrmacher and Weyns, 2018; Zeigler et al., 2000). L'originalité de cette proposition réside dans sa capacité à réconcilier les paradigmes apparemment contradictoires du couplage fort et du couplage faible (Abar et al., 2017; Steinbrink et al., 2018), en proposant un langage unifié dont la sémantique opérationnelle permet de couvrir l'ensemble du continuum des design patterns multi-niveaux identifiés dans la figure 2.1 et de faciliter les transitions fluides entre ces patterns au cours de l'exécution.

Cette approche s'articule autour de deux composantes fondamentales et complémentaires : un cadre de couplage sémantique qui assure la cohérence conceptuelle des interactions inter-niveaux, et un système de transition dynamique entre échelles qui permet l'adaptation flexible des modèles aux besoins de simulation évolutifs. La présentation de cette proposition méthodologique s'organise, donc, en suivant ces deux parties principales : la première partie expose le cadre de couplage proposé,

détaillant les mécanismes de contraintes sémantiques et de coordination temporelle et spatiale qui garantissent la cohérence des interactions multi-niveaux ; la seconde partie présente le cadre de transition entre échelles, explicitant les approches multi-échelles développées et les méthodes de transition dynamique offertes aux modélisateurs qui permettent l'adaptation flexible des modèles aux contextes variables de simulation.

3.1 Cadre de couplage

Le cadre de couplage proposé dans cette thèse constitue le fondement théorique et méthodologique permettant d'assurer la cohérence sémantique des interactions entre niveaux hétérogènes d'abstraction au sein d'un modèle multi-niveaux unifié. Cette approche répond directement aux limitations identifiées précédemment dans les approches existantes de *comodeling* et de *capture/release*, en proposant une architecture conceptuelle qui transcende la dichotomie traditionnelle entre couplage fort et couplage faible (Courdier et al., 2002; Kaye, 2003; Quesnel et al., 2009; Rijpma, 1997).

Pour illustrer concrètement l'application de ces contraintes, nous nous appuierons sur un modèle proies-prédateurs (illustré en Figure 3.1) multi-niveaux organisé selon trois échelles d'abstraction distinctes. Au niveau micro, des agents individuels (`Prey` et `Predator`) possèdent des comportements autonomes de déplacement, chasse, fuite et reproduction. Chaque agent dispose d'attributs individuels (énergie, position, état comportemental) et interagit directement avec son environnement local. Au niveau méso, ces agents s'organisent en groupes agrégés (`PreyHerd` et `PredatorPack`) représentant des dynamiques collectives émergentes telles que la formation de troupeaux, les stratégies de chasse coordonnée, ou les migrations saisonnières. Enfin, au niveau macro, les populations globales (`EcosystemState`) modélisent les équilibres démographiques, les cycles prédateur-proie, et la dynamique des ressources à l'échelle du système complet.

L'objectif de ce cadre est de fournir une couche sémantique formelle qui unifie les différents design patterns multi-niveaux tout en préservant l'autonomie conceptuelle des niveaux constitutifs. Dans notre exemple proies-prédateurs, cette unification doit permettre de gérer simultanément les interactions directes entre proies et prédateurs individuelles (niveau micro), les dynamiques de groupe comme la formation de meutes de chasse (niveau méso), et les oscillations populationnelles caractéristiques des systèmes écologiques (niveau macro), tout en maintenant la cohérence sémantique entre ces trois échelles d'observation (Morvan, 2012; Steinbrink et al., 2018; Uhrmacher and Weyns, 2018; Zeigler et al., 2000).

Cette formalisation vise plusieurs objectifs pratiques. D'abord, elle offre un cadre générique applicable à différents types de modèles socio-environnementaux, facilitant ainsi l'intégration de modèles multi-niveaux et la réutilisation des patterns de couplage. Ensuite, elle établit un vocabulaire commun précis qui facilite la communication entre les modélisateurs et la documentation des choix de conception.

3.1.1 Contraintes sémantiques

La notion de contraintes sémantiques constitue le cœur conceptuel du cadre de couplage proposé, définissant un ensemble de règles formelles qui gouvernent les interactions entre niveaux d'abstraction tout en préservant la cohérence ontologique

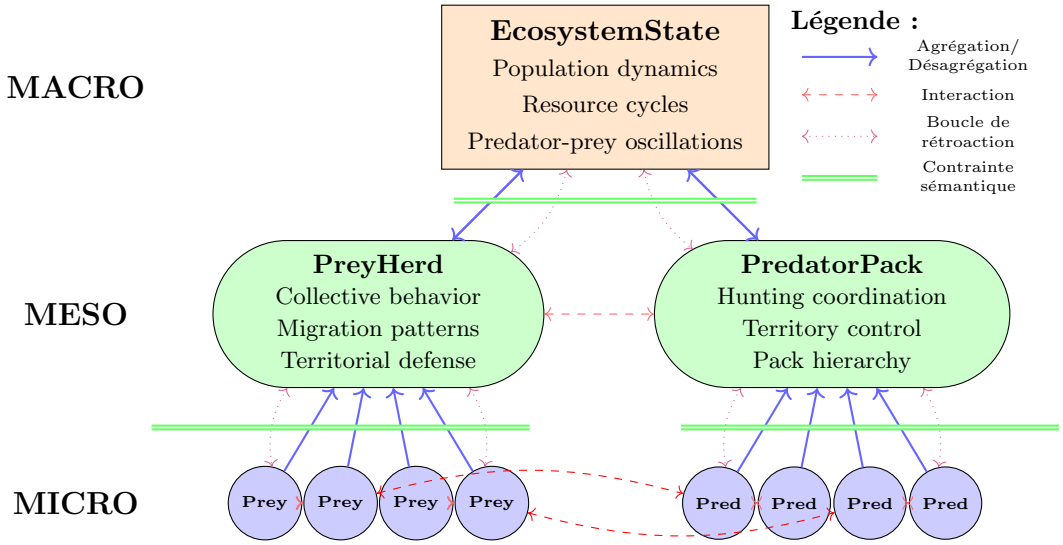

FIGURE 3.1 : Architecture multi-niveaux du modèle proies-prédateurs illustrant les trois échelles d'abstraction (micro, méso, macro), les entités à chaque niveau, leurs interactions et les contraintes sémantiques qui assurent la cohérence conceptuelle lors des transitions inter-niveaux. Les flèches bleues représentent les processus d'agrégation/délégation, les flèches rouges les interactions prédateur-proie, les flèches violettes les boucles de rétroaction, et les lignes vertes, les contraintes sémantiques de préservation ontologique et comportementale.

du modèle global. Dans le contexte de notre modèle proies-prédateurs, ces contraintes assurent que les comportements individuels des agents `Prey` et `Predator` restent cohérents avec les dynamiques collectives observées au niveau des `PreyHerd` et `PredatorPack`, et que ces dynamiques de groupe s'articulent de manière logique avec les équilibres populationnels représentés par l'`EcosystemState` (Brugière et al., 2022; Lara and Vangheluwe, 2002).

Le développement de contraintes sémantiques repose sur une formalisation qui établit les correspondances conceptuelles entre les entités, les attributs et les processus représentés à différents niveaux d'abstraction. Dans notre exemple proies-prédateurs, cette formalisation s'articule autour de trois dimensions principales :

Les **contraintes d'entités** définissent les relations de correspondance entre les agents de différents niveaux. Ces contraintes peuvent se formaliser selon plusieurs types de relations :

— Les **relations d'agrégation** qui permettent de décrire le passage d'un ensemble d'agents vers un niveau supérieur où ils constituent une entité collective.
— Les **relations de délégation** qui permettent de décrire le processus inverse, où une entité de niveau supérieur génère ou se décompose en agents de niveau inférieur, sans imposer de contrainte stricte sur le type d'agents générés.

Par exemple, comme illustré par la figure 3.2, la relation d'agrégation spécifie qu'un `PreyHerd` ne peut être constitué que d'agents `Prey` ; de même, un `PredatorPack` ne peut agréger que des agents `Predator`. L'`EcosystemState` peut agréger des agents `PredatorPack` et `PreyHerd` au niveau macro, mais peut également se décomposer au niveau méso en l'un ou l'autre de ces types d'agents grâce à sa relation de délégation. Cette architecture impose que les transitions directes entre niveaux méso et micro passent obligatoirement par les mécanismes d'agrégation/délégation définis

explicitement, évitant ainsi les incohérences de type dans les transitions multi-échelles. Un `PreyHerd` ne peut donc se transformer directement en agents `Prey` que s'il existe une relation de délégation explicite, ce qui n'est pas le cas dans cet exemple par choix de conception.

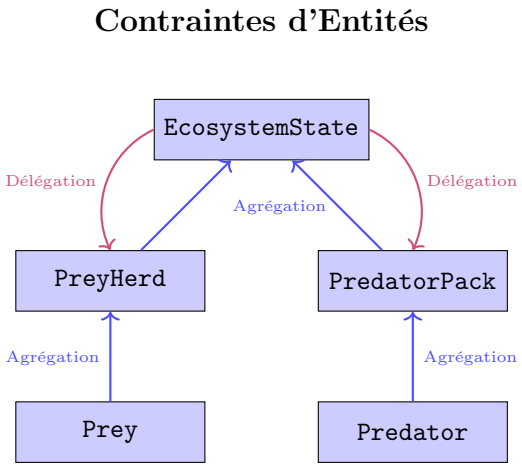

FIGURE 3.2 : Illustration des contraintes d'entités appliquées au modèle proies-prédateurs. Les contraintes d'entités définissent les relations structurelles entre niveaux (agrégation ascendante et délégation descendante), assurant la cohérence typologique des transitions.

Les **contraintes d'attributs** spécifient les règles de transformation des propriétés des agents lors des transitions inter-niveaux. Ces contraintes définissent comment les attributs individuels (position, énergie, état) sont modifiés ou agrégés lors du passage d'un niveau à un autre.

En reprenant le modèle proies-prédateurs et en se concentrant sur les niveaux de `Prey`, comme illustré dans la figure 3.3, l'énergie d'un `PreyHerd` est égale à la somme de l'énergie de tous les `Prey` qui le constituent, et cette valeur est mise à jour explicitement à l'agrégation de chaque nouvel individu. La forme spatiale du `PreyHerd` correspond à l'enveloppe convexe de toutes les positions des agents le constituant. De même, l'agrégation en `EcosystemState` maintient une biomasse totale qui est préservée lors de la désagrégation, et une position aléatoire dans l'espace de simulation est choisie pour initialiser les nouveaux agents créés.

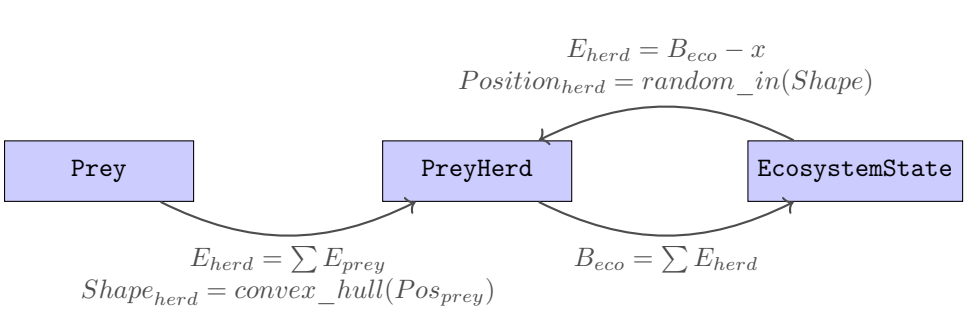

FIGURE 3.3 : Illustration des contraintes d'attributs appliquées au modèle proies-prédateurs. Ces contraintes spécifient les règles de transformation des propriétés (énergie, position, forme spatiale) lors des transitions entre niveaux, garantissant la conservation ou la transformation cohérente des attributs.

Enfin, les **contraintes de processus** encadrent les interactions inter-niveaux et assurent la cohérence comportementale des agents ainsi que des phénomènes émergents multi-échelles. Ces contraintes opèrent selon deux mécanismes complémentaires illustrés dans la Figure 3.4 :

— **Coordination descendante :** Lorsqu'un agent `PreyHerd` adopte un comportement de migration collective (comportement au niveau méso), les agents `Prey` individuels qui le composent doivent adapter leurs comportements de déplacement pour maintenir la cohésion du groupe, tout en préservant leurs capacités de réaction individuelle aux menaces locales. Cette influence peut s'exprimer par une pondération des décisions individuelles et collectives.

— **Émergence ascendante :** Inversement, les comportements individuels des agents `Prey` peuvent influencer les décisions collectives au niveau `PreyHerd`. Par exemple, si une proportion significative d'agents individuels détectent une menace et amorcent un mouvement de fuite, cela peut déclencher automatiquement un comportement de migration collective au niveau supérieur.

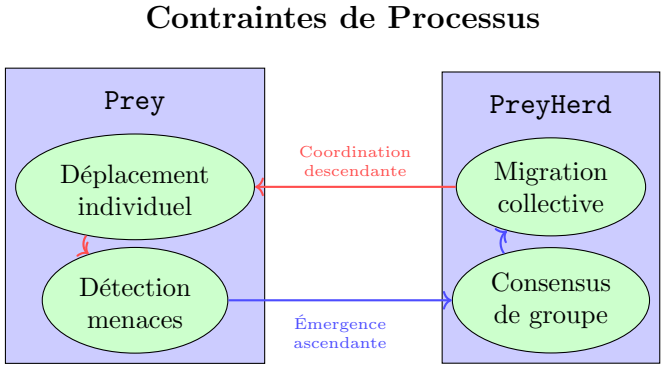

FIGURE 3.4 : Illustration des contraintes de processus bidirectionnelles appliquées au modèle proies-prédateurs. La coordination descendante permet au niveau collectif d'influencer les comportements individuels, tandis que l'émergence ascendante permet aux comportements individuels de déclencher des décisions collectives. Les boucles internes montrent les interactions au sein de chaque niveau.

Cette contrainte de cohérence comportementale bidirectionnelle assure que les décisions prises à un niveau se traduisent de manière réaliste aux autres niveaux, sans supprimer l'autonomie décisionnelle des agents ni empêcher l'émergence de nouveaux comportements collectifs.

La formalisation des contraintes sémantiques présentée précédemment établit les fondements conceptuels nécessaires pour assurer la cohérence ontologique des interactions inter-niveaux. Cependant, ces contraintes ne constituent que la base statique du cadre proposé (le *Meta-Meta-Model* d'un modèle multi-niveaux selon Lara and Vangheluwe (2002)). Pour permettre la pleine expressivité des ML-ABM, il convient d'examiner comment ces contraintes s'articulent concrètement lors des processus de transition des agents entre niveaux d'abstraction. Cette problématique soulève des enjeux techniques et méthodologiques spécifiques concernant la préservation de l'information, la gestion des incompatibilités, et l'orchestration des changements d'échelle (Bonté et al., 2009; Brugière et al., 2022; Morvan, 2012).

3.1.2 Coordination temporelle et spatiale

La modélisation multi-niveaux des systèmes socio-environnementaux requiert la mise en place de mécanismes sophistiqués de synchronisation multi-échelle, capables de gérer l'hétérogénéité intrinsèque des processus en termes de temporalités et de spatialités. Cette section développe un cadre méthodologique permettant d'orchestrer la coexistence et l'interaction de dynamiques opérant à des échelles radicalement différentes, depuis les processus microscopiques (interactions individuelles localisées se déroulant en quelques secondes) jusqu'aux phénomènes macroscopiques (tendances globales émergentes évoluant sur plusieurs années).

L'articulation entre les contraintes sémantiques établies dans la section précédente et leur implémentation spatio-temporelle constitue un enjeu central. Les contraintes d'entités, d'attributs et de processus définissent le cadre ontologique des interactions inter-niveaux ; il s'agit maintenant de spécifier comment ces interactions se matérialisent concrètement dans leurs dimensions spatiale et temporelle. Cette coordination doit répondre à trois exigences fondamentales : (i) assurer la cohérence causale des événements à travers les échelles temporelles, (ii) maintenir l'intégrité géométrique des représentations spatiales multi-résolutions, et (iii) préserver l'autonomie comportementale des entités tout en garantissant leur synchronisation appropriée (Steinbrink et al., 2018; Zeigler et al., 2000).

Les systèmes socio-environnementaux présentent une complexité particulière due à la coexistence de processus aux rythmes temporels extrêmement variés – depuis les décisions individuelles instantanées jusqu'aux cycles écosystémiques pluriannuels (Liu et al., 2007; Wu, 2004) – et aux échelles spatiales s'étendant du local (parcelle, bâtiment) au global (bassin versant, continent) (Verburg et al., 2013). Cette hétérogénéité multi-échelle nécessite des mécanismes de coordination capables de gérer efficacement les transitions entre niveaux tout en préservant la richesse informationnelle de chaque échelle.

3.1.2.a Coordination temporelle

La coordination temporelle dans les modèles multi-niveaux vise à orchestrer l'évolution simultanée de processus opérant à des vitesses d'exécutions différentes tout en maintenant leur cohérence causale. Cette coordination doit pouvoir répondre à trois principaux défis liés au temps dans un couplage hétérogène :

— **Granularité temporelle :** Chaque modèle peut utiliser un pas de temps différent ou une échelle temporelle propre. Par exemple, un modèle macro peut évoluer par pas de 1 an alors qu'un micro-modèle détaillé simule des pas de 1 jour. Cette différence de "granularité" temporelle doit être gérée pour pouvoir aligner les évolutions des modèles malgré leurs résolutions différentes.

— **Synchronisation des avancées :** Les modèles doivent être synchronisés de sorte qu'aucun ne prenne de l'avance ou du retard par rapport aux autres. En pratique, cela signifie qu'il faut coordonner les cycles d'exécution (ou *pas* de simulation) des différents modèles.

— **Alignement des événements :** Les événements discrets se produisant au sein de chaque modèle (par exemple naissance ou mort d'un agent, pic d'activité, etc.) doivent être alignés temporellement afin d'être pris en compte correctement à travers le couplage. Cela implique de gérer les cas d'événements simultanés

ou d'ordonnancement d'événements interdépendants provenant de modèles distincts, pour préserver la causalité et l'ordre logique des interactions.

a) Formalisation des échelles temporelles. La hiérarchie temporelle d'un système multi-niveaux se caractérise par l'attribution à chaque niveau de modelisation d'un pas de temps spécifique reflétant la vitesse caractéristique des processus à cette échelle. Cette formalisation s'illustre généralement en trois paramètres temporels : Δt_{micro}, Δt_{meso} et Δt_{macro}, représentant respectivement les incréments temporels de niveaux micro, méso et macro.

La relation d'ordre $\Delta t_{\text{micro}} \ll \Delta t_{\text{meso}} \ll \Delta t_{\text{macro}}$ établit une hiérarchie temporelle stricte où les processus de niveau inférieur évoluent plus rapidement que ceux des niveaux supérieurs. Cette disparité temporelle se traduit par l'existence de ratios $r_{i,j} = \frac{\Delta t_j}{\Delta t_i}$ quantifiant le nombre d'itérations du niveau i nécessaires pour une itération du niveau j.

Dans le contexte du modèle proies-prédateurs introduit précédemment (section 3.1), la hiérarchisation temporelle se concrétise ainsi :

— $\Delta t_{\text{micro}} = 1$ minute : fréquence de mise à jour des comportements individuels (déplacement, détection, fuite/chasse des agents `Prey` et `Predator`)
— $\Delta t_{\text{meso}} = 1$ heure : périodicité de réorganisation des structures collectives (`PreyHerd`, `PredatorPack`)
— $\Delta t_{\text{macro}} = 1$ jour : intervalle d'actualisation des équilibres écosystémiques globaux (`EcosystemState`)

Ces valeurs impliquent des ratios temporels $r_{\text{micro,meso}} = 60$ et $r_{\text{meso,macro}} = 24$, illustrant la cascade temporelle caractéristique des systèmes multi-échelles.

La gestion de ces ratios soulève plusieurs défis techniques et conceptuels. D'une part, la propagation d'information entre niveaux doit respecter la causalité temporelle : un événement survenant au niveau micro au temps t ne peut influencer le niveau macro qu'à partir du temps $t + \Delta t_{\text{macro}}$. D'autre part, la granularité temporelle doit être suffisamment fine pour capturer les phénomènes critiques sans imposer une charge computationnelle prohibitive.

La contrainte de consistance temporelle s'exprime formellement par le principe suivant : un niveau k ne peut progresser de son pas temporel Δt_k que si tous les niveaux inférieurs ont complété les itérations correspondantes.

Cette formalisation garantit qu'aucune information utile des niveaux inférieurs n'est omise lors de la mise à jour d'un niveau supérieur. L'utilisation de ratios entiers simplifie considérablement l'implémentation de cette contrainte, mais le formalisme reste applicable aux cas de ratios non entiers moyennant des mécanismes d'interpolation temporelle appropriés.

b) Mécanismes de synchronisation. L'orchestration temporelle des processus multi-échelles peut suivre trois paradigmes de synchronisation distincts, chacun présentant des avantages et contraintes spécifiques selon le contexte d'application.

Synchronisation par verrouillage strict (lock-step). Cette approche impose une progression temporelle centralisée et synchronisée globalement où tous les niveaux avancent selon un calendrier commun dicté par un orchestrateur central. L'algorithme suivant formalise ce mécanisme :

```
1: procedure SynchronousUpdate(t_max, levels, Δt)
2:     t ← 0
3:     cycle ← 0
4:     while t < t_max do
5:         for all level ∈ levels do
6:             ratio ← Δt[level]/Δt[micro]
7:             if cycle mod ratio = 0 then
8:                 UpdateEntities(level, t)
9:                 CollectStatistics(level)
10:                if level < |levels| then          ▷ Pas le niveau le plus haut
11:                    AggregateToUpperLevel(level, level + 1)
12:                end if
13:            end if
14:        end for
15:        t ← t + Δt[micro]
16:        cycle ← cycle + 1
17:    end while
18: end procedure
```

Cette synchronisation stricte garantit plusieurs propriétés essentielles :

— **Déterminisme** : l'ordre d'exécution étant fixé, les résultats sont reproductibles
— **Cohérence causale** : aucun niveau ne peut "dépasser" un autre temporellement
— **Simplicité d'implémentation** : la logique de synchronisation est centralisée

Cependant, cette approche peut présenter des limitations en termes d'efficacité computationnelle. Si mal optimisé, comme dans le formalisme ci-dessus, les niveaux supérieurs, évoluant lentement, sont vérifiés à chaque pas de temps élémentaire, générant un surcoût non négligeable. La complexité temporelle s'établit à $O(N \cdot \frac{t_{max}}{\Delta t_{micro}})$ où N représente le nombre total d'entités à travers tous les niveaux.

Synchronisation événementielle. Le paradigme événementiel abandonne la notion de temps global au profit d'une progression décentralisée, et asynchrone, pilotée par les interactions et inspiré de l'approche DEVS (Zeigler et al., 1997). Chaque niveau maintient sa propre horloge locale et communique avec les autres via des événements horodatés. L'architecture repose sur une file de priorité globale ordonnant les événements par timestamp :

```
1: procedure EventDrivenSimulation(t_max, levels)
2:     eventQueue ← InitializePriorityQueue()
3:     InitializeEvents(eventQueue, levels)
4:     while eventQueue.notEmpty() and eventQueue.peek().time < t_max do
5:         event ← eventQueue.pop()
6:         targetLevel ← event.target
7:         currentTime[targetLevel] ← event.time
8:         newEvents ← ProcessEvent(event, targetLevel)
9:         for all e ∈ newEvents do
10:            if ValidateCausality(e, currentTime) then
11:                eventQueue.insert(e)
12:            end if
13:        end for
14:        CheckInterLevelInteractions(targetLevel, levels, eventQueue)
15:    end while
```

16: **end procedure**

Cette approche offre une efficacité computationnelle supérieure lorsque les interactions inter-niveaux sont sporadiques. La complexité dépend du nombre d'événements E plutôt que du temps simulé : $O(E \log E)$. Néanmoins, la gestion manuelle de la causalité devient critique. Le système doit garantir qu'aucun événement ne puisse influencer rétroactivement un état déjà calculé, nécessitant des mécanismes sophistiqués de validation temporelle (Fujimoto, 1990). De plus, bien que compatible avec le formalisme précédemment établi, cette approche peut permettre des transgression de celui-ci, notamment dans la relation d'ordre d'execution entre les niveaux.

Synchronisation hybride adaptative. L'approche adaptative combine les avantages des deux paradigmes précédents en modulant dynamiquement la stratégie de synchronisation. Le système évalue en continu des métriques de criticité pour ajuster la fréquence de synchronisation :

1: **procedure** AdaptiveSync(t_{max}, $levels$)
2: $syncMode \leftarrow$ SYNCHRONOUS
3: $activityMetrics \leftarrow$ InitializeMetrics($levels$)
4: **while** $t < t_{max}$ **do**
5: UpdateActivityMetrics($activityMetrics$, $levels$)
6: $syncMode \leftarrow$ DetermineSyncMode($activityMetrics$)
7: **if** $syncMode =$ SYNCHRONOUS **then**
8: PerformSynchronousStep($levels$)
9: **else**
10: PerformEventDrivenStep($levels$)
11: **end if**
12: AdjustTemporalResolution($levels$, $activityMetrics$)
13: **end while**
14: **end procedure**
15: **function** AdjustTemporalResolution($level$, $metrics$)
16: $intensity \leftarrow$ ComputeChangeIntensity($level$, $metrics$)
17: $\Delta t_{effective} \leftarrow \Delta t_{base} \times (1 - 0.9 \times intensity)$
18: $level.timeStep \leftarrow \max(\Delta t_{min}, \Delta t_{effective})$
19: **end function**

Cette stratégie permet d'optimiser le compromis entre précision et performance. En *période de faible activité* (peu d'événements, états stables), la synchronisation s'espace, réduisant la charge computationnelle et améliorant la vitesse d'exécution du modèle intégré. L'état de *faible activité* se traduit par des seuils de transition entre modes synchrone et événementiel qui sont définis par les modélisateurs avec des métriques telles que le taux d'événements par unité de temps, le nombre d'interactions inter-niveaux, ou la variance des états des agents. Par exemple, un seuil typique pourrait être fixé à dix événements par seconde simulée pour basculer en mode événementiel.

Cependant, cette flexibilité s'accompagne d'une complexité d'implémentation considérable. Les outils existants peinent à supporter nativement ces mécanismes (à la fois centralisés et décentralisés), imposant des développements techniques sophistiqués et une utilisation applicative complexe. De plus, la rigueur programmatique requise reste difficilement accessible aux modélisateurs non spécialistes en informatique. Ils doivent, par exemple, pouvoir traiter tout à la fois la gestion des métriques d'activité, l'ajustement continu des paramètres, la synchronisation des frontières entre zones de

résolutions différentes, etc ; limitant de facto l'adoption de cette approche. Enfin, si mal implémentée, cette approche peut dégrader les résulats du modéle intégré aussi bien sur ses résultats que sur ses performances.

c) Approche privilégiée. Bien qu'imparfaite, la première approche de gestion temporelle utilisant une *horloge centrale* pilotant l'ensemble des modèles couplés semble l'approche la plus prometteuse pour le développement d'outils pour les modélisateurs. Concrètement, il s'agit d'introduire un ordonnanceur global (ou *scheduler* central) qui régit l'avancement de la simulation partagée. Chaque modèle, qu'il soit macro ou micro, voit son évolution temporelle contrainte par cette horloge commune : le temps global est découpé en unités élémentaires (par exemple le plus petit dénominateur commun des pas de temps de tous les modèles), et à chaque unité globale de temps l'ordonnanceur décide quelle(s) composante(s) doit(doivent) avancer.

Dans la pratique, chaque sous-modèle conserve son propre mécanisme interne d'évolution (par exemple une boucle par pas de temps ou un calendrier d'événements). Cependant, ces mécanismes sont *soumis* au scheduler central, qui joue le rôle de méta-simulateur. Ainsi, un micro-modèle peut très bien avoir un scheduler interne (éventuellement à temps continu par exemple) pour gérer ses agents, mais c'est le scheduler central qui déclenche l'exécution de ce scheduler au bon moment. Cette stratégie permet de garantir une exécution chronologiquement cohérente de l'ensemble des modèles : aucun modèle ne peut dépasser le temps global courant tant que le scheduler central ne l'y autorise pas, assurant par construction la synchronisation.

En plus de la robustesse de cette architecture, cette approche est intéressante par sa simplicité conceptuelle : il suffit de déclarer le pas de temps de chaque niveau et le système gère automatiquement la synchronisation. Contrairement aux approches événementielles ou adaptatives qui nécessitent une expertise technique poussée, l'horloge centrale offre un modèle mental clair et prévisible. Les modélisateurs peuvent se concentrer sur la logique de leur modèle plutôt que sur les mécanismes de synchronisation.

De plus, cette approche facilite le débogage et la validation des modèles. L'exécution déterministe permet de reproduire exactement les mêmes séquences d'événements, ce qui est crucial pour identifier et corriger les problèmes ou explorer le modèle intégré. La trace d'exécution est linéaire et peut être facilement visualisée, contrairement aux exécutions événementielles où l'ordre des événements peut varier selon des conditions subtiles.

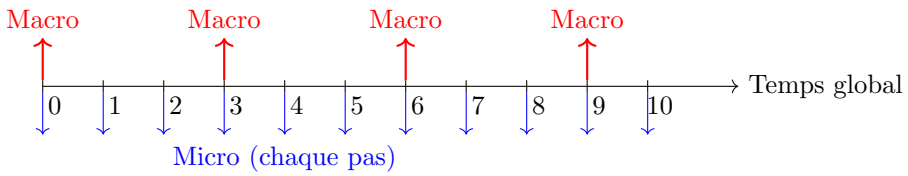

FIGURE 3.5 : Illustration de la synchronisation temporelle par une horloge centrale. L'exemple montre un macro-modèle mis à jour par pas de trois unités de temps (flèches rouges) tandis qu'un micro-modèle est exécuté à chaque unité de temps (flèches bleues). L'ordonnanceur central coordonne l'ensemble : le macro-modèle n'avance qu'aux cycles 0,3,6,... et reste en attente entre-temps pendant que le micro-modèle effectue plusieurs pas intermédiaires.

La Figure 3.5 illustre ce principe de synchronisation. Le temps global sert de

référence absolue : chaque modèle a des instants d'activation prédéfinis (par exemple le macro-modèle agit aux cycles multiples de trois, le micro-modèle à chaque cycle). Entre deux activations du macro-modèle, le micro-modèle réalise plusieurs itérations, ce qui correspond par exemple à affiner une dynamique rapide à l'échelle locale pendant que le modèle global reste quasi-statique à court terme. Au cycle $t = 3$ (point de synchronisation), le scheduler central déclenche l'exécution du macro-modèle en s'assurant qu'il dispose des dernières informations du micro-modèle jusqu'au temps $t = 3$. Réciproquement, le micro-modèle intègre les éventuels décisions ou événements survenus au niveau macro à cet instant synchronisé.

Pour illustrer la coordination temporelle, considérons le modèle proie-prédateur formalisé précédemment à trois niveaux et illustré par la figure 3.6. Au niveau macro, est modélisé l'évolution globale de l'écosystème avec un pas de temps de un jour, tandis qu'au niveau méso, on simule le comportement collectif des meutes de proies et de prédateurs avec un pas de temps de une heure, enfin, au niveau micro, on simule les interactions individuelles de proies et prédateurs avec un pas de temps de 1 minute.

Cette hiérarchie temporelle n'est pas arbitraire mais reflète les échelles naturelles des processus modélisés. Les interactions individuelles (chasse, fuite) se déroulent en quelques secondes, les réorganisations de groupes (formation de meutes, dispersion de troupeaux) nécessitent des dizaines de minutes à quelques heures, tandis que les équilibres populationnels évoluent sur des jours voire des semaines.

L'ordonnanceur central peut être configuré pour représenter une unité de temps égale à une minute. Ainsi, en 60 cycles de l'horloge centrale (une heure), le micro-modèle aura simulé l'évolution locale (naissances, prédations, déplacements, etc.), puis le méso-modèle sera activé tous les 60^{e} cycle pour mettre à jour les meutes de chaque population sur la base de ce qui s'est passé durant l'heure écoulée dans le micro-modèle. Et de même avec le niveau macro tous les 1440^{e} cycles (24 heures). Si un événement macro (par exemple une modification du milieu à l'échelle régionale) doit survenir à une date donnée, il sera programmé sur l'horloge centrale (par exemple au cycle correspondant), garantissant que les modèles micro et méso en tiennent compte exactement au bon moment.

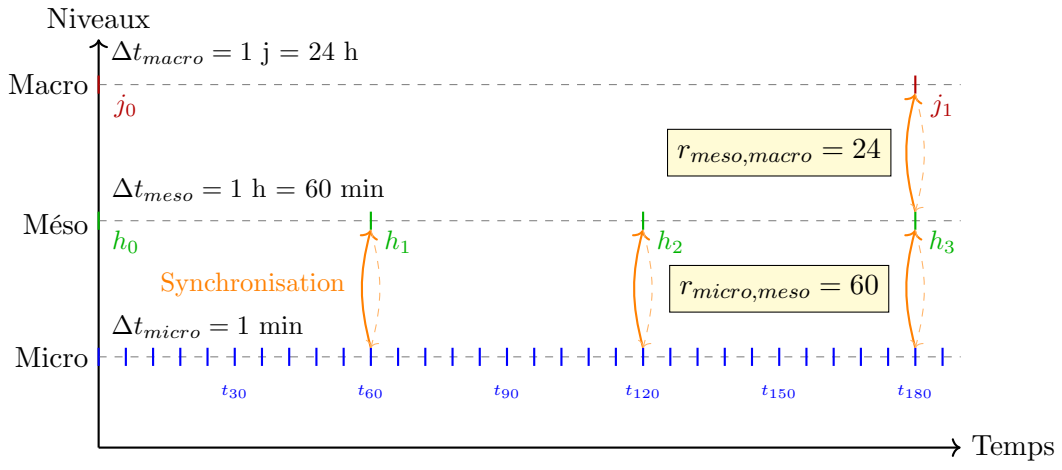

FIGURE 3.6 : Hiérarchie temporelle multi-niveaux illustrant les pas de temps caractéristiques et les ratios entre échelles. Les flèches oranges indiquent les moments de synchronisation où les informations sont agrégées du niveau inférieur vers le niveau supérieur, et les informations désagrégées dans le sens inverse.

Ce dispositif assure un *alignement des événements* : les occurrences significatives à un niveau temporel se répercutent sur l'autre niveau de manière synchrone. En somme, la gestion temporelle proposée, fondée sur une horloge commune et des événements synchronisés, offre un cadre rigoureux et simple pour éviter les dérives entre les modèles et assurer que l'enchaînement des causes et effets reste cohérent à travers les échelles du modèle.

3.1.2.b Coordination spatiale

La coordination spatiale constitue le second pilier fondamental de notre cadre méthodologique pour la gestion multi-échelles. Au-delà de la synchronisation temporelle, elle vise à garantir la cohérence géométrique et topologique des représentations spatiales à travers les différents niveaux d'abstraction. Cette section développe un formalisme permettant de spécifier et vérifier automatiquement les relations spatiales inter-niveaux, offrant ainsi aux modélisateurs un cadre déclaratif pour exprimer leurs contraintes spatiales sans recourir à des implémentations techniques complexes.

De façon analogue au temps, la dimension spatiale du couplage multi-modèle requiert une attention particulière afin que la cohérence entre les niveaux soit préservée. Deux problèmes majeurs sont posés : d'une part la *continuité spatiale* entre modèles, et d'autre part la *gestion d'espaces hétérogènes* en termes de représentation.

Continuité spatiale : Lorsque les modèles couvrent des portions d'espace qui s'articulent ensemble (par exemple un micro-modèle simule une région particulière à l'intérieur du domaine d'un macro-modèle), la continuité aux frontières doit être assurée. Ainsi, aucun "vide" ni chevauchement indésirable ne doit exister : un agent ou un élément géographique qui sort de l'espace du micro-modèle doit entrer dans l'espace du macro-modèle associé (ou inversement) sans incohérence. La continuité spatiale concerne également la cohérence des propriétés environnementales : si une zone A est considérée par le macro-modèle comme ayant une certaine caractéristique (par exemple type de sol, climat, ressource disponible), ces caractéristiques doivent être retrouvées dans le micro-modèle qui opère *dans* A, éventuellement avec plus de détails.

Espaces hétérogènes : Des représentations spatiales différentes peuvent être employées par les modèles couplés : par exemple, l'un peut utiliser une grille régulière de cellules, l'autre un espace continu ; ou bien des projections et unités différentes comme des coordonnées géographiques, pour l'un, et des coordonnées abstraites pour l'autre. Cette hétérogénéité pose le défi de l'interopérabilité spatiale : comment faire correspondre une position ou une étendue spatiale dans un modèle à une position équivalente dans l'autre modèle ? Ces transformations (changement d'échelle, d'unités, autre) doivent être définies et elles doivent préserver les relations topologiques importantes (telles que l'inclusion, le voisinage, les distances relatives) entre les entités des deux modèles.

Pour répondre à ces défis, il est nécessaire d'établir un système de *contraintes spatiales* qui constitue un langage déclaratif permettant aux modélisateurs de spécifier explicitement les relations qui doivent être satisfaites entre les différents espaces des différents modèles intégrés.

Ces contraintes agissent comme des invariants garantissant que la compatibilité des deux représentations spatiales soit maintenue tout au long de la simulation. Deux contraintes fondamentales sont à considérer :

— **Contraintes de localisation** : définissent où un sous-modèle s'inscrit dans l'espace d'un modèle englobant (par exemple "le micro-modèle occupe la cellule (i,j) du macro-modèle")
— **Contraintes de résolution** : spécifient les ratios d'échelle entre les représentations spatiales (par exemple "1 unité macro = 100 unités micro")

L'avantage de ce langage déclaratif est qu'il permet aux modélisateurs d'exprimer leurs intentions de manière claire et vérifiable, sans avoir à implémenter manuellement des mécanismes complexes de transformation spatiale. Par exemple, dans le modèle proies-prédateurs, un modélisateur pourrait déclarer :

— **Contraintes de localisation** : "L'espace de chasse du `PredatorPack` correspond exactement à la zone forestière du niveau macro"
— **Contraintes de résolution** : "La résolution du niveau micro est dix fois plus fine que celle du niveau méso"

Concrètement, il est nécessaire qu'un ensemble de contraintes soient définies, contraintes que le moteur de simulation devra respecter ou vérifier.

La première contrainte concerne l'**espace** des modèles. Par exemple, on peut stipuler que "l'espace du micro-modèle est entièrement contenu dans la zone Z du macro-modèle". Formellement, si E_μ désigne l'espace du micro-modèle et E_M celle du macro-modèle ; si $Z \subseteq Es_M$ désigne une sous-région spécifique du macro-modèle (une cellule particulière ou un polygone représentant une zone d'intérêt), la contrainte de localisation du micro-modèle s'écrit :

$$Es_\mu = Z \subseteq Es_M.$$

Autrement dit, l'espace micro doit coïncider exactement avec la région Z de l'espace macro. Une telle contrainte assure qu'aucun décalage spatial n'existe : les frontières de E_μ correspondent aux frontières internes de E_M.

La seconde contrainte importante concerne l'**échelle** spatiale. Si le macro-modèle représente la zone Z sous la forme d'une grille G de $n \times m$ cellules, tandis que le micro-modèle utilise une grille plus fine ou un espace continu, une correspondance d'échelle entre E_μ et Z doit être établie. Par exemple, il peut être imposé (selon des choix de modélisation) que la résolution du micro-modèle soit un multiple de celle du macro-modèle, ou qu'un facteur d'échelle σ soit défini tel qu'une unité de distance dans le macro équivaut à σ unités dans le micro. Cette contrainte peut être exprimée formellement par une relation du type :

$$\text{résolution}(Ec_\mu) = \frac{1}{\sigma} \times \text{résolution}(Z),$$

ce qui signifie que l'espace micro subdivise l'espace macro de Z selon le facteur σ. En pratique, σ pourrait être, par exemple, dix si le micro-modèle utilise des cellules de dix mètres de côté alors que chaque cellule du macro-modèle représente cent mètres (ainsi 10×10 m $= 100$ m, concordant avec une cellule macro).

Au-delà de la position spatiale et de l'échelle, ce langage de contraintes spatiales permet que toutes les règles nécessaire à la cohérence spatiale soit exprimée. Des contraintes de continuité aux frontières peuvent ainsi être spécifiées , ou encore des contraintes de non-chevauchement si plusieurs micro-modèles sont insérés dans des zones voisines du macro-modèle (aucun E_μ distinct ne doit occuper, même partiellement, la même portion de E_M).

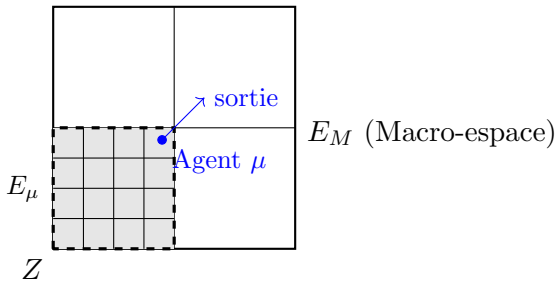

FIGURE 3.7 : Schéma de coordination spatiale entre un macro-espace (E_M) et un micro-espace (E_μ). L'espace macro E_M (grand carré, divisé en 4 cellules) contient une région Z (en gris) au sein de laquelle est défini le micro-espace E_μ (quadrillage fin à l'intérieur de Z). Les contraintes spatiales assurent que E_μ correspond exactement à Z (mêmes frontières) et définissent la relation d'échelle entre la grille macro et la grille micro. Un agent μ (point bleu) évoluant dans E_μ près de la frontière se trouve ainsi aussi à la frontière de Z dans le modèle macro. S'il traverse cette frontière (flèche bleue), cela signale qu'il quitte le micro-modèle pour potentiellement entrer dans un autre contexte spatial.

La Figure 3.7 illustre ce principe avec un exemple simplifié. Le macro-modèle possède un espace E_M divisé en régions (par exemple des cellules carrées). Le micro-modèle opère à l'intérieur d'une région Z particulière, son propre espace E_μ étant confondu à Z. Un quadrillage plus fin à l'intérieur de Z représente une résolution plus détaillée avec le micro-modèle. Dans cet exemple, Z pourrait représenter un territoire spécifique (une parcelle), et E_μ modélise en détail ce territoire. Les agents du micro-modèle (des proies et prédateurs locaux) se déplacent dans E_μ ; à tout instant, leur position correspond implicitement à une localisation dans E_M (via l'identification de E_μ avec Z). Ainsi, un prédateur local proche de la lisière de E_μ est aussi situé en bordure de la zone Z dans le macro-modèle. Ce dispositif prévient toute divergence : il n'est pas possible qu'un agent soit à une coordonnée spatiale qui ne soit pas inclue dans l'espace de l'autre modèle. Si un agent micro dépasse les limites de E_μ (sortie de la flèche bleue sur la figure), cela viole la contrainte de confinement spatiale et un mécanisme de transfert est déclenché (par exemple, l'aggrégation de cet agent vers un agent troupeau plus macro, ou son apparition dans une zone macro adjacente). Ce point particulier relève des *transitions dynamiques entre échelles*, traité dans la section suivante (3.2.1).

L'application de ces contraintes dans l'exemple proies-prédateurs permet d'illustrer concrètement leur utilité. Imaginons qu'un `PredatorPack` opère dans une zone forestière spécifique du modèle macro. Les contraintes spatiales garantissent que :

— Tous les `Predator` individuels du pack restent dans les limites de cette zone forestière ;

— Si la zone forestière est modifiée au niveau macro (déforestation, expansion), l'espace disponible pour le pack est automatiquement ajusté ;

— Les ressources (proies disponibles) sont cohérentes entre la représentation agrégée au niveau macro et la distribution détaillée au niveau micro ;

— Un prédateur qui sort de la zone forestière est automatiquement géré (soit agrégé dans un pack voisin, soit désagrégé comme individu isolé).

Sans ces contraintes formalisées, maintenir cette cohérence nécessiterait un code complexe et sujet aux erreurs, avec des vérifications manuelles à chaque pas de temps.

En résumé, la coordination spatiale proposée établit un cadre déclaratif et vérifiable

pour gérer les relations entre espaces hétérogènes. Ce langage de contraintes spatiales permet aux modélisateurs d'exprimer clairement leurs intentions tout en bénéficiant de garanties formelles sur la cohérence spatiale. Couplée à l'horloge centrale pour la dimension temporelle, cette approche assure une intégration multi-niveau robuste et maintenable.

3.2 Cadre de transition entre échelles

3.2.1 Approches multi-échelles

La gestion des transitions inter-niveaux constitue l'un des aspects les plus techniques de la modélisation multi-niveaux, nécessitant une articulation fine entre les contraintes sémantiques formalisées précédemment (section 3.1.1) et leur mise en œuvre opérationnelle lors des changements d'échelle. Ces processus de transition impliquent l'application concrète des contraintes d'entités (relations d'agrégation et de délégation), l'exécution des contraintes d'attributs (transformations et préservations de propriétés), et l'activation des contraintes de processus (coordination des comportements multi-niveaux) dans des contextes dynamiques.

L'analyse des mécanismes de transition révèle deux dimensions principales qui structurent leur implémentation :

— La gestion de la préservation et transformation des propriétés lors des passages inter-niveaux par le respect et l'application des contraintes d'attributs précédemment établies ;
— La définition et vérification des contraintes d'entités pour assurer la compatibilité entre agents candidats aux transitions.

3.2.1.a Préservation et transformation des propriétés lors des transitions

L'application pratique des contraintes d'attributs constitue le cœur opérationnel des mécanismes de transition, car elle détermine concrètement comment les propriétés des agents évoluent lors de leur passage d'un niveau d'abstraction à un autre. Ces contraintes doivent être adaptées en règles de transition spécifiques à chaque contexte d'application, selon les besoins du modélisateur, révélant la nécessité d'une approche souple et configurable qui respecte cette diversité de logiques d'implémentation.

Dans le contexte du modèle proies-prédateurs, l'application des contraintes d'attributs lors de l'agrégation énergétique illustre parfaitement cette diversité d'approches possibles, comme présenté dans la Figure 3.8. La contrainte générale ($E_{herd} = f(E_{prey1},$
$E_{prey2}, ..., E_{preyn})$) peut être implémentée selon trois stratégies différentes, chacune reflétant des hypothèses théoriques distinctes sur la nature des interactions énergétiques au sein des groupes.

La première application en transformation *linéaire* implémente la contrainte d'attributs sous sa forme la plus stricte, où chaque unité d'énergie individuelle est préservée exactement lors de l'agrégation. Cette approche convient aux modèles bioénergétiques stricts où les lois de conservation constituent des invariants physiques incontournables.

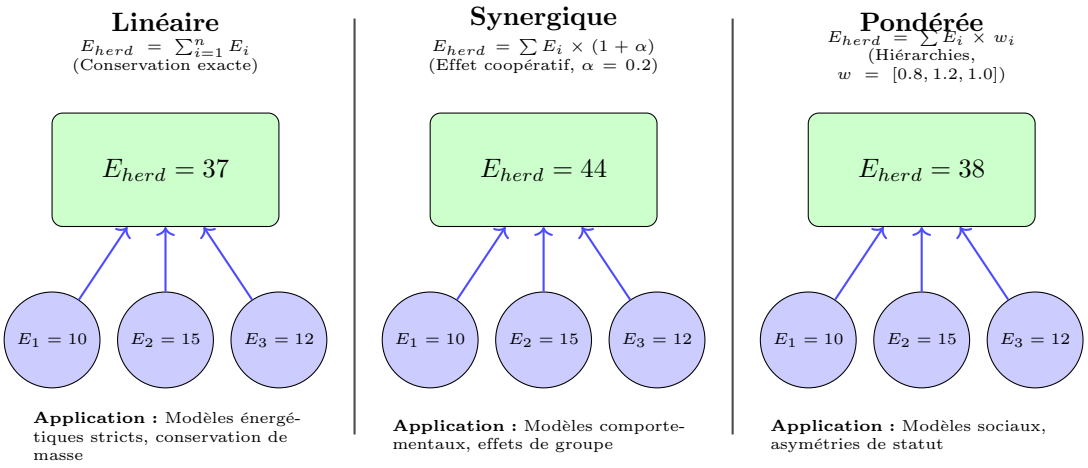

FIGURE 3.8 : Illustration de l'application des contraintes d'attributs selon trois stratégies de transformation distinctes. Chaque stratégie implémente la contrainte générale $E_{herd} = f(E_{prey_i})$ selon des règles spécifiques adaptées aux hypothèses théoriques du domaine d'application, démontrant la flexibilité nécessaire dans l'implémentation des contraintes sémantiques.

La deuxième application en transformation *synergique* modifie la contrainte d'attributs pour intégrer les effets coopératifs. Cette adaptation reconnaît que la simple agrégation ne capture pas les dynamiques émergentes des systèmes sociaux où la coordination peut générer des gains d'efficacité collectifs.

La troisième application en transformation *pondérée* adapte la contrainte d'attributs pour refléter les asymétries internes et les hiérarchies sociales. Cette implémentation reconnaît que certains individus peuvent exercer une influence disproportionnée sur les propriétés collectives (Abar et al., 2017).

Cette diversité d'implémentation des contraintes d'attributs révèle la nécessité d'un framework suffisamment expressif et souple pour permettre aux modélisateurs de spécifier leurs propres règles de transformation sans être contraints par des formulations prédéfinies. Le cadre proposé permet cette configurabilité en séparant la définition sémantique des contraintes (le *quoi*) de leur implémentation technique (le *comment*), préservant ainsi le choix des mécanismes de transformation appropriés à chaque domaine d'étude.

3.2.1.b Vérification des contraintes d'entités et compatibilité

L'application pratique des *contraintes d'entités* lors des transitions entre niveaux nécessite des mécanismes de vérification qui assurent le respect des relations d'agrégation et de délégation définies précédemment. Ces vérifications impliquent l'évaluation de conditions d'éligibilité des agents candidats aux transitions.

Par exemple, la relation d'agrégation des `Prey` en `PreyHerd` du modèle d'exemple peut se formaliser de la manière suivante : $PreyHerd = \{Prey_i | conditions\}$.

Ces conditions s'articulent autour de critères d'éligibilité spécifiques qui varient selon les approches théoriques adoptées par les modélisateurs. Cette variabilité se manifeste dans les multiples façons de spécifier les conditions d'appartenance aux

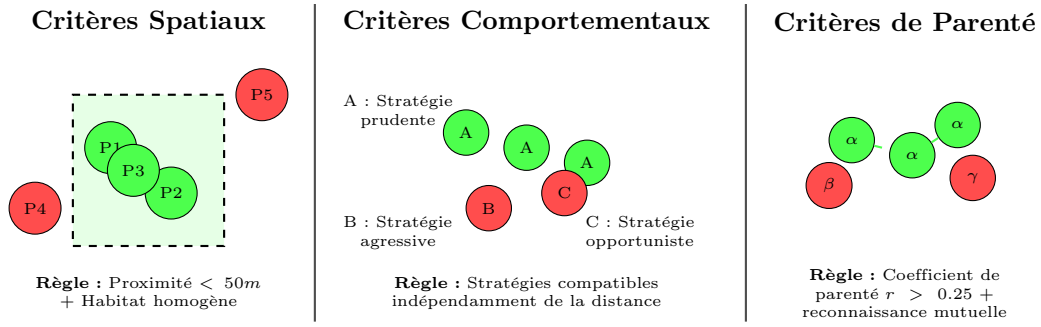

FIGURE 3.9 : Illustration de la variabilité des critères d'éligibilité pour la formation de `PreyHerd` selon différentes approches théoriques. Les critères spatiaux privilégient la proximité géographique, les critères comportementaux se focalisent sur la compatibilité des stratégies, tandis que les critères de parenté valorisent les liens génétiques et la reconnaissance mutuelle. Cette diversité nécessite une configurabilité des mécanismes de transition.

groupes, comme illustré dans la Figure 3.9. Ces critères peuvent se décrire en trois catégories principales :

— Les **critères spatiaux** implémentent la relation d'agrégation en vérifiant que $d(Prey_i, centroid) \leq \delta_{spatial}$, où $\delta_{spatial}$ représente la distance maximale autorisée pour l'appartenance au groupe. Cette vérification s'appuie sur une conception géographique de la cohésion sociale où la proximité physique constitue la condition nécessaire et suffisante pour l'agrégation.

— Les **critères comportementaux** modifient la relation d'agrégation pour intégrer la compatibilité stratégique :
$PreyHerd = \{Prey_i | strategy_i \in compatible\_strategies\}$. Cette approche privilégie la dimension comportementale plutôt que spatiale des agents pour évaluer la formation de groupes cohésifs (Railsback and Grimm, 2019).

— Les **critères de parenté** enrichissent la relation d'agrégation en intégrant les liens génétiques et sociaux : $PreyHerd = \{Prey_i | r_{ij} > r_{threshold} \wedge mutual\_recognition_{ij}\}$, où r_{ij} représente le coefficient de parenté entre individus et $mutual\_recognition_{ij}$ leur capacité de reconnaissance mutuelle. Cette approche s'inspire des théories de la sélection de parentèle et de l'altruisme réciproque.

Évidemment, ces critères ne sont pas exclusifs et, dans des applications réelles de ML-ABM, les modélisateurs doivent utiliser une combinaison de ces critères pour décrire précisément les phénomènes multi-niveaux à explorer par le modèle. Par exemple, un modèle écologique réaliste pourrait combiner des critères spatiaux (proximité géographique), comportementaux (compatibilité des rythmes d'activité) et de parenté (préférence pour les apparentés) pour modéliser la formation de groupes sociaux complexes.

Ces critères s'appliquent également aux transitions inverses (de type *délégation*). Par exemple, la désagrégation d'un `EcosystemState` en `PreyHerd` pourrait utiliser une combinaison de critères spatiaux (distribution géographique des nouveaux groupes), énergétiques (répartition de la biomasse totale) et temporels (moments de formation des groupes selon les saisons).

3.2.2 Méthodes de transition dynamique entre échelles

Les mécanismes de transition dynamique entre échelles constituent l'aboutissement fonctionnel du cadre méthodologique proposé. L'objectif de cette section est de formaliser les mécanismes qui permettent aux entités de naviguer fluidement entre les différents niveaux d'abstraction tout en préservant la cohérence sémantique établie par les contraintes formalisées précédemment.

Le cadre proposé structure les transitions dynamiques autour de trois composantes fondamentales : (i) les *déclencheurs de transition*, qui déterminent *quand* une transition doit s'opérer, (ii) les *stratégies de transition*, qui spécifient *comment* la transition s'effectue, et (iii) les *mécanismes d'orchestration*, qui coordonnent l'ensemble du processus au niveau du système global.

3.2.2.a Déclencheurs de transition

Les déclencheurs de transition constituent les mécanismes de détection qui identifient les moments où un agent à un certain niveau doit initier un changement d'échelle. Ces déclencheurs, définis explicitement par le modélisateur, peuvent s'appuyer sur l'évaluation continue de métriques spécifiques au contexte du modèle, permettant une adaptation automatique et contextuelle de l'architecture multi-niveaux ; ou du résultat du comportement d'un des agents de la simulation.

La typologie des déclencheurs peut, ainsi, être organisée selon deux catégories principales : le déclencheur vient du modèle global, ou de l'agent lui-même.

Déclencheurs globaux (contextuels). Cette première catégorie regroupe les déclencheurs qui émanent du modèle global et s'imposent aux agents selon des critères définis au niveau du système. Ces déclencheurs permettent d'orchestrer des transitions coordonnées répondant à des contraintes systémiques ou des événements environnementaux.

Dans le contexte du modèle proies-prédateurs, plusieurs types de déclencheurs globaux peuvent être identifiés :

— **Déclencheurs démographiques globaux** : Lorsque la densité globale d'agents dans une zone dépasse un seuil critique, le système peut imposer une agrégation pour optimiser les performances. Par exemple, si la densité de `Prey` dans une région R dépasse $\rho_{max} = 100$ agents/km^2, le système déclenche automatiquement la formation de `PreyHerd` pour tous les agents de cette zone, indépendamment de leur volonté individuelle.

— **Déclencheurs temporels systémiques** : Le modèle peut imposer des transitions selon des cycles temporels prédéfinis reflétant des phénomènes naturels. Par exemple, au début de chaque saison migratoire (définie par le modélisateur), tous les agents `Prey` isolés sont automatiquement agrégés en `PreyHerd` migratoires, reproduisant les dynamiques saisonnières observées dans les écosystèmes réels.

— **Déclencheurs spatiaux** : Comme décrit dans l'exemple de la section 3.1.2.b, si un agent sort d'une zone spatiale définie (une zone forestière au niveau méso dans l'exemple) il sera automatiquement géré par le modèle, soit agrégé dans un pack voisin, soit désagrégé comme individu isolé.

Ces déclencheurs globaux agissent comme des contraintes descendantes (*top-down*) qui s'imposent aux agents et garantissent la cohérence globale du système multi-niveaux.

Déclencheurs locaux (agent-centrés). La seconde catégorie comprend les déclencheurs initiés par les agents eux-mêmes, selon leur état interne et leurs perceptions locales. Cette approche *bottom-up* permet une adaptation organique et émergente de l'architecture multi-niveaux, reflétant les décisions autonomes des entités simulées.

Les agents peuvent initier des transitions selon plusieurs critères internes :

— **Déclencheurs basés sur l'état interne** : Un agent `Prey` peut décider de rejoindre un `PreyHerd` lorsque son niveau d'énergie E_i descend sous un seuil $E_{critique}$, cherchant la protection du groupe pour optimiser sa recherche de nourriture. Cette décision reste individuelle et contextuelle à l'agent.
— **Déclencheurs basés sur les interactions entre agents** : Les interactions entre agents peuvent déclencher des transitions. Un agent `Predator` isolé détectant un `PredatorPack` réussi (taux de chasse $\tau_{success} > 0.7$) peut demander à s'y intégrer, initiant ainsi sa propre transition vers le niveau méso.

Cette dualité entre déclencheurs globaux et locaux permet une grande flexibilité dans la modélisation des transitions, combinant le contrôle systémique nécessaire à la cohérence globale avec l'autonomie décisionnelle caractéristique des systèmes multi-agents et laissant apparaitre des phénomènes émergents (Bonabeau, 2002; Iskandar et al., 2024).

3.2.2.b Stratégies de transition

Les stratégies de transition constituent l'implémentation concrète des mécanismes de changement d'échelle, entièrement définie par les modélisateurs selon les spécificités de leur domaine d'application. Ces stratégies prennent la forme de règles de transition, (qu'elles soient mathématiques, algorithmiques ou autre) qui doivent respecter les contraintes sémantiques établies précédemment. La localisation et la nature de ces règles dépendent directement du type de déclencheur activé.

La distinction fondamentale entre transitions descendantes (*top-down*) et ascendantes (*bottom-up*) est importante à souligner et structure, en partie, l'implémentation des règles de transition :

— **Transitions descendantes (déclencheurs globaux)** : Les règles de transition sont définies au niveau cible de la transition. Par exemple, pour une agrégation forcée de `Prey` en `PreyHerd`, les règles d'agrégation sont implémentées dans l'entité `PreyHerd` elle-même, qui "capture" les agents individuels selon des critères définis par le modélisateur.
— **Transitions ascendantes (déclencheurs locaux)** : Les règles sont implémentées au sein des agents initiateurs comme un comportement ou une action de l'agent. Par exemple, une `Prey` souhaitant rejoindre un groupe possède ses propres règles internes déterminant quand et comment initier cette transition, ainsi que les critères d'évaluation des groupes potentiels.

Application des approches multi-échelles. C'est dans ces fonctions de transition que les modélisateurs appliquent les contraintes développées dans la section 3.2.1. Les règles de transition incarnent les mécanismes de préservation et transformation des propriétés lors des changements d'échelle, ainsi que la vérification des contraintes d'entités pour assurer la compatibilité des transitions.

Les modélisateurs doivent expliciter trois types de règles complémentaires : (1) les règles de *vérification des contraintes d'entités*, qui déterminent l'éligibilité des agents pour une transition donnée ; (2) les règles de *transformation des attributs*, qui spécifient comment les propriétés sont préservées ou transformées lors de la transition, appliquant directement les concepts de la section ; et (3) les règles de *coordination comportementale* qui assurent la cohérence des processus entre niveaux.

Cette architecture permet aux modélisateurs d'encoder leur expertise disciplinaire dans des règles spécifiques tout en garantissant le respect des contraintes formelles du système multi-niveaux. La flexibilité offerte permet d'adapter ces règles selon le domaine : un écologue pourrait privilégier des règles basées sur les interactions trophiques, tandis qu'un épidémiologiste se concentrerait sur les taux de contact et les probabilités de transmission.

Orchestration des transitions L'orchestration des transitions s'intègre naturellement dans le mécanisme de synchronisation temporelle par horloge centrale présenté dans la section 3.1.2.a. Cette intégration garantit que les transitions s'exécutent de manière cohérente avec l'évolution temporelle du système global, évitant ainsi les incohérences causales et les conflits de synchronisation.

Dans cette architecture unifiée, le scheduler central — déjà chargé de coordonner temporellement les différents niveaux de simulation — prend également en charge la planification des transitions inter-niveaux. Centraliser cette logique dans un composant unique simplifie la gestion des interdépendances et assure une cohérence systémique.

L'orchestration suit alors un cycle d'exécution séquentiel structuré en cinq étapes (illustré dans la Figure 3.10) :

1. **Initialisation du pas de simulation** : déclenchement du nouveau pas d'horloge, collecte des événements synchrones.
2. **Mise à jour des variables globales** : actualisation des indicateurs d'état partagés (environnement, métriques agrégées, etc.).
3. **Sélection des sous-modèles à exécuter** : détermination de l'ordre d'exécution des couches ou groupes d'agents pour ce pas.
4. **Vérification des conditions de transition** : pour chaque sous-modèle retenu, évaluation des règles globales de transition. Si une condition de transition est satisfaite, la fonction correspondante (agrégation ou désagrégation) est immédiatement exécutée.
5. **Exécution des sous-modèles** : lancement de la dynamique interne de chaque sous-modèle selon l'ordre défini. Au sein de cette phase, des transitions supplémentaires peuvent être déclenchées par les agents locaux, puis résolues avant de passer au sous-modèle suivant.

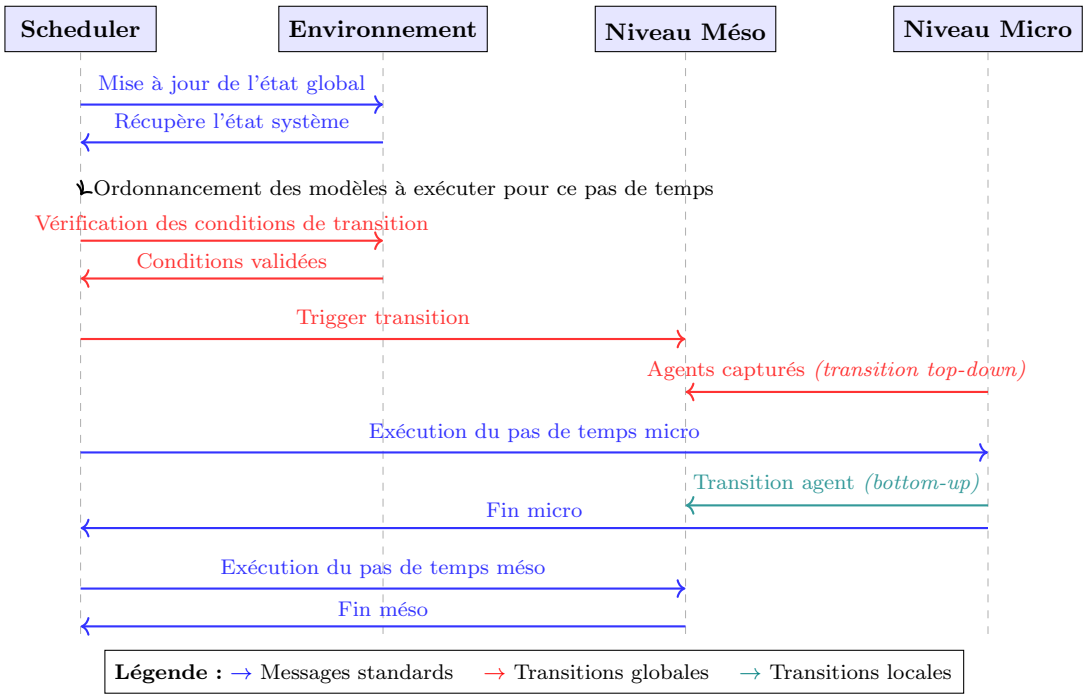

FIGURE 3.10 : Diagramme UML de séquence du processus d'orchestration des transitions dans un système à deux niveaux. Le scheduler central coordonne l'exécution séquentielle des cinq étapes : initialisation, mise à jour globale, sélection de l'ordre d'exécution, vérification des conditions avec déclenchement des transitions globales, et exécution des niveaux avec possibilité de transitions locales.

3.3 Conclusion

Le cadre méthodologique développé dans ce chapitre répond de manière systématique aux limitations identifiées dans l'analyse des approches ML-ABM existantes, tout en proposant une vision unifiée qui transcende les dichotomies traditionnelles du domaine. Cette section synthétise les contributions apportées par cette proposition et positionne ces avancées par rapport aux solutions existantes, avant d'identifier les défis pratiques que soulève leur implémentation concrète.

3.3.1 Synthèse des contributions et positionnement par rapport aux approches existantes

L'analyse critique menée au chapitre précédent, dans la section 2.4.1, a révélé un ensemble de limitations structurelles dans les approches ML-ABM contemporaines, limitations qui trouvent désormais des réponses concrètes dans le cadre proposé dans ce chapitre. La contribution principale de cette thèse réside dans la résolution systématique de ces verrous méthodologiques par une approche conceptuellement unifiée.

La rigidité architecturale qui caractérisait les design patterns existants trouve une réponse dans le système formalisé de contraintes sémantiques. Contrairement aux approches traditionnelles qui imposaient des choix architecturaux définitifs (*Zoom*, *Russian Dolls*, *Collaboration*) et en amont du développement des modèles, ce cadre permet une adaptabilité évolutive où les modélisateurs peuvent faire évoluer

l'architecture multi-niveaux selon les besoins de leur exploration. Les contraintes sémantiques (contraintes d'entités, d'attributs et de processus développée dans la section 3.1.1) offrent un vocabulaire formel suffisamment expressif pour décrire l'ensemble du continuum des patterns existants tout en autorisant des configurations hybrides innovantes.

Cette flexibilité résout également le problème de migration architecturale entre paradigmes. Là où le passage du *comodeling* au *capture/release* nécessitait une refonte complète, le cadre unifié permet des transitions fluides entre différents modes de couplage au sein d'un même modèle, selon les phases de simulation ou les besoins des utilisateurs.

La gestion manuelle des échelles spatio-temporelles, qui constituait un frein majeur à l'adoption des ML-ABM, trouve une réponse dans les mécanismes de coordination automatisée proposés (section 3.2.1). Le scheduler central et le système de contraintes spatiales déclaratives automatisent des aspects jusqu'alors laissés à la charge des modélisateurs, réduisant significativement les risques d'erreur et la complexité d'implémentation. Le système de contraintes spatiales déclaratives répond spécifiquement aux difficultés d'interopérabilité entre espaces hétérogènes, permettant aux modélisateurs d'exprimer leurs intentions de couplage spatial via un langage formel tandis que le système se charge automatiquement des transformations géométriques et des vérifications de cohérence topologique.

Ainsi, l'absence de directives standardisées pour le choix d'un pattern ou d'un outil approprié est résolue par l'unification conceptuelle proposée. Plutôt que de contraindre les modélisateurs à choisir a priori entre des paradigmes incompatibles, cette méthodologie offre un continuum de possibilités où les choix architecturaux émergent naturellement des contraintes sémantiques spécifiées. La difficulté de communication interdisciplinaire, particulièrement problématique dans les projets socio-environnementaux, trouve une réponse dans la séparation claire entre spécification sémantique et implémentation technique (la séparation entre le *quoi* et le *comment*). Les experts d'un domaine peuvent exprimer leurs contraintes dans leur vocabulaire conceptuel, tandis que le système assure automatiquement la traduction technique.

Comparée au ***comodeling*** (Huynh, 2016), l'approche conserve les avantages du couplage faible tout en résolvant ses principales limitations. La complexité d'usage est drastiquement réduite par l'automatisation des aspects techniques, tandis que l'absence de structure grammaticale explicite est comblée par le système de contraintes formalisées. Le cadre proposé peut d'ailleurs être vu comme une généralisation du comodeling, où ce dernier devient un cas particulier d'application des contraintes de collaboration et avec une gestion simplifiée des contraintes temporelles et spatiales.

Face au ***capture/release*** (Vo, 2012), le cadre transcende les contraintes hiérarchiques strictes en permettant des structures multi-niveaux flexibles. L'intégration forte au sein d'un modèle unique, source de rigidité dans le *capture/release*, est remplacée par une modularité préservant la réutilisabilité. Les transitions dynamiques proposées généralisent les mécanismes de capture et release tout en autorisant des patterns plus sophistiqués.

Le tableau 3.1 synthétise cette évolution conceptuelle et technique par rapport aux approches existantes de la plateforme GAMA. Cette comparaison révèle que le cadre proposé combine et étend les avantages des deux approches tout en résolvant leurs limitations respectives, offrant ainsi une solution unifiée plus robuste et accessible.

TABLE 3.1 : Analyse comparative détaillée des approches de coordination multi-niveaux

Critère	**Cadre proposé**	**Comodeling**	**Capture/Release**
Capacités architecturales			
Flexibilité des patterns	✓	∼	×
Migration entre paradigmes	✓	×	×
Contraintes sémantiques formelles	✓	×	×
Coordination spatio-temporelle			
Gestion automatique des échelles	✓	×	✓
Contraintes spatiales déclaratives	✓	×	∼
Synchronisation adaptative	✓	∼	×
Transitions dynamiques			
Déclencheurs configurables	✓	×	∼
Stratégies de transition flexibles	✓	×	✓
Orchestration centralisée	✓	×	×
Aspects pratiques			
Complexité d'usage	Faible∼Moyenne	Élevée	Moyenne
Réutilisabilité des composants	Moyenne	Élevée	Faible
Expressivité	Élevée	Moyenne	Moyenne
Intégration GAMA native	✓	✓	✓

3.3.2 Vers une validation expérimentale

Bien que le cadre proposé apporte des réponses conceptuelles robustes aux limitations identifiées, il convient de reconnaître que la contribution de ce chapitre demeure essentiellement méthodologique et théorique. La validation de ces propositions nécessite une implémentation concrète qui transforme ces concepts formalisés en outils opérationnels accessibles aux modélisateurs.

Le défi principal réside dans l'extension du langage GAML pour intégrer nativement les mécanismes proposés. Cette extension doit permettre aux modélisateurs de spécifier les contraintes sémantiques de manière déclarative, tout en automatisant et masquant les aspects techniques complexes de coordination spatio-temporelle et de gestion des transitions. L'enjeu consiste à concevoir une syntaxe intuitive qui masque la complexité du cadre formel tout en préservant son expressivité et une grande flexibilité d'utilisation, permettant ainsi une adoption par la communauté élargie des modélisateurs GAMA.

L'intégration à la plateforme GAMA soulève également des défis architecturaux spécifiques. L'extension doit s'articuler harmonieusement avec l'écosystème existant, préservant la compatibilité avec les modèles actuels tout en offrant les nouvelles capacités multi-niveaux.

Ces considérations motivent la nécessité d'une validation expérimentale sous la forme d'une implémentation au sein de la plateforme GAMA, ce qui constitue l'objet du chapitre suivant. Cette validation s'articule autour de l'implémentation concrète du cadre proposé sous forme d'une extension GAML, incluant la conception de nouvelles constructions syntaxiques pour les contraintes sémantiques et les mécanismes de coordination. L'évaluation portera sur la validation fonctionnelle via la réimplémentation de modèles existants (COMOKIT, SIMPLE), l'analyse de l'utilisabilité pour différents profils de modélisateurs, et la démonstration de l'accessibilité des nouveaux outils.

4

Implémentation et validation

4.1 Intégration à la plateforme GAMA

L'illustration concrète du framework proposé s'appuie sur un cas d'usage représentatif des défis rencontrés dans la modélisation multi-niveaux : l'intégration de deux modèles à base d'agents développés de manière autonome. Le premier modèle explore les dynamiques de recherche alimentaire d'une colonie de fourmis au sein de leur environnement spatial, tandis que le second modèle décrit un système proie-prédateur classique. L'objectif d'intégration consiste à établir une correspondance sémantique entre les agents `ant` du premier modèle et les agents `prey` du second modèle, permettant alors, l'émergence d'interactions entre ces modèles où les dynamiques de prédation influencent les comportements des fourmis.

Cette section présente l'intégration de ces modèles afin d'illustrer comment l'écriture unifiée proposée dans cette thèse sépare le *quoi* (logique de couplage) du *comment* (infrastructure). Elle introduit successivement : (i) des alias pour le modèle intégrateur et les sous-modèles, (ii) des imports configurés *in situ*, (iii) une centralisation des paramètres et une horloge centrale assurant la coordination temporelle, (iv) la spécialisation d'espèces pour exprimer les interactions inter-modèles, et (v) l'intégration de *capture/release* pour des dynamiques multi-niveaux.

4.1.1 Espace de noms unifié : alias du modèle intégrateur et des sous-modèles

Dans le contexte du *comodeling*, un **alias** désigne le mécanisme linguistique permettant d'associer un identifiant court aux modèles importés via le mot-clé `as`. Cette fonctionnalité constitue un élément central du couplage faible entre sous-modèles, car elle permet de facilement référencer les sous-modèles couplés et accéder à leurs `experiment` d'interface pour manipuler et orchestrer leurs interactions.

Dans le *comodeling "classique"*, l'usage des alias est essentiellement cantonné aux *imports* de sous-modèles ; ici, ce mécanisme est étendu au *classname* (`model` en en-tête de chaque modèle) du modèle intégrateur lui-même. L'alias `IntegratedModel`,

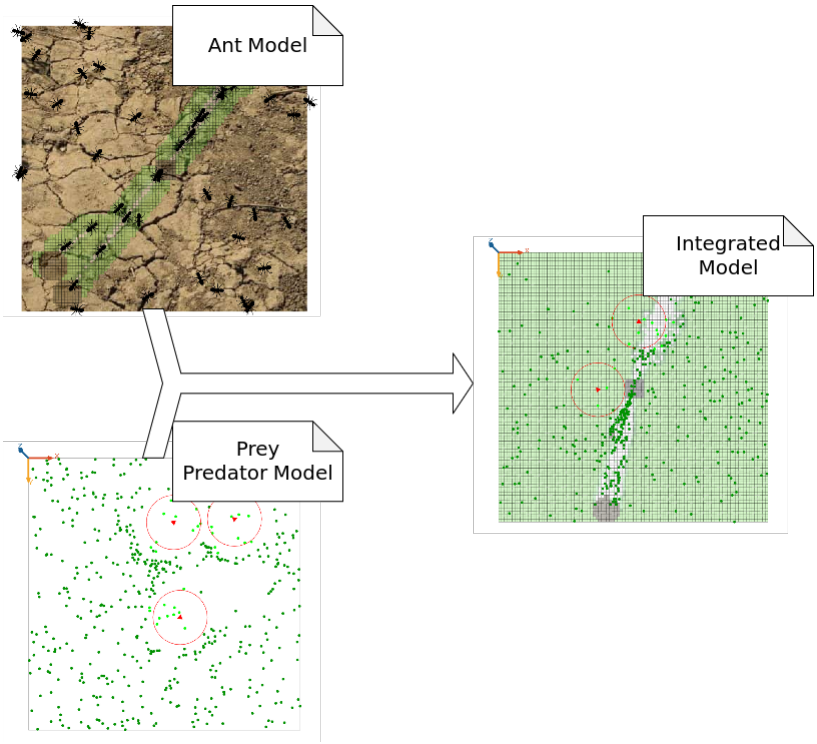

FIGURE 4.1 : Intégration de deux modèles ABM hétérogènes dans un modèle unifié. À gauche, les deux modèles sources : (haut) *Ant Model* où la colonie forme et suit une piste de phéromones (traces vertes), (bas) *Prey Predator Model* où des proies (points verts) évoluent sous la pression de prédateurs (zones d'influence matérialisées par des disques rouges). À droite, *Integrated Model* où le comportement des fourmis est perturbée dans les zones d'influence des prédateurs, illustrant l'intégration des agents `ants` et `prey`.

conjointement aux alias `AntModel` et `PreyPredModel`, offre un adressage uniforme de tous les éléments du modèle (variables globales, géométrie, espèces, fonctions utilitaires, etc.) et simplifie l'usage de références croisées entre les modèles intégrés.

```
model comodel_mix_behaviors as IntegratedModel

import "Ants Foraging.gaml" as AntModel
import "Prey Predator.gaml" as PreyPredModel
```

LISTING 4.1 – Alias explicite du modèle intégrateur

Ces alias (`IntegratedModel`, `AntModel`, `PreyPredModel`) constituent la base des sections suivantes : ils définissent un espace de nommage clair, ce qui rend la configuration *in situ* des imports plus compacte, plus robuste et plus lisible.

4.1.2 Imports configurés *in situ* et interfaces simplifiées

L'import est, également, enrichi pour configurer chaque modèle lors de son import, sans multiplier les fichiers d'interfaces ni disperser la logique dans des `experiment` et fichiers séparées. Les paramètres d'exécution et la surcharge d'espèces sont déclarés localement, rendant l'intention de couplage explicite et immédiatement lisible en plus de centraliser l'ensemble de la logique métier d'intégration dans un unique fichier.

Dans le *comodeling*, le couplage s'appuyait sur des *fichiers d'interface* (généralement un par sous-modèle) au sein desquels des `experiment` spécifiques jouaient le rôle d'adaptateurs logiciels. Ces expériences servaient l'**orchestration normalisée** des interactions entre sous-modèles : elles prenaient en charge l'initialisation des composants importés, le passage de paramètres (par exemple, listes d'agents ou indicateurs agrégés), harmonisaient les unités et pas de temps, etc. Cette couche d'interface centralisait la logique d'adaptation, au prix toutefois d'une dispersion de la logique du couplage dans plusieurs fichiers (interfaces, expériences et fragments d'import).

La réécriture unifie cette étape au sein d'un *fichier intégrateur unique*. La configuration est réalisée au point d'import, et la coexistence avec les autres éléments intégrés permet de réutiliser des attributs et croiser des références, simplifiant et fluidifiant d'autant le travail de modélisation.

```
model comodel_mix_behaviors as IntegratedModel

import "Ants Foraging.gaml" as AntModel {
    float step <- IntegratedModel.step / 2;
    int gridsize <- IntegratedModel.worldSize;
}
```

Listing 4.2 – Import et configuration locale du modèle des fourmis

Dans le Listing 4.2, le modèle *Ants Foraging* est importé avec l'alias `AntModel`. Également, la coordination des échelles temporelles et spatiales sont explicitement données pour le sous-modèle importé en se basant sur des variables du modèle importateur.

```
model comodel_mix_behaviors as IntegratedModel

import "Prey Predator.gaml" as PreyPredModel {
    float step <- IntegratedModel.step;
    geometry shape <- IntegratedModel.shape;

    // Add model's function (i.e getters)
    list<predator> get_predator { return list(predator); }
}
```

Listing 4.3 – Import et configuration locale du modèle proies-prédateurs

Le Listing 4.3 illustre l'import du modèle *Prey Predator* avec l'alias `PreyPredModel`. L'initialisation établit d'emblée, là encore, la cohérence spatio-temporelle (pas de temps et géométrie hérités du modèle intégrateur), tandis que deux accesseurs utilitaires exposent proprement les populations d'intérêt (proies et prédateurs) pour la visualisation et l'instrumentation.

Il convient de préciser que la redéfinition du pas de temps lors de l'import (variable `step` redéfinie dans les listing 4.2 et 4.3) ne constitue pas une obligation du framework. Cette configuration explicite est toutefois *recommandée* car elle centralise et rend immédiatement lisible l'ensemble des relations temporelles entre sous-modèles au sein du fichier intégrateur, facilitant ainsi la compréhension et la maintenance de la logique de couplage. En l'absence de redéfinition explicite, le framework préserve automatiquement le pas de temps défini originellement dans chaque sous-modèle, garantissant la compatibilité ascendante avec les modèles existants tout en permettant une adoption progressive de cette approche unifiée.

4.1.3 Centralisation des paramètres et initialisation unifiée

Une autre amélioration, découlant des points précédents, concerne la centralisation des paramètres en un lieu unique : le *modèle intégrateur*. Celui-ci devient alors le point de configuration principal, où sont définis les paramètres propres au modèle intégré (comme sa taille, la forme de l'espace, la durée d'un pas de temps, etc.) ainsi que l'initialisation synchronisée des sous-modèles importés.

```
global {
    // Création de variables globales du modèle intégré
    int worldSize <- 100;
    geometry shape <- square(worldSize);

    float step <- 1#min;

    // Initialisation du modèle intégré
    init {
        // Création des sous-modèles précédemment importés
        create AntModel;
        create PreyPredModel;
    }
}
```

Listing 4.4 – Paramètres globaux et initialisation unique du modèle intégré

Le Listing 4.4 montre cette centralisation des paramètres spatiaux (`worldSize`, `shape`) qui ont été réutilisés lors de l'import des sous-modèles (Listings 4.2 et 4.3) et la création des sous-modèles importés dans le `init` du modèle.

4.1.4 Ordonnancement multi-échelles et horloge centrale

La coordination temporelle est assurée par une horloge centrale (décrite dans la section 3.1.2.a) : le modèle intégrateur fournit un point de référence (`Integrated Model.step`), à partir duquel chaque sous-modèle peut déclarer sa propre granularité relative. Ensuite, le framework s'occupe (depuis ces valeurs ou en recupérant la valeur du `step` de chaque sous-modèle) de re-séquencer l'exécution de l'ensemble des modèles intégrés. Cela retire, ainsi, aux modélisateurs la gestion bas niveau de la synchronisation entre sous-modèles.

```
model comodel_mix_behaviors as IntegratedModel

import "Ants Foraging.gaml" as AntModel {
    // [...]
    // Dans l'import des fourmis
    float step <- IntegratedModel.step / 2;
}

import "Prey Predator.gaml" as PreyPredModel {
    // [...]
    // Dans l'import proies-prédateurs
    float step <- IntegratedModel.step;
}
```

Listing 4.5 – Synchronisation temporelle relative aux sous-modèles

Le Listing 4.5 explicite une fréquence plus fine pour les agents fourmis (un demi-pas du modèle intégrateur) et une fréquence de base pour les agents proies/prédateurs. Le détail d'ordonnancement est encapsulé par le framework ; le *quoi* du couplage se concentre sur les interactions, non sur la micro-gestion des pas.

4.1.5 Spécialisation d'espèces et articulation inter-modèles

Cette amélioration formalise un schéma de spécialisation au sein du modèle intégré : le modélisateur peut définir une nouvelle espèce qui *hérite* (via la facet `parent`) d'une espèce d'un sous-modèle afin de l'adapter au contexte d'intégration. Cette spécialisation peut prendre différentes formes selon les besoins du modélisateur allant de la modification d'affichage au retravail complet du comportement de l'agent en passant par, plus systématiquement, la modification des intéractions de l'agent pour intégrer les agents des autres modéles couplés.

```
import "Prey Predator.gaml" as PreyPredModel {
    // [...]
    substitute prey as IntegratedModel.escaping_ant;
}

species escaping_ant parent: PreyPredModel.prey mirrors:
   AntModel.ants {
    // Replace default prey's movement
    reflex move {
        if (self.is_chased) {
            // Prey escaping
            target.location <- location;
        } else {
            // Ant Movement
            do target.move;
            location <- target.location;
        }
    }
}
```

LISTING 4.6 – Surcharge d'espèce et liens inter-modèles

Dans le Listing 4.6, `escaping_ant` est une espèce spécialisée de `PreyPredModel.prey` d'origine du modèle de prédation, et `mirrors: AntModel.ants` associe à chaque proie un *miroir* d'une fourmi issue de `AntModel` (un lien de correspondance entre sous-espèces). La logique métier (le couplage entre les agents `ants` et `prey`) s'écrit directement en termes de comportement de l'agent spécialisé (déplacement de l'agent et mise à jour de la position du miroir), sans manipuler d'API bas niveau de couplage.

Le mécanisme de `substitute` permet la substitution transparente d'espèces lors de l'intégration de sous-modèles. Cette fonctionnalité résout élégamment le problème de la spécialisation comportementale sans modification des modèles sources, en permettant au modélisateur de *remplacer* une espèce existante par une espèce spécialisée pour ce couplage au moment de l'importation. La syntaxe `substitute ant as IntegratedModel.specialized_ant` dans le bloc d'import du sous-modèle indique que tous les agents de type `ant` du sous-modèle importé seront automatiquement remplacés par des instances de l'espèce `specialized_ant` définies dans le modèle intégrateur lors de l'exécution du modèle multi-niveaux. Cette substitution s'opère

de manière transparente pour le sous-modèle, qui continue de faire référence à ses agents `ant` originaux, mais manipule en réalité les instances spécialisées enrichies des capacités de couplage.

4.1.6 Opérationnalisation des dynamiques multi-niveaux par *capture/release*

L'intégration des sous-modèles au sein d'un *unique* modèle rend immédiatement opérationnels les mécanismes de *capture/release* (présentés au Chapitre 2.3.4) pour enrichir les dynamiques multi-niveaux. Premièrement, l'une des contraintes fondamentales de *capture/release* (à savoir que ces opérations ne sont applicables qu'à l'intérieur d'un même modèle) est ici levée par construction : le couplage préalable fournit l'espace d'exécution unifié requis, de sorte que la capture d'agents et leur relâchement ultérieur peuvent être spécifiés de manière déclarative et sûre.

De plus, l'emploi de *capture/release* dans ce cadre couplé permet de pleinement exploiter l'expressivité de ce framework pour formuler des dynamiques multi-échelles, qui constituent la finalité première de cet outil. Il devient simple, par exemple, d'opérationnaliser des phases d'agrégation/désagrégation temporaires (constitution d'un collectif transitoire, mise à l'abri ponctuelle, regroupement fonctionnel), d'orchestrer des changements de rôle et de régime (prise en charge d'agents par une entité de niveau supérieur durant un intervalle limité), ou d'implémenter des calculs méso ou macro-échelles sur des ensembles capturés avant restitution au niveau micro. L'horloge centrale et la centralisation des paramètres assurent que ces transitions se produisent à des instants bien définis, dans un espace et avec des unités, qui sont cohérents avec l'ensemble du modèle intégré.

Troisièmement, *capture/release* fournit un socle adapté pour exprimer, au sein du modèle couplé, les contraintes formalisées dans le chapitre méthodologique : les conditions d'entrée et de sortie (seuils d'état, événements, fenêtres temporelles), domaines spatiaux de validité (zones d'autorité de l'hôte, géométries de capture), invariants de conservation (identité des agents, quantités conservées), politiques d'ordonnancement (priorités, fréquences) et garanties d'intégrité (restauration des attributs lors de la désaggrégation, cohérence des références). La spécification de ces contraintes demeure locale et lisible : la *logique sémantique* de la transition est explicitée au niveau des espèces concernées, tandis que l'infrastructure d'exécution (déjà unifiée par le couplage) en assure la mise en œuvre de façon déterministe et traçable.

Ainsi, l'articulation "couplage logiciel (*comodeling*) → dynamiques multi-niveaux (*capture/release*)" constitue une stratégie générale : le couplage fournit l'unicité de contexte exigée par *capture/release* ; le *capture/release* apporte, en retour, le langage opérationnel pour modéliser des reconfigurations multi-niveaux sophistiquées, conformes aux contraintes sémantiques, spatiales et temporelles exigées par la méthodologie ML-ABM, sans réintroduire de complexité logicielle superflue.

4.1.7 Expérimentation et rendu du modèle intégré

Enfin, comme pour les modèles du *comodeling* ou n'importe quel modèle GAMA, une `experiment` unique permet de simuler et de visualiser le modèle intégré avec les différents outils de visualisation de la plateforme, en combinant les alias et les

accesseurs déclarés à l'importation des sous-modèles. Le résultat est une exploitation *standard* des capacités de sortie, sans traitement *ad hoc*.

```
experiment coupled_model type: gui {
    output {
        display map {
            // Affichage des agents spécialisé
            species escaping_ant;

            // Affichage des agents des sous-modèles
            species PreyPredModel.predator;
        }
    }
}
```

LISTING 4.7 – Expérience et affichage unifiés

Le Listing 4.7 illustre (i) l'affichage direct de l'espèce couplée `specialised_ant`, et (ii) la récupération des populations proies/prédateurs via les alias et accesseurs définis dans `PreyPredModel`. Il est bon de souligner que l'écriture de l'expérimentation et de l'affichage d'un modèle multi-niveaux s'écrit avec les mêmes syntaxe et constructions que celles utilisées pour tout modèle GAMA standard, qu'il soit mono-niveau ou multi-niveaux, masquant ainsi intégralement la complexité architecturale sous-jacente au bénéfice d'un usage homogène des modèles.

4.1.8 Impacts sur l'ergonomie et le coût d'adoption

L'ensemble de ces évolutions transforme fondamentalement le travail de couplage (et de l'usage du *comodeling*) pour le modélisateur en réduisant drastiquement la barrière d'entrée technique. Là où l'approche initiale nécessitait une expertise avancée en programmation et en architecture logicielle, cette syntaxe permet aux modélisateurs de se concentrer sur les aspects conceptuels (le *quoi* du couplage) de leur domaine d'application.

Cette simplification ne constitue pas une réduction des capacités du système, mais plutôt une réorganisation de la complexité. Les aspects techniques complexes (gestion des cycles de vie des agents, synchronisation des pas de temps, correspondances spatiales) sont désormais assurés par ce framework plutôt que délégués aux modélisateurs. Mais les modélisateurs doivent toujours assurer et expliciter le comportement, les interactions des agents issus des sous-modèles et les dynamiques de modélisation au sein du modèle couplé.

4.2 Application aux design patterns ML-ABM

Cette section démontre comment le framework développé permet à un modèle d'évoluer naturellement le long du continuum des architectures ML-ABM identifiées dans la littérature (section 2.2). À travers l'exemple du modèle *Ant Foraging* de GAMA, nous illustrons comment les transitions entre architectures *Zoom*, *Russian Dolls* et *Collaboration* répondent à des besoins de modélisation plutôt qu'à des contraintes techniques.

4.2.1 Modèle de base et formalisation

Le modèle *Ant Foraging* simule le comportement de recherche de nourriture des fourmis via un mécanisme de stigmergie basé sur les phéromones. Dans sa version originale, ce modèle opère exclusivement au niveau microscopique, où chaque fourmi est un agent autonome naviguant dans l'environnement. Ce modèle canonique servira de fondation pour illustrer les transitions vers des architectures multi-niveaux progressivement plus complexes.

Ce modèle simule une colonie de fourmis explorant un environnement spatial à la recherche de sources de nourriture. Les fourmis naviguent individuellement dans un espace bidimensionnel, déposent des phéromones pour marquer leurs trajectoires, et retournent au nid pour déposer la nourriture collectée. L'objectif principal est de modéliser l'émergence de pistes optimisées de recherche collective via les mécanismes de stigmergie.

Le modèle comprend trois types d'entités principales :

— **Fourmis** (`ant`) : agents mobiles caractérisés par leur position, leur direction de déplacement, leur charge de nourriture transportée, et leur état comportemental (recherche, retour au nid) ;
— **Sources de nourriture** (`food_source`) : entités statiques créées au démarrage de la simulation et définies par leur position et leur quantité de nourriture disponible ;
— **Nid** (`nest`) : entité centrale où les fourmis déposent la nourriture collectée et où elles sont initialement créées.

L'environnement spatial est discrétisé en cellules portant des concentrations de phéromones qui s'évaporent au cours du temps et influencent les déplacements des fourmis.

Le modèle opère sur un espace bidimensionnel de dimension paramétrable (dans notre cas, une grille de 100×100 cellules). Chaque pas de simulation représente une unité de temps arbitraire (ici, un pas de temps dure *une seconde*) pendant laquelle chaque fourmi peut se déplacer sur la carte et effectuer une action (collecter, déposer, se déplacer).

Processus comportementaux. Les fourmis suivent un cycle comportemental simple basé sur des règles locales :

1. **Exploration** : recherche de nourriture en privilégiant les zones à forte concentration de phéromones ;
2. **Collecte** : prélèvement de nourriture lorsqu'une source est découverte ;
3. **Retour** : navigation vers le nid en déposant des phéromones de marquage ;
4. **Dépôt** : transfert de la nourriture au nid et reprise du cycle d'exploration.

Les décisions de déplacement combinent trois composantes : attraction par les phéromones, mouvement aléatoire de diffusion, et orientation directionnelle vers les objectifs connus.

Le comportement collectif émerge de l'interaction entre les dépôts de phéromones individuels et les réponses comportementales locales. Sans coordination explicite, le système génère des pistes efficaces reliant le nid aux sources de nourriture, avec optimisation progressive des trajets selon leur utilisation collective.

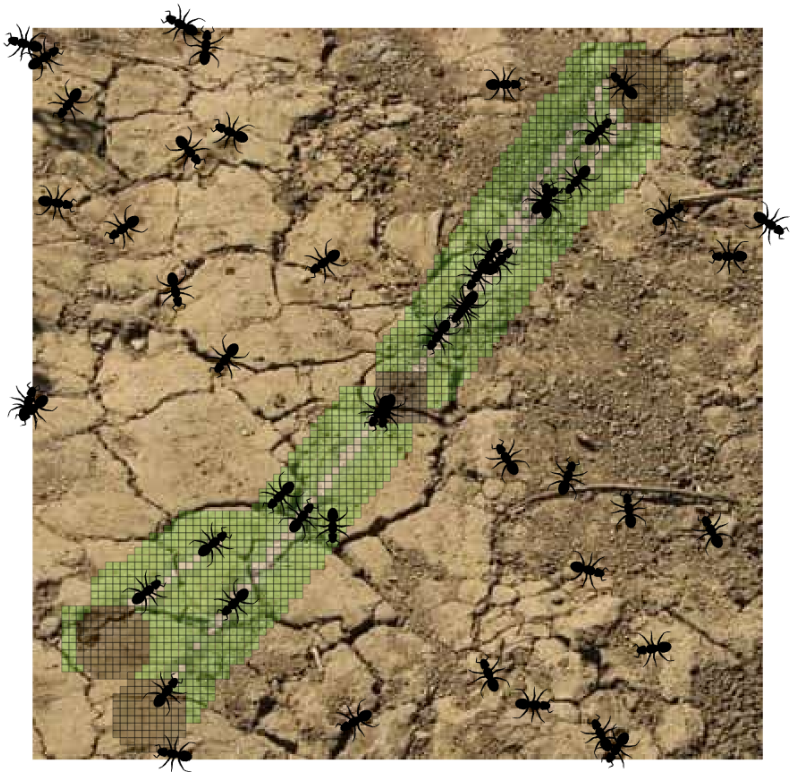

FIGURE 4.2 : Capture d'écran du modèle *Ant Foraging*. Le `nest` est situé au centre de l'image ; les points sombres correspondent aux `food_source`, et les traces vertes superposées matérialisent les concentrations de phéromones déposées puis suivies par les agents `ant`.

La figure 4.3 présente le modèle *Ant Foraging* et servira de référence pour examiner comment les extensions multi-niveaux modifient et enrichissent ces dynamiques fondamentales.

4.2.2 Architecture Zoom : gestion multi-vue et transitions utilisateur

4.2.2.a Motivation et évolution naturelles

L'architecture Zoom émerge naturellement lorsque les modélisateurs souhaitent analyser leur système à différents niveaux de granularité selon le contexte d'observation. Son utilisation permet de répondre à des questions de modélisation comme :

Comment optimiser l'analyse selon le niveau de détail requis ?

Dans notre cas d'étude, cette nécessité apparaît quand l'analyse détaillée des comportements individuels (pistes de phéromones, congestion locale) doit alterner avec une vue d'ensemble des performances de collecte.

Pour répondre à ce besoin, nous introduisons une représentation mésoscopique où la colonie de fourmis est abstraite en une entité unique dotée de "bras" dynamiques représentant les flux de recherche. Cette dualité permet à l'utilisateur de basculer entre une vue détaillée (toutes les fourmis individuelles) et une vue agrégée (la colonie comme blob avec des extensions), optimisant ainsi les ressources computationnelles et la charge cognitive selon les besoins d'analyse.

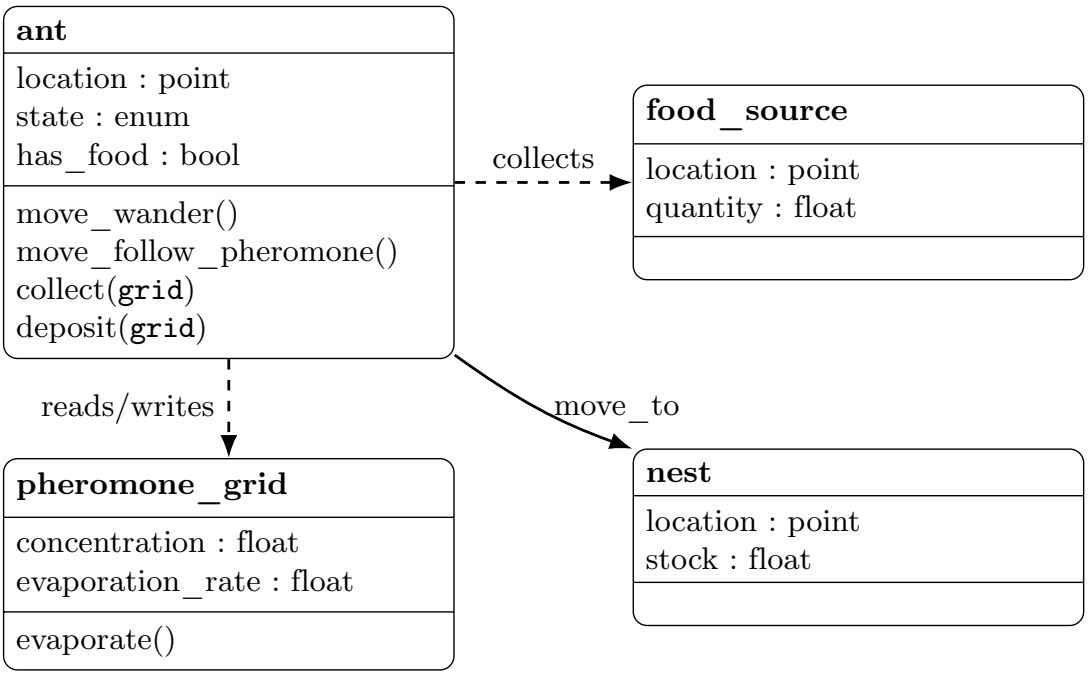

FIGURE 4.3 : Architecture UML du modèle de base *Ant Foraging*.

4.2.2.b Mécanismes de transition et contraintes

La transition du niveau micro vers le niveau méso suit un processus rigoureux en trois étapes : capture de l'état microscopique, transformation des données, et création de la représentation mésoscopique. Cette transformation conserve deux contraintes fondamentales du système.

Premièrement, la **conservation de la matière** (*contraintes de cohérence*) impose que la quantité totale de nourriture dans le système reste identique avant et après la transition. Toute la nourriture portée individuellement par les fourmis doit se retrouver dans le stock agrégé de l'agent colonie. Deuxièmement, la **cohérence spatiale** exige que la position de l'agent colonie corresponde au centre de gravité de la distribution des fourmis individuelles, préservant ainsi la localisation géographique du groupe et la cohérence entre les différentes vues du modèle.

La transition micro vers méso nécessite qu'au moins une fourmi soit présente dans le système et qu'aucun agent colonie ne soit déjà actif (*conditions d'exécution*). Après transition, le système doit contenir exactement un agent colonie dont le stock correspond à la nourriture totale précédemment distribuée. Les cas particuliers peuvent être traités spécifiquement comme : l'absence de fourmis (création d'une colonie sans extensions), et des positions aberrantes de la colonie (recalage automatique).

Enfin, les transitions ne s'effectuent qu'entre deux pas de temps (méchanisme induit par le fonctionnement de GAMA), et á la demande de l'utilisateur (comme par exemple le modélisateur qui clique sur un bouton précis). Le modèle ne décide pas automatiquement de basculer d'une vue à une autre, mais c'est l'utilisateur qui pilote ces transitions selon ses besoins.

4.2.2.c Implémentation de la représentation mésoscopique

L'architecture multi-niveau nécessite d'abord l'importation du modèle microscopique de base, puis la définition d'une représentation mésoscopique alternative.

```
model AntForagingZoom as ZoomModel

// Import du modèle microscopique existant
import "ant_foraging_micro.gaml" as MicroModel {}

global {
    bool micro_active <- true;  // Contrôle du niveau actif

    init {
        // Démarrage par défaut au niveau micro
        create MicroModel;
    }
}
```

Listing 4.8 – Import du modèle micro et structure globale

Cette structure d'import permet de réutiliser le modèle de fourmis existant tout en définissant une architecture multi-niveaux par-dessus. Le niveau mésoscopique abstrait la colonie de fourmis en une entité unique dotée de "bras" dynamiques qui représentent les directions principales d'exploration collective. Cette représentation simplifie drastiquement les complexités visuelle et computationnelle tout en conservant les propriétés émergentes essentielles du système.

```
species anthill {
    point center <- {50, 50};
    list<arm> active_arms;
    float collected_food <- 0.0;  // Stock agrégé de toute la
   colonie

    reflex update_arms {
        ask active_arms {
            do extend;  // Les bras s'étendent vers les zones
   prometteuses
            if (found_food) {
                // La nourriture collectée par un bras est ajout
   ée au stock global
                myself.collected_food <- myself.collected_food +
    do collect;
                do leave_pheromones;
            }
        }
    }

    // Micro-species
    species arm parent: MicroModel.ant schedule: [] {
        point origin;    // Point de départ du bras (centre de
   la colonie)
        float strength;  // Intensité abstraite représentant le
   nombre de fourmis
        bool found_food <- false;

        // Reste de la logique de comportement du bras
```

```
        // [...]
    }
}
```

Listing 4.9 – Structure du niveau mésoscopique

Ce code définit la représentation mésoscopique où la colonie devient un agent unique avec des extensions spatiales (les bras) qui évoluent dynamiquement. Chaque bras représente un flux d'exploration collectif plutôt que des fourmis individuelles.

La transition du niveau micro vers le niveau méso implique plusieurs étapes critiques pour maintenir la cohérence du modèle :

```
global {
    // [...]
    action aggregate_to_meso {
        // Analyse de la distribution spatiale des fourmis
        map<point, int> clusters <-
            compute_clusters(MicroModel.ant collect each.
    location);
        float total_food <-
            sum(MicroModel.ant collect each.carrying_food);

        // Création du niveau méso avec transfert d'état
        create anthill number: 1
            with: ( collected_food: total_food,
                location: clusters.keys mean_of each )
        {
            // Création des bras selon les zones d'activité dé
    tectées
            loop arm_cluster over: clusters.keys {

                // Utilisation des mecanismes de Capture/Release
                capture MicroModel.ant where (each distance_to
    cluster < 10) as: arm {
                    // Centre de l'agent anthill
                    origin <- myself.location;
                    shape <- cluster.envelope;
                    // Nombre de fourmis dans ce cluster
                    strength <- ant count (each distance_to
    cluster < 10);
                }
            }

            active_arms <- arm;
        }
    }
}
```

Listing 4.10 – Transition micro vers méso avec vérifications

Cette implémentation illustre comment la transition préserve les deux contraintes fondamentales : le respect des contraintes de cohérence (la conservation de la nourriture qui est transférée entre les niveaux, et la cohérence spatiale). La détection de clusters permet de créer des bras orientés vers les zones où les fourmis étaient concentrées, préservant ainsi l'information spatiale importante.

La transition inverse (désagrégation du niveau méso vers micro) suit la même logique conceptuelle : distribution spatiale des fourmis capturées dans et autour des bras, répartition de la nourriture stockée, et vérification des contraintes de cohérence, le tout avec l'usage du *statement* `release`. Par souci de concision, l'implémentation ne sera pas détaillée ici.

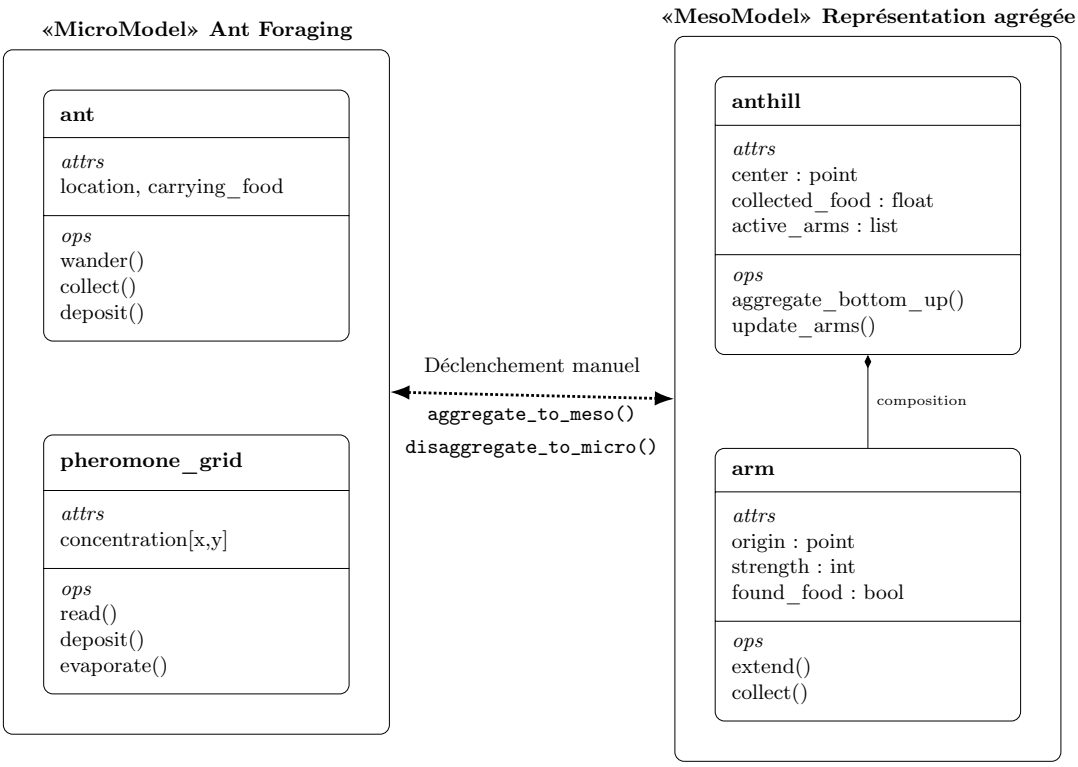

FIGURE 4.4 : Architecture UML du pattern *Zoom* : bascule entre les représentations micro ↔ méso pilotée par l'utilisateur.

Le modèle se termine par une `experiment` permettant de visualiser ce nouveau modèle et de déclencher manuellement les transitions entre niveaux :

```
experiment zoom_interface type: gui {
    action zoom_to_meso {
        do aggregate_to_meso;
        micro_active <- false;
    }

    action zoom_to_micro {
        do disaggregate_to_micro;
        micro_active <- true;
    }

    output {
        display main_view {
            // Affichage des éléments du modele importé
            species MicroModel.food_source;
            species MicroModel.nest;

            species MicroModel.ant;

            // Affichage du nouveau niveau méso
```

```
            species anthill;
            species arm;

            // Intéraction de transition entre les niveaux
            event #mouse_down {
              if (micro_active) {
                  do zoom_to_meso;
              } else {
                  do zoom_to_micro;
              }
            }
        }
    }
}
```

Listing 4.11 – Expérience et contrôles utilisateur

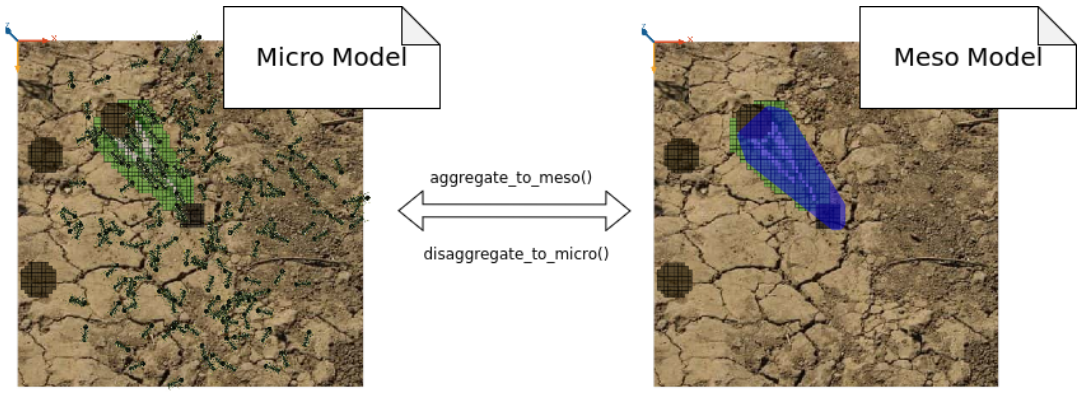

Figure 4.5 : Implémentation du pattern *Zoom*.
À gauche (vue micro), les fourmis individuelles interagissent avec les sources de nourriture et suivent les pistes de phéromones (zones vertes) sur la grille spatiale. À droite (vue méso), la colonie est agrégée en fourmilière de plus haut niveau (en bleu) résumant les flux d'exploration. Les flèches centrales indiquent les transitions déclenchées par l'utilisateur via les fonctions dans l'`experiment` : `aggregate_to_meso()` et `disaggregate_to_micro()`.

4.2.3 Architecture Russian Dolls : coexistence et rétroactions contrôlées

4.2.3.a Évolution naturelle vers la coexistence

L'architecture *Russian Dolls* émerge lorsque les modélisateurs sont limités par l'alternance (destructrice) entre niveaux du *Zoom* qui masque des phénomènes importants. La question de recherche évolue vers :

Comment capturer les rétroactions en temps réel entre les comportements individuels et les stratégies collectives ?

Contrairement au *Zoom* où un seul niveau est actif à la fois, cette architecture permet aux deux niveaux de coexister et de s'influencer mutuellement en temps réel. Les fourmis individuelles opèrent autour des bras de la colonie agrégée, créant une

dynamique où le comportement émergeant du niveau micro informe continuellement le niveau méso, qui en retour structure les comportements individuels.

4.2.3.b Mécanismes de synchronisation bidirectionnelle

Dans cette architecture, les niveaux coexistent de manière permanente : les fourmis individuelles continuent d'évoluer pendant que l'agent colonie agrégé maintient une représentation globale mise à jour en temps réel. Cette coexistence permet des influences bidirectionnelles continues entre les échelles.

La synchronisation repose sur deux mécanismes complémentaires qui assurent la stabilité du système. D'une part, l'agrégation contrôlée limite la fréquence de mise à jour de l'agent colonie pour éviter les oscillations numériques. D'autre part, les rétroactions descendantes (du niveau méso vers micro) sont bornées par des seuils pour prévenir les comportements chaotiques.

4.2.3.c Implémentation de la coexistence temporelle

L'architecture *Russian Dolls* étend le modèle *Russian Dolls* en maintenant les deux niveaux simultanément actifs. Cette coexistence nécessite une évolution du modèle précédent et l'ajout de mécanismes de synchronisation entre les niveaux.

```
model AntForagingRussianDolls as RussianDollsModel

// Réutilisation du modèle micro existant
import "ant_foraging_micro.gaml" as MicroModel {}

global {
    init {
        // Création simultanée des deux niveaux
        create anthill number: 1;
        create MicroModel;
    }
}
```

Listing 4.12 – Import et structure pour la coexistence

Le couplage bidirectionnel s'articule autour de deux flux d'information principaux : influence descendante (top-down) et remontée d'informations (bottom-up). La première peut s'écrire ainsi :

```
species anthill {
    float stock <- 0.0;
    float influence_threshold <- 50.0;
    int update_frequency <- 5;  // Contrôle de la fréquence d'
   agrégation

    reflex aggregate_bottom_up when: cycle mod update_frequency
   = 0 {
        list<agent> ants_in_arms <- MicroModel.ant where (each
   distance_to self < 10);

        // Remontée micro -> méso : agrégation de l'état des
   fourmis
        stock <- sum(ants_in_arms collect each.carrying_food);
```

```
        // Adaptation des bras selon la distribution réelle des
    fourmis
        loop single_arm over: active_arms {
            list<ant> ants_in_zone <- MicroModel.ant
                where (each distance_to single_arm < 10);
            // Force du bras = nb fourmis
            single_arm.strength <- length(ants_in_zone);
            // Update arm's shape
            single_arm.shape <- ants_in_zone.envelope;
        }
    }
    // [...]
}
```

LISTING 4.13 – Synchronisation top-down

En revanche, l'implémentation *bottom-up* vient modifier le comportement propre des agents `ant`. De fait, le modélisateur doit créer une nouvelle espèce d'agent fourmis, qui hérite de tous les comportements des fourmis, mais dont le comportement est spécialisé pour respecter l'influence de l'agent `anthill`. Les fourmis individuelles restent autonomes tout en respectant l'influence du niveau méso :

```
import "ant_foraging_micro.gaml" as MicroModel {
    // Remplace les agents 'ant' par les nouveaux '
    specialised_ants'
    substitute ant as RussianDollsModel.specialised_ants;
}

// [...]

species specialised_ants parent: MicroModel.ant {
    anthill colony <- first(anthill);

    // Influence méso -> micro
    // Surcharge l'état 'wandering' pour rester dans la zone des
    bras
    state wandering initial: true {
    do wander on:
        // Récupère le bras le plus proche
        // et considère une surface plus large autour du bras
        any(colony.active_arms where (each distance_to self <
    10)).shape + 10;

    // Garde le reste de la logique identique
    float pr <- (ant_grid(location)).road;
    transition to: carryingFood when: has_food;
    transition to: followingRoad when: (pr > 0.05) and (pr < 4);
  }
}
```

LISTING 4.14 – Spécialisation de l'espèce `ant` et synchronisation bottom-up

Ce code illustre la synchronisation bidirectionnelle : l'agent colonie observe continuellement l'état des fourmis individuelles (agrégation top-down) et influence en retour le comportement d'une nouvelle espèce spécialisée (`specialized_ant`) par sa propre position.

Cette implémentation montre comment les fourmis conservent leur autonomie comportementale tout en étant subtilement influencées par les structures mésoscopiques. L'attraction vers les bras de la colonie créé une coordination émergente sans contrainte rigide.

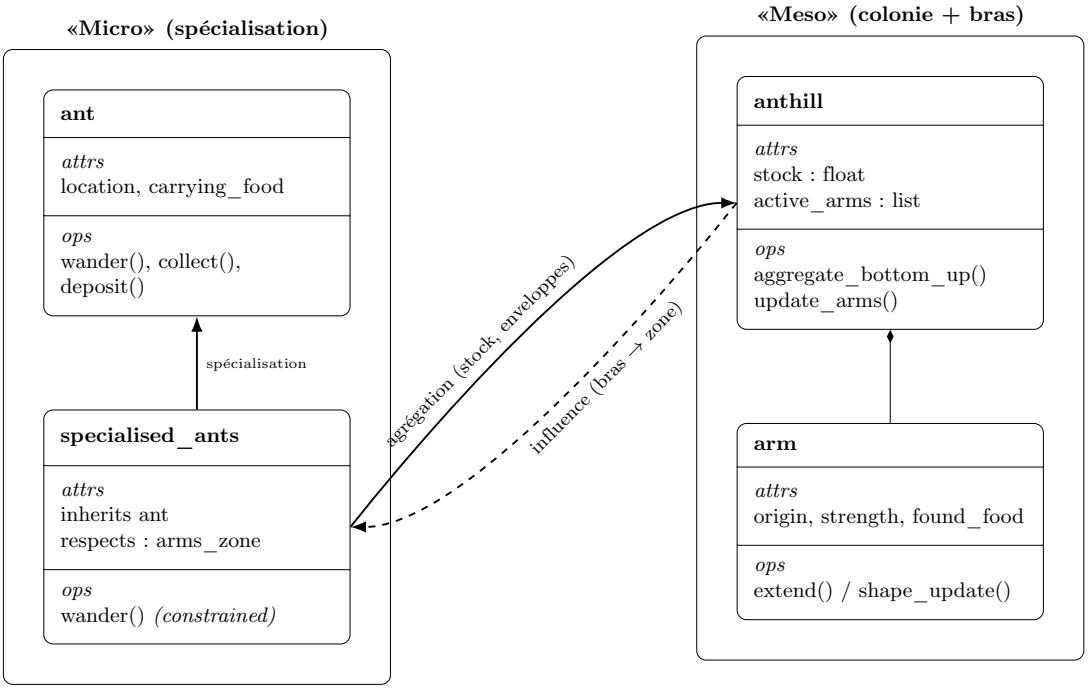

FIGURE 4.6 : Architecture UML du pattern *Russian Dolls* implémenté : coexistence micro–méso avec flux de données bottom-up (agrégation) et top-down (influence).

Enfin, l'`experiment` peut être adaptée pour permettre d'observer les agents de ce modèle et enlever les commandes de bascule entre les niveaux :

```
experiment russian_dolls_interface type: gui {
    output {
        // Affichage simultané des deux niveaux
        display main_view {
            // Affichage des espèces micro
            species MicroModel.food_source;
            species MicroModel.nest;

            // Affichage des espèces défini dans ce modèle
            species anthill;
            species anthill.arm transparency: 0.3;
            species specialised_ants transparency: 0.3;
        }
    }
}
```

LISTING 4.15 – Expérience pour l'architecture *Russian Dolls*

Il est utile de remarquer que le passage de l'architecture *Zoom* vers *Russian Dolls* s'est effectué par évolution progressive du code existant, sans remise à zéro. La transition a consisté, principalement, à cesser de détruire les agents micro lors du basculement vers le niveau méso, puis à ajouter les mécanismes de synchronisation bidirectionnelle. Les structures de données (agents fourmis, colonie, bras) ont été

réutilisées, seuls les réflexes de coordination ont été ajoutés. Cette capacité d'évolution incrémentale démontre la flexibilité du framework : les modélisateurs peuvent faire évoluer leur architecture selon leurs besoins de recherche sans repartir de zéro, ou être limités par des choix logiciels faits au début du développement.

FIGURE 4.7 : Implémentation du pattern *Russian Dolls*.
Les deux niveaux coexistent simultanément : l'essaim de fourmis individuelles (niveau micro) se déplace dans l'environnement et interagit avec les `food_source`, tandis que les bras de la fourmilière (en bleu) influence la dynamique collective (les fourmis sont attirées). La grille/zone verte environnante matérialise les pistes de phéromones.

4.2.4 Architecture Collaboration : couplage faible entre modèles hétérogènes

4.2.4.a Évolution vers l'intégration externe

L'utilisation de l'architecture *Collaboration* permet, par exemple, de répondre à une question comme :

Comment intégrer l'impact de dynamiques externes complexes dans ce modèle de fourmis ?

Cette évolution naturelle survient quand les modélisateurs veulent complexifier leur système pour qu'il n'évolue plus en vase clos, mais en interaction avec un(des) autre(s) modèle(s).

Dans notre cas, nous introduirons un modèle Prey-Predator où les colonies de fourmis deviennent des proies potentielles. Cette intégration illustre la capacité du framework à coupler des modèles conceptuellement distincts, et à des échelles différentes, via une interface de couplage minimale, préservant l'autonomie de chaque sous-système, et par une évolution du modèle déjà créé.

4.2.4.b Mécanismes de couplage faible avec un système externe

L'architecture *Collaboration* étend le modèle *Russian Dolls* existant en y intégrant un système externe autonome. Cette extension nécessite la définition d'un contrat d'interface qui spécifie les échanges d'information tout en préservant l'autonomie de chaque sous-système.

Le contrat d'interface définit trois éléments essentiels : les sorties que le système de fourmis transmet vers le modèle externe (nombre d'agents, positions, etc), les entrées qu'il reçoit en retour (événements de prédation, intensité des menaces), et le lien entre sous-espèces de chaque modèle (une colonie de fourmis va avoir un lien direct avec les agents `prey`).

Cette approche de couplage faible permet d'intégrer des dynamiques complexes (comme un modèle proies-prédateurs) sans modifier le fonctionnement interne de chaque modèle, préservant ainsi la modularité et la réutilisabilité des composants.

4.2.4.c Implémentation du couplage multi-modèles

Tout comme les patterns précédents, l'architecture *Collaboration* utilise les mécanismes natifs de GAMA pour créer un couplage entre modèles hétérogènes. Le framework permet d'importer plusieurs modèles indépendants et de les faire interagir via des espèces spécialisées de couplage.

```
model comodel_mix_behaviors as CollaborationModel

// Import du modèle Russian Dolls avec configuration temporelle
import "ant_foraging_russian-dolls.gaml" as AntModel {
    float step <- CollaborationModel.step / 2;  //
    Synchronisation temporelle
    int gridsize <- CollaborationModel.worldSize;
}

// Import du modèle proies-prédateurs
import "Prey Predator.gaml" as PreyPredModel {
    float step <- CollaborationModel.step;
    geometry shape <- CollaborationModel.shape;

    // Fonctions d'accès aux populations
    list<predator> get_predator { return list(predator); }
}

global {
    int worldSize <- 100;
```

```
    geometry shape <- square(worldSize);

    init {
        // Création des deux sous-modèles
        create AntModel;
        create PreyPredModel;
    }
}
```

LISTING 4.16 – Structure de couplage avec imports configurés

Il est important de noter que l'implémentation présentée (Listing 4.16) fait le choix de traiter l'ensemble du modèle *Russian Dolls* (fourmis individuelles + colonie agrégée + bras) comme un modèle complet à coupler avec le modèle proies-prédateurs. Néanmoins, le framework aurait permis d'intégrer directement les mécanismes de prédation directement dans ce modèle. Le choix retenu illustre la flexibilité méthodologique offerte : selon le contexte de recherche, les modélisateurs peuvent soit étendre un modèle existant, soit coupler des systèmes développés indépendamment.

4.2.4.d Mécanisme de couplage par miroir

Le couplage entre les fourmis et les proies utilise le concept de `mirrors` de la plateforme GAMA qui établit une correspondance biunivoque entre agents. Et ce framework permet de le faire entre des agents de modèles différents :

```
model comodel_mix_behaviors as CollaborationModel

import "ant_foraging_russian-dolls.gaml" as AntModel {
    // [...]
    // Enlève les agents ‘ants‘ mirror-és
    init_nbr_ants <- 0;
}

import "Prey Predator.gaml" as PreyPredModel {
    // [...]
    // Remplace les agents ‘prey‘ par les nouveaux ‘prey_ant‘
    substitute prey as CollaborationModel.specialized_prey_ant;
}

// [...]

species specialized_prey_ant parent: PreyPredModel.prey mirrors:
    AntModel.specialised_ants{
    // Intégration du comportement entre les modèle intégrés
    reflex coupled_move {
        if (is_chased) {
            // Mode fuite : la colonie suit le mouvement d'é
   vasion de la proie
            target.location <- location;
        } else {
            // Mode normal : comportement de colonie avec
   synchronisation
            location <- target.location;  // Synchronisation
   vers le modèle PreyPredModel
        }
```

```
    }
}
```

LISTING 4.17 – Espèce de couplage avec synchronisation comportementale

Ce mécanisme crée une dualité conceptuelle : chaque fourmie est également une proie dans le système prédateur-proie. L'agent `specialized_prey_ant` hérite des comportements du modèle de proie tout en étant en miroir (*mirrored*) avec une fourmi du sous-modèle. La synchronisation se fait naturellement via la référence 'target' qui pointe vers l'agent miroir.

4.2.4.e Propagation comportementale multi-niveaux

Ce couplage permet une propagation naturelle des comportements entre systèmes. Lorsqu'une proie est chassée par un prédateur dans le modèle `PreyPredModel`, cette information se répercute automatiquement sur la colonie, puis sur les fourmis qui adopte un comportement de fuite. Inversement, les déplacements autonomes des fourmis influencent les positions des espèces aux niveaux supérieurs, créant une interaction bidirectionnelle fluide.

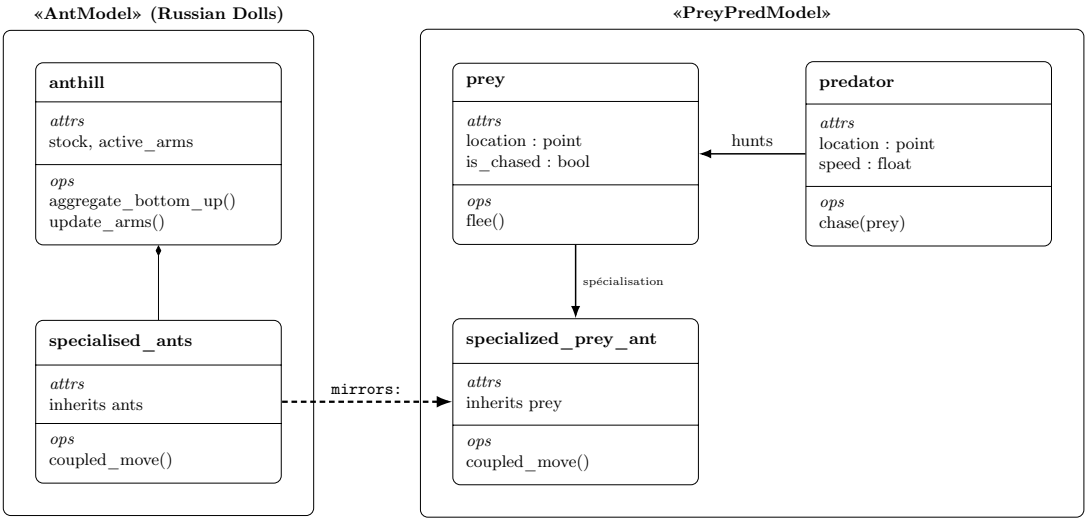

FIGURE 4.8 : Architecture UML du pattern *Collaboration* : couplage faible entre le modèle de fourmis sous sa forme *Russian Dolls* de la section précédente et un modèle proies–prédateurs via `mirrors`.

Cette approche préserve l'autonomie conceptuelle de chaque modèle : le système de fourmis et de colonie conserve sa logique de recherche de nourriture, tandis que le système proies-prédateurs maintient ses dynamiques de chasse. Le couplage n'intervient qu'au niveau des positions et états comportementaux.

```
experiment collaboration_interface {
    output {
        display main_view {
            // Affichage des agents de couplage
            species AntModel.specialised_ants;
            species specialised_colony;

            // Affichage des agent 'predator'
            species PreyPredModel.predator;
```

```
        }
    }
}
```

Listing 4.18 – Expérience pour l'observation du couplage

Cette implémentation illustre comment GAMA simplifie la création d'architectures *Collaboration* et sa migration depuis d'autres design pattern. Contrairement aux approches traditionnelles qui nécessitent des interfaces complexes et des protocoles de synchronisation élaborés, le framework permet un couplage direct via les ajouts de ce framework (comme `import` et `substitute`) ainsi que les mécanismes natifs (`import` et `mirrors`). Cette simplicité réduit significativement la barrière technique pour l'intégration de modèles hétérogènes, démocratisant l'accès aux architectures ML-ABM complexes.

4.3 Discussion

Les exemples d'implémentation ont montré que le framework proposé permet de parcourir sans rupture le continuum des design patterns ML-ABM (section 2.1.1). Cette section consolide la validité méthodologique de l'approche au regard des défis posés (coordination sémantique, orchestration spatio-temporelle, transitions d'échelle), puis examine sa portée opérationnelle au travers du projet COMOKIT et, enfin, dessine des perspectives de recherche et d'ingénierie.

4.3.1 Validation méthodologique des contributions

4.3.1.a Réponse aux problématiques identifiées

Les travaux présentés dans cette thèse apportent des réponses concrètes aux trois défis majeurs identifiés dans l'introduction : la coordination sémantique entre niveaux hétérogènes, la synchronisation spatio-temporelle des processus multi-échelles, et la gestion dynamique des transitions entre échelles d'observation.

La problématique de **coordination sémantique** trouve une réponse dans le système de contraintes formalisé au chapitre précédent (section 3.1.1) et mis en œuvre dans ce chapitre. Les mécanismes d'import et de spécialisation d'agents permettent aux modélisateurs de spécifier explicitement les relations conceptuelles entre niveaux sans avoir à maîtriser les aspects techniques du couplage logiciel. Cette approche déclarative simplifie considérablement la création de modèles intégrés tout en préservant la rigueur sémantique nécessaire à leur validation (Morvan, 2012).

Le défi de la **synchronisation spatio-temporelle** (section 3.1.2.b) se voit traité par l'automatisation des mécanismes de coordination proposés. Contrairement aux approches antérieures qui laissaient cette charge aux modélisateurs, le framework développé intègre nativement des stratégies de synchronisation adaptatives. Les exemples présentés démontrent comment les correspondances spatiales et les ratios temporels peuvent être spécifiés de manière intuitive et cohérente avec le reste de la modélisation au sein de la plateforme GAMA et du langage GAML, réduisant significativement les risques d'erreur et la complexité d'implémentation (Taillandier et al., 2019).

La **gestion dynamique des transitions** est assurée par le mécanisme robuste du *capture/release* avec le couplage des modèles via une simplification du *comodeling*.

Cette coordination d'outils constitue l'innovation majeure de cette approche. Les exemples d'implémentation montrent comment un même modèle peut évoluer fluidement entre les différents design patterns selon les besoins du modélisateur, sans nécessiter de refonte architecturale. Cette flexibilité répond directement aux besoins évolutifs des projets de recherche en systèmes socio-environnementaux, où les questions de recherche peuvent nécessiter des changements d'échelle d'observation (Verburg et al., 2013).

4.3.1.b Application de la méthodologie proposée

L'implémentation concrète du framework valide empiriquement l'applicabilité de la méthodologie développée au chapitre 3. Les trois design patterns (*Zoom*, *Russian Dolls*, *Collaboration*) ont été intégralement implémentés en utilisant le cadre unifié proposé, démontrant sa capacité à couvrir l'ensemble du continuum des approches ML-ABM existantes.

Ce cadre unifié intègre et étend les approches complémentaires *comodeling* et *capture/release* préexistants dans GAMA (Huynh, 2016; Vo, 2012). Du *comodeling*, il absorbe les mécanismes d'interfaces inter-modèles et de couplage de composants (modèles) autonomes, tout en automatisant la coordination spatio-temporelle qui nécessitait auparavant une architecture logicielle complexe. Du *capture/release*, il préserve les capacités de transfert d'agents et de changements d'échelle dynamiques, en levant la contrainte de co-localisation forcée dans un modèle unique. L'originalité réside, ainsi, dans la déclarativité des correspondances sémantiques et la gouvernance automatique du temps multi-échelles, transformant des mécanismes techniques en constructions conceptuelles accessibles.

L'architecture du framework, basée sur les concepts de contraintes sémantiques et de coordination spatio-temporelle, se traduit opérationnellement par des extensions du langage GAML qui préservent la simplicité syntaxique tout en offrant une expressivité accrue. Les modélisateurs peuvent désormais spécifier des modèles multi-niveaux complexes avec une syntaxe déclarative intuitive, comme illustré par les exemples de code présentés précédemment (section 4.2).

4.3.1.c Évolution fluide le long du continuum ML-ABM

Les trois design pattern utilisés forment le continuum des types d'architectures multi-niveau des modèles à base d'agents (Section 2.1.2) où chaque étape répond à des besoins de modélisation spécifiques plutôt qu'à des contraintes techniques :

1. **Zoom** : *Comment optimiser l'analyse selon le niveau de détail requis ?* (section 4.2.2)
2. **Russian Dolls** : *Comment capturer les rétroactions en temps réel entre les comportements individuels et les stratégies collectives ?* (section 4.2.3)
3. **Collaboration** : *Comment intégrer l'impact de dynamiques externes complexes dans ce modèle de fourmis ?* (section 4.2.4)

Cette progression conserve une syntaxe GAML unifiée et accessible :

```
model integrated as IntegratedModel

// Import avec configuration temporelle
```

```
import "micro_model.gaml" as MicroModel {
    // Synchronisation temporelle explicite/automatique
    float step <- IntegratedModel.step / 2;
}

global {
    float step <- 1#mn;
    init {
        // Création standard des niveaux (évolution progressive)
        create MicroModel; // Zoom : seulement un niveau à la
   fois
        create MesoModel; // Russian Dolls : + le méso en
   coexistence
        create specialised_agent; // Collaboration : + agent de
   couplage
    }

    // Interactions via réflexes familiers
    reflex synchronize { /* logique de couplage encapsulée */ }
}

// Spécialisation d'agents via définition familière
species specialised_agent parent: MicroModel.agent { /* [...] */
   }
```

Listing 4.19 – Syntaxe cohérente pour toutes les architectures

Chaque design pattern peut se construire sur ou évoluer vers une autre de ces architectures. Le modèle *Russian Dolls* étend le *Zoom* en supprimant la destruction des agents et en ajoutant la synchronisation bidirectionnelle. L'architecture *Collaboration* encapsule le système *Russian Dolls* complet comme composant autonome dans un système plus large.

Cette flexibilité facilite l'exploration incrémentale de configurations alternatives tout en préservant les investissements antérieurs de développement. Les choix de développement ne sont plus guidés par des contraintes techniques, mais par des questions ou des besoins de recherche par le modélisateur qui peut ainsi, par exemple :

1. Commencer par un modèle *Zoom* simple pour valider les mécanismes de base ;
2. Évoluer vers *Russian Dolls* pour étudier les rétroactions multi-niveaux ;
3. Intégrer ce modèle dans une architecture *Collaboration* pour explorer l'impact de perturbations externes ;
4. Revenir à une architecture plus simple si les besoins évoluent.

La validation de cette approche méthodologique s'étend par delà les exemples données. Cette implémentation permet d'envisager la réécriture simplifiée de modèles existants complexes, tels que ceux développés dans les projets ESCAPE, COMOKIT ou SIMPLE mentionnés en introduction. Cette perspective de migration facilite l'adoption progressive du framework par la communauté scientifique tout en valorisant les investissements antérieurs.

4.3.2 Application de la modélisation ML-ABM : l'exemple de COMOKIT

L'analyse des design patterns ML-ABM présentée dans le chapitre 2 pourrait suggérer que les modélisateurs doivent choisir a priori entre des approches architecturales distinctes et mutuellement exclusives. Cependant, comme souligné dans la section 2.4, la réalité pratique de la modélisation de systèmes complexes révèle que les modèles opérationnels nécessitent fréquemment des architectures hybrides combinant plusieurs paradigmes de couplage selon les spécificités des sous-systèmes modélisés (Chapuis et al., 2021a).

Cette hybridation architecturale reflète la nature intrinsèquement hétérogène des systèmes socio-environnementaux, où différents processus opèrent selon des logiques temporelles et spatiales distinctes. Par exemple, dans le contexte épidémiologique, la modélisation des comportements individuels de mobilité peut bénéficier d'une approche *Zoom* pour gérer les transitions entre niveaux de détail, tandis que l'intégration des politiques publiques multi-échelles nécessite une architecture *Collaboration* préservant l'autonomie des processus décisionnels à chaque niveau administratif. D'autres modèles multi-échelles de propagation infectieuse en milieu clos (Le-Kim et al., 2016) illustrent ces défis d'intégration de dynamiques individuelles et collectives.

Le framework développé dans cette thèse répond précisément à cette nécessité d'hybridation en proposant un langage unifié qui permet la coexistence et l'articulation de différents paradigmes de couplage au sein d'un même modèle intégré. Cette flexibilité constitue un atout majeur pour la modélisation de systèmes réels, qui ne se conforment que rarement aux typologies théoriques pures.

4.3.2.a Couplage micro-méso dans COMOKIT

Le projet COMOKIT illustre parfaitement cette nécessité d'hybridation architecturale. Comme mentionné en introduction (section 1.1.3.b), la seconde version de COMOKIT propose trois modèles distincts (micro, méso, macro) qui, bien qu'opérant à des échelles différentes, demeurent découplés et nécessitent des implémentations séparées (Taillandier et al., 2024b). Cette limitation technique constitue un frein à l'exploration des interactions dynamiques entre échelles, particulièrement cruciales pour comprendre l'efficacité différentielle des politiques de contrôle épidémique selon les contextes locaux.

L'application du framework développé permet de réviser cette architecture en exploitant une approche de *couplage sélectif spatial*. Plutôt que de dupliquer l'ensemble du territoire à différentes résolutions, cette stratégie consiste à maintenir une représentation méso globale tout en instanciant des modèles micro détaillés uniquement pour les zones critiques nécessitant une analyse fine, comme les établissements hospitaliers où les dynamiques de transmission présentent des spécificités importantes.

Ce choix de modélisation utilise la capacité des agents `people` à exister simultanément sous deux représentations : une représentation micro détaillée (chaque agent a un rôle et un travail -docteur, patient, agent d'entretien-) et une représentation méso agrégée (dynamiques populationnelles, indicateurs territoriaux).

```
model COMOKIT_Integrated as ComokitIntegratedModel

// Import des modèles COMOKIT existants
```

```
import "../Micro/Models/Experiments/No_intervention.gaml" as
    MicroModel {
     float step <- 10#mn;
}
import "../Meso/Models/Experiments/Baseline/No containment.gaml"
     as MesoModel {
     float step <- ComokitIntegratedModel.step;
     // Enforce same health policy between levels
     Policy BuildingPolicy <- MesoModel.Policy;
}

global {
     float step <- 1#h;

     init {
         create MesoModel number: 1;
         // Initialisation du niveau micro uniquement
         // pour les hopitaux
         create MicroModel over: MesoModel.building where (each.
    type = "hospital");
     }
}
```

LISTING 4.20 – Structure de base pour l'intégration COMOKIT micro-méso

Cette architecture hybride combine efficacement un mélange entre les patterns *Zoom* et *Russian Dolls* : le niveau méso fournit la dynamique épidémiologique générale de la population, tandis que les instances micro permettent d'analyser avec précision les processus critiques dans les hôpitaux. La synchronisation temporelle automatique (pas de dix minutes pour le micro, une heure pour le méso) garantit la cohérence des dynamiques inter-échelles.

Le couplage dynamique repose sur la capacité des agents `people` à persister entre leurs représentations méso (agents de population générale) et micro (agents spécialisés dans les contextes hospitaliers). Cette transition s'opère automatiquement selon leur localisation spatiale, exploitant les mécanismes `capture/release` pour assurer une continuité (sémantique et applicative, les attributs de l'agent méso sont conservés dans l'agent micro et inversement) de l'identité épidémiologique des agents.

```
import "../Micro/Models/Experiments/No_intervention.gaml" as
    MicroModel {
     float step <- 10#mn;

     substitute DefaultWorker as ComokitIntegratedModel.
    specialised_micro_people;
}

// [...]

// Top-down dynamic
reflex transition_to_micro {
     // Check for each micro model
     loop micro_model over: MicroModel {
         // Capture each ‘people‘ inside a micro model
         loop meso_people over: MesoModel.people where (each
    inside self.shape) {
             capture meso_people as specialised_micro_people {
```

```
                location <- meso_people.location;
                // Start micro dynamic, depending on the job of
    the agent
                do init_micro_state;
            }
        }
    }
}

species specialised_micro_people parent: DefaultWorker {

    // [...]

    // Replace deleting action by releasing them in the meso
    level
    action exit_building {
        release self as: MesoModel.people in: MesoModel.world;
        // Set travel and destination
        target <- any_location_in(self.target_building);
    }
}
```

Listing 4.21 – Transition de niveaux des agents `people`

Cette implémentation illustre parfaitement l'hybridation des design patterns dans un contexte opérationnel réel. Le mécanisme de `capture` transforme les agents `people` du niveau méso en agents `specialised_micro_people` lorsqu'ils pénètrent dans les zones hospitalières, leur conférant des comportements détaillés, adaptés aux spécificités de ces environnements (protocoles de soin, circuits patients, interactions soignants-patients). Inversement, l'action `exit_building` redéfinie utilise le mécanisme `release` pour réintégrer les agents dans la population générale du niveau méso, préservant leur état épidémiologique acquis durant leur séjour hospitalier.

Cette architecture présente l'avantage de concentrer la complexité computationnelle sur les zones où elle apporte une valeur analytique réelle, tout en maintenant une vision globale cohérente des dynamiques épidémiologiques. Elle répond directement aux limitations de l'architecture COMOKIT actuelle en permettant l'exploration des dynamiques inter-niveaux entre les évolutions hospitalières (niveau micro) et les tendances populationnelles générales (niveau méso), ouvrant ainsi de nouvelles perspectives pour l'évaluation des politiques de gestion des capacités hospitalières en contexte épidémique.

4.3.2.b Gains opérationnels de l'intégration

Cette intégration multi-niveaux de COMOKIT offre plusieurs avantages opérationnels significatifs par rapport à l'architecture actuelle découplée. Premièrement, elle permet l'exploration des **rétroactions dynamiques** entre comportements individuels et politiques collectives, actuellement impossible avec des modèles séparés. Cette capacité est cruciale pour évaluer l'efficacité adaptative des interventions sanitaires face à l'évolution des comportements populationnels.

Deuxièmement, l'intégration facilite la **calibration cohérente** des paramètres entre échelles, actuellement problématique lorsque les modèles micro et méso utilisent des sources de données distinctes (Taillandier et al., 2024b). Le partage d'une

base d'agents commune garantit la cohérence des hypothèses démographiques et comportementales entre niveaux.

Troisièmement, cette approche simplifie l'**exploration scénaristique** en permettant aux utilisateurs de modifier dynamiquement le niveau de détail selon leurs besoins analytiques. Les décideurs politiques peuvent ainsi examiner les implications locales de décisions macro sans recourir à des modèles distincts nécessitant des paramétrages séparés.

Il convient de préciser que, bien que l'implémentation technique de cette intégration multi-niveaux ait été réalisée et validée fonctionnellement, les contraintes temporelles de cette thèse n'ont pas permis de conduire une exploration expérimentale approfondie du modèle intégré produit. L'effort de développement s'est concentré sur la démonstration de faisabilité technique et la validation des mécanismes de couplage proposés, sans procéder à une analyse systématique des dynamiques épidémiologiques émergentes spécifiques à l'architecture multi-niveaux.

4.3.3 Vers une démocratisation des approches multi-niveaux

Les travaux présentés dans cette thèse s'inscrivent dans une perspective plus large de démocratisation des approches de modélisation multi-niveaux. Historiquement, la complexité technique des ML-ABM a limité leur adoption aux équipes disposant d'une expertise avancée en génie logiciel, créant une barrière significative entre les besoins des domaines d'application et les outils disponibles (Taillandier et al., 2019).

Le framework développé dans cette thèse contribue à réduire cette barrière en proposant une approche déclarative qui masque la complexité technique tout en préservant l'expressivité nécessaire aux applications avancées. Cette simplification syntaxique, couplée à l'intégration native dans GAMA, facilite l'appropriation progressive des concepts multi-niveaux par des communautés de modélisateurs non-informaticiens pour qui le besoin de ces architectures logicielles complexes rendait impossible le développement de modèles multi-niveaux.

Cette démocratisation ouvre des perspectives prometteuses pour l'application des ML-ABM à de nouveaux domaines d'application, notamment en sciences sociales et environnementales où la prise en compte simultanée de multiples échelles d'observation constitue un enjeu méthodologique central. L'accessibilité accrue de ces outils peut catalyser le développement de modèles intégrés plus riches et plus réalistes.

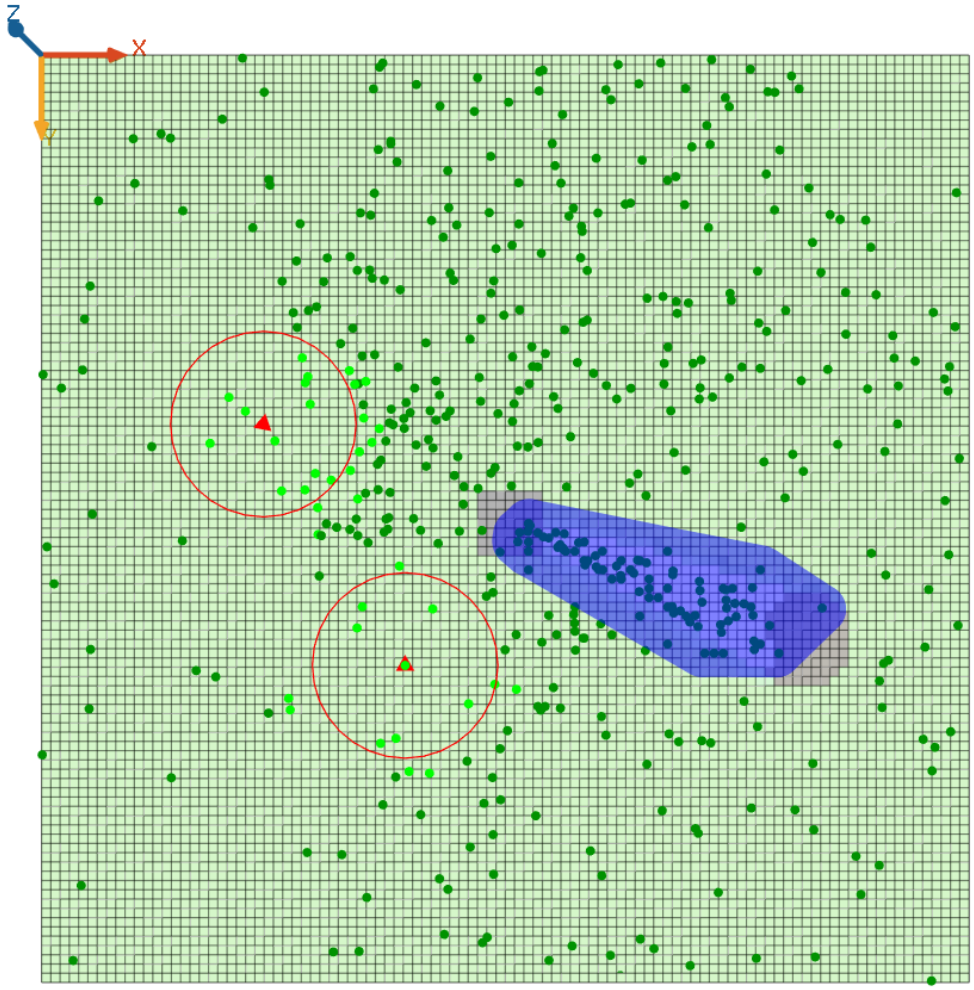

FIGURE 4.9 : Implémentation du pattern *Collaboration*.
Le modèle de fourmis (version *Russian Dolls*) est faiblement couplé à un modèle proies–prédateurs. La fourmilière agrégée est représentée par un polygone bleu semi-transparent. Les triangles rouges figurent les prédateurs, tandis que leurs cercles rouges indiquent leur rayon de perception. Les fourmis–proies apparaissent sous forme de points verts sur la grille et adoptent un comportement couplé : elles se déplacent selon le comportement standard des fourmis, sauf lorsqu'un agent est marqué comme *is_chased* (dans le modèle `prey` couplé), auquel cas il fuit activement le prédateur.

5
Conclusion et perspectives

Cette thèse a exploré la problématique du couplage dynamique et du passage à l'échelle de modèles socio-environnementaux (SES) complexes à travers le développement d'un cadre méthodologique pour la modélisation multi-niveaux à base d'agents (ML-ABM). Les travaux présentés ont abouti à la conception et à l'implémentation d'outils facilitant la création de modèles ML-ABM au sein de la plateforme GAMA (Taillandier et al., 2012), répondant ainsi aux défis identifiés dans la modélisation intégrée des systèmes complexes. Notamment, ces outils permettent de résoudre des limitations concrètes comme l'impossibilité dans COMOKIT v2 (Taillandier et al., 2024b) de faire communiquer les niveaux micro, méso et macro au sein d'une simulation intégrée.

5.1 Synthèse des contributions

5.1.1 Bilan condensé des contributions

Cette thèse a produit un ensemble de contributions structurées autour de trois axes principaux : conceptuel, méthodologique et technique. Le tableau 5.1 présente une vue synoptique de ces contributions avant leur développement détaillé.

5.1.2 Contributions conceptuelles

La contribution conceptuelle principale de cette thèse réside dans l'unification des paradigmes ML-ABM existants au sein d'un cadre théorique cohérent. Contrairement aux approches antérieures qui traitaient les patterns *Zoom*, *Russian Dolls* et *Collaboration* comme des architectures distinctes et incompatibles, cette thèse a démontré qu'ils constituent des manifestations particulières d'un système plus général de contraintes sémantiques.

Cette unification s'appuie sur la formalisation de trois types de contraintes fondamentales : les *contraintes d'entités* (définissant les correspondances conceptuelles entre agents de niveaux différents), les *contraintes d'attributs* (spécifiant les relations

TABLE 5.1 : Bilan synthétique des contributions de la thèse

Dimension	Contributions principales
Conceptuelle	— Séparation du *quoi* et *comment* du couplage de modèle multi-niveaux — Facilitation de l'exploration itérative des systèmes complexes — Réconciliation des paradigmes de couplages fort et faible
Méthodologique	— Système de contraintes sémantiques déclaratives pour les relations inter-niveaux — Mécanismes automatisés de coordination spatio-temporelle — Architecture adaptative permettant l'évolution dynamique entre patterns multi-niveaux
Technique	— Intégration native dans la plateforme de modélisation et simulation à base d'agent GAMA — Extension du langage GAML — Application unifiée réutilisant et améliorant le *comodeling* et *capture/release*
Empirique	— Validation sur la réimplémentation de composants de COMOKIT — Démonstration de la réduction de la complexité d'implémentation — Preuve de concept d'architectures hybrides ML-ABM

entre propriétés des agents multi-niveaux), et les *contraintes de processus* (déterminant les interactions entre dynamiques opérant à différentes échelles).

Ces travaux ont également établient une séparation claire entre la spécification des relations inter-niveaux (le *quoi* du couplage) et leur implémentation technique (le *comment* du couplage). Cette distinction conceptuelle permet aux modélisateurs de se concentrer sur les aspects scientifiques de leur modélisation sans être contraints par les détails techniques de l'implémentation multi-niveaux.

5.1.3 Contributions méthodologiques

Du point de vue méthodologique, cette thèse a développé un système complet de gestion des relations multi-niveaux, articulé autour de trois composants principaux.

Le système de contraintes sémantiques déclaratives constitue le cœur méthodologique de l'approche. Il permet aux modélisateurs d'exprimer formellement les relations conceptuelles entre niveaux de modélisation sans recours à une programmation technique complexe. Ces contraintes sont ensuite automatiquement traduites en mécanismes de couplage opérationnels par le framework.

Les mécanismes de coordination spatio-temporelle automatisée résolvent la problématique cruciale de la synchronisation multi-échelles. L'horloge centrale proposée gère automatiquement les ratios temporels entre niveaux, tandis que le système d'espaces et les contraintes précédentes assure la cohérence spatiale des transformations multi-niveaux.

Les méthodes de transition dynamique entre échelle complètent, enfin, le dispositif méthodologique en permettant des passages fluides entre différents modes de

visualisation et d'interaction avec le modèle multi-niveaux. Ces transitions peuvent être déclenchées selon des conditions prédéfinis ou à la demande de l'utilisateur.

5.1.4 Contributions techniques

L'implémentation technique des contributions s'articule autour de l'extension native de la plateforme GAMA et du langage GAML. Cette extension comprend de nouveaux mots-clés, structures de données et mécanismes d'exécution spécifiquement conçus pour faciliter la création de modèles ML-ABM.

L'architecture logicielle développée intègre de manière unifiée les approches existantes de *comodeling* et de *capture/release*, tout en les étendant pour supporter les nouveaux paradigmes de couplage proposés. Cette intégration permet aux modélisateurs de bénéficier de la maturité des outils existants tout en accédant aux nouvelles fonctionnalités développées.

L'implémentation a notamment été pensée pour minimiser la courbe d'apprentissage et faciliter l'adoption d'usage des ML-ABM par la communauté des modélisateurs. Les interfaces logicielles proposées restent proches du langage GAML standard, permettant une migration progressive depuis les approches mono-niveau vers les architectures multi-niveaux.

5.1.5 Validation empirique

La validation empirique des contributions s'appuie sur un ensemble de cas d'étude couvrant les trois patterns ML-ABM classiques ainsi que des configurations hybrides innovantes.

Pour le pattern *Zoom*, l'implémentation a démontré la capacité du framework à gérer des transitions fluides entre niveaux de détails, avec maintien de la cohérence des propriétés agrégées lors des changements d'échelles.

Le pattern *Russian Dolls* a été validé par la création de modèles emboîtés où les dynamiques de chaque niveau influencent et sont influencées par les niveaux associés, illustrant la gestion bidirectionnelle des influences multi-niveaux.

Le pattern *Collaboration* a fait l'objet d'une validation spécifique à travers la réimplémentation simplifiée de composants du projet COMOKIT. Cette réimplémentation a montré une réduction significative de la complexité de développement tout en préservant l'expressivité scientifique du modèle original.

La validation a également porté sur des architectures hybrides combinant plusieurs patterns au sein d'un même modèle, démontrant la flexibilité et l'extensibilité de l'approche proposée.

5.2 Limites et perspectives

Les contributions présentées dans cette thèse, bien qu'apportant des réponses concrètes aux défis identifiés dans la modélisation multi-niveaux à base d'agents, s'accompagnent de limitations méthodologiques et techniques qui méritent une analyse critique approfondie. Cette section examine ces limites avant de dessiner les perspectives de recherche et de développement qu'elles ouvrent pour l'évolution future du domaine.

5.2.1 Limites de la validation empirique

La validation empirique développée dans cette thèse présente des limitations méthodologiques substantielles qui affectent la généralisation des résultats obtenus. La validation s'est principalement appuyée sur la réimplémentation de composants du modèle COMOKIT, un projet développé par l'équipe de la plateforme GAMA elle-même. Cette approche constitue un biais de validation interne qui limite la portée des conclusions.

COMOKIT représente un cas d'étude particulier développé par des modélisateurs informaticiens experts de la plateforme GAMA et familiers des bonnes pratiques d'architecture de modélisation. De surcroît, COMOKIT v2 (Taillandier et al., 2024b), version utilisée pour cette validation, avait déjà été conçu avec une structuration par niveaux, facilitant naturellement son adaptation au framework proposé. Cette double spécificité (expertise technique des développeurs et architecture favorable aux approches multi-niveaux) limite considérablement la représentativité de cette validation pour des modèles développés sans considération particulière de niveau. La validation sur ce seul projet ne permet, donc, pas d'évaluer la robustesse du framework dans des contextes non contrôlés, avec des modélisateurs présentant des niveaux d'expertise variables ou des approches de modélisation différentes. Cette limitation est d'autant plus significative que l'objectif affiché du framework est de faciliter l'accès aux techniques ML-ABM pour des profils non-informaticiens.

L'analyse des données collectées lors de la conférence GAMA Days 2024 (Brugière and Nguyen-Ngoc, 2024) révèle d'ailleurs ce paradoxe : alors que 75% des participants expriment un intérêt pour des outils ML-ABM simplifiés, seulement 30% ont effectivement utilisé des modèles multi-niveaux, et ce sous-ensemble est très majoritairement composé de développeurs de la plateforme GAMA. Cette disproportion entre intérêt théorique et adoption effective souligne la nécessité d'une validation élargie impliquant des utilisateurs représentatifs de la diversité de la communauté de modélisation.

Une validation externe approfondie reste donc nécessaire pour confirmer l'accessibilité effective du framework. Cette validation devrait porter sur plusieurs dimensions critiques actuellement non mesurées : la courbe d'apprentissage réelle pour des utilisateurs novices, la capacité du framework à s'adapter à des paradigmes de modélisation non anticipés lors de son développement, l'identification de limitations pratiques dans l'harmonisation des patterns ML-ABM, et l'évaluation de métriques quantitatives précises telles que le temps de développement, le taux d'erreur, et la qualité des modèles produits.

5.2.2 Performance et passage à l'échelle

L'évaluation systématique de la performance computationnelle n'a pas été abordée dans cette thèse, ce qui constitue une limitation importante pour l'adoption du framework dans des contextes de simulation intensive. Ce positionnement découle d'un choix délibéré : les efforts se sont concentrés sur l'accessibilité et l'expressivité de l'écriture ML-ABM, ainsi que sur la cohérence sémantique des mécanismes de couplage. Cette approche génère néanmoins une incertitude concernant le surcoût introduit par les couches d'abstraction (coordination automatique, orchestration centrale, interfaces de couplage) comparé à une implémentation *ad hoc* optimisée ; bien que probablement plus efficace grâce à son implémentation plus bas niveaux. Cette préoccupation revêt une importance particulière dans un contexte où les approches

multi-niveaux sont fréquemment associées à des stratégies d'optimisation destinées à contourner les limites de ressources des simulations monolithiques.

L'absence d'évaluation de performance constitue une limite actuelle, sans le remettre en cause, du cadre proposé. Les contributions de cette thèse fournissent la base sémantique et outillée nécessaire pour conduire des études de performance reproductibles et des optimisations ciblées. Un protocole d'évaluation robuste est nécessaire pour pouvoir comparer différentes implémentations (monolithique, *ad hoc*, framework) sur des cas d'étude variés.

Au-delà de l'évaluation, les architectures ML-ABM présentent des caractéristiques intrinsèque favorables à l'optimisation. L'encapsulation des différents niveaux en sous-modèles autonomes facilite leur déploiement sur des infrastructures de calcul distribué, où chaque nœud de calcul peut prendre en charge un ou plusieurs sous-modèles selon les besoins en ressources. Cette modularité architecturale constitue un avantage significatif par rapport aux approches monolithiques et s'aligne avec les travaux récents de Grosjean et al. (2025, 2024) sur la distribution de simulations GAMA.

La compatibilité conceptuelle avec ces travaux de distribution ouvre des perspectives prometteuses pour le déploiement à grande échelle sans modifications architecturales majeures du framework. La séparation entre modèle intégrateur et sous-modèles spécialisés permet d'optimiser les communications inter-modèles selon les patterns d'interaction spécifiques à chaque architecture ML-ABM, facilitant l'adaptation aux contraintes de performance des différents contextes d'application.

5.2.3 Dépendance exclusive à la plateforme GAMA

L'ancrage exclusif du framework dans l'écosystème GAMA constitue une limitation structurelle majeure qui restreint sa portée pour la communauté élargie de la modélisation à base d'agents. Cette spécialisation, bien qu'elle permette une intégration native optimale avec les approches existantes de *comodeling* et de *capture/release*, limite l'impact potentiel des contributions développées.

Cette dépendance se manifeste à deux niveaux distincts. La méthodologie conceptuelle développée, étant par nature indépendante de quelconque plateforme de modélisation, présente une portabilité théorique vers d'autres environnements de modélisation. Les concepts de *contraintes sémantiques*, de *coordination spatio-temporelle*, et d'*architecture de design patterns ML-ABM* constituent des abstractions suffisamment générales pour être transposées vers NetLogo, MASON, Repast, ou d'autres frameworks spécialisés.

En revanche, la syntaxe présentée et son implémentation présentent une dépendance forte aux spécificités de GAML et aux infrastructures techniques de GAMA. L'extension du langage GAML, les mécanismes d'import configuré, la centralisation de paramètres, et l'orchestration automatique des couplages du *comodeling* et du *capture/release* constituent des développements intimement liés à l'architecture logicielle de GAMA. Le portage vers d'autres plateformes nécessiterait une ré-implémentation substantielle, impliquant un effort de développement conséquent pour chaque plateforme cible.

5.2.4 Perspectives d'interopérabilité multi-plateformes

Le dépassement de la limitation à la plateforme GAMA constitue une perspective de développement majeure qui pourrait amplifier significativement l'impact des contributions de cette thèse. Principalement, elle ouvre la voie vers l'intégration d'autres plateformes comme sous-modèles au sein de GAMA.

L'enjeu de l'interopérabilité dépasse les seules considérations techniques pour toucher aux questions de standardisation et de mutualisation des efforts dans la communauté de recherche en ML-ABM. Le développement d'interfaces et de protocoles standardisés pour la coordination de modèles multi-plateformes pourrait bénéficier à l'ensemble de la communauté, indépendamment des choix technologiques spécifiques de chaque équipe de recherche.

Cette standardisation pourrait également porter sur la formalisation de bonnes pratiques de modélisation multi-niveaux, dépassant les spécificités techniques pour établir des recommandations méthodologiques transversales. Cette démarche contribuerait à la maturation du domaine en facilitant la communication entre équipes, la validation des modèles, et l'accumulation de connaissances scientifiques reproductibles et comparables.

L'enjeu de l'interopérabilité répond également aux besoins croissants de modélisation pluridisciplinaire dans l'étude des systèmes socio-environnementaux. L'intégration avec des simulateurs spécialisés (modèles climatiques, simulateurs hydrauliques, modèles économétriques, outils de dynamique des populations, etc) permettrait de développer des modèles multi-niveaux d'une précision et d'un réalisme nouveau. Cette capacité d'orchestration de simulateurs experts issus de différentes disciplines constitue un levier majeur pour aborder la complexité des systèmes réels, où les dynamiques sociales, environnementales, économiques et politiques s'articulent à des échelles multiples et hétérogènes.

5.3 Conclusion

Cette thèse s'est attachée à résoudre la problématique du couplage dynamique et du passage à l'échelle de modèles socio-environnementaux complexes à travers le développement d'un cadre méthodologique pour la modélisation multi-niveaux à base d'agents (ML-ABM). Les travaux présentés ont abouti à la conception et à l'implémentation d'outils facilitant la création de modèles ML-ABM au sein de la plateforme GAMA, répondant ainsi aux défis identifiés dans la modélisation intégrée des systèmes complexes.

Le premier chapitre a établi le cadre général de la recherche en présentant la modélisation intégrée des systèmes socio-environnementaux comme un défi majeur pour la communauté scientifique. Cette thèse a démontré l'importance cruciale du couplage multi-niveaux dans l'exploration et la compréhension des interactions entre dynamiques humaines et naturelles, en illustrant cette nécessité à travers les projets ESCAPE, COMOKIT et SIMPLE. Les enjeux de recherche fondamentaux ont été identifiés : l'intégration de modèles hétérogènes, la gestion des échelles spatiale et temporelle, et l'importance du couplage multi-niveaux.

Le chapitre 2 a proposé une analyse critique exhaustive des approches ML-ABM existantes. Une typologie des architectures ML-ABM a été établie en présentant les trois design patterns fondamentaux : *Zoom* (section 2.2.1), *Russian Dolls*

(section 2.2.2), et *Collaboration* (section 2.2.3). Leurs forces et limitations respectives ont été analysées en détails, puis les solutions apportées par la plateforme GAMA ont été examinées, notamment les mécanismes de *comodeling* et de *capture/release*. Cette analyse a révélé les limitations persistantes des approches existantes et a mis en évidence la nécessité d'un cadre méthodologique unifié (section 2.4).

Le chapitre 3 a développé la contribution méthodologique principale de cette thèse. Un cadre de couplage unifié basé sur des contraintes sémantiques formalisées a été proposé (section 3.1), permettant de spécifier de manière déclarative les relations inter-niveaux. Un système de coordination temporelle et spatiale automatisée qui résout les problèmes de gestion manuelle des échelles identifiés dans l'état de l'art a été présenté. Enfin, un cadre de transition entre échelles permettant des passages dynamiques entre différents patterns architecturaux au cours d'une même simulation a également été développé (section 3.2). Cette proposition méthodologique harmonise les dichotomies traditionnelles entre couplage fort et couplage faible pour offrir une vision intégrée de la modélisation multi-niveaux.

Le chapitre 4 a présenté la validation empirique et technique des contributions théoriques. L'intégration complète du cadre proposé à la plateforme GAMA a été détaillée (section 4.1), incluant l'extension du langage GAML et du *comodeling* avec des constructions spécialisées pour la modélisation multi-niveaux. Un espace unifié de noms, des mécanismes d'imports configurés *in situ*, une centralisation des paramètres et une initialisation unifiée ont été développés. Un système d'ordonnancement multi-échelles avec horloge centrale a été implémenté et l'opérationnalisation des dynamiques multi-niveaux par *capture/release* a été développée. L'application pratique de ces outils aux trois design patterns ML-ABM a ensuite été démontrée (section 4.2), validant ainsi la généralité et la flexibilité de l'approche proposée. La discussion a souligné la validation méthodologique des contributions et a illustré les réalités pratiques de la modélisation ML-ABM à travers l'exemple concret de COMOKIT.

Enfin, ce dernier chapitre a présenté une synthèse des contributions structurées selon leurs dimensions conceptuelle, méthodologique, technique et empirique. La portée et les limites de l'unification conceptuelle proposée ont été analysées, ainsi que les contraintes de l'implémentation dans GAMA. Les limites techniques et méthodologiques ont été identifiées, comme le manque d'évalution de la performance du couplage, et la complexité résiduelle du *framework* développé. Des perspectives de recherche et de développement ont été proposées incluant une validation empirique élargie, une intégration avec les travaux de Grosjean et al. (2025, 2024) sur la distribution de modèles, et une interopérabilité du framework avec d'autres plateformes de simulation.

Les contributions de cette thèse répondent de manière systématique aux limitations identifiées dans l'analyse des approches ML-ABM existantes. La rigidité architecturale qui caractérisait les design patterns traditionnels trouve une réponse dans le système de contraintes sémantiques formalisées, permettant une adaptabilité évolutive où les modélisateurs peuvent faire évoluer l'architecture multi-niveaux selon leurs besoins d'exploration. L'unification conceptuelle développée réconcilie les paradigmes apparemment contradictoires du couplage fort et du couplage faible, accompagnée d'une simplification technique significative grâce à l'intégration native dans GAMA.

Bibliographie

Abar, S., Theodoropoulos, G. K., Lemarinier, P., and O'Hare, G. M. (2017). Agent based modelling and simulation tools : A review of the state-of-art software. *Computer Science Review* 24, 13–33

Abouaissa, H., JOLLY, D., BENASSER, A., et al. (2006). Macro-micro simulation of traffic flow. *IFAC Proceedings Volumes* 39, 351–356

Adam, C., Taillandier, P., and Dugdale, J. (2017). Comparing agent architectures in social simulation : Bdi agents versus finite-state machines

Aguilera, P. A., Fernández, A., Fernández, R., Rumí, R., and Salmerón, A. (2011). Bayesian networks in environmental modelling. *Environmental Modelling & Software* 26, 1376–1388

Alqurashi, R. and Altman, T. (2019). Hierarchical agent-based modeling for improved traffic routing. *Applied Sciences* 9, 4376

Anderson, P. (1999). Perspective : Complexity theory and organization science. *Organization science* 10, 216–232

Anh, N. T. N., Zucker, J. D., Du, N. H., Drogoul, A., and An, V. D. (2012). Hybrid equation-based and agent-based modeling of crowd evacuation on road network. In *Proc. ICCS*. 456–466

Anh, T. T. Q., Nguyen-Huu, T., and Nguyen-Ngoc, D. (2024). Agent-based modeling approach in single fish population management. In *International Conference on Computational Data and Social Networks* (Springer), 302–316

Auger, P., Parra, R., Poggiale, J.-C., Sánchez, E., and Nguyen-Huu, T. (2008). Aggregation of variables and applications to population dynamics. In *Structured population models in biology and epidemiology* (Springer). 209–263

Aumpuchin, P., Chaiprom, U., and Fuvattanasilp, V. (2024). BiodeVRestorer : An Immersive VR Experience for Biodiversity Restoration. In *GAMA Days 2024*, ed. A. Grignard (Online, France)

Bae, J. W., Paik, E., Kim, K., Singh, K., and Sajjad, M. (2016). Combining microsimulation and agent-based model for micro-level population dynamics. *Procedia Computer Science* 80, 507–517

Barreteau, O., Bots, P., Daniell, K., Etienne, M., Perez, P., Barnaud, C., et al. (2013). Participatory approaches. *Simulating social complexity : A handbook* , 197–234

Batty, M. (2007). *Cities and complexity : understanding cities with cellular automata, agent-based models, and fractals* (The MIT press)

Bellifemine, F., Bergenti, F., Caire, G., and Poggi, A. (2005). Jade—a java agent development framework. In *Multi-agent programming* (Springer). 125–147

Berkes, F. and Folke, C. (1998). Linking social and ecological systems for resilience and sustainability. *Linking social and ecological systems : management practices and social mechanisms for building resilience* 1, 4

Biré, L., Phung, Q. N., Taillandier, P., Phung, D. A., Nguyen, N. D., and Drogoul, A. (2025). Rác, a serious agent-based simulation game to drive discussion on waste management in vietnamese irrigation systems. *Journal of Artificial Societies and Social Simulation* 28, 4

Bonabeau, E. (2002). Agent-based modeling : Methods and techniques for simulating human systems. *Proceedings of the national academy of sciences* 99, 7280–7287

Bonté, B., Duboz, R., Quesnel, G., and Müller, J. P. (2009). Recursive simulation and experimental frame for multiscale simulation

Bosmans, S., Bogaerts, T., Casteels, W., Mercelis, S., Denil, J., and Hellinckx, P. (2022). Adaptivity in multi-level traffic simulation using experimental frames. *Simulation Modelling Practice and Theory* 114, 102395

Bousquet, F. and Le Page, C. (2004). Multi-agent simulations and ecosystem management : a review. *Ecological modelling* 176, 313–332

Brockelman, W. Y., Nathalang, A., and Gale, G. A. (2011). The mo singto forest dynamics plot, khao yai national park, thailand. *Natural History Bulletin of the Siam Society* 57, 35–55

Brugière, A. and Nguyen-Ngoc, D. (2024). Investigating coding preferences for coupling multi-level models in the gama platform. In *GAMA Days 2024*

Brugière, A., Nguyen-Ngoc, D., and Drogoul, A. (2022). Handling multiple levels in agent-based models of complex socio-environmental systems : A comprehensive review. *Frontiers in Applied Mathematics and Statistics* 8, 1020353

Brugière, A., Pham, M. D., Chapuis, K., Drogoul, A., Gaudou, B., Grignard, A., et al. (2019). Experimenting the impact of pedestrianisation on urban pollution using tangible agent-based simulations : Application to hoan kiem district, hanoi, vietnam. In *International Workshop on Complex Systems Modelling & Simulation* (Springer), 43–77

Bruner, G. (2008). *HEC RAS, River Analysis System Users Manual, Versión 4.0.* Tech. rep., US Army Corps of Engineers.(http ://www. hec. usace. army. mil)

Brunsdon, C. and Singleton, A. (2015). *Geocomputation : a practical primer* (Sage)

Butner, J. D., Cristini, V., and Wang, Z. (2017). Development of a three dimensional, multiscale agent-based model of ductal carcinoma in situ. In *2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)* (IEEE), 86–89

Chapuis, K., Elwaqoudi, T. A., Brugière, A., Daudé, E., Drogoul, A., Gaudou, B., et al. (2019a). An agent-based co-modeling approach to simulate the evacuation of a population in the context of a realistic flooding event : a case study in hanoi (vietnam). In *International Workshop on Complex Systems Modelling & Simulation* (Springer), 79–108

Chapuis, K., Elwaqoudi, T. A., Brugière, A., Daudé, E., Drogoul, A., Gaudou, B., et al. (2019b). An agent-based co-modeling approach to simulate the evacuation of a population in the context of a realistic flooding event : A case study in hanoi (vietnam). In *International Workshop on Complex Systems Modelling & Simulation* (Springer), 79–108

Chapuis, K., Taillandier, P., Gaudou, B., Brugière, A., and Drogoul, A. (2021a). Comokit : un environnement générique et modulaire pour analyser les impacts des politiques d'intervention contre l'épidémie de covid-19. In *Journées Francophones sur les Systèmes Multi-Agents (JFSMA) 2021*

Chapuis, K., Taillandier, P., Gaudou, B., Brugière, A., Drogoul, A., Araldi, A., et al. (2021b). Using the comokit model to study the impact of the morpho-functional organization of cities on the spread of covid-19. In *6th International Workshop on Agent-Based Modelling of Urban Systems (ABMUS@ AAMAS 2021)*. 61–65

Coates, G., Dugdale, J., and Hanachi, C. (2024). Simulation for crisis and disaster management. *Simulation* 100, 333–334

Combes, M., Grigné, C., Husson, L., Conrad, C., Le Yaouanq, S., Parenthoën, M., et al. (2012). Multiagent simulation of evolutive plate tectonics applied to the thermal evolution of the earth. *Geochemistry, Geophysics, Geosystems* 13

Concepcion, A. I. and Zeigler, B. P. (1988). Devs formalism : A framework for hierarchical model development. *IEEE Transactions on Software Engineering* 14, 228–241

Corchado, J. M., Mata, A., and Rodriguez, S. (2009). Osm : A multi-agent system for modeling and monitoring the evolution of oil slicks in open oceans. In *Advanced Agent-Based Environmental Management Systems* (Springer). 91–117

Cossentino, M., Gaud, N., Galland, S., Hilaire, V., and Koukam, A. (2007). A holonic metamodel for agent-oriented analysis and design. *Holonic and Multi-Agent Systems for Manufacturing* , 237

Courdier, R. (2003). *Simulation Multi-Agents : Architecture, Ingénierie et Expérimentations*. Accreditation to supervise research, Université de la Réunion

Courdier, R., Guerrin, F., Andriamasinoro, F. H., and Paillat, J.-M. (2002). Agent-based simulation of complex systems : application to collective management of animal wastes. *Journal of Artificial Societies and Social Simulation* 5, 30–56

Crociani, L., Vizzari, G., Yanagisawa, D., Nishinari, K., and Bandini, S. (2016). Route choice in pedestrian simulation : Design and evaluation of a model based on empirical observations. *Intelligenza Artificiale* 10, 163–182

Crutzen, P. J. and Brauch, H. G. (2016). *Paul J. Crutzen : A pioneer on atmospheric chemistry and climate change in the Anthropocene*, vol. 50 (Springer)

Dahmann, J. S. and Morse, K. L. (1998). High level architecture for simulation : An update. In *Proceedings. 2nd International Workshop on Distributed Interactive Simulation and Real-Time Applications (Cat. No. 98EX191)* (IEEE), 32–40

Daudé, E., Chapuis, K., Taillandier, P., Tranouez, P., Caron, C., Drogoul, A., et al. (2019). Escape : exploring by simulation cities awareness on population evacuation. In *ISCRAM*

David, D., Payet, D., and Courdier, R. (2011). Réification de zones urbaines émergentes dans un modèle simulant l'évolution de la population à la réunion. In *JFSMA*. 63–72

De la Torre, R., Onggo, B. S., Corlu, C. G., Nogal, M., and Juan, A. A. (2021). The role of simulation and serious games in teaching concepts on circular economy and sustainable energy. *Energies* 14, 1138

Diepart, J.-C., Ngin, C., and Oeur, I. (2019). Struggles for life : Smallholder farmers' resistance and state land relations in contemporary cambodia. *Journal of Current Southeast Asian Affairs* 38, 10–32

Drogoul, A., Amouroux, E., Caillou, P., Gaudou, B., Grignard, A., Marilleau, N., et al. (2013). Gama : A spatially explicit, multi-level, agent-based modeling and simulation platform. In *International Conference on Practical Applications of Agents and Multi-Agent Systems* (Springer), 271–274

Drogoul, A., Huynh, N. Q., and Truong, Q. C. (2016). Coupling environmental, social and economic models to understand land-use change dynamics in the mekong delta. *Frontiers in environmental science* 4, 19

Drogoul, A., Taillandier, P., Brugière, A., Martinez, L., Sillano, L., Lesquoy, B., et al. (2025). Coupling agent-based simulations and vr universes : the case of gama and unity. *arXiv preprint arXiv :2502.07405*

Duboz, R. (2004). Intégration de modeles hétérogenes pour la modélisation et la simulation de systemes complexes. *Application à la modélisation multiéchelles en écologie marine*

Duboz, R., Versmisse, D., Quesnel, G., Muzy, A., and Ramat, E. (2006). Specification of dynamic structure discrete event multiagent systems. *Simulation Series* 38, 103

Edmonds, B. and Moss, S. (2004). From kiss to kids–an 'anti-simplistic'modelling approach. In *International workshop on multi-agent systems and agent-based simulation* (Springer), 130–144

Elsawah, S., Filatova, T., Jakeman, A. J., Kettner, A. J., Zellner, M. L., Athanasiadis, I. N., et al. (2020). Eight grand challenges in socio-environmental systems modeling. *Socio-Environmental Systems Modelling* 2, 16226–16226

Forrester, J. W. (1997). Industrial dynamics. *Journal of the Operational Research Society* 48, 1037–1041

Fujimoto, R. M. (1990). Parallel discrete event simulation. *Communications of the ACM* 33, 30–53

Gain, A. K., Hossain, S., Benson, D., Di Baldassarre, G., Giupponi, C., and Huq, N. (2021). Social-ecological system approaches for water resources management. *International journal of sustainable development & world ecology* 28, 109–124

Gaudou, B., Huynh, N. Q., Philippon, D., Brugière, A., Chapuis, K., Taillandier, P., et al. (2020). Comokit : A modeling kit to understand, analyze, and compare the impacts of mitigation policies against the covid-19 epidemic at the scale of a city. *Frontiers in public health* 8, 563247

Gil-Quijano, J., Louail, T., and Hutzler, G. (2010). From biological to urban cells : Lessons from three multilevel agent-based models. *Principles and Practice of Multi-Agent Systems* , 620

Giupponi, C., Ausseil, A.-G., Balbi, S., Cian, F., Fekete, A., Gain, A. K., et al. (2022). Integrated modelling of social-ecological systems for climate change adaptation. *Socio-Environmental Systems Modelling* 3, 18161–18161

Gotts, N. M., van Voorn, G. A., Polhill, J. G., de Jong, E., Edmonds, B., Hofstede, G. J., et al. (2019). Agent-based modelling of socio-ecological systems : Models, projects and ontologies. *Ecological Complexity* 40, 100728

Grignard, A., Macià, N., Alonso Pastor, L., Noyman, A., Zhang, Y., and Larson, K. (2018). Cityscope andorra : a multi-level interactive and tangible agent-based visualization. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*. 1939–1940

Grignard, A., Nguyen-Huu, T., Gaudou, B., Nguyen-Ngoc, D., Brugière, A., Dang-Huu, T., et al. (2020). Cityscope hanoi : interactive simulation for water management in the bac hung hai irrigation system. In *2020 12th International Conference on Knowledge and Systems Engineering (KSE)* (IEEE), 153–158

Grimm, V., Berger, U., DeAngelis, D. L., Polhill, J. G., Giske, J., and Railsback, S. F. (2010). The odd protocol : a review and first update. *Ecological modelling* 221, 2760–2768

Grimm, V., Railsback, S. F., Vincenot, C. E., Berger, U., Gallagher, C., DeAngelis, D. L., et al. (2020). The odd protocol for describing agent-based and other simulation models : A second update to improve clarity, replication, and structural realism. *Journal of Artificial Societies and Social Simulation* 23

Groeneveld, J., Müller, B., Buchmann, C. M., Dressler, G., Guo, C., Hase, N., et al. (2017). Theoretical foundations of human decision-making in agent-based land use models–a review. *Environmental modelling & software* 87, 39–48

Grosjean, L., Drogoul, A., Herrmann, B., Huynh, N. Q., Lang, C., Marilleau, N., et al. (2025). Distribution Model : Separation of Concerns to Facilitate the Distribution of Agent-Based Models. In *PAAMS 2025* (Lille, France)

Grosjean, L., Huynh, N. Q., Lang, C., Marilleau, N., Philippe, L., Bénédicte, H., et al. (2024). Modèle de distribution : une approche pour faciliter la distribution de modèles à base d'agents. *Journées Francophones sur les Systèmes Multi-Agents (JFSMA)* 1, 67–76

Gutknecht, O. and Ferber, J. (2000). The madkit agent platform architecture. In *Workshop on infrastructure for scalable multi-agent systems at the international conference on autonomous agents* (Springer), 48–55

Hall, R., Pauls, K., McCulloch, S., and Savage, D. (2011). *OSGi in action : Creating modular applications in Java* (Manning Publications Co.)

Henderson, L. F. (1974). On the fluid mechanics of human crowd motion. *Transportation research* 8, 509–515

Himmelspach, J. and Uhrmacher, A. M. (2009). The james ii framework for modeling and simulation. In *2009 International Workshop on High Performance Computational Systems Biology* (IEEE), 101–102

Hjorth, A., Head, B., Brady, C., and Wilensky, U. (2020). Levelspace : A netlogo extension for multi-level agent-based modeling. *Journal of Artificial Societies and Social Simulation* 23

Huynh, Q.-N. (2016). *CoModels, engineering dynamic compositions of coupled models to support the simulation of complex systems.* Ph.D. thesis, Université Pierre et Marie Curie-Paris VI

Iskandar, R., Dugdale, J., Beck, E., and Cornou, C. (2024). Agent-based simulation of seismic crisis including human behavior : application to the city of beirut, lebanon. *Simulation* 100, 357–377

Janssen, M. A. and Ostrom, E. (2006). Empirically based, agent-based models. *Ecology and society* 11

Kadanoff, L. P. (2009). More is the same ; phase transitions and mean field theories. *Journal of Statistical Physics* 137, 777–797

Kafando, R., Ho, T. V., and Nguyen, M. H. (2019). An agent-based simulation for studying air pollution from traffic in urban areas : The case of hanoi city. *International Journal of Advanced Computer Science and Applications* 10

Kang, J.-Y. and Aldstadt, J. (2019). Using multiple scale spatio-temporal patterns for validating spatially explicit agent-based models. *International Journal of Geographical Information Science* 33, 193–213

Karsakov, A., Moiseev, A., Mukhina, K., Ankudinova, I. N., Ignatieva, M. A., Krotov, E., et al. (2016). Toolbox for visual explorative analysis of complex temporal multiscale contact networks dynamics in healthcare. *Procedia Computer Science* 80, 2107–2118

Kaye, D. (2003). *Loosely coupled : the missing pieces of Web services* (RDS Strategies LLC)

Keeling, M. J. and Rohani, P. (2008). *Modeling infectious diseases in humans and animals* (Princeton university press)

Kelly, R. A., Jakeman, A. J., Barreteau, O., Borsuk, M. E., ElSawah, S., Hamilton, S. H., et al. (2013). Selecting among five common modelling approaches for integrated environmental assessment and management. *Environmental modelling & software* 47, 159–181

Kermack, W. O. and McKendrick, A. G. (1927). A contribution to the mathematical theory of epidemics. *Proceedings of the royal society of london. Series A, Containing papers of a mathematical and physical character* 115, 700–721

Khiewbanyang, N., Khudamrongsawat, J., Saralamba, C., and Nathalang, A. (2017). The relationships between host tree characteristics and liana climbing success at mo singto forest dynamics plot, khao yai national park, thailand. *Tropical Natural History* 17, 1–10

Kiselev, A. V., Karbovskii, V. A., and Kovalchuk, S. V. (2016). Agent-based modelling using ensemble approach with spatial and temporal composition. *Procedia Computer Science* 80, 530–541

Krajzewicz, D., Hertkorn, G., Rössel, C., and Wagner, P. (2002). Sumo (simulation of urban mobility)-an open-source traffic simulation. In *Proceedings of the 4th middle East Symposium on Simulation and Modelling (MESM20002)*. 183–187

Kuhl, F., Weatherly, R., and Dahmann, J. (1999). *Creating computer simulation systems : an introduction to the high level architecture* (Prentice Hall PTR)

Lagadeuc, Y. and Chenorkian, R. (2009). Les systèmes socio-écologiques : vers une approche spatiale et temporelle. *Natures Sciences Sociétés* 17, 194–196

Lara, J. d. and Vangheluwe, H. (2002). Atom 3 : A tool for multi-formalism and meta-modelling. In *Fundamental Approaches to Software Engineering : 5th International Conference, FASE 2002 Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2002 Grenoble, France, April 8–12, 2002 Proceedings 5* (Springer), 174–188

Le-Kim, T., Nguyen-Thi-Ngoc, A., Nguyen-Ngoc, D., Quang, N. H., and Amouroux, E. (2016). A multi-scale model for spreading of infectious disease in an office building. In *2016 IEEE RIVF International Conference on Computing & Communication Technologies, Research, Innovation, and Vision for the Future (RIVF)* (IEEE), 199–204

Lepagnot, J. and Hutzler, G. (2009). A multiscale agent-based model for the simulation of avascular tumour growth. *Journal of Biological Physics and Chemistry* 9, 17–25

Lesquoy, B., Taillandier, P., Gaudou, B., Chapuis, K., Grignard, A., Huynh, N. Q., et al. (2024). Behind the scenes : an overview of the gama platform's environment and development practices. In *GAMA Days 2024*

Lèye, B., Nguyen, D. N., Monga, O., Garnier, P., and Nunan, N. (2015). Simulating biological dynamics using partial differential equations : application to decomposition of organic matter in 3d soil structure. *Vietnam Journal of Mathematics* 43, 801–817

Lin, Y., Hilaire, V., Gaud, N., and Koukam, A. (2011). K-crio : An ontology for organizations involved in product design. In *Digital Information and Communication Technology and Its Applications*, eds. H. Cherifi, J. M. Zain, and E. El-Qawasmeh (Berlin, Heidelberg : Springer Berlin Heidelberg), 362–376

Lippe, M., Bithell, M., Gotts, N., Natalini, D., Barbrook-Johnson, P., Giupponi, C., et al. (2019). Using agent-based modelling to simulate social-ecological systems across scales. *GeoInformatica* 23, 269–298

Liu, J., Dietz, T., Carpenter, S. R., Folke, C., Alberti, M., Redman, C. L., et al. (2007). Coupled human and natural systems. *AMBIO : a journal of the human environment* 36, 639–649

Lu, Y., Basak, K., Carrion, C., Loganathan, H., Adnan, M., Pereira, F. C., et al. (2015). Simmobility mid-term simulator : A state of the art integrated agent based demand and supply model. *Transportation Research Board 94th Annual MeetingTransportation Research Board*

Marcenac, P. and Giroux, S. (1998). Geamas : A generic architecture for agent-oriented simulations of complex processes. *Applied Intelligence* 8, 247–267

Mathieu, P., Morvan, G., and Picault, S. (2018). Multi-level agent-based simulations : Four design patterns. *Simulation Modelling Practice and Theory* 83, 51–64

McConnell, S. (2004). *Code complete* (Pearson Education)

Mercure, J.-F., Pollitt, H., Bassi, A. M., Viñuales, J. E., and Edwards, N. R. (2016). Modelling complex systems of heterogeneous agents to better design sustainability transitions policy. *Global environmental change* 37, 102–115

Missaoui, E., Mazigh, B., Bhiri, S., and Hilaire, V. (2017). Ncrio : A normative holonic metamodel for multi-agent systems. In *Hybrid Artificial Intelligent Systems*, eds. F. J. Martínez de Pisón, R. Urraca, H. Quintián, and E. Corchado (Cham : Springer International Publishing), 638–649

Morvan, G. (2012). Multi-level agent-based modeling-a literature survey. *arXiv preprint arXiv :1205.0561*

Nguyen, N. D. (2010). *Coupling equation-based and individual-based models in the study of complex systems : a case study in theoritical population ecology.* Ph.D. thesis, Paris 6

Nguyen, N. D., Drogoul, A., and Auger, P. (2008). Methodological steps and issues when deriving individual based-models from equation-based models : A case study in population dynamics. *Intelligent Agents and Multi-Agent Systems* , 295

Nguyen, N. D., Phan, T. H. D., Nguyen, T. N. A., Drogoul, A., and Zucker, J.-D. (2010). Disk graph-based model : a graph theoretical approach for linking agent-based models and dynamical systems. In *2010 IEEE RIVF International Conference on Computing & Communication Technologies, Research, Innovation, and Vision for the Future (RIVF)* (IEEE), 1–4

Nguyen, N. D., Taillandier, P., Drogoul, A., and Auger, P. (2012). Inferring equation-based models from agent-based models : A case study in competition dynamics. *Principles and Practice of Multi-Agent Systems* , 413

Nguyen-Trong, K., Nguyen-Thi-Ngoc, A., Nguyen, D. N., and Dinh-Thi-Hai, V. (2017). Optimization of municipal solid waste transportation by integrating gis analysis, equation-based, and agent-based model. *Waste management* 59, 14–22

Parr, T., Sajid, N., and Friston, K. J. (2020). Modules or mean-fields ? *Entropy* 22, 552

Pike, T. (2019). Multi-level mesa. *arXiv preprint arXiv :1904.08315*

Plews-Ogan, E., Mariola, M. J., and Ananta, A. (2017). Polyculture, autonomy, and community : the pursuit of sustainability in a northern thai farming village. *International Journal of Agricultural Sustainability* 15, 418–431

Quesnel, G., Duboz, R., and Ramat, É. (2009). The virtual laboratory environment– an operational framework for multi-modelling, simulation and analysis of complex dynamical systems. *Simulation Modelling Practice and Theory* 17, 641–653

Rahman, M. M., Nguyen, R., and Lu, L. (2022). Multi-level impacts of climate change and supply disruption events on a potato supply chain : An agent-based modeling approach. *Agricultural Systems* 201, 103469

Railsback, S. F. and Grimm, V. (2019). *Agent-based and individual-based modeling : a practical introduction* (Princeton university press)

Reynolds, C. W. (1987). Flocks, herds, and schools : A distributed behavioral model. *Computer Graphics* 21

Rijpma, J. A. (1997). Complexity, tight–coupling and reliability : Connecting normal accidents theory and high reliability theory. *Journal of contingencies and crisis management* 5, 15–23

Sangsupan, H. A., Hibbs, D. E., Withrow-Robinson, B. A., and Elliott, S. (2021). Effect of microsite light on survival and growth of understory natural regeneration during restoration of seasonally dry tropical forest in upland northern thailand. *Forest Ecology and Management* 489, 119061

Schlüter, M., Baeza, A., Dressler, G., Frank, K., Groeneveld, J., Jager, W., et al. (2017). A framework for mapping and comparing behavioural theories in models of social-ecological systems. *Ecological economics* 131, 21–35

Schulze, J., Müller, B., Groeneveld, J., and Grimm, V. (2017). Agent-based modelling of social-ecological systems : achievements, challenges, and a way forward. *Journal of Artificial Societies and Social Simulation* 20

Schwarz, N., Dressler, G., Frank, K., Jager, W., Janssen, M., Müller, B., et al. (2020). Formalising theories of human decision-making for agent-based modelling of social-ecological systems : practical lessons learned and ways forward. *Socio-Environmental Systems Modelling* 2, 16340–16340

Serugendo, G. D. M., Irit, M.-P., and Karageorgos, A. (2006). Self-organisation and emergence in mas : An overview. *Informatica* 30, 45–54

Servat, D., Perrier, E., Treuil, J.-P., and Drogoul, A. (1998). When agents emerge from agents : Introducing multi-scale viewpoints in multi-agent simulations. In *International Workshop on Multi-Agent Systems and Agent-Based Simulation* (Springer), 183–198

Simon, H. A. (1991). The architecture of complexity. In *Facets of systems science* (Springer). 457–476

Soares, G., Kokkinogenis, Z., Macedo, J. L., and Rossetti, R. J. (2013). Agent-based traffic simulation using sumo and jade : an integrated platform for artificial transportation systems. In *Simulation of Urban MObility User Conference* (Springer), 44–61

Steinbrink, C., van der Meer, A. A., Cvetkovic, M., Babazadeh, D., Rohjans, S., Palensky, P., et al. (2018). Smart grid co-simulation with mosaik and hla : a comparison study. *Computer Science-Research and Development* 33, 135–143

Steiniger, A., Krüger, F., and Uhrmacher, A. M. (2012). Modeling agents and their environment in multi-level-devs. In *Proceedings of the 2012 Winter Simulation Conference (WSC)* (IEEE), 1–12

Ta, X. H., Gaudou, B., Longin, D., and Ho, T. V. (2017a). Emotional contagion model for group evacuation simulation. *Informatica : an international journal of computing and informatics* 41, 169–182

Ta, X. H., Longin, D., Gaudou, B., Ho, T. V., and Nguyen, M.-H. (2017b). Modeling and simulation of the effects of social relation and emotion on decision making in emergency evacuation. *International Journal of Advanced Computer Science and Applications* 8, 371–392

Taillandier, F., Di Maiolo, P., Taillandier, P., Jacquenod, C., Rauscher-Lauranceau, L., and Mehdizadeh, R. (2021). An agent-based model to simulate inhabitants' behavior during a flood event. *International Journal of Disaster Risk Reduction* 64, 102503

Taillandier, P., Brugière, A., and Drogoul, A. (2024a). Simple toolchain : Coupling gama simulation with vr universe. In *GAMA Days 2024*

Taillandier, P., Chapuis, K., Gaudou, B., Brugière, A., and Drogoul, A. (2024b). Comokit v2 : A multi-scale approach to modeling and simulating epidemic control policies. *Plos one* 19, e0299626

Taillandier, P. and Drogoul, A. (2010). From gis data to gis agents, modeling with the gama simulation platform. In *Technical Forum Group on Agent and Multi-agent-based Simulation : 1st meeting collocated with Eumas.* vol. 10

Taillandier, P., Gaudou, B., Grignard, A., Huynh, Q.-N., Marilleau, N., Caillou, P., et al. (2019). Building, composing and experimenting complex spatial models with the gama platform. *GeoInformatica* 23, 299–322

Taillandier, P., Vo, D.-A., Amouroux, E., and Drogoul, A. (2010). Gama : bringing gis and multi-level capabilities to multi-agent simulation. In *European Workshop on Multi-Agent Systems*

Taillandier, P., Vo, D.-A., Amouroux, E., and Drogoul, A. (2012). Gama : a simulation platform that integrates geographical information data, agent-based modeling and multi-scale control. In *Principles and Practice of Multi-Agent Systems : 13th International Conference, PRIMA 2010, Kolkata, India, November 12-15, 2010, Revised Selected Papers 13* (Springer), 242–258

Tchappi, I. H., Galland, S., Kamla, V. C., and Kamgang, J. C. (2018). A brief review of holonic multi-agent models for traffic and transportation systems. *Procedia computer science* 134, 137–144

Thinh, D.-T., Dong, H.-V., Doanh, N.-N., and Anh, N.-T.-N. (2017). Coupling statistical and agent-based models in the optimization of traffic signal control. In *International Conference on Industrial Networks and Intelligent Systems* (Springer), 197–211

Thuy, N. P. (2018). *Marker Assisted Selected for Enhanced Zinc Content in Rice : Nguyen Phuong Thuy.* Ph.D. thesis, University of Agricultural Sciences, GKVK.

Tiansawat, P., Elliott, S. D., and Wangpakapattanawong, P. (2022). Climate niche modelling for mapping potential distributions of four framework tree species : implications for planning forest restoration in tropical and subtropical asia. *Forests* 13, 993

Tisue, S. and Wilensky, U. (2004). Netlogo : A simple environment for modeling complexity. In *International conference on complex systems* (Boston, MA), vol. 21, 16–21

Uhrmacher, A. M. and Weyns, D. (2018). *Multi-Agent systems : Simulation and applications* (CRC press)

Van Toll, W., Braga, C., Solenthaler, B., and Pettré, J. (2020). Extreme-Density Crowd Simulation : Combining Agents with Smoothed Particle Hydrodynamics. In *MIG 2020 - 13th ACM SIGGRAPH Conference on Motion, Interaction and Games* (Virtual Event, United States : ACM), 1–10. doi: 10.1145/3424636.3426896

Verburg, P. H., Erb, K.-H., Mertz, O., and Espindola, G. (2013). Land system science : between global challenges and local realities. *Current opinion in environmental sustainability* 5, 433–437

Vo, D. A. (2012). *An operational architecture to handle multiple levels of representation in agent-based models.* Ph.D. thesis, Paris 6

Vo, D.-A., Drogoul, A., and Zucker, J.-D. (2012a). An operational meta-model for handling multiple scales in agent-based simulations. In *2012 IEEE RIVF International Conference on Computing & Communication Technologies, Research, Innovation, and Vision for the Future* (IEEE), 1–6

Vo, D.-A., Drogoul, A., Zucker, J.-D., and Ho, T.-V. (2012b). A modelling language to represent and specify emerging structures in agent-based model , 212–227

Voinov, A. and Shugart, H. H. (2013). 'integronsters', integral and integrated modeling. *Environmental Modelling & Software* 39, 149–158

Wall, F. (2016). Agent-based modeling in managerial science : an illustrative survey and study. *Review of Managerial Science* 10, 135–193

Wang, Z. and Maini, P. K. (2017). Editorial special section on multiscale cancer modeling. *IEEE Transactions on Biomedical Engineering* 64, 501–503

Wu, J. (2004). Effects of changing scale on landscape pattern analysis : scaling relations. *Landscape ecology* 19, 125–138

Xiong, M., Tang, S., and Zhao, D. (2013). A hybrid model for simulating crowd evacuation. *New Generation Computing* 31, 211–235

Yang, L., van Dam, K. H., Anvari, B., and Zhang, L. (2021). Multi-level agent-based simulation for supporting transit-oriented development in beijing. In *International Workshop on Agent-Based Modelling of Urban Systems (ABMUS)*. 17

Zeigler, B. P., Moon, Y., Kim, D., and Ball, G. (1997). The devs environment for high-performance modeling and simulation. *IEEE Computational Science and Engineering* 4, 61–71

Zeigler, B. P., Praehofer, H., and Kim, T. G. (2000). *Theory of modeling and simulation* (Academic press)

Zhanli Sun, I. L., Millington, J. D., Lauf, S., and Magliocca, N. R. (2016). Simple or complicated agent-based models ? a complicated issue. *Environmental Modelling & Software* 86, 56–67