Séparation des préoccupations en épidémiologie

Thi-Mai-Anh Bui

THÈSE DE DOCTORAT DE
l'UNIVERSITÉ PIERRE ET MARIE CURIE

Spécialité : **Informatique**

École Doctorale Informatique, Télécommunications et Électronique (Paris)

Présentée par

BUI Thi Mai Anh

Pour obtenir le grade de

DOCTEUR de l'UNIVERSITÉ PIERRE ET MARIE CURIE

Sujet de la thèse :

Séparation des Préoccupations en Épidémiologie

soutenue le 09 Décembre 2016 devant le jury composé de :

Directeur de thèse :	**Mikal Ziane**	Maître de Conférences HDR, Université Paris Descartes, UMR 7606 LIP6 (UPMC/CNRS)
Encadrants de thèse :	**Serge Stinckwich**	Maître de Conférences, Université de Caen Normandie, UMI 209 UMMISCO (IRD/UPMC)
	Benjamin Roche	Chargé de Recherches, IRD, UMI 209 UMMISCO (IRD/UPMC)
Rapporteurs :	**Mireille Blay-Fornarino**	Professeur des Universités, Université de Nice - Sophia Antipolis, UMR 7271 I3S (UNS/CNRS)
	Eric Ramat	Professeur des Universités, Université du Littoral Côte d'Opale, EA 4491 LISIC
Examinateurs :	**Bernard Cazelles**	Professeur des Universités, Université de Pierre et Marie Curie, UMI 209 UMMISCO (IRD/UPMC)
	Laurence Duchien	Professeur des Universités, Université Lille 1, UMR 9189 CRIStAL
	Sébastien Mosser	Maître de Conférences, Université de Nice - Sophia Antipolis, UMR 7271 I3S (UNS/CNRS)

Université Pierre & Marie Curie - Paris 6
Bureau d'accueil, inscription des doctorants
Esc G, 2ème étage
15 rue de l'école de médecine

75270-PARIS CEDEX 06 Tél. Secrétariat : 01 44 27 28 10
Fax : 01 44 27 23 95
Tél. pour les étudiants de A à EL : 01 44 27 28 07
Tél. pour les étudiants de EM à MON : 01 44 27 28 05
Tél. pour les étudiants de MOO à Z : 01 44 27 28 02
E-mail : scolarite.doctorat@upmc.fr

Résumé

Thèse intitulée : Séparation des Préoccupations en Épidémiologie

La modélisation mathématique est une méthode puissante et souvent utilisée dans de nombreuses disciplines scientifiques pour étudier des systèmes complexes. Des années récentes, elle est largement utilisée pour effectuer des recherches sur la modélisation des maladies infectieuses afin d'étudier les mécanismes de transmission, d'explorer les caractéristiques des épidémies, de prévoir l'évolution de l'épidémie et d'évaluer des stratégies pour trouver un meilleur programme-contrôle. Tandis que la modélisation mathématique se focalise sur la construction d'un modèle conceptuel, la modélisation informatique permet d'implémenter ce modèle comme un programme de simulation (modèles informatiques). Les modèles informatiques donnent des résultats numériques qui sont utilisés pour étudier l'évolution de la dynamique de population, estimer des paramètres et comparer l'efficacité des différentes stratégies de contrôle. Cependant, la complexité apparente de certains modèles épidémiologiques peut les rendre moins attractifs pour les épidémiologistes en raison de la compétence de programmation exigée pour les implémenter. Combler le fossé entre les modèles conceptuels et leurs simulations est l'un des problèmes de la modélisation. Le passage d'un modèle mathématique à informatique peut entraîner quelques problèmes liés au manque d'expressivité, à la possibilité de réutilisation et à l'adaptabilité aux nouvelles situations à cause des détails d'implémentation.

Les langages métiers sont souvent utilisés pour addresser ces problèmes en séparant deux aspects de la modélisation : la spécification (modèles conceptuels) et la simulation (modèles informatiques). Un langage métier est un langage de programmation ou de modélisation qui offre, à travers des notations et abstractions appropriées, un pouvoir d'expression restreint à un domaine particulier. Grâce à sa focalisation sur un domaine, un langage métier est une solution représentant explicitement la sémantique du domaine. Dans cette perspective, nous développons un langage métier, appelé Kendrick, dédié à la modélisation épidémiologique, couplé avec une plate-forme de simulation. Ce langage permet de formaliser de façon descriptive des modèles épidémiologiques reposant sur l'architecture en compartiments et de produire ensuite plein versions dépendant de l'objectif de la modélisation comme : déterministe, stochastique, individus basés ainsi que des versions en langages de programmation généralistes.

Un autre problème de la modélisation en épidémiologie est le mélange des aspects de domaine qui doivent être séparés. En effet, un modèle épidémiologique pourrait se composer de plusieurs aspects liés à la distribution de la population dans l'espace, à la décomposition de la population par les intervalles d'âge, sexes, espèces ou souches

virales etc. Ces aspects (ou préoccupations) peuvent s'interagir entre eux pour représenter des scénarios plus complexes de l'épidémiologie, par exemple la transmission de multiples maladies infectieuses dans l'espace. Afin de faciliter l'écriture et l'évolution des modèles, il est crucial de pouvoir définir une préoccupation avec aussi peu de dépendances avec d'autres que possible et de pouvoir les combiner aussi librement que possible. Nous abordons ces défis en proposant un méta-modèle mathématique commun qui peut représenter les modèles ainsi que les préoccupations. Nous définissons ensuite les opérateurs qui permettent de combiner des préoccupations ainsi que de les appliquer dans un modèle. Le langage Kendrick simplifie donc la programmation des simulations épidémiologiques en décomposant un modèle monolithique hautement-couplé en préoccupations modulaires. Cela rend alors plus facile la construction des modèles complexes de l'épidémiologie où plusieurs préoccupations sont considérées en même temps.

Mots clefs : modélisation épidémiologique, modèles à base des compartiments, langages métiers, séparation des préoccupations

---

Separation of concerns in epidemiologie

Abstract

Mathematical and computational models have become widely used and demanded tools for examining mechanisms of transmission, exploring characteristics of epidemics, predicting future courses of an outbreak and evaluating strategies to find a best control-program. While the mathematical modelling focuses on establishing a conceptual model, the computational modelling helps implement such model as a simulation program. Numerical results given by simulation models can be used to investigate the evolutionary dynamics, to estimate parameters and to compare the effectiveness of different potential control strategies. However, the apparent complexity of epidemic models can decrease the attractiveness of epidemiologists because of the programming competence required for implementing such models. One of the problems of modelling is bridging the gap between conceptual models (i.e compartmental models of epidemiology) and their computer simulation (through deterministic, stochastic or agent-based implementation). In fact, going from a conceptual model to a computational one requires some programming skills, which could be very rudimentary for the deterministic implementation to extremely complex for agent-based models. Complex algorithms, where all implementation specificities cannot be detailed, can host a large number of code portion that could impact on the disease dynamics simulated. Thus such complex models can suffer of a lack of validation that breaks the link with the abundant literature on mathematical epidemiology.

Domain Specific Languages (DSLs) are often used to address such difficulties by separating two concerns of modelling, specification (conceptual model) and implementation (computational model). As opposed to General-purpose Programming Languages (GPLs), DSLs are higher level languages that provide a more expressive syntax based on abstractions and notations representing directly the concepts of studied domain. Because of its sharp focus, DSLs facilitate quick and effective software development, yielding programs that are easy to understand and maintain. In this perspective, we develop a DSL called Kendrick targeted to the epidemiological modelling and coupled with a simulation platform that allows the study of such models. The modelling language allows to specify compartmental models of epidemiology using a high-level expressive syntax. Then a single specified model can be simulated in different approaches i.e deterministic, stochastic and agent-based simulations.

An important issue needs to be addressed in the context of epidemiological modelling is the heterogeneities introduced by separate concerns. A concern is any aspect of a model that can be meaningfully combined with other concerns or at least with models. An epidemiological model may include several concerns related to the distribution of the studied population into spatial regions, age intervals, sexes, species or viral strains... Some concerns may be defined separately, for example a general concept of spatial distribution may be defined independently of the concept of species. It may however happen that some species involved in the studied disease are not uniformly distributed on space. This is caused by the spatial heterogeneities, without those the spatial concern is useless. In order to facilitate the specification of models and their evolution, it is crucial to be able to define concerns with as few dependencies with each other as possible and to combine them as freely as possible. We address such challenges by proposing a common mathematical meta-model that supports both concerns and models and enabling their compositions by some operators. We then implement our proposal language Kendrick based on this meta-model. The language simplifies the construction of complex epidemiological models by decomposing them into modular concerns, by which common concerns can be reused across models and can be easily changed. These include the spatial structure, the strains of pathogen, the transmission cycle and the control strategies.

Keywords: epidemiological modelling, compartmental models, domain specific language, separation of concerns

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction

Sommaire

1.1 Le contexte

Depuis le début des années 1980, les populations humaines se trouvent dans une période d'émergences et de ré-émergences de maladies infectieuses particulièrement préoccupante [67, 79, 113]. Ces nouvelles menaces, comme le SIDA au début des années 1980 [75] ou la dengue aujourd'hui sur le pourtour méditerranéen [58], nécessite de rapidement comprendre les mécanismes de transmission et de pathogénicité de ces agents infectueux afin d'en mitiger les effets dévastateurs.

La nécessité des modèles en épidémiologie a été comprise très tôt par la communauté scientifique. Le premier modèle mathématique a été proposé par Daniel Bernoulli en 1760 pour défendre la pratique de l'inoculation contre la variole [37]. Son travail a été pionnier, car son argument repose de façon essentielle sur une démarche de nature mathématique, marquant ainsi la naissance d'un domaine dénommé aujourd'hui « biomathématique ». Près de deux siècles plus tard, Kermack et McKendrick ont analysé en profondeur le concept des modèles en compartiments en utilisant un ensemble d'équations différentielles ordinaires pour décrire le déroulement d'une épidémie [70]. C'est ce qui est appelé aujourd'hui le modèle SIR. L'architecture de ces modèles, basée sur une classification des individus en Susceptibles (pas encore infectés), Infectieux (pouvant transmettre la maladie) et Rétablis (étant immunisés contre le pathogène) proche du statut infectieux et donc compréhensible par le corps médical, est encore aujourd'hui le formalisme le plus utilisé [9, 69].

L'un des premiers apports significatif des modèles en épidémiologie a été la caractérisation d'un effet de seuil pour la survenue éventuelle d'une épidémie. En effet, l'épidémiologie mathématique a démontré que pour qu'une épidémie éclate, le nombre d'infections qu'un individu malade peut provoquer durant sa période infectieuse doit être supérieur à 1 [38]. Grâce à l'analyse de ces modèles, il a été possible d'exprimer mathématiquement, sous la forme d'une variable appelé R_0, l'expression de ce seuil en fonction des paramètres des populations et des pathogènes [60]. L'avènement de ces modèles mathématiques en épidémiologie a donc permis intuitivement d'identifier sur quels paramètres agir pour faire passer cette valeur en dessous de 1 et donc de pouvoir éviter une épidémie. Parmi ces paramètres, le nombre d'individus susceptibles, pouvant être grandement réduit grâce à la vaccination, est particulièrement important. Aujourd'hui encore, c'est ce phénomène de seuil qui est utilisé comme base pour concevoir les politiques de santé publique sur les maladies à prévention vaccinale [18, 77].

Néanmoins, ces modèles ont leurs limitations. En effet, la plus grande partie de la théorie autour de ces modèles est basée sur la rougeole qui est pour bien des raisons une des maladies les plus faciles à étudier (pathogène en stase évolutive, forte contagiosité, facilité de détection des symptômes, existence de longues séries temporelles avant et après vaccination, etc.) [3, 120]. A l'inverse de la rougeole, la grande majorité des maladies infectieuses possèdent des spécificités bien plus complexes à intégrer dans une approche de modélisation. Sans vouloir être exhaustif, il y a les processus évolutifs, comme l'échappement immunitaire des variants antigéniques de la grippe [53], écologiques, comme la présence des multiple routes de transmission du choléra [117], sociaux, comme les contacts entre les différentes classes d'âges pour les maladies infantiles [83], ou encore de recueil de données, comme pour les cas asymptomatiques chez la dengue [28]. En conséquence, ces modèles peuvent connaître une sophistication considérable par l'ajout de nombreux autres statuts infectieux (latence, asymptomatique, porteurs

sains, etc.) ainsi que par l'intégration de différents aspects de l'épidémiologie (structure sociale comme sexe, structure d'âge, spatial, stratégies de contrôle etc.), ce qui rend leur utilisation compliquée pour des non-initiés. **Un premier espace de variation** comprend donc les différents type de schéma infectieux (SIR, SEIR, ...) et ces différents aspects épidémiologiques.

Une fois que les modèles ont été formalisés pour répondre à une question précise, ils peuvent être implémentés avec de nombreux outils mathématiques différents. De manière chronologique, l'évolution au cours du temps de la dynamique d'un modèle épidémiologique a été abordée avec des approches déterministes, basées sur un système d'équations différentielles ordinaires (EDO) ou d'équations aux dérivées partielles (EDP) lorsque des processus graduels sont considérés. Ceux-ci pourraient être analysés analytiquement (recherche d'équilibre, diagramme de bifurcation, etc...) et/ou être résolus grâce à des techniques numériques d'intégration fournissant une approximation continue de la dynamique de population [56]. Cette approche étudie le modèle au niveau macroscopique et est particulièrement utile pour comprendre la dynamique moyenne. Si les modèles déterministes sont aisés à étudier, les modèles stochastiques sont en revanche censés être plus réalistes. Dans ce cas, le modèle est formulé de manière probabiliste, souvent comme un processus de Markov avec des variables discrètes dont la dynamique stochastique est obtenue et analysée le plus souvent en épidémiologie par des techniques de simulation (par exemple : avec les algorithmes de Gillespie [54]). Enfin, un modèle multi-agents, généralement stochastique, est parfois utilisé pour pouvoir toucher un niveau plus détaillé au sein duquel les individus de la population interagissent les uns avec les autres [106].

En fonction des objectifs, une approche de modélisation que l'on pense la plus appropriée va être choisie. Par exemple, on utilisera l'approche déterministe pour comprendre le comportement général du modèle (comme la dynamique épidémique au travers des diagrammes de bifurcation). Les approches stochastique et multi-agents, en revanche, permettront de capturer les stochasticités qui sont naturelles lorsque l'on étudie la propagation de la transmission dans un contexte plus réaliste.

Un second espace de variation en modélisation épidémiologique est donc constitué par ce qu'on pourrait appeler le paradigme (déterministe, stochastique, agents) ainsi que par l'algorithme de simulation choisi. **Un dernier espace de variation** concerne tout ce qui est par ailleurs nécessaire dans la chaine de modélisation et de simulation et notamment la visualisation des résultats, leur stockage éventuel etc.

1.2 Le problème

Un grand nombre de travaux en génie logiciel a porté sur les logiciels dont les différents espaces de variation (ou aspects) sont d'une part entremêlés et d'autre part éparpillés dans le code (ou les différents artéfacts) au lieu d'être modulaires c'est-à-dire bien localisés dans des modules (classes, fonctions ...) et séparés les uns des autres. Le fait que les aspects se croisent (*cross-cut*) alors qu'ils devraient pouvoir être compris et en principe devraient pouvoir varier largement indépendamment les uns des autres induit de nombreux défauts : le code est plus difficile à comprendre, à modifier et à réutiliser [64][72].

Une préoccupation est un aspect qu'il est difficile de rendre modulaire avec les abstractions classiques de la programmation (fonctions et classes). La séparation des préoccupations vise à surmonter cette difficulté de façon a éviter les défauts que nous venons de citer [64]. C'est dans ce cadre que se situe cette thèse, bien que la démarche que nous allons proposer, comme nous le verrons, se démarque très nettement des techniques classiques de séparations des préoccupations qui opèrent typiquement au niveau du code.

Plus précisément cette thèse vise à séparer les aspects épidémiologiques (le premier espace de variation que nous avons identifié) des autres aspects (ceux des deux autres espaces de variation) mais surtout de les séparer les uns des autres. En fait les deux enjeux se rejoignent car pour séparer les aspects épidémiologiques les uns des autres nous les formaliserons ce qui prépare naturellement leur séparation des autres aspects.

Dans la suite de ce document et sauf mention contraire nous appellerons donc aspects, ou préoccupations, les aspects épidémiologiques : le statut infectieux (SIR ...), la distribution de la population étudiée dans l'espace, selon les âges, les sexes, les espèces (hôtes, vecteurs), les souches virales ... En effet, comme nous le verrons, ces aspects sont typiquement éparpillés et entremêlés dans les modèles épidémiologique, y compris dans les modèles mathématiques. Dans leurs implémentations, ces aspects sont de plus mêlés aux autres espaces de variations que nous avons mentionnés et exprimés dans des langages de relativement bas niveau comme MATLAB ce qui exige des compétences en programmation. On peut le constater par exemple sur les programmes du site internet de l'ouvrage classique de Keeling et al. "Modeling Infectious Diseases in Humans and Animals" [1].

Les techniques classiques de séparation des préoccupations, notamment la programmation par aspects, se situent à niveau d'abstraction trop bas pour pouvoir tirer pleinement profit de la sémantique d'un domaine applicatif particulier comme la modélisation épidémiologique. Nous avons donc adopté un point de vue plus abstrait.

Notre problème principal consistera donc à déterminer si on peut donner une structure aux modèles épidémiologiques. Au minimum on cherchera un opérateur qui soit une loi de composition interne sur l'ensemble des aspects. De préférence on aimerait que l'opérateur soit associatif et commutatif pour ne pas avoir à se préoccuper des parenthèses ni de l'ordre des aspects dans un modèle. Un problème secondaire consistera à séparer l'expression de cette structure des détails d'implémentation.

Une difficulté nous attend : les aspects de la modélisation épidémiologique paraissent a priori difficiles à séparer du statut infectieux. Considérons par exemple un modèle incluant la distribution spatiale de la population dans certaines régions. Si cette distribution était complètement indépendante du statut infectieux des individus, elle n'aurait guère d'intérêt : autant étudier la population dans son ensemble plutôt que de la découper inutilement en régions. Si un modèle inclut un aspect, c'est parce que ce dernier présente, au moins potentiellement, des hétérogénéités par rapport au statut infectieux. C'est donc qu'il y a des interdépendances entre le statut infectieux et cet aspect, ce qui a priori rend difficile leur séparation.

1.3 Notre proposition

Notre premier objectif est donc de définir de façon aussi générale que possible l'ensemble des aspects en modélisation épidémiologique et de le structurer. Idéalement les modèles seraient même entièrement composés de ces aspects et du schéma infectieux (SIR ou une de ses variantes).

Nous avons mis de côté les modèles agents, trop complexes, pour nous focaliser sur les modèles déterministes et stochastiques en observant que l'on peut passer, sous certaines hypothèses d'une représentation à l'autre. Nous nous concentrons donc sur les modèles stochastiques, bien qu'ils soient parfois exprimés sous forme d'ODE, les hypothèses de stochasticité (distribution de Poisson, ...) étant classiques et donc implicites.

Ces modèles peuvent être représentés par des automates stochastiques dont les états sont les compartiments et dont les transitions sont étiquetées par les taux (les éléments strictement positifs) dans la matrice des taux de transition de la chaîne de Markov. Or nous avons constaté que les aspects qui nous intéressent peuvent eux-aussi être représentés par des automates stochastiques.

En cherchant dans la littérature, nous avons trouvé que ces automates avaient justement été dotés, bien que dans un domaine complètement différent, de l'opérateur idéal : un opérateur tensoriel (la somme tensorielle $\oplus$ dans le cas des CTMC) [95, 96]. Un opérateur tensoriel est idéal parce qu'il donne à l'ensemble des aspects une structure de monoïde.

Nous avons donc pu proposer une définition très générale des aspects et des modèles (des combinaisons d'aspects) en leur donnant les propriétés voulues pour la séparation des préoccupations et ce d'autant plus que l'opérateur tensoriel peut être rendu commutatif.

Toutefois il nous fallait encore permettre d'introduire des hétérogénéités par rapport au statut infectieux comme nous l'avons souligné plus haut. Nous avons pour cela réutilisé la notion de taux fonctionnel de [96] qui consiste à permettre à la définition de taux de transition de dépendre de l'ensemble des états de l'automate. Le point à souligner est que les modèles proprement dits, c'est-à-dire la définition des aspects et leur composition, ne sont pas concernés par ces dépendances induites par le taux fonctionnel qui sont différées à une phase d'instanciation.

De façon à séparer l'expression des modèles des détails d'implémentation nous avons par ailleurs défini et implémenté un langage de domaine (DSL) Kendrick, au dessus d'une plate-forme de simulations déterministes, stochastiques et d'un certaine façon multi-agents (*individual-based*) ainsi qu'une exportation des modèles vers C/C++.

Nous avons réalisé une validation préliminaire de notre démarche en la comparant à la démarche classique en MATLAB sur une série des modèles de plus en plus complexes. Pour cela nous avons mesuré de façon objective un certain nombre de critères liés à la séparation des préoccupations ou à leur réutilisabilité.

Le résultat est sans appel. En MATLAB, les aspects sont éparpillés dans tout le code et mêlés les uns au autre et au reste du code. Chaque passage d'une version à la suivante des modèles induit des changements dans tout le code et à un niveau d'abstraction très

bas. Au contraire, dans les modèles exprimés en Kendrick, les aspects sont exprimés de façon modulaire, concise, dans un langage proche du domaine. Le couplage entre les aspects est minimal. Enfin, l'impact des changements successifs (en passant d'un modèle à la version suivante) est clairement moins important avec Kendrick.

1.4 La structure de la thèse

La présentation des travaux réalisés au cours de cette thèse est organisée de la manière suivante :

Dans le deuxième chapitre, nous rappelons les connaissances de base des modèles de l'épidémiologie et quelques outils qui ont été développés pour leur expression. Pour ce faire, nous commençons par présenter les objectifs de la modélisation de l'épidémiologie et ce qu'on peut faire avec des modèles. Ensuite, nous introduisons les modèles compartimentaux, en définissant les compartiments usuels que l'on rencontre régulièrement dans la littérature, puis en présentant quelques modèles utilisant ce formalisme. En plus de ces modèles de base, nous présentons également certains modèles plus complexes considérant divers aspects complexes en épidémiologie, comme les aspects multi-espèces hôtes/multi-souches de pathogènes, multi-groupes d'âge, l'hétérogénéité spatiale etc. Nous faisons un point sur la grande variabilité des modèles épidémiologiques et décrivons les différentes techniques de simulation.
À la fin de ce chapitre, nous portons un regard attentif à quelques outils de modélisation à la disposition des chercheurs en sciences de la santé publique, y compris les langages de modélisation mathématique, les outils de modélisation intégrés et notamment les langages de modélisation dédiés au domaine épidémiologique. Nous concluons ce chapitre avec la justification de notre choix de développer un langage métier qui permet de faciliter la spécification et la modélisation des processus épidémiologiques.

Nous avons choisi de proposer un outil de modélisation sous forme d'un langage métier qui se nomme Kendrick. Le premier prototype du langage sera présenté dans le chapitre 3. Nous nous focalisons sur la conception d'un langage qui est dédié à la modélisation des modèles à base d'équations en épidémiologie. Notre objectif est de fournir une syntaxe de haut niveau permettant aux modélisateurs de pouvoir facilement spécifier leurs modèles mathématiques sans avoir besoin de faire du codage. Les modèles développés avec Kendrick, ensuite, peuvent être traduits vers différents types de simulation déterministe, stochastique ou multi-agents ainsi que de pouvoir générer le code des simulateurs en C/C++. Ce langage est conçu à l'aide d'un méta-modèle qui se base sur les concepts comme *Population, Compartment, Equation, Variable*.
La seconde partie de ce chapitre a pour but de faire un retour d'expérience sur la conception du premier prototype de langage afin de confirmer qu'il peut s'adapter aux différentes problématiques courantes en épidémiologie. Nous évoquons alors le problème posé par la définition des modèles qui se compose de plusieurs aspects épidémiologiques, auquel le premier méta-modèle proposé ne peut pas répondre. Enfin, nous argumentons la pertinence de proposer un nouveau méta-modèle qui devrait être aussi générique que possible pour pouvoir représenter divers aspects du domaine.

La difficulté de la modélisation en épidémiologie sera détaillée dans la première partie du chapitre 4. Nous analysons un exemple pour justifier le problème causé par le mélange des préoccupations et expliquons la nécessité de la séparation des préoccupations

en épidémiologie. Le reste du chapitre 4 est consacré à la description détaillée de l'approche que nous proposons pour séparer des préoccupations autant que possible afin de faciliter l'écriture des modèles et leur évolution.
Premièrement, nous présentons le méta-modèle mathématique qui capture la dynamique des populations à base de compartiments. Deuxièmement, nous donnons la définition formelle des préoccupations et décrivons certains types de dépendance entre elles. Les préoccupations sont vues comme des transformations de modèle et jugées comme des automates stochastiques. Un modèle sera considéré comme un réseau d'automates stochastiques [1]. Nous proposons ensuite un moyen pour la composition des préoccupations et révélons comment elles interagissent dans un modèle.

Le chapitre 5 se focalise sur la présentation du langage de modélisation Kendrick qui est construit en se basant sur le méta-modèle décrit dans le chapitre 4. Tout d'abord, nous clarifions les objectifs de notre langage, puis décrivons sa syntaxe abstraite et concrète ainsi que ses sémantiques. Les limitations de l'implémentation sont discutées à la fin du chapitre.

Dans le chapitre 6, nous prenons quelques cas d'études en épidémiologie pour vérifier et valider l'approche proposée. Cette étape permet de valider que les modèles générés sont cohérents avec les dynamiques épidémiques attendues.

Finalement, nous dressons le bilan de nos recherches dans la conclusion et décrivons une série de perspectives sur lesquelles peut déboucher ce travail.

1. Stochastic Automata Network

Chapitre 2

Modèles et Outils de l'Épidémiologie

Sommaire

Dans ce chapitre, nous allons décrire brièvement l'approche de modélisation mathématique de la transmission de maladies infectieuses. Nous nous concentrons sur les modèles dit compartimentaux, sur lesquels la plupart des travaux en épidémiologie sont basés. Nous étendrons cette présentation à certains aspects utilisés en épidémiologie tels que la structure d'âge, l'hétérogénéité spatiale, la structure sociale, etc. Les différentes techniques utilisées pour la simulation de ces modèles sont présentées dans la seconde partie. Enfin, nous aborderons quelques outils de modélisation et de simulation généralement utilisés en épidémiologie y compris les langages de modélisation mathématiques, les outils intégrés et les langages métiers.

2.1 Le processus de modélisation en épidémiologie

L'épidémiologie est l'étude de la distribution des maladies et des facteurs qui les influencent. Elle vise à la compréhension des causes des maladies et à l'amélioration de leur traitement et des moyens de prévention. Elle permet de proposer des stratégies nécessaires pour lutter contre la propagation d'une épidémie : traitements curatifs ou préventifs, mise en quarantaine des individus infectés, vaccination ... Cependant, en fonction de la maladie et ses caractéristiques comme le mécanisme de transmission, la probabilité d'infection, le temps d'incubation ou le temps avant la rémission n'ont pas le même impact sur la propagation de la maladie (en termes de nombre d'infectés par exemple). Il est souvent nécessaire d'utiliser une technique spécifique ou un mélange de plusieurs pour trouver la stratégie optimale.

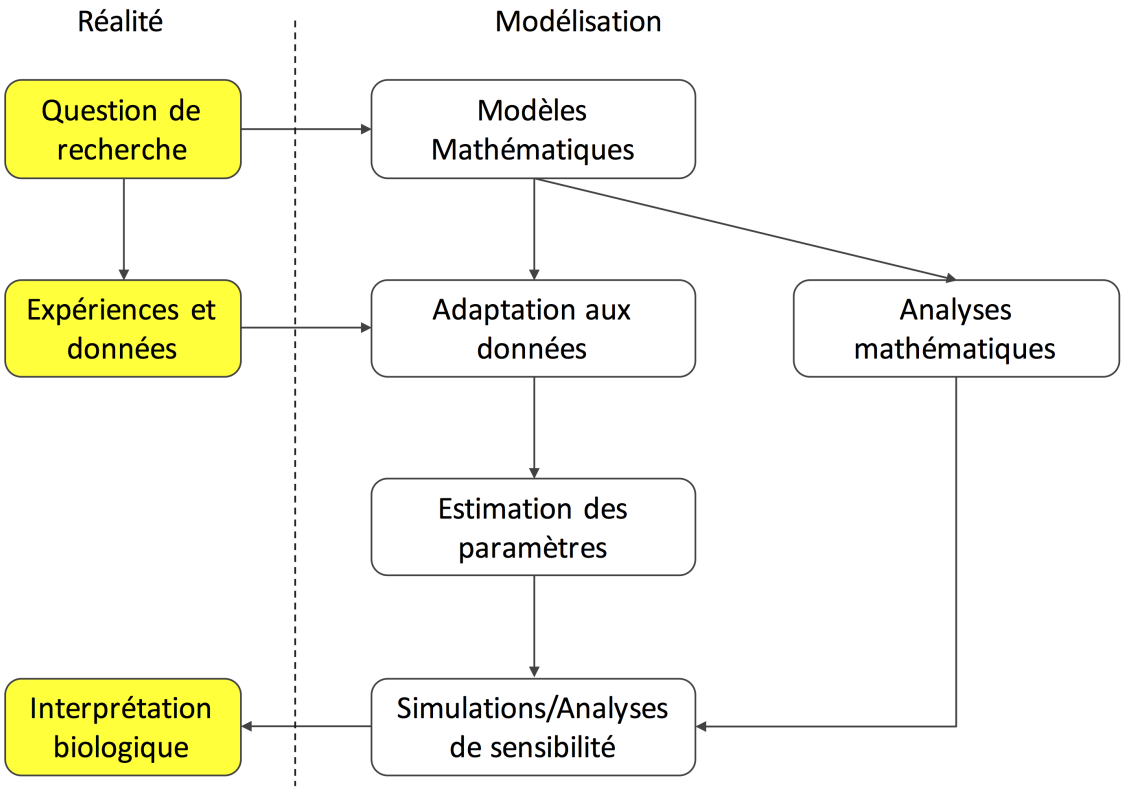


Figure 2.1 – Le processus de modélisation épidémiologique

La modélisation permet d'abord de formaliser les processus épidémiologiques (transmission, guérison etc.), puis le formalisme mathématique permet de les analyser, voire de prévoir leurs conséquences sur la dynamique de l'épidémie. Un modèle mathématique est une description d'un système utilisant les outils et formalismes des mathématiques. L'activité de développement de ces modèles est appelé *modélisation mathématique*.

Le modèle n'est néanmoins pas la réalité et n'est pas supposé la reproduire de façon parfaite. Il doit reproduire au mieux les caractéristiques du phénomène étudié en fonction des objectifs fixés dans le cadre de l'étude. De façon générale, le choix des processus que l'on modélise dépend des questions que l'on se pose. Partant d'une question, on élabore un dispositif *in silico* nous permettant de répondre au mieux à celle-ci.

Le processus de modélisation épidémiologique est représenté dans la figure 2.1. Il s'agit en grande partie du processus décrit dans [30, 81]. Ce processus commence généralement par une description du problème basée sur les connaissances des experts du domaine. La description du système est ensuite encodée en utilisant des outils mathématiques. La traduction vers un modèle mathématique doit être faite avec un objectif précis ou une question de départ claire. Le modèle ne devrait intégrer que les composants qui sont pertinents pour répondre à la question de recherche. Une fois que la formalisation du modèle est complétée, il peut être étudié à l'aide d'outils mathématiques ou informatiques. Voici quelqu'uns des traitements possibles une fois qu'un modèle épidémiologique a été établi :

- On peut réaliser une analyse théorique sur le modèle afin de pouvoir comprendre l'évolution globale de la taille de l'épidémie (par exemple, en terme du nombre d'infectés), les effets de la saisonnalité etc.
- Grâce au modèle, on peut estimer les valeurs des paramètres pour déterminer lesquelles donnent la meilleure approximation par rapport aux observations et décrivent le plus correctement le mécanisme de propagation observée.
- On peut également réaliser des simulations sur le modèle pour étudier la sensibilité des paramètres et évaluer l'efficacité des stratégies de contrôle.
- Quand le portrait général de l'épidémie à laquelle on fait face est disponible (quelles pathogènes, leurs routes de transmission, période d'incubation, période d'infection, etc.) on peut effectuer des prédictions.

2.2 Les modèles à compartiments

La naissance de l'épidémiologie est attribuée à Daniel Bernoulli, qui présenta en 1760 un modèle dont le principal objectif était de savoir si la variolisation (l'inoculation du pus d'une personne atteinte de variole) était plus avantageuse ou plus risquée pour les personnes ayant contracté cette maladie [20, 37]. Près de deux siècles plus tard, Kermack et McKendrick ont formalisé le concept des modèles à compartiments (ou modèles compartimentaux) en utilisant un ensemble d'équations différentielles ordinaires pour décrire le déroulement d'une épidémie [70]. Cette approche est devenue extrêmement populaire et puissante dans les années 80/90 avec les travaux de Bob May et Roy Anderson, qui ont permis de jeter les principes de la modélisation moderne de l'épidémiologie [25, 30, 62, 81].

2.2.1 Le modèle SIR

Le modèle décrit à l'origine par Kermack et McKendrick est aujourd'hui appelé : SIR (Susceptible, Infected, Recovered). Les individus sont donc catégorisés en trois compartiments selon leur statut infectieux, entre *Susceptibles* - ceux qui ne sont pas exposés à la maladie et peuvent être contaminés, *Infectieux* qui ont contracté et peuvent transmettre l'infection et *Rétablis* (ou *Guéris*) qui ont acquis une immunité (définitive ou passagère) (Figure 2.2). On représente généralement le taux de passage des individus d'un compartiment à l'autre. L'évolution du système est assurée par deux transitions : la transition entre individus du compartiment S au compartiment I en fonction des contacts entre individus susceptibles et infectieux et la transition des infectieux dans le compartiment R dès qu'ils sont guéris et deviennent immunisés. Le modèle de Kermack

Figure 2.2 – Le modèle compartimental SIR

et McKendrick suppose que la population est **constante** (pas de naissances ou morts) et les compartiments sont tous **homogènes**, c'est-à-dire que dans le compartiment S, tous les individus ont la même susceptibilité à la maladie. De la même façon, tous les individus infectieux ont la même probabilité de transmettre la maladie lors de la rencontre d'un susceptible et le même taux de guérison. On suppose également que toutes les paires d'individus ont la même probabilité d'établir un contact (loi d'action de masse).

La dynamique du système est réglée par le système d'équations différentielles ordinaires suivant :

$$\begin{cases} \frac{dS}{dt} &= -\beta SI \\ \frac{dI}{dt} &= \beta SI - \gamma I \\ \frac{dR}{dt} &= \gamma I \end{cases} \tag{2.1}$$

où S, I, R dénotent le nombre de susceptibles, d'infectieux et de guéris et on suppose que la démographie n'est pas prise en compte.

β et γ sont les deux paramètres du modèle. Le passage des individus des compartiments S, I dans le compartiment suivant est réglé par les taux de transfert qui sont formulés par des taux de la taille des compartiments (par exemple, βIS, γI).

Avec ce modèle, il est possible de poser des questions clés pour étudier la propagation d'une maladie infectieuse :

- *Si un nouveau agent pathogène est introduit dans une population saine, est-ce qu'il peut provoquer une épidémie ?*
- *Si oui, comment la maladie se propage dans le temps ?*
- *Quelle proportion de la population sera finalement infectée ?*

Nous allons maintenant étudier quelques paramètres du modèle, ceux qui permettent de répondre a ces questions.

2.2.2 La force d'infection

Tout d'abord, on se focalise sur la description du processus de transmission. On définit la « force d'infection », dénotée par λ, comme le taux par habitant auquel les individus susceptibles contractent l'infection. λ est une fonction centrale pour modéliser la dynamique de transmission. Elle résume le processus de transmission entre les susceptibles et les infectieux, et dépend du taux de prévalence [1] de l'infection dans la population

1. En épidémiologie, la prévalence est une mesure de l'état de santé d'une population, dénombrant le nombre de cas de maladies à un instant donné ou sur une période donnée.

$I(t)/N(t)$, le taux de contact c et la probabilité de transmission par contact β [86].

$$\lambda(t) = \beta c' \frac{I(t)}{N(t)} \tag{2.2}$$

où $I(t)$ et $N(t)$ sont respectivement le nombre d'infectés et le total de la population à un instant t.

La transmission entre les infectieux et les susceptibles dépend également de comment la structure de contact est mesurée par rapport à la taille de la population. Il y a deux approches possibles, la transmission *fréquence-dépendant* et *densité-dépendant* :

- La transmission densité-dépendant : Le taux de contact est proportionnel à la densité/la taille de la population. Ainsi, $c' = cN$ et l'équation 2.2 se réduit à $\lambda(t) = \beta c I(t)$.
- La transmission fréquence-dépendant : Le taux de contact est supposé être indépendant de la taille de la population, donc $c' = c = const$ et l'équation 2.2 s'écrit $\lambda(t) = \beta c \frac{I(t)}{N(t)}$.

Pour simplifier les formules, on suppose que β est le produit de deux paramètres : le taux de contact c et la probabilité de transmission par contact. β est également appelé le taux de transmission. Comme on la déjà mentionné, dans une population constante et homogène, le taux de contact et la probabilité de transmission sont supposés constants et identiques pour toutes paires d'individu susceptible-infectieux.

2.2.3 Le taux de guérison

Le taux de guérison, souvent noté par γ, mesure la vitesse pour être guéri d'un individu infectieux. Il est constant dans le temps et pareil pour tous les individus. Son inverse, $1/\gamma$ détermine la durée moyenne de la période infectieuse.

2.2.4 Le taux de reproduction de base

Étudier le système d'équations différentielles permet d'établir le « théorème du seuil », ce qui nous donne le taux de reproduction de base $\mathcal{R}_0$. C'est l'étude des équilibres du système qui permet d'introduire la notion de $\mathcal{R}_0$.

$\mathcal{R}_0$ mesure le nombre d'infections secondaires suite à l'introduction d'un individu infecté dans une population hôte entièrement constituée d'individus susceptibles [9]. $\mathcal{R}_0$ permet de répondre à la question importante que l'épidémiologiste souvent se pose lors de l'étude de la propagation d'une maladie : « Est-ce qu'il va y avoir une épidémie ou non ? ». Dans le modèle SIR, en supposant la transmission densité-dépendante, on va examiner le système d'equations différentielles (2.1) pour déduire la forme du $\mathcal{R}_0$. On commence par l'équation de I :

$$\frac{dI}{dt} = \beta SI - \gamma I = I(\beta S - \gamma) \tag{2.3}$$

Car une épidémie implique que le nombre de malades dans le compartiment I augmente, c'est-à-dire $dI/dt > 0$, donc $\beta S > \gamma$. On obtient la condition pour qu'il y ait une épidémie :

$$\beta S/\gamma > 1 \tag{2.4}$$

On appelle la quantité $\beta S/\gamma$ le nombre de reproduction que l'on note $\mathcal{R}$. Nous avons donc une épidémie dès que $\mathcal{R} > 1$. L'expression du $\mathcal{R}_0$ se déduit de celle du $\mathcal{R}$ dans laquelle le nombre de susceptibles S est remplacé par la taille de la population totale N, comme $\mathcal{R}_0$ se définit par rapport à une population entièrement susceptible :

$$\mathcal{R}_0 = \beta N/\gamma \tag{2.5}$$

Dans le cas de la transmission fréquence-dépendante, la formule du $\mathcal{R}_0$ est :

$$\mathcal{R}_0 = \frac{\beta}{\gamma} \tag{2.6}$$

Quand $R_0 > 1$, chaque individu infectieux contaminera plus d'individu susceptible en moyenne et la maladie se propagera à un nombre grandissant d'individus jusqu'à ce que le nombre de susceptibles soit trop faible et que $\mathcal{R} < 1$. Au contraire, quand $\mathcal{R}_0 < 1$, seuls quelques individus seront infectés avant que la propagation de la maladie ne s'arrête d'elle-même. Les équations (2.5, 2.6) montrent que le taux de reproduction de base $\mathcal{R}_0$ dépend du taux de transmission β. Comme β dépend non seulement des propriétés de la maladie mais aussi de la population hôte considérée, il est à noter que le $\mathcal{R}_0$ n'est en aucun cas une caractéristique d'une maladie donnée mais d'une maladie donnée dans une population hôte donnée. Ainsi, une même maladie pourra avoir deux $\mathcal{R}_0$ différents dans deux populations hôtes différentes et deux maladies différentes pourront avoir deux $\mathcal{R}_0$ différents dans une même population hôte [9].

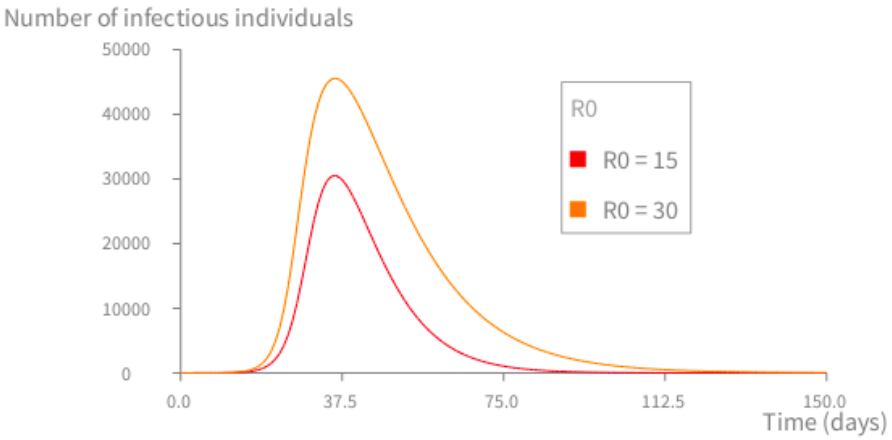


Figure 2.3 – La dynamique infectieuse du modèle de la rougeole quand $R\mathcal{R}_0$ est 15 et 30.

La figure 2.3 représente la dynamique infectieuse du modèle de la rougeole (voir le chapitre 3) en fonction de la valeur de $\mathcal{R}_0$. $\mathcal{R}_0 = 15$ signifie que chaque individu infectieux peut infecter en moyenne 15 susceptibles. Quand $\mathcal{R}_0$ augmente, le nombre d'individus infectieux augmente.

2.3 Les extensions du modèle SIR

2.3.1 La modification du cycle de transmission

Le modèle SIR est considéré comme le modèle de base de l'épidémiologie. En fonction des caractéristiques de la maladie à étudier, on va formuler d'autres modèles en ajou-

tant différents statuts infectieux. Par exemple, si la maladie conduit à une infection permanente, le modèle SI peut être utilisé. Dans le cas où la maladie ne confère pas une immunité, le modèle SIS est plus approprié. Dans ce modèle, on présume qu'après l'étape infectieuse, l'individu guérit et redevient aussitôt susceptible, c'est-à-dire qu'il peut à nouveau être contaminé. En cas d'immunité temporelle, on peut utiliser le modèle SIR en ajoutant un taux de perte d'immunité qui décrit la vitesse à la quelle les individus sortent de la classe *Rétabli* pour entrer à la classe *Susceptible* (le modèle SIRS). En outre, on peut supposer qu'après avoir été contaminé, l'individu susceptible entre dans une période d'incubation et devient infecté par l'ajout d'un autre compartiment E (*Exposé*).

Cela permet de distinguer deux étapes : devenir infecté et être infectieux. Lorsqu'un individu du compartiment S est exposé à la maladie, il ne devient pas capable de la transmettre immédiatement car le pathogène ne s'est pas encore suffisamment répliqué dans l'individu. Après une période de latence, il passe au compartiment I et peut infecter d'autres individus. La Figure 2.4 représente le schéma du modèle $SEIR$ avec démographie (c'est-à-dire tenant compte de la mortalité naturelle et de la natalité).

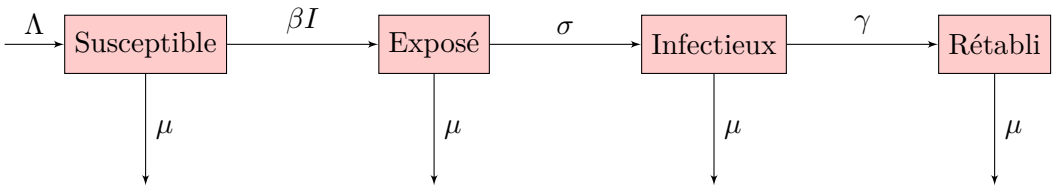


Figure 2.4 – Le modèle à base de compartiment SEIR

De nombreuses autres étapes peuvent être ajoutées (ou enlevées) à partir de la configuration de base d'un modèle SIR pour exprimer d'autres cycles de transmission voire des stratégies de contrôle comme la vaccination [3, 10, 18, 116], l'isolation ou la quarantaine [14, 47, 61] etc. Par exemple, le modèle $SEIR$ peut être étendu pour prendre en compte la vaccination à la naissance en disant que une proportion de susceptibles sont immunisés dès leur naissance. Le point commun de ces modèles est que la population se compose d'au moins deux compartiments *Susceptible* et *Infectieux*. Autrement dit, tous les modèles épidémiologiques doivent étendre le modèle SI.

2.3.2 Prise en considération de plusieurs aspects

Dans les modèles en compartiments se basant sur la structure SIR, la population est décomposée en fonction des statuts infectieux et est supposée être répartie de manière homogène dans l'espace étudié (toutes les paires d'individus ont la même probabilité d'établir un contact). Néanmoins, la modélisation des maladies infectieuses demande d'étudier non seulement les cycles de transmission mais aussi l'influence des caractéristiques (âge, genre, etc.) ou des comportements (voyager, aller à l'école/au travail, rester à la maison lors d'une épidémie etc.) des individus sur la propagation de l'infection. L'ajout de ces aspects conduit à la décomposition des compartiments *S-I-R* en plus petits groupes.

Par exemple, pour étudier des maladies infectieuses dont l'infection est transmise par

Table 2.1 – Aspects de l'épidémiologie que l'on retrouvent le plus souvent dans la littérature

Aspects	Caractéristiques
Saisonnalité	La transmission d'infection change au cours du temps (hétérogénéité de la transmission d'infection).
Multi-hôtes	Dans les maladies vectorielles ou zoonotiques, la transmission se fait entre plusieurs groupes d'espèces (hétérogénéité des hôtes).
Multi-souches	Une maladie peut être causée par plusieurs souches pathogènes ou plusieurs maladies sont étudiée en même temps. L'hétérogénéité des pathogènes se focalise sur la modélisation de l'interaction entre les souches d'infection concurrentes.
Structure d'âge	Étude des maladies en fonction de la structure d'âge des hôtes, souvent pour les maladies enfantines comme rougeole, varicelle (hétérogénéité des hôtes).
Structure de risque	Les maladies sexuellement transmissibles exigent d'étudier l'interaction entre de différents groupes à risque (groupe à haut/faible risque)(hétérogénéité des hôtes).
Structure Spatiale	Étude de la distribution de la population dans l'espace et le déplacement des individus entre les différentes zones graphiques pour modéliser la propagation spatiale de la maladie (hétérogénéité spatiale)
Stratégies de contrôle	Étude de l'efficacité de différents types de stratégies de contrôle (comme la vaccination, quarantaine/isolation etc.)

voie sexuelle, les compartiments *S-I-R* devront nécessairement être décomposés par sexe. Dans ce cas-là, chaque compartiment du modèle se compose des individus ayant le même sexe et le même statut infectieux. La population devient hétérogène à cause de l'introduction de l'appartenance sexuelle.

Le tableau 2.1 résume certains aspects de l'épidémiologie qui l'on retrouvent souvent dans la littérature [9, 69, 81]. Les hétérogénéités étant dû à l'âge, le sexe, les espèces, les souches virales, l'espace ou les différences de comportement entre les individus peuvent causer des variabilités chez les paramètres de la maladie tels que l'infectivité ou la susceptibilité. L'exemple suivant décrit la formalisation mathématique d'un modèle compartimental dans lequel l'on considère plusieurs de ces aspects simultanément.

Exemple : formalisation mathématique d'un modèle SIR multi-espèces Considérons un modèle multi-espèces dans lequel les individus de la population se décomposent en n espèces différentes [2]. On suppose que le cycle de transmission de la maladie étudiée se représente par un modèle SIR et que la transmission se fait à la fois au sein d'une même espèce et entre espèces différentes. La considération de l'aspect multi-espèces dans le modèle conduit à la division des compartiments *S-I-R* en plus petits groupes S_i, I_i, R_i, qui représentent respectivement les individus susceptibles, infectieux et rétablis de l'espèce i.

La formalisation mathématique de ce modèle est la suivante :

$$\begin{cases} \frac{dS_i}{dt} &= -\sum_{j=1}^{n} \frac{\beta_{ij}}{N_j} I_j S_i \\ \frac{dI_i}{dt} &= \sum_{j=1}^{n} \frac{\beta_{ij}}{N_j} I_j S_i - \gamma_i I_i \\ \frac{dR_i}{dt} &= \gamma_i I_i \end{cases} \tag{2.7}$$

Les paramètres β, γ, N du modèle SIR original (voir la section 2.2.1) deviennent hétérogènes, $\beta_{ij}, \gamma_i, N_j$, dans ce modèle. γ_i dénote le taux de guérison qui diffère pour chaque espèce, γ devient donc un vecteur. β_{ii} est le taux de transmission au sein de l'espèce i et β_{ij} représente celui entre espèces i et j. Le paramètre β devient une matrice à deux dimensions qui capture la transmission d'infection entre espèces différentes. En comparant avec la formule 2.2, dans ce modèle, $\lambda_i = \sum_{j=1}^{n} \frac{\beta_{ij}}{N_j} I_j$ prend en compte l'interaction entre les groupes d'espèces ($\beta_{ij} I_j / N_j$ compte l'infection causée par le contact entre un susceptible de la classe i et les infectieux de la classe j).

De même façon, si on veut considérer un aspect spatial en plus de l'aspect multi-espèces, la population doit être décomposée en groupes plus petits, tels que S_{sp}, I_{sp}, R_{sp} où s dénote une espèce et p dénote une zone géographique (telles que pays, villes, villages...) [13, 14].

L'interaction entre les espèces différentes d'une même zone géographiques et entre les zones doit aussi être considérée. Cela peut par exemple être représenté dans la formule de λ de la façon suivante :

$$\lambda_{sp} = \sum_{j}^{zones} \sum_{i}^{espèces} \beta_{isj} I_{ij} / N_{ij}$$

où chaque terme $\beta_{isj} I_{ij} / N_{ij}$ décrit l'infection causée par le contact entre un susceptible de l'espèce s dans la zone p et les infectieux de l'espèce i dans la zone j).

On peut faire varier la formule de λ pour capturer d'autres types d'interaction qui sont dû à la présence de plusieurs aspects dans le modèle. Nous reviendrons à ce modèle dans le chapitre 4.

En résumé, on retrouve les points communs qui suivent dans les modèles où plusieurs aspects épidémiologiques sont introduits :

- Il y a plusieurs façon de diviser une population (non seulement par le biais des statuts infectieux, mais aussi en utilisant d'autres attributs de l'individu).
- Un modèle peut introduire plusieurs paramètres d'hétérogénéité comme le taux de transmission, les périodes infectieuses, le taux de natalité, de mortalité, etc.
- La formule de la force d'infection du modèle peut varier en fonction des mécanismes de transmission causés par chaque aspect. Le plus, on introduit d'aspects, la plus complexe il devient difficile d'exprimer la force d'infection.

2.4 Simulation des modèles épidémiologiques

Comme on a déjà dit précédemment, un modèle est une représentation d'un système réel. Le modélisateur a la possibilité pour répondre aux questions qu'il se pose d'utiliser

des simulations (Figure 2.1). Il faut noter que le modèle mathématique peut être analysé également sans utiliser des simulations (par exemple pour calculer la valeur de R_0 ou déterminer des conditions de stabilité) mais nous n'aborderons pas cet aspect dans le cadre de la thèse.

Une simulation est définie comme une expérimentation effectuée sur un modèle. Les modélisateurs en épidémiologie ou en écologie généralement considère trois paradigmes de simulations : simulation déterministe, simulation stochastique et enfin simulation multi-agents.

Le modèle déterministe est généralement connu dans la littérature sous le vocable "*population-level model*[2]" tandis que le modèle stochastique est appelé "*individual-based model*[3]" ou individu-centré. Il faut distinguer ces modèles individus-centrés de ceux utilisant des agents "*agent-based model*[4], où la sémantique de chaque agent est défini par un algorithme et qui sont alors très difficilement traitable de manière analytique[23].

2.4.1 Simulations déterministes

Dans un cadre déterministe, les équations différentielles constituent l'outil mathématique idéal pour décrire les modèles en compartiments. La recherche analytique de solutions peut être alors plus ou moins complexe selon les équations que l'on choisit pour décrire les modalités de passage d'un compartiment à l'autre. Dans le cadre de cette thèse, nous nous focalisons exclusivement sur les modèles à compartiments de l'épidémiologie formulés par des équations différentielles ordinaires (EDO). La raison de ce choix est qu'elles représentent la majorité des modèles de la littérature.

Certains modèles épidémiologiques peuvent être formulés également par des équations différentielles partielles (EDP). Par exemple pour représenter des modèles structurés par l'âge, où l'on veut tenir compte du fait que l'âge des individus augmente au cours du temps. Cependant, les méthodes de résolution d'EDP sont difficiles à implémenter et nécessitent une discrétisation. Dans le cas de modèles structurés par l'âge, on résoud ce problème en regroupant les individus en classes d'âge, et on utilise des EDO pour représenter leurs passage d'une classe d'âge à une autre.

Lorsque la résolution des équations s'avère trop difficile (voire impossible) de manière analytique, on peut faire appel à des techniques d'intégration numérique. Ces techniques (dont les méthodes d'Euler ou Runge-Kutta sont les plus connues) permettent, par une discrétisation des calculs, d'obtenir des solutions approchées [56]. On choisit comme condition initiale une valeur pour chaque variable, puis on reproduit les changements d'état de ces variables en calculant leurs valeurs par itérations successives. L'algorithme qui décrit la façon de calculer les valeurs successives est un modèle de simulation qui représente le modèle analytique et donne accès à une approximation des solutions de celui-ci. Les résultats sont constants (i.e. déterministes) pour toutes simulations avec certaines conditions initiales. Ce modèle reflète donc la dynamique sans hasard du système.

2. PLM
3. IBM
4. ABM

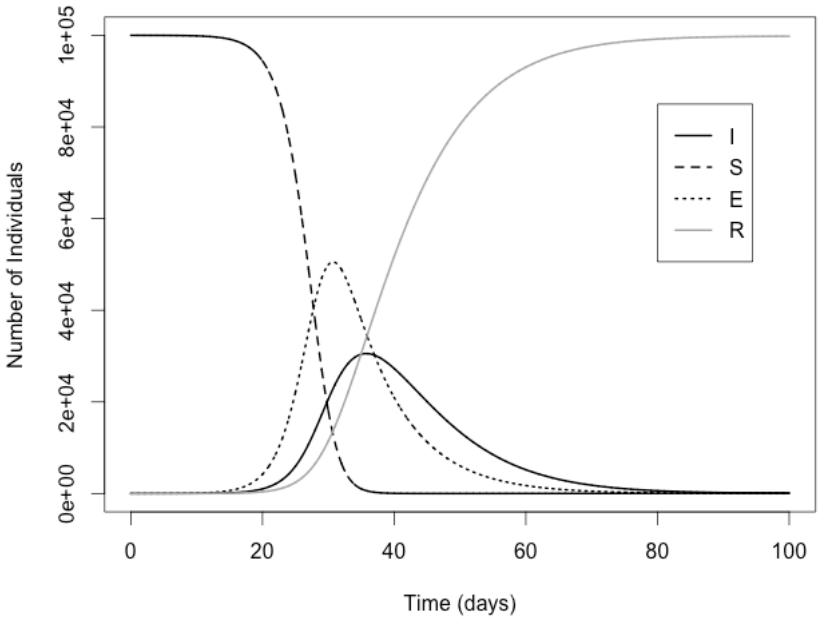


Figure 2.5 – L'évolution au cours du temps des compartiments du modèle SEIR déterministe. Le système d'équations est résolu par la méthode Runge-Kutta. $S = 99999$, $E = 0$, $I = 1$, $R = 0$, $\beta = 0.0000214$, $1/\gamma = 7$ jours, $1/\sigma = 8$ jours, $\mu = 1/(78 * 365)$ jour^{-1}, $N = 100000$. Courbes générées par R [26]

Le modèle déterministe est souvent le premier outil que va utiliser l'épidémiologiste pour étudier une nouvelle épidémie afin de déterminer la courbe globale d'infection et de trouver les meilleures valeurs des paramètres du modèle qui minimisent l'écart entre la courbe théorique et les valeurs observées. La figure 2.5 représente les résultats déterministes du modèle $SEIR$ obtenus en utilisant la méthode *Runge-Kutta* pour résoudre le système d'équations différentielles :

$$\begin{cases} \frac{dS}{dt} &= \mu N - \beta SI - \mu S \\ \frac{dE}{dt} &= \beta SI - \sigma E - \mu E \\ \frac{dI}{dt} &= \sigma E - \gamma I - \mu I \\ \frac{dR}{dt} &= \gamma I - \mu R \end{cases} \tag{2.8}$$

Le modèle déterministe est également utile pour évaluer l'efficacité des stratégies de contrôle sur le long terme. La figure 2.6 représente l'impact de la vaccination en comparant deux valeurs différentes du taux de vaccination. Dans le cas où la population est vaccinée, on voit que le nombre d'infectés est réduit.

2.4.2 Simulations stochastiques (ou individus centrés)

Tandis que les modèles déterministes permettent de calculer l'équilibre endémique (par l'étude du taux de reproduction de base $\mathcal{R}_0$) ainsi que de prévoir les conditions de la propagation de l'épidémie, le passage à des simulations stochastiques est reconnu comme plus réaliste pour comprendre et prédire la dynamique des maladies infectieuses. Les

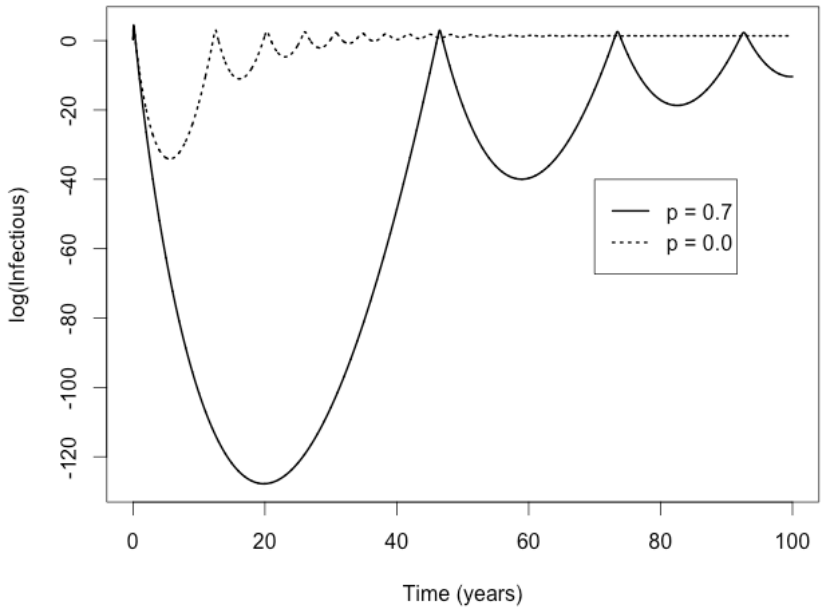


Figure 2.6 – La dynamique de l'infection du modèle SEIR avec vaccination à la naissance. Le modèle est étudié pendant 100 ans : $S = 99999, E = 0, I = 1, R = 0, \beta = 0.00782, 1/\gamma = 7/365ans, 1/\sigma = 8/365ans, \mu = 1/78ans^{-1}, N = 100000$. La courbe noire affiche la dynamique déterministe de l'infection avec la population de 70% vaccinée. La courbe grise affiche le résultat de la population non vaccinée. Courbes générées par R. L'axe des ordonnées utilise une échelle *log* pour la lisibilité [26].

modèles stochastiques sont une façon naturelle de modéliser l'évolution d'une épidémie : chaque individu à une certaine probabilité de la transmettre et de guérir. Une partie importante de l'étude de ces problèmes stochastiques va être de déterminer *si, quand la taille de la population augmente, ils convergent vers un problème déterministe* [89]. Un moyen pour rendre un modèle stochastique est d'ajouter du bruit dans les équations différentielles du modèle déterministe. Un autre moyen plus connu se base sur la stochasticité démographique qui est définie comme des fluctuations se produisant par les différences aléatoires entre les individus d'une même population. À la différence d'un modèle déterministe, ce type de modèle utilisera des nombre de susceptibles, d'infectieux ou de rétablis qui seront des entiers et chaque simulation sera différente de la précédente.

De manière plus formelle, un modèle stochastique peut être formulé comme un processus de Markov, où la probabilité faire une transition vers un nouvel état ne dépend que de l'état courant du système. Il peut être formulé de deux façons diffèrentes : processus à temps continu (CTMC[5]) ou processus à temps discret (DTMC[6]), selon les choix de modélisation. Les modèles qui sont définis de façon déterministe sous forme d'équations différentielles peuvent être interprétées à partir du point de vue stochastique en fournissant une certaine hypothèse sur la distribution de probabilité En général, les taux de transfert linéaires dans les équations différentielles correspondent aux temps d'attente avec des distributions exponentielles négatives [62]. Par exemple, le taux de transfert γI correspond à $P(t) = e^{-\gamma t}$ qui représente la probabilité que les individus restent dans la classe I après t unités de temps dès qu'ils se sont entrés dans cette classe ($1/\gamma$ est le temps moyen d'attente). La probabilité pour que les individus quittent la

5. Continuous Time Markov Chain
6. Discrete Time Markov Chain

classe I pour entrer dans la classe R est : $P(t) = 1 - e^{-\gamma t}$.

Le fait de choisir le processus de Markov permet de pouvoir étudier les modèles stochastiques de façon analytique et/ou numérique (par des simulations). Pour simuler les chaines de Markov, on utilise des algorithmes de simulation stochastiques parmi lesquels la méthode directe de Gillespie est la plus utilisée dans le domaine de la modélisation épidémiologique [54, 69]. L'algorithme direct de Gillespie est décrit comme suit :

Algorithm 1: L'algorithme direct de Gillespie

entrée: Un ensemble des événements $\mathcal{E} = \{E_1, E_2, \ldots, E_n\}$

$t \leftarrow T_{min}, e \leftarrow 0$

répéter

  Calculer les taux auxquels les événements se produisent : $R_1, R_2, \ldots, R_n$.

  Calculer le taux total de tous les événements : $R_{total=\sum_{m=1}^{n} R_m}$.

  Générer deux nombres aléatoires : $r_1, r_2 \in [0..1]$.

  Calculer l'intervalle du temps pour passer à un nouvel événement : $\delta t = \frac{-1}{R_{total}} log(r_1)$.

  $P = r_2 * R_{total}$.

  si $\sum_{m=1}^{p-1} R_m < P \leq \sum_{m=1}^{p} R_m$ **alors**

    $e \leftarrow p$

  finsi

  $t \leftarrow t + \delta t$.

  Exécuter l'événement E_e.

jusqu'à $t > T_{max}$

Les termes linéaires dans les équations différentielles du modèle déterministe (par exemple, βSI, γI dans le cas du modèle SIR) correspondent aux taux des événements dans le modèle stochastique. L'algorithme direct de Gillespie suppose que sur du temps continu (approximé) les évènements arrivent les uns après les autres. À chaque pas de temps, on considère que l'évènement ayant le taux le plus fort est aussi celui qui a la plus forte probabilité d'arriver.

Les modèles sont généralement exécutés pour un grand nombre de fois afin d'estimer le résultat moyen et la variation autour de cette moyenne. Ce résultat moyen généré par les nombreuses simulations fournit des prédictions comparables à celles du modèle déterministe particulièrement lorsque la population est grande. Par exemple, la figure 2.7 donne 100 simulations stochastiques de l'infection du modèle $SEIR$ en comparant avec le résultat déterministe généré par la méthode Runge-Kutta avec même configurations (comme indiqué dans la figure 2.5). Il faut noter que pour le modèle $SEIR$, nous avons 8 événements avec les taux qui suivent :

- $S \rightarrow S + 1 : \mu N$
- $S \rightarrow S - 1 : \mu S$
- $S \rightarrow S - 1, E \rightarrow E + 1 : \beta SI$
- $E \rightarrow E - 1, I \rightarrow I + 1 : \sigma E$
- $E \rightarrow E - 1 : \mu E$
- $I \rightarrow I - 1, R \rightarrow R + 1 : \gamma I$
- $I \rightarrow I - 1 : \mu I$
- $R \rightarrow R - 1 : \mu R$

À chaque itération, le temps et le nombre d'individus dans chaque classe S, E, I, R va être mis à jour en fonction de l'événement choisi.

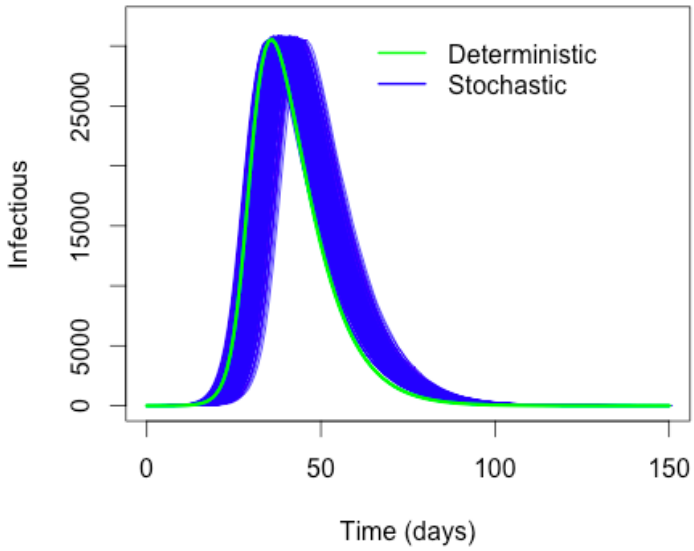


Figure 2.7 – 200 simulations stochastiques de l'infection du modèle $SEIR$ générées par la méthode directe de Gillespie en utilisant le langage R

Dans le cas de la méthode directe de Gillespie, le temps de calcul nécessaire pour simuler un scénario particulier d'une maladie augmente linéairement avec la taille de la population. En effet, dans une large population, l'infection et la guérison devraient être beaucoup plus fréquentes que dans une petite population. Cette augmentation des taux conduit à une diminution du pas de temps moyen δt et donc à une augmentation du nombre d'itérations nécessaires pour simuler le modèle dans une période spécifiée. Il existe des améliorations de cet algorithme appelé méthode τ-*leap* [69] qui est nettement plus rapide avec un temps de simulation car moins dépendant de la taille de la population. On suppose que pendant un temps suffisamment petit, le nombre de fois que chaque événement se passe, suit la loi de Poisson. Cependant cette méthode fournit des approximations moins précises.

2.4.3 Simulations multi-agents

Récemment, les modélisateurs en épidémiologie se sont intéressé à des modèles avec un niveau de détail plus important en utilisant des simulations multi-agents[27, 42, 48, 55, 106]. Ces détails en terme d'attributs ou de comportements peuvent être facilement ajoutés à un groupe ou même à un individus. Le principal inconvénient de cette approche est qu'elle ne peut produire que des résultats numériques et la validité de ce type de modèle reste difficile à prouver.

Il y a parfois une confusion dans la littérature [57, 69, 106, 107, 109] entre ce type de modèle et les modèles que l'on appellent individus-centrés [7] Afin de faire clairement la

7. Les modèles multi-agents sont parfois également mentionnés comme individus-centrés, mais cela ne correspond pas aux modèles stochastiques (appelés également individus-centrés) dont nous avons parlés dans la section précédente.

distinction avec le modèle stochastique abordé dans la section précédente nous utilisons seulement le terme **modèle multi-agents** pour désigner ce type de modèle.

Un modèle multi-agents en épidémiologie peut être formulé de façon simple comme suit. En fonction de l'état courant, chaque individu est assigné à certains comportements, par exemple, contacter d'autres individus pour transmettre l'infection, entrer dans une période d'infection mais pas encore infectieux (capacité de transmettre l'infection à d'autres), être guéri, la naissance d'un nouveau individu, la morte naturelle. D'autres comportements peuvent être considérés qui augmentent le nombre d'attributs des individus. En supposant également que le modèle représente un processus de Markov à temps continu, P_i dénote la probabilité que l'état d'un individu reste en i après une période de temps δt. On peut alors poser comme hypothèse que le temps passé dans chacun des états suit une distribution exponentielle (cela résulte de la propriété de Markov). Pour accélérer la vitesse quand on fait au niveau des individus, on considère un temps discret où plusieurs évènements vont pouvoir arriver à un même temps donné - un pas de temps choisi suffisamment petit. $P_i = e^{-\delta t \times rate}$ où $rate$ est le taux auquel l'individu va passer à l'état suivant. La probabilité pour que cet individu change son état est :

$$P = 1 - e^{\delta t \times rate} \tag{2.9}$$

δt est une intervalle de temps (ou un pas de temps). Dans la modélisation des maladies infectieuses, il faut choisir un pas de temps suffisamment petit de telle sorte qu'au plus d'un changement d'état se produit pendant cet intervalle de temps. Par exemple, car la population humaine se trouve dans un cycle quotidien, le choix d'un pas de temps d'un jour ou d'un demi-jour est peut-être suffisamment petit. Dans ce type de modèle, les taux des transfert sont interprétés au niveau des individus (tandis que dans la méthode directe de Gillespie, on considère les taux des événements au niveau de la population, correspondant aux termes linéaires dans les EDO de l'approche déterministe). À chaque itération, tous les individus sont étudiés ; en fonction de la probabilité calculée, leur état peut être mis à jour ou non ($P > rand()$). C'est une simple formule du modèle multi-agents qui a été introduite dans plusieurs travaux en épidémiologie [57, 69, 106, 107, 109]. Dans un modèle multi-agents plus complexe, un individu a généralement beaucoup plus d'attributs et donc possède beaucoup plus de comportements, surtout quand les facteurs sociaux, comme classes sociales, ethnies, activités quotidiennes (aller à l'école, au travail, voyager, etc.), interactions sociales etc., sont considérés lors de l'étude de la propagation des maladies [42].

Modèle stochastique et modèle multi-agents reposent donc sur les mêmes concepts (i.e une chaîne de Markov à temps continu). Il est donc possible d'étudier ce modèle de façon analytique et/ou numérique. Cependant, dans un cas plus complexe, le comportement des agents est alors une fonction des attributes d'individu, des caractéristiques de l'environnement etc., Les modèles multi-agents sont formulés en général par des algorithmes qui implémentent des comportments, des interactions entre agents et avec l'environnement. Ces modèles ne pas généralement analysable de façon analytique et sont hors du champ de cette thèse.

On peut se poser une question sur la différence entre cette approche et la méthode directe de Gillespie qui a été introduite dans la section précédente. Ces deux méthodes se basent sur le même principe (processus de Markov à temps continu) cependant la dernière interprète les modèles au niveau des individus. Chaque individu a sa propre valeur d'attribut et sa propre probabilité pour changer son état. On peut donc capturer

des stochasticités au niveau plus détaillé.

2.5 Outils développés pour la modélisation informatique de l'épidémiologie

Dans le but de faciliter la construction et la simulation des modèles épidémiologique, des outils informatiques ont été développé. Ces outils doivent permettre aux modélisateurs de pouvoir spécifier leurs modèles mathématiques et puis de lancer les simulations pour étudier et analyser la propagation de la transmission des maladies infectieuses. Nous allons faire ici un panorama rapide des outils pertinents dédié à l'épidémiologie dans la section suivante. Ce panorama permettra d'identifier les enjeux important liés à ces outils et une liste de critères pour l'outil que nous souhaitons développer par la suite.

2.5.1 Langages de modélisation mathématique

Les épidémiologistes dans leur grande majorité à l'occasion de leurs travaux de recherches, utilisent des langages de modélisation mathématiques comme : MATLAB [8], Modelica [9], R [10] etc.

D'une part, ces langages permettent de facilement spécifier des modèles sous forme mathématique comme ceux qui utilisent des équations différentielles, puis ensuite de les résoudre analytiquement. D'autre part, ces langages disposent pour la plupart d'une boucle d'évaluation interactive [11] permettant à des personnes qui ne sont pas des informaticiens de développer rapidement leurs modèles et de les exploiter. Cette approche interactive favorise plutôt une écriture des programmes sous forme de scripts. Enfin l'intégration de ces langages avec des outils statistiques et de visualisation des résultats les rend plus facile d'accès aux épidémiologistes ou biologistes.

Ces langages permettent facilement de faire des simulations déterministes mais les simulations stochastiques ou multi-agents restent plus difficiles car nécessitant une plus grand effort de programmation surtout dans le cas de modèles complexes couplants plusieurs aspects (par exemple modèles spatial ou bien nécessitant des stratégies de contrôles). Dans ce cas-là, pour faciliter la construction des modèles, les chercheurs ont développé des bibliothèques dédiées à résoudre certains problèmes spécifiques de la modélisation en épidémiologie. Parmi ces outils on peut citer : *Epipy* [12] - un outil Python de visualisation de données épidémiologique [105, 104], *GillespieSSA* [13] - un package R pour générer des simulations stochastiques utilisant les algorithmes de Gillespie [93, 94] et Simecol [14] [92] - un framework orienté objet pour la modélisation écologique en R.

La figure 2.8 représente un exemple d'utilisation de la bibliothèque *GillespieSSA* (ici

8. `http://www.mathworks.com/products/matlab/`
9. `https://www.modelica.org/`
10. `https://www.r-project.org/`
11. REPL (Read-Eval print loop)
12. `http://cmrivers.github.io/epipy`
13. `http://CRAN.R-project.org/package=GillespieSSA`
14. `http://www.jstatsoft.org/v22/i09/`

on a utilisé la méthode directe de Gillespie présentée dans la section 2.4.2).

```
1 parms <- c(beta=0.001, gamma=0.1)
2 x0  <- c(S=499, I=1, R=0)
3 a   <- c("beta*S*I", "gamma*I")
4 nu  <- matrix(c(-1, 0, +1, -1, 0, +1),nrow=3, byrow=TRUE)
5 out <- ssa(x0, a, nu, parms, tf=100, simName="SIR model")
6 ssa.plot(out)
```

Figure 2.8 – Script R de simulation stochastique du modèle SIR utilisant le package *GillespieSSA*

Ces bibliothèques spécifiques au domaine permettent de réduire l'activité de programmation mais en général la sémantique du domaine (concepts et relation entre eux) est cachée par l'implémentation. Par exemple, dans le script 2.8, la sémantique des compartiments S, I, R et les transitions qui représentent le cycle de transmission sont cachés par la signature de la fonction `ssa` du package *GillespieSSA*. De plus, ces bibliothèques ne favorisent pas de bonnes pratiques de programmation comme celles liés à la réutilisation.

2.5.2 Outils de modélisation intégrés

Dans cette section, nous établissons une liste des logiciels intégrés (intégré dans le sens l'ensemble des fonctionnalités de modélisation, visualisation et d'analyse forme un tout cohérent prêt à l'emploi) de modélisation permettant l'intermédiaire d'une interface utilisateur graphique de construire ses modèles. On va s'attarder plus précisément sur trois outils qui nous semblent importants : STEM, GLEaMviz et FRED.

STEM

STEM - *Spatiotemporal Epidemiological Modeler* est un logiciel libre destiné à la modélisation de la propagation temporelle/spatiale des maladies infectieuses pour les humains ou animaux [46, 50]. STEM décompose une modélisation en trois niveaux : graphes, modèles et scénarios.

Un graphe consiste en un ensemble de nœuds qui représentent des entités spatiales avec un forme et une position spatiale (coordonnées géographiques). Les arêtes entre nœuds représentent soit des processus de transports ou encore des dépendances spatiales ou hiérarchiques (par exemple, un sous-graphe dans un nœud).

Des étiquettes peuvent être associées à un nœud ou une arête, qui portent des informations pertinentes pour le scénario à modéliser (par exemple, la taille d'une population, la taille d'une région). L'état courant d'une population (le nombre d'individus de chaque compartiment) est inclut aussi dans l'étiquette du nœud représentant cette population. Plusieurs étiquettes peuvent être associées à un nœud, qui donne des informations sur d'autres épidémies (qui se passe en même temps) ou d'autres populations (car un nœud peut représenter un sous-graphe dans la structure hiérarchique).

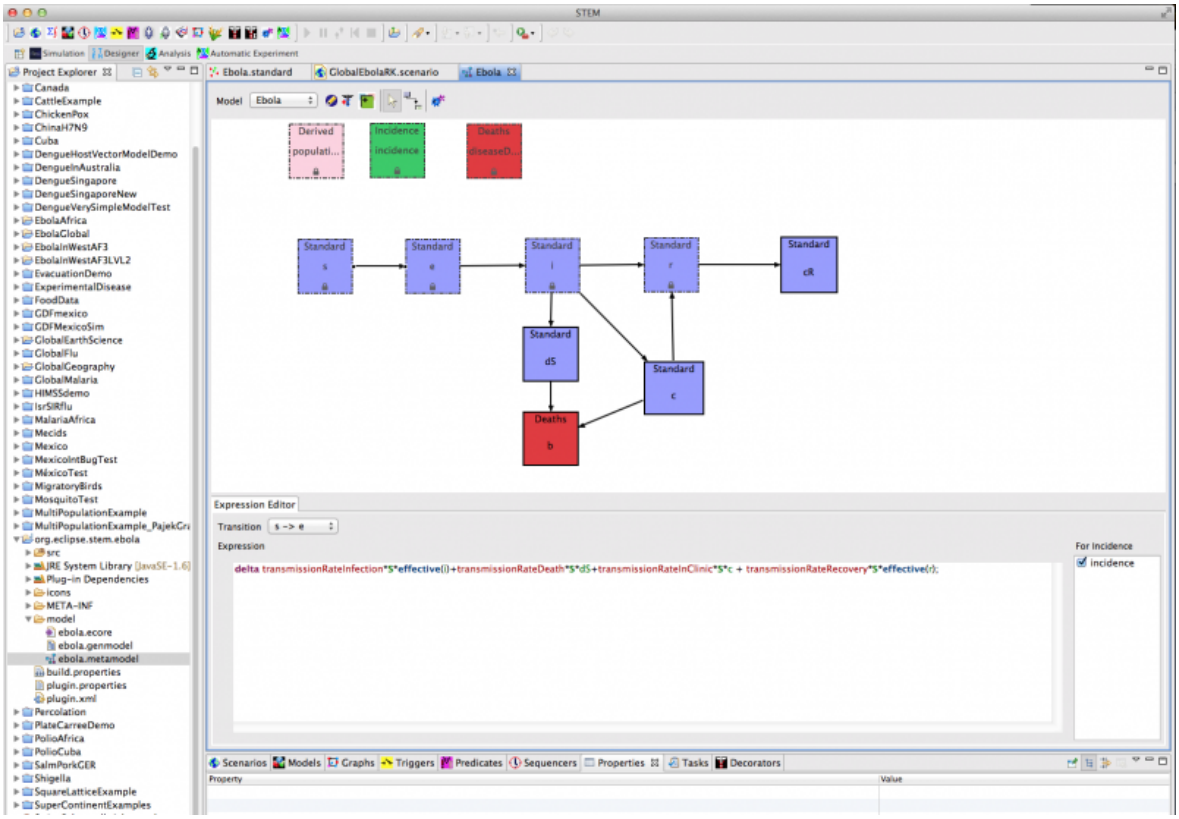


Figure 2.9 – L'interface graphique d'utilisateur de STEM

Un modèle STEM est constitué d'au moins un graphe et autant de combinations de décorateurs que possible. Il y a deux types de décorateurs : décorateur de population qui permet de spécifier des informations d'une population (la taille initiale de chaque compartiment) et décorateur d'épidémie qui donne la structure à base des compartiments de la maladie étudiée. Des décorateurs vont *décorer* des graphes du modèle, par exemple, en assignant les valeurs aux étiquettes existantes des nœuds ou en ajoutant de nouvelles étiquettes. Des modèles différents peuvent être organisés de façon hiérarchique à l'intérieur d'un autre modèle.

Un scénario va être construit à partir d'au moins un modèle, une série temporelle et un résolveur d'équations différentielles. D'autre components sont également disponibles dans un scénario pour analyser et visualiser les résultats.

STEM fournit deux structures compartimentales de base pour modéliser des maladies infectieuses : SIR et $SEIR$, qui peuvent être modifiées par l'utilisateur pour ajouter des statuts supplémentaires. Les modèles sont ensuite simulés de façon déterministe ou stochastique. Lors de la simulation, à chaque itération, chaque nœud est visité tour à tour et les nouvelles valeurs pour chaque étiquette associée à ce nœud sont calculées et enregistrées. L'état des étiquettes est mis à jour et le temps de simulation est augmenté d'un pas de temps dès que tous les nœuds sont visités. Ce processus est itératif jusqu'à ce qu'il soit arrêté par l'utilisateur.

L'interaction avec STEM passe principalement par une interface graphique qui permet aux utilisateurs de pouvoir construire leur modèles visuellement (Figure 2.9). La structure de graphe hiérarchique de STEM permet de capturer l'hétérogénéité de la population (un nœud du graphe peut s'associer à plusieurs populations en même temps,

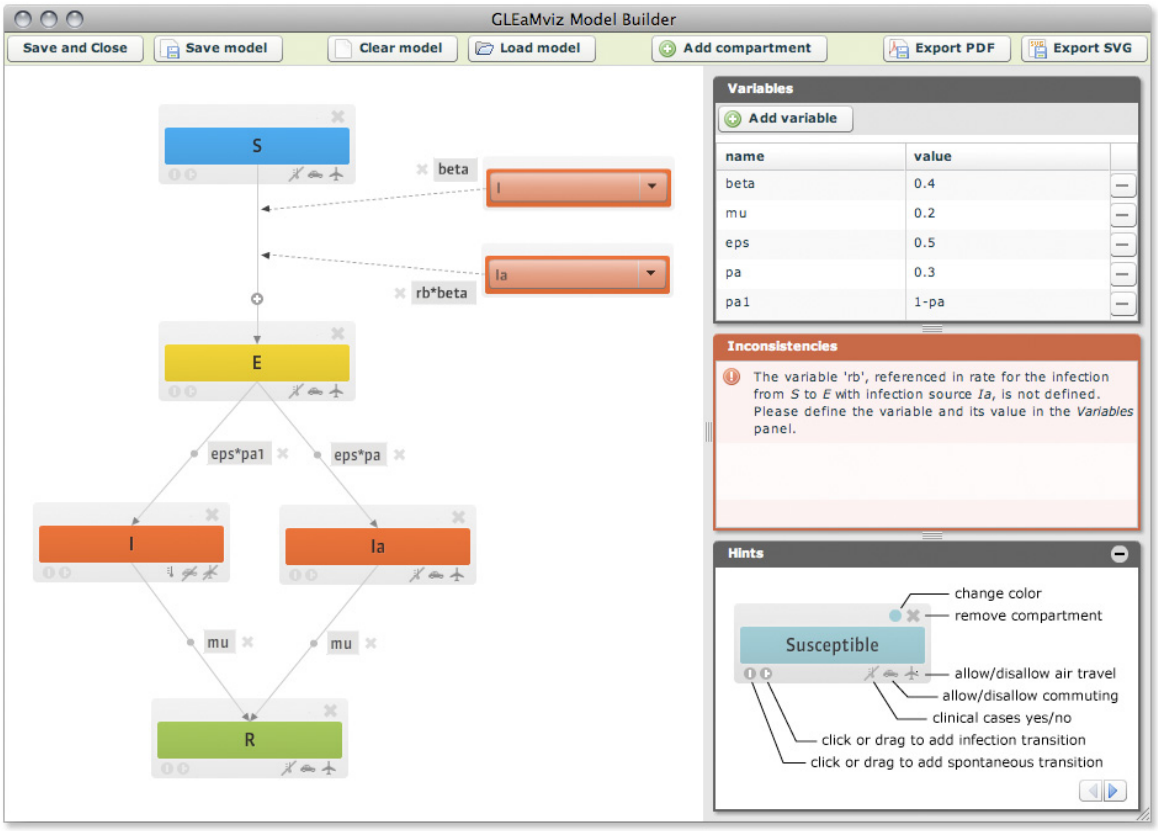


Figure 2.10 – Interface graphique de GLEAMviz-client

par exemple : population humaine, population animale), ce qui donne la capacité de pouvoir modéliser des scénarios complexes comme : la transmission par vecteurs (paludisme, dengue, grippe aviaire) ou l'immunité croisée (maladies multiples).

GLEaMviz

GLEaMviz [15] est bien connue dans la communauté car il permet de faire des simulations globales de types pandémie. Il s'agit d'une application de type client-serveur dédiée à la modélisation spatiale/temporelle des maladies infectieuses[16, 118]. L'architecture de cet outil se compose de trois components : une application client, un middleware proxy et un moteur de simulation. Un modèle GLEaM utilise une approche de type méta-population dans laquelle le monde est divisé en des régions géographiques dont chacune est une sous-population d'individus. Les connections entre sous-populations représentent les flux des individus qui passent d'une région à une autre. Un modèle est formé de trois couches : un premier niveau contenant des donnés sur la population, un deuxième qui ajoute des informations de mobilités sur les personnes entre régions géographiques et enfin le troisième pour exprimer la structure des compartiments de la maladie étudiée.

L'application de client GLEaM permet aux utilisateurs de pouvoir spécifier un modèle compartimental qui représente la maladie par un graphe (Figure 2.10). Le lien entre deux compartiments représente une transition. Le taux de transition est indiqué sur ce

15. Global Epidemic and Mobility visualization

lien. Une fois que la structure compartimentale du modèle est construite, l'utilisateur va spécifier la structure de la population et les données de la mobilité, puis configurer la simulation et les envoie au serveur pour lui demander d'exécuter. Le serveur va traiter la demande d'un client et lance une ou plusieurs simulations stochastiques. L'épidémie se propage entre les sous-populations en raison du voyage des individus infectieux d'une région infectée à une région non-infectée et du voyage des individus susceptibles d'une région (infecté ou non-infecté) à une région infectée. La dynamique de l'infection va être modélisée pour chaque sous-population en tenant compte de la propagation de la maladie à l'intérieur de cette sous-population et la propagation spatiale entre les sous-populations.

GLEaMviz ne permet que de modéliser des maladies infectieuses dont la transmission d'infection est directe, c'est-à-dire sans vecteurs (comme la dengue, le paludisme etc.). Dans le modèle GLEaM, chaque sous-population est supposée homogène. Il n'est donc pas possible de tenir compte de l'hétérogénéité de la population due à des attributs comme le sexe, l'âge, etc.

FRED

Alors que les deux précédents outils, se focalise sur la modélisation déterministe et/ou stochastique, FRED [16] est une approche de simulation multi-agents. Ce logiciel libre vise à étudier la propagation de la grippe et à évaluer l'efficacité des stratégies de contrôle comme la vaccination ou les politiques de fermeture d'école [55].

Dans un modèle FRED, chaque individu agit comme un agent autonome qui possède des attributs démographiques (sexe, âge, école, profession etc.), statuts clinique (susceptibilité, niveau symptomatique, infectivité, immunité, etc.), des activités sociales (activités domestiques, activités lié à l'écoles ou à son lieu de travail etc.) et quelques autres propriétés liés à la santé publique comme la probabilité d'être vacciné. L'hétérogénéité de la population est donc capturée à un niveau de détail très important.

FRED effectue une simulation à temps discret avec un pas de temps d'un jour. À chaque itération, chaque agent interagit avec d'autres agents qui partagent les mêmes lieux d'activités, par exemple, les enfants interagissent avec leur camarades à l'école. Si un agent infectieux a un contact avec un agent susceptible, la maladie est possiblement transmise. Chaque événement de transmission de l'infection est enregistré, ce qui permet d'évaluer l'efficacité de plusieurs mesures de contrôle possibles et leur impact sur les sous-populations spécifiques. Les agents peuvent dynamiquement modifier leurs activités quotidiennes, par exemple, en voyageant ou en décidant de rester à la maison lorsqu'ils sont malades.

FRED suppose que toutes les interactions spécifiques concernant la propagation de la maladie chez les agents (infectivité, immunité, etc.) se produisent dans des lieux spécifiques comme la maison, le lieu de travail, à l'école, chez les voisins. Chaque type de lieu représente un environnement distinct pour la propagation de l'infection. En fonction de ses activités personnelles, chaque agent possède un horaire quotidien qui indique les lieux qu'il visite régulièrement. Dans chaque lieu, un agent a une probabilité de contact différente. Le processus pour contracter une ou des maladies (FRED peut modéliser

16. Framework for Reconstructing Epidemic Dynamics

plusieurs maladies en même temps) va être défini pour chaque agent, normalement selon le schéma S-E-I-R. D'autres schémas sont disponibles pour l'utilisateur comme S-E-I-R-S, S-I-S, S-E-I-S ou S-I-R-S en modifiant la valeur des paramètres. La spécification d'un modèle dans FRED décrit la maladie, la population, les agents (individus), les données des activités et les configurations des paramètres. L'outil ne fournit pas une interface graphique d'utilisateur comme STEM or GLEAMviz mais permet de générer des scripts pour visualiser le résultat de simulation dans un autre outil de visualisation.

FRED fournit un vrai modèle à base des agents pour étudier l'impact de la structure de population et la distribution spatiale (sur certains lieux) sur la propagation de l'infection entre les regions des États-Unis. Un point très fort du modèle mutli-agents est qu'on peut capturer l'hétérogénéité de la population à un niveau de détail très fin (c'est-a-dire qu'un individu peut avoir de nombreux attributs). Cependant, comme nous avons abordé dans la section 2.4, l'effet stochastique influence considérablement la validité des résultats surtout dans le cas où la taille de la population est petite. Cela veut dire que les résultats des simulations (du même modèle) peuvent être très variés. Comprendre la nature de ces variations est toujours l'objet de recherche importante.

Il existe aussi d'autres outils dans le domaine comme :

InfluSim [40, 41], un outil qui fournit aussi une interface pour simuler la propagation de la grippe dans une population homogène utilisant l'approche déterministe. La structure d'âge peut être prise en compte dans la population mais la structure spatiale ou d'autres formes d'hétérogénéité et de stochasticité ne le sont pas.

FluTE [27] - un outil pour la modélisation stochastique de la grippe aux États-Unis, dans lequel la population est structurée à un niveau très détaillé : domicile, voisins, lieux de communauté. FluTe fournit un fichier de configuration textuel aux utilisateurs pour spécifier leur modèle en déclarer des paramètres, le taux de reproduction de base et quelques moyens d'intervention. Le résultat est également donné sous forme d'un fichier textuel qui demande d'être analysé et visualisé par d'autres outils.

GEM[17] [43] - un outil permet de modéliser de façon stochastique la propagation de la grippe au niveau globale en utilisant l'approche méta-population dans laquelle la population est décomposée en sous-populations homogènes. Les connections entre sous-populations représente la mobilité des individus, ce qui est spécifié par un réseau aérien de 155 régions métropolitaines dans le monde. Le modèle compartimental se limite à S-E-I-R. L'utilisateur va configurer les paramètres et tester les stratégies d'intervention prédéfinies.

EpiFast [22] - une application web qui étudie l'impact du réseau de contact entre les individus sur la propagation d'une épidémie. Le modèle à compartiments considéré dans EpiFast est S-E-I-R. Le réseau de contact est défini en utilisant un graphe dans lequel chaque individu est un nœud et une arête représente le contact entre deux individus. EpiFast fournit aussi également un algorithme parallèle pour accélérer considérablement la vitesse de la simulation. L'utilisateur va configurer le modèle SEIR en initialisant les paramètres et les stratégies d'intervention prédéfinies.

VLE - VLE (Virtual Laboratory Environment) [98] est une plateforme de modélisation et de simulation permettant de coupler plusieurs formalismes de manière uniforme.

17. Global Epidemic Model

Table 2.2 – Comparaison des outils de modélisation/simulation en épidémiologie

	Aspects	FluTE	GLEAMviz	STEM	FRED	GEM
Approche de simulation	Déterministe	-	-	+	-	+
	Stochastique	-	+	+	-	-
	Multi-agents	+	-	-	+	-
Hétérogénéité des hôtes	Multi-espèces	-	-	+	-	-
	Multi-souches	-	-	+	+	-
	Structure sociologique	+	-	+	+	-
Hétérogénéité spatiale	Meta-population	+	+	+	+	+
	Réseaux de contacts	+	-	-	+	-
	Mobilité des individus	+	+	+	+	+
Stratégies de contrôle	Vaccination	+	+	+	+	+
	Quarantaine/Isolation	+	+	+	+	+
Saisonnalité		-	+	-	-	-

VLE repose essentiellement sur DEVS[121] un cadre formel dédié à la modélisation de systèmes à événements discrets. Il offre une vision modulaire et hiérarchique des systèmes dynamiques : un système peut être décomposé en plusieurs sous-systèmes couplés entre eux. Il par exemple possible de décrire un système d'équations différentielles avec DEVS et de spécifier la résolution numériques à l'aide d'événements discrets [24].

Discussion

Le point commun des outils dont nous avons parlé dans cette section est qu'ils permettent aux modélisateurs de pouvoir construire leurs modèles en utilisant un environnement de développement ou de modélisation la plupart du temps graphique. L'utilisation d'une interface graphique permet naturellement de simplifier la tâche de construction d'un modèle. L'utilisateur peut être guidée dans sa démarche par un workflow spécifique imposé par l'environnement et les modèles peuvent être composés de façon visuelle en glissant/déplaçant les éléments graphiques. Ces outils permettent également aux utilisateurs de pouvoir se focaliser sur les aspects conceptuels en cachant les détails de l'implémentation des moteurs de simulation.

Lors de l'étude de ces outils, nous nous sommes intéressé à leur potentiel pour exprimer les différents aspects de la modélisation épidémiologique. Cela est résumé dans le tableau 2.2.

Un problème important est que ces outils constituent des *boîtes noires* pour les utilisateurs. On connait les entrées et les sorties mais les fonctionnalités sont cachées et donc l'ajout d'extensions difficile.

Par exemple, dans la plupart de cas, les stratégies de contrôle sont prédéfinies dans les outils. Les utilisateurs peuvent configurer juste les paramètres concernés et voir l'effet sur la simulation. Il est difficile de pouvoir ajouter une nouvelle stratégie ou modifier celles qui existent. Autre exemple : dans GLEAMviz, la formule mathématique qui représente l'interaction entre les individus lors de leurs voyages est fixée. L'outil ne

permet aux utilisateurs que de configurer les paramètres. Cela simplifie la construction des modèles mais restreint la flexibilité des outils. On n'a pas beaucoup de choix pour pouvoir faire varier les modèles.

Ce qui manque dans l'ensemble de ces approches est un niveau d'abstraction qui, d'un côté, fournit un degré de flexibilité suffisant pour pouvoir exprimer différents aspects du domaine et de l'autre côté, permet de générer efficacement différents types de simulation.

2.5.3 Langages métiers pour la modélisation épidémiologique

En plus des logiciels indiqués dans la section précédente, un autre choix pour construire les modèle est d'utiliser un langage métier ou **Domain Specific Language** (DSL). Un langage métier est un langage de programmation et de modélisation qui offre à travers des notations et abstractions appropriées, un pouvoir d'expression restreint à un domaine spécifique [59]. L'idée principale des DSL est de cacher les détails de l'implémentation et de fournir une syntaxe plus descriptive que les langages de programmation généraux (comme Java, C/C++ etc.), ce qui les rend donc plus accessible aux experts du domaine et facilite la construction des modèles. Dans cette partie, nous faisons une liste des DSL qui été conçus pour l'épidémiologie.

Ronald

Ronald est un langage métier dédié à la modélisation des interactions entre l'infection du paludisme et un traitement médical [11]. En fonction des parasites paludéens, l'utilisateur va spécifier les médicaments correspondants (y compris la spécification des composants médicaux et leur quantité) et étudier leur impact pharmacocinétique contre ces parasites, puis proposer un régime de traitement. Ronald est un langage embarqué utilisant le langage hôte Groovy. À partir d'un modèle Ronald, on peut générer une simulation Fortran et sortir les résultats sous forme d'une documentation en LaTeX. L'application intègre des outils qui permettent d'analyser les résultats et répondre à des questions telles que : *Après combien de jours le traitement medical reste efficace contre une certaine souche pathogène ?* Ce DSL ne semble plus développé.

Frabjous

Frabjous est un langage de modélisation qui permet de spécifier des modèles agents, couplés avec une plateforme qui est capable de générer le code exécutable pour ces modèles utilisant Haskell/Yamba [18] - un langage fonctionnel [108]. La plateforme Frabjous se compose de deux components : un compilateur qui génère le code Haskell/Yamba à partir de la spécification d'un modèle en Frabjous et un ensemble de bibliothèques en Haskell/Yamba destiné à manipuler le modèle généré. L'objectif est de pouvoir spécifier des modèles à deux niveaux. D'abord, les utilisateurs utilisent Frabjous pour développer rapidement un modèle simple (où les agents ne possèdent que le statut infectieux

18. Yamba est un langage métier embarqué dans le langage fonctionnel Haskell (`https://wiki.haskell.org/Haskell`), destiné à la programmation des systèmes hybrides (considérant à la fois le temps continu et discret).

comme attribut). Ce modèle est ensuite traduit vers le langage métier Yamba. Puis, les utilisateurs peuvent utiliser l'environnement de programmation de Haskell/Yamba pour l'améliorer et de le rendre plus sophistiqué (par exemple par l'ajout d'attributs et comportements aux agents).

Le langage métier Frabjous se consiste de cinq éléments qui aident à spécifier un modèle multi-agents : *modèle, agents, diagrammes, messages, réseaux d'agents*. Un *modèle* Frabjous encapsule tout ce qu'il faut pour exécuter une simulation à base d'agents. Chaque agent a un identifiant unique, un état et un ou plusieurs *diagrammes* d'état-transition qui décrivent comment cet agent passe d'un état à un autre. Les agents communiquent par l'envoi de *messages*. Chaque message contient un déclencheur (*trigger* en anglais) qui soit initialise le processus d'envoi d'un message à un autre agent soit active une transition pour passer d'un état à un autre. Par exemple, dans le modèle SIR, un agent *Infectieux* peut envoyer un message à un autre agent pour lui demander d'établir un contact. Dans le cas où l'agent qui reçoit le message est *Susceptible*, il va changer d'état et devenir *Infectieux* avec une certaine probabilité. Le message est envoyé soit à tous les agents, soit à un agent qui est choisi aléatoirement, ou à un agent dans le réseau de contact de l'envoyeur. Un *réseau de contact* décrit la façon dont certains agents (dans ce réseau) sont reliés les uns aux autres. Chaque réseau possède un identifiant unique, un ensemble d'identifiants des agents et une structure qui définit la méthode pour identifier les voisins d'un agent dans ce réseau.

En résumé, Frabjous est un langage de modélisation essentiellement basé sur une approche multi-agents. Les interactions entre les individus de la population sont capturées en définissant un ou plusieurs réseaux de contact. La combination de plusieurs diagrammes permet d'étudier plusieurs maladies en même temps. Frabjous permet de spécifier des modèles simples où les agents ne possèdent que le statut infectieux comme un attribut, d'autres attributs comme age, sexe, etc. ne sont pas mentionnés. Le passage à des modèles plus complexes se fait en utilisant directement l'environnement de programmation du langage fonctionnel Haskell/Yamba.

Ocelet

Ocelet est un langage métier dédié à la modélisation et la simulation de dynamiques paysagères [35, 36] en utilisant des graphes. Ces graphes visent à représenter les différentes interactions qui peuvent intervenir dans un modèle. Un modèle Ocelet est un programme constitué de différents éléments : des définitions d'entités qui vont être utilisées dans le modèle, des définitions de relations à partir desquelles on pourra construire des graphes d'interaction et au moins un scénario décrivant les séquences d'opérations à effectuer lors des simulations.

Les *entités* d'Ocelet correspondent aux sommets de graphes qui sont constitués d'individus. Ces entités possèdent un état sous la forme de *propriétés*, des services d'entité qui sont des fonctions de consultation ou de transition permettant respectivement de lire l'état des entités et de le modifier. Elles peuvent interagir les unes avec les autres lorsque elles sont reliées à travers les arcs d'un graphe d'interaction. En Ocelet, on doit définir chaque type d'entité qu'on veut utiliser dans un modèle. Une fois qu'un type d'entité a été défini, il est possible de créer un ou plusieurs exemplaires d'entités correspondant à cette definition.

Le concept *relation* est utilisé pour définir un type de graphe d'interaction. Il est ensuite possible de créer une ou plusieurs instances de graphes de ce type. Une relation définit d'une manière générique les arcs d'un graphe et peut posséder certains propriétés (valeurs attachées à un arc) et services (des fonctions exécutée sur un arc). Une fois qu'un graphe est constitué, on peut consulter ou modifier son état en agissant sur les entités qu'il contient ou sur ses arcs.

Un *scénario* est une suite ordonnée d'opérations à travers laquelle les divers constituants d'un modèle peuvent être construits et faire évoluer leurs états. Il contient donc le programme proprement dit qui est exécuté lors d'une simulation. Il est possible de définir plusieurs scénarios dans un même modèle. Chaque modèle possède un scénario principal qui sera considéré comme le point d'entrée pour les simulations, depuis lequel on pourra faire appel aux autres scénarios.

La dynamique d'un système se représente par un ensemble des EDO. À partir de ces équations, on va extraire des services d'entité et services de relation pour construire le modèle Ocelet. Même si Ocelet est plus axé sur la modélisation des dynamiques paysagères, il est néanmoins possible de l'utiliser pour décrire certains modèles de l'épidémiologie comme le modèle SIR ou plus complexe, le modèle du paludisme pour modéliser la propagation du virus au sein de certaines espèces de moustique[35]. Le moteur d'exécution va résoudre les EDO et donner des résultats déterministes.

EpiSpec

EpiSpec [65] est un langage de spécification formelle pour décrire des propriétés d'un modèle agents pour l'épidémiologie.

Dans un modèle EpiSpec, chaque agent est défini par une structure spatio-temporelle probabiliste. Les agents se trouvent dans une espace à trois dimensions qui représente leur position géographique. Le temps est supposé homogène et continu. Chaque agent possède certains événements associés aux probabilités qui sont implicitement déterminées à partir du formalisme mathématique des modèles épidémiologiques. EpiSpec est construit à partir de propositions et expressions atomiques utilisant la logique du premier ordre. Les expressions EpiSpec sont utilisées pour décrire de nombreux indicateurs qui intéressent le modélisateur comme la proportion des infectées par rapport aux susceptibles dans un endroit donné à différents points de temps, le nombre de nouveaux infectés à un endroit donné dans une période de temps donnée ou enfin le moment où le nombre d'infection a atteint une valeur maximum ou minimum.

Les propositions EpiSpec sont utilisées pour capturer des informations plus complexes concernant la transmission spatiale d'un modèle épidémiologique comme le taux auquel les infections se propagent à travers les régions géographiques en tenant compte de l'interaction entre ces régions. Les propriétés des modèles sont représentées utilisant les formules probabilistes de EpiSpec.

Par exemple, on peut représenter une propriété comme : la probabilité pour que le nombre d'infectés atteigne le seuil (C) est de 80% puis, dans les trois semaines qui suivent, le nombre d'infectés dépasse 90% de la taille de la population.

Une vérification formelle est ensuite peut-être effectuée sur les modèles stochastiques

à base d'agents EpiSpec en utilisant des techniques issues du *model-checking* afin de calculer la probabilité pour que le modèle stochastique satisfasse à la spécification. Puis on compare le résultat estimé avec le seuil.

PEPA et Bio-PEPA

Les algèbres de processus définissent une famille de langages formels permettant de modéliser les systèmes concurrents ou distribués. Récemment de nombreuses études portent sur l'utilisation de ces algèbres dans le contexte des systèmes biologiques. Ces systèmes peuvent être représentés généralement par des systèmes concurrents : les espèces sont considérées comme des processus qui sont capable d'interagir les uns avec les autres et les reactions sont modélisées comme des actions de ces processus. Du point de vue de la modélisation épidémiologique, ce genre de langage offre la possibilité de décrire les individus dans la population comme des composants distincts et de spécifier précisément des interactions entre eux.

PEPA [19] est une algèbre de processus dédiée à la modélisation de la performance des systèmes concurrents. Les systèmes sont représentés sous forme de composition de composants séquentiels qui réalisent des actions. Dans PEPA, chaque action est supposée avoir une durée qui est représentée par une variable aléatoire (δt) suivant une distribution exponentielle négative. PEPA a été appliqué dans le contexte de la biologie pour représenter des réseaux chimiques, même si certaines caractéristiques des modèles biochimiques ne peuvent pas être représentées correctement : comme la description des cœfficients stœchiométriques [20] et la loi de la cinétique chimique [21]. Dans PEPA, on suppose que les réactions sont élémentaires avec un taux constant, ce qui n'est pas bon surtout dans le cas où il exige le taux exprimé par une fonction temporelle. De plus, la représentation des événements de naissance et de mort n'est pas facile avec PEPA. La raison est qu'il ne permet pas les processus de création et de suppression. Cela signifie qu'un individu ne peut pas se reproduire et un individu qui est mort ne peut pas être supprimé. Enfin les populations structurées ne peuvent pas être directement représentées en PEPA. Bien qu'il soit possible de proposer des solutions artificielles pour surmonter ces limitations (voir [19]), elles rend les modèles plus complexes et moins accessibles aux les biologistes/épidémiologistes.

Bio-PEPA est une modification du PEPA dédiée à la modélisation des réseaux biochimiques [32]. La dynamique du système est décrit alors par des lois cinétiques en utilisant un ensemble de réactions chimiques. Les modèles compartimentaux de l'épidémiologie peuvent être représentés par des réactions chimiques [22], ce qui fait de BIO-PEPA un bon candidat pour modéliser les maladies infectieuses.

Par exemple, le modèle $SEIR$ représenté par les équations (3.1) peut être formulé par les réactions chimiques :

$$\emptyset \to S; S + I \to E + I; S \to \emptyset; E \to I; E \to \emptyset; I \to R; I \to \emptyset; R \to \emptyset$$

où les réactions comme $\emptyset \to S$, $S \to \emptyset$ représentent les événements de naissance et de

19. Performance Evaluation Process Algebra
20. proportion d'une espèce chimique qui participe à une réaction
21. étude de la vitesse des réactions chimiques
22. Sous certaines conditions, on peut montrer qu'un système décrit sous forme de réactions chimiques est équivalent à un système décrit en utilisant des ODE et vice versa[88].

mort.

Dans Bio-PEPA, plusieurs fonctionnalités ont ajoutées pour résoudre les limitations de PEPA concernant la représentations des modèles biochimiques. Premièrement, Bio-PEPA permet d'exprimer les cœfficients stœchiométriques et le rôle de différents espèces dans une réaction donnée, par exemple, le réactif, le produit de réaction ... Deuxièmement, dans Bio-PEPA, les taux des réactions ne sont pas représentés directement dans la syntaxe des components comme PEPA mais définis par des fonctions temporelles associés aux actions du modèle. L'introduction des taux fonctionnels est une contribution essentielle qui permettent d'exprimer n'importe quelle loi cinétique des réseaux biochimiques. Troisièmement, Bio-PEPA permet de représenter des modèles multi-compartiments, ce qui n'est pas possible avec PEPA.

La notion de *compartiment* est introduite dans Bio-PEPA comme un conteneur d'espèces. Une espèce peut se déplacer entre des compartiments [32]. Elle a été ensuite remplacée par une notion plus générique de *location*[23] [31]. Chaque *location* a un nom explicite et unique dans le système. Pour indiquer une espèce appartenant à une telle *location*, on utilise l'opérateur @ suivi par le nom de cette *location*. Afin de pouvoir capturer des structures plus complexes, une *location* de Bio-PEPA est structurée de façon hiérarchique qu'on appelle *location tree* dont les nœuds représentent des sous-locations, les feuilles correspondants aux espèces. Le fait qu'une même molécule à différentes locations est représentée comme des espèces différentes semble un choix raisonnable du point de vue des systèmes biologiques, car une molécule peut apparaitre dans des réactions différentes en fonction du compartiment dans lequel elle se trouve.

Regardons maintenant plus précisément comment Bio-PEPA modélise des modèles compartimentaux. Prenons l'exemple de $SEIR$. Il y a quatre espèces : S (susceptible), Exp (exposé), Inf (infectieux) et R (rétabli). Les hétérogénéités de la population sont exprimés sous forme de *locations*. Une sous-population correspond à une location dans laquelle les individus sont de différentes espèces.

Les interactions entre les espèces sont :

1. Contact entre un susceptible et un infectieux : $S + Inf \rightarrow Inf + Exp$
2. L'apparition des symptômes chez un exposé : $Exp \rightarrow Inf$
3. L'immunité à l'infection : $Inf \rightarrow R$
4. Toutes les espèces peuvent donner une naissance dans S : $S \rightarrow 2S, Exp \rightarrow Exp + S, Inf \rightarrow Inf + S, R \rightarrow R + S$
5. Toutes les espèces peuvent mourir : $S \rightarrow 0, Exp \rightarrow 0, Inf \rightarrow 0, R \rightarrow 0$

La figure 2.11 représente le code source du modèle $SEIR$ écrit avec Bio-PEPA. L'opérateur $\odot$ indique que l'espèce participe à la réaction sans changer sa concentration (dans le cas de l'épidémiologie, c'est sans changer le nombre d'individus). s_0, e_0, i_0, r_0 sont le nombre initial des espèces S, Exp, Inf, R, respectivement.

Les fonctions des réactions sont définies séparément par la syntaxe : `kineticLawOf [nom de fonction]: [formule]`. Par exemple, les taux fonctionnels des réactions du modèle $SEIR$ sont définis comme indiqué sur la figure 2.12. Une fois que le modèle est complètement défini (y compris l'initialisation des paramètres), Bio-PEPA peut générer des simulations stochastiques utilisant l'algorithme direct de Gillespie ou afficher les dynamiques déterministes en traduisant les réactions vers des EDO[33].

23. Nous utilisons le mot *location* en anglais

$$
\begin{array}{rcl}
S & \stackrel{def}{=} & (contact,1) \downarrow S + (birth,1) \uparrow S + (dieS,1) \downarrow S \\
Exp & \stackrel{def}{=} & (contact,1) \uparrow Exp + (incubation,1) \downarrow Exp \\
 & + & (dieExp,1) \downarrow Exp + (birth,1) \odot Exp \\
Inf & \stackrel{def}{=} & (contact,1) \odot Inf + (incubation,1) \uparrow Inf + (recover,1) \downarrow Inf \\
 & + & (dieInf,1) \downarrow Inf + (birth,1) \odot Inf \\
R & \stackrel{def}{=} & (recover,1) \uparrow R + (dieR,1) \downarrow R + (birth,1) \odot R \\
SEIR & \stackrel{def}{=} & S(s_0) \underset{*}{\bowtie} Exp(e_0) \underset{*}{\bowtie} Inf(i_0) \underset{*}{\bowtie} R(r_0)
\end{array}
$$

Figure 2.11 – Modèle SEIR en Bio-PEPA

```
kineticLawOf contact: beta * S * Inf;
kineticLawOf birth: mu * (S + Exp + Inf + R);
kineticLawOf incubation: sigma * E;
kineticLawOf recover: gamma * I;
kineticLawOf dieS: mu*S;
kineticLawOf dieExp: mu*Exp;
kineticLawOf dieInf: mu*Inf;
kineticLawOf dieR: mu*R;
```

Figure 2.12 – Taux fonctionnels des réactions du modèle SEIR

Dans le cas où la population est structurée par d'autres attributs des individus comme : sexe, âge, espace,... il faut specifier explicitement le taux fonctionnel d'une même réaction pour chaque *location* s'ils sont différents en fonction des *locations*. Par exemple, considérons la population est distribuée sur deux régions géographiques $Zone_1$ et $Zone_2$. On suppose que la probabilité d'infection des individus dans la $Zone_1$ est différente de ceux dans la $Zone_2$. Cela veut dire que la fonction *contact* (de la réaction $S + Inf \rightarrow Inf + Exp$) doit être spécifiée pour chaque *location* $Zone_1$ et $Zone_2$:

```
kineticLawOf contact1: beta1 * Inf@Zone1 * S@Zone1;
kineticLawOf contact2: beta2 * Inf@Zone2 * S@Zone2;
```

Il faut aussi spécifier de façon explicite ces réactions pour les espèces dans la figure 2.11 :

$$
\begin{array}{rcl}
S & \stackrel{def}{=} & (contact1,1) \downarrow S@Zone1 + (contact2,1) \downarrow S@Zone2 \\
 & + & (birth,1) \uparrow S + (dieS,1) \downarrow S \\
SEIRspatial & \stackrel{def}{=} & S@Zone1(s_0) \underset{*}{\bowtie} Exp@Zone1(e_0) \underset{*}{\bowtie} Inf@Zone1(i_0) \\
 & \underset{*}{\bowtie} & R@Zone1(r_0) \underset{*}{\bowtie} S@Zone2(s_0) \underset{*}{\bowtie} \ldots
\end{array}
$$

Il faut le faire pour toutes les autres réactions si il y hétérogénéité d'une *location* à une autre. Par exemple, la réaction $(birth,1) \uparrow S$ sans la notation $@location$ veut dire que cette action est pareil pour les espèces S dans toutes les zones. Si le taux de naissance des individus dans $zone1$ est différent de ceux dans $zone2$, il faut spécifier deux lois cinétiques $birth1$ et $birth2$, donc deux réactions $(birth1,1) \uparrow S@Zone1$ et

$(birth2, 1) \uparrow S@Zone2$. Dans des scénarios plus complexes, la taille et la complexité du modèle croisent linéairement avec le nombre de compartiments, ce qui implique que l'on arrive rapidement à une situation que les outils actuels de Bio-PEPA ne sont pas capables de traiter.

Discussion

Les langages métiers permettent de séparer ce qui relève du domaine de ce qui concerne plus la programmation. Ils sont donc normalement plus facile à utiliser pour des personnes qui n'ont pas de compétence en informatique.

Généralement, le développement d'un langage se décompose en deux parties : (a) la construction de la syntaxe abstraite (qu'on appelle également *méta-modèle* du langage) qui se détermine les concepts du domaine et les relations qu'ils établissent entre eux (b) la construction de la sémantique du langage. En raison de la séparation abordée ci-dessus, la syntaxe abstraite d'un langage métier peut être réutilisée pour plusieurs modules sémantiques. C'est-à-dire qu'un modèle peut être traduit vers différentes sortes de modèle comme des modèles de simulation, des modèles écrits dans un langage générique etc, ce qui rend les langages métiers plus flexible que les logiciels intégrés que nous avons vu précédemment. De plus, l'utilisation de logiciels intégrés à l'aide d'une interface graphique cache parfois la sémantique des concepts du domaine.

Par exemple, dans des outils comme STEM ou GLEAMviz, le mécanisme de la transmission spatiale est implémenté comme une fonctionnalité interne du système. Les utilisateurs se focalisent sur la spécification d'une carte graphique sur laquelle ils vont étudier la propagation des épidémies ainsi que les paramètres concernés. Les formules mathématiques sont cachées aux modélisateurs. Dans le cadre des langages métiers, grâce à une syntaxe plus déclarative se basant sur les concepts du domaine, on peut définir ce qui concerne le domaine de façon explicite (par exemple, composer directement des équations, des formules mathématiques de la transmission etc.). C'est la raison pour laquelle les modèles sont plus accessibles aux experts du domaine d'un certain point de vue.

Récemment, les langages métiers ont été largement utilisés dans la modélisation écologique/biologique, mais seulement quelques-uns d'entre eux ont porté sur le domaine de l'épidémiologie. Parmi ceux qui se sont représentés dans cette section, certains langages ne sont pas dédiés au domaine de l'épidémiologie comme Ocelet, Bio-PEPA, d'autres visent à résoudre des besoins plus concrèt comme Ronald pour modéliser l'effet des traitements médicales dans la lutte contre le paludisme, Frabjous pour spécifier un modèle à base d'agents en épidémiologie ou EpiSpec pour décrire un modèle multi-agents de façon formelle. Ocelet et Bio-PEPA peuvent être utilisés pour modéliser la propagation des épidémies car ils ont été construits en se basant sur des approches qui sont également appliquées pour formuler des modèles mathématiques de l'épidémiologie : EDO ou réactions chimiques. Néanmoins, puisqu'ils se situent dans un autre domaine, la représentation des aspects spécifiques de l'épidémiologie peut s'avérer plus compliquée que nécessaire.

Dans le cadre de la thèse, nous visons à développer un langage de modélisation dédié à la modélisation des modèles à base des compartiments en épidémiologie. Nous nous focalisons exclusivement sur les modèles qui sont formulés par des ODE. Les objectifs

du langage proposé sont de :

- Faciliter la construction des modèles à compartiments de l'épidémiologie en offrant aux utilisateurs une syntaxe déclarative des concepts du domaine.
- Produire les différentes formes de simulation comme déterministe, stochastique, multi-agents ainsi que de pouvoir générer automatiquement à partir du DSL un programme dans un langage généraliste comme C/C++ ou R.
- Fournir des moyens pour visualiser les résultats de simulation.
- Pouvoir exprimer les aspects de l'épidémiologie, l'un de façon séparée de l'autre et permettre de construire les modèles par la composition des aspects modulaires.

Le prototype que nous allons présenter dans le chapitre 3, va essayer de résoudre les trois premiers objectifs. Nous examinerons les lacunes de ce première version et nous proposerons un deuxième version de notre langage qui sera abordé dans les chapitres 4, 5 et 6.

2.6 Conclusion

Dans ce chapitre, nous avons rappelé les éléments de base de la modélisation en épidémiologie. Nous avons vu que pour étudier une maladie infectieuse, on construit généralement un modèle à base d'équations différentielles afin de représenter les cycles de transmission de cette maladie. Les modèles les plus utilisés dans ce domaine sont les modèles en compartiments dont le modèle le plus simple est *SIR*. Les modèles mathématiques sont ensuite implémentés par des programmes afin de déterminer leurs dynamiques avec en utilisant différents paradigmes : déterministe, stochastique ou encore multi-agents.

Le mécanisme de transmission des maladies est influencé par les interactions entre les individus-hôtes. On a besoin d'ajouter des aspects supplémentaires pour étudier l'effets de différentes sortes d'interactions. Au fur et à mesure de leurs utilisations, le modèles épidémiologique ont connu une sophistication de plus en plus considérable. Une telle explosion de la complexité rend la modélisation parfois inaccessible pour les non-informaticiens. Il exige donc de développer des outils de modélisation qui aident à faciliter la construction et la simulation de ces modèles. Le travail de cette thèse a pour but de construire un langage de modélisation dédié au domaine de l'épidémiologie. Le choix de développer un langage métier a été principalement motivé par deux objectifs : (1) séparer ce qui concerne le domaine de ce qui concerne la programmation, l'un pouvant varier de façon indépendante de l'autre ; (2) permettre de formuler les modèles de façon explicite et faciliter des changements grâce à une syntaxe descriptive qui est plus accessible aux experts du domaine.

Chapitre 3

Vers un premier Méta-modèle pour l'Épidémiologie

Sommaire

Dans ce premier chapitre, nous décrivons le premier prototype d'un langage métier dédié à la modélisation mathématique de l'épidémiologie que nous avons réalisé. Notre objectif est de pouvoir fournir une syntaxe de haut niveau permettant aux modélisateurs de pouvoir facilement spécifier leurs modèles mathématiques sans avoir besoin nécessairement d'utiliser un langage de programmation. Nous avons conçu le langage de modélisation Kendrick et la plate-forme sous-jascente permettant de générer des simulations pour étudier le comportement de ces modèles. Pour y parvenir, nous proposons un méta-modèle qui représente la syntaxe abstraite du langage et l'ensemble des opérations sémantiques qui traduisent un modèle en Kendrick vers des implémentations de simulation déterministe, stochastique ou multi-agents et également la possibilité de génére le code des simulateurs en C/C++. Nous proposons également une syntaxe concrète au dessus de la syntaxe abstraite afin de faciliter la tâche du modélisateur. Pour illustrer le fonctionnement de ce premier prototype du langage de modélisation Kendrick, nous utiliserons quelques modèles épidémiologiques emblématiques.

Dans la deuxième partie de ce chapitre, nous faisons un retour d'expérience sur la conception du langage pour voir s'il peut s'adapter aux différents cas d'usages de l'épidémiologie. Nous nous focalisons sur les problèmes posés par les aspects que nous avions décrit dans le chapitre 1 et auxquels le premier méta-modèle proposé ne peut pas répondre de manière satisfaisante. Puis, nous présentons les exigences pour un nouveau méta-modèle qui devrait être aussi générique que possible pour pouvoir représenter les

divers aspects du domaine.

Publication :
T.M.A BUI, S. Stinckwich, M. Ziane, B. Roche, T.V. HO, KENDRICK *: A domain specific language and platform for epidemiological modelling.* 11^{th} IEEE-RIVF International Conference on Computing and Communication Technologies, RIVF-2015, IEEE, 2015, pp. 132-137.

3.1 Le premier prototype du langage de modélisation Kendrick

Cette section est consacrée à la description du premier prototype de Kendrick, le langage métier pour aider à la modélisation épidémiologique. La conception de Kendrick correspond à deux objectifs que nous avons identifiés à la fin du chapitre précédent :

- Proposer un langage de haut niveau qui facilite la description des modèles en utilisant les concepts de base de l'épidémiologie.
- Fournir des outillages de haut niveau permettant aux épidémiologistes/biologistes/mathématiciens (i.e experts du domaine) d'utiliser des modèles écrits avec ce language afin de produire différentes formes de simulation en vue de leurs analyses ultérieures.

La décision d'utiliser un langage métier est le résultat d'un compromis : nous nous trouvons à mi-chemin entre la bibliothèque de fonctions pour un langage généraliste et une plateforme intégrée de modélisation regroupant des outils de modélisation graphique ou des modèles prédéfinis que l'on peut paramétrer.

Avec une bibliothèque de fonctions ou API [1], on peut ajouter de nouvelles structures de données et de nouvelles fonctions à un langage de programmation existant. Cela revient en quelque sorte à étendre les possibilités d'un langage (souvent un langage généraliste comme Java, C/C++ etc...) en proposant des modules pré-programmés que l'on peut assembler et paramétrer tout en bénéficiant de la richesse d'expressivité de ce langage. Cette forme d'extension offre un gain de temps important : on n'a pas besoin de programmer soi-même un certain nombre de fonctions, ni d'acquérir l'expertise nécessaire pour les programmer. C'est aussi potentiellement un gain en qualité puisque ces bibliothèques parfois complexes doivent faire l'objet de tests et doivent en principe respecter des spécifications précise et documentées. Cependant, cette forme d'extension n'affecte en rien la sémantique de base du langage auquel elle est dédiée. En effet, une API développée en Java ou C++ apportera de nouvelles classes et fonctions, mais restera fondamentale orientée objet, et une bibliothèque pour Lisp apportera de nouvelles fonctions mais restera fondamentalement basée sur la programmation fonctionnelle.

En ce qui concerne les plateformes intégrées disposant d'une interface graphique d'utilisateur (GUI [2]) dédiées à la modélisation épidémiologique (voir Section 2.5.2), elles permettent de spécifier des modèles et de rendre transparent les modules de simulation mais sont souvent des outils fermés, i.e. une boîte noire dont les utilisateurs ne connaissent pas le fonctionnement interne. Ainsi, les bibliothèques et les plateformes actuelles souffrent du manque d'une sémantique particulière qui reflètent les concepts de l'épidémiologie et les relations entre eux. Comme dans [63], Holst et Belete ont montré qu'un langage métier est une solution représentant explicitement la sémantique du domaine. Ce type de langage peut être utilisé non seulement pour composer des modèles mais également pour valider les comportements et relations entre des composants constitués, autrement dit pour étendre l'expressivité sémantique de ce langage.

Proposer un langage métier c'est proposer une nouvelle forme d'expression, et cela se justifie particulièrement si cette forme d'expression reflète et accompagne une nouvelle façon d'approcher la conception d'un programme. La conception de modèle épidémiologique faisant appel à la représentation des équations différentielles nécessite une

1. Application Programming Interface
2. Graphical User Interface

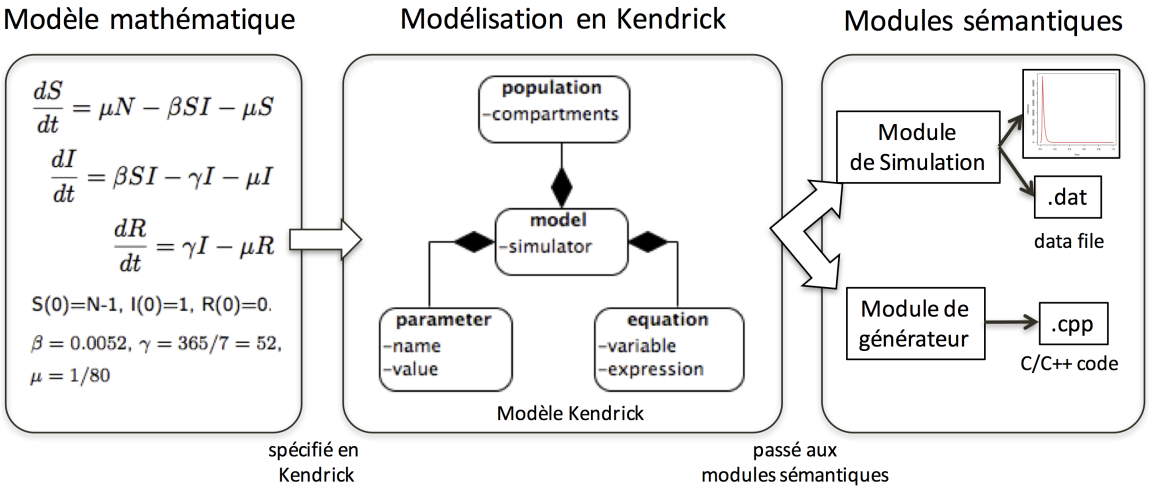


Figure 3.1 – Vue générale de la plateforme Kendrick

réflexion particulière sur le système que l'on cherche à modéliser : il faut par exemple réfléchir à la représentation des compartiments qui sont reliés aux variables des EDO, il faut aussi imaginer les processus qui vont être mis en œuvre sur les équations pour extraire les transitions qui représentent les relations entre ces compartiments etc. Cette façon de concevoir reflète une sémantique particulière qu'un langage métier est mieux à même d'intégrer qu'une bibliothèque de fonctions de type API.

Suivant les recommendations pour le développement de langages métiers proposées par Parr, T. [91], la conception du langage de modélisation Kendrick se décompose en deux parties :

- La première partie définit la syntaxe abstraite du langage qui fournit des concepts adaptés à la modélisation mathématique des modèles de l'épidémiologie ainsi que la syntaxe concrete du langage.
- La deuxième partie vise à établir l'ensemble des opérations sémantiques qui permettent d'exécuter une simulation d'un modèle ou traduisent ce modèle vers un langage de programmation généraliste comme C/C++.

La Figure 3.1 représente l'architecture générale de Kendrick et de la plateforme de simulation associée. Nous présentons dans la suite les concepts de base de Kendrick et leurs relations ainsi que les opérations sémantiques que l'on peut effectuer sur un modèle en Kendrick. Enfin, on s'intéresse à quelques exemples emblématiques dans le domaine pour montrer comment utiliser le langage obtenu.

3.1.1 Le méta-modèle de Kendrick

Le langage Kendrick est construit en se basant sur cinq concepts de base : *modèle, population, compartiment, paramètre et équation* qui sont représentés par des classes. Pour chaque concept, nous présentons également la syntaxe concrète pour le définir en Kendrick (l'implémentation sera détaillée ultérieurement).

Modèle

Construire un modèle à l'aide de Kendrick revient à écrire un programme dont l'exécution permettra de simuler l'évolution des variables de ce modèle. Un tel programme est constitué de différents éléments : des définitions de population et de compartiments qui vont être utilisées dans le modèle, des définitions d'équations différentielles qui représentent la dynamique de population, des définitions d'un moteur de simulation pour générer la dynamique déterministe ou stochastique du modèle. À cela peuvent s'ajouter quelques éléments complémentaires, comme par exemple les paramètres, leurs fonctions et la configuration initiale du modèle (les valeurs initiales des compartiments) ainsi que du moteur de simulation (durée d'exécution, algorithme utilisé). Ensuite, pour visualiser les résultats de simulation, on peut ajouter au modèle une visualisation des résultats sous forme de diagrammes. Les modèles en Kendrick sont définis de façon textuelle au travers d'un script.

Un modèle en Kendrick est constitué de la façon suivante [3] :

```
model := KEModel new.
model population: [a population]
model addEquation: [an equation]
model addParameter: [name] value: [a constant or an expression]
model run: [algorithm] from: [t_min] to: [t_max] step: [a step]
model plot: [compartment list]
model generate: [langage] algorithm: [algorithm]
```

Chaque modèle est une instance de la classe `KEModel` [4]. Il est relié aux définitions des autres éléments pour constituer un modèle épidémiologique.

Population

Chaque modèle de Kendrick est étudié sur une seule population. La population se compose des individus hôtes qui sont catégorisés en compartiments selon leur état clinique. Durant l'exécution d'une simulation, à chaque pas de temps, on va conserver la taille de chaque compartiment et l'état de chaque individu.

Une population est définie selon le modèle suivant :

```
population := KEPopulation new.
population compartments: [compartment symbols]
population at: [a status] put: [a value]
```

par exemple :

```
population := KEPopulation new.
population compartments: #(S I R).
population at: #S put: 10000
```

3. Les éléments entre crochets représentent les paramètres en général.
4. Les classes de Kendrick sont préfixées avec KE

Pour tenir compte de l'hétérogénéité de la population résultant par exemple de sa structuration en *âge, sexe, espace, etc.*, la valeur d'un compartiment peut être un vecteur (de valeurs entières). Par exemple : `population at: #S put: #(999 1000 2000)`. Cela veut dire que le compartiment S de la population se subdivise en trois parties contenant respectivement $999, 1000, 2000$ individus.

Compartiment

Un compartiment est considéré comme une sous-population dans laquelle tous les individus ont le même état infectieux. Chaque compartiment est caractérisé par une étiquette (état clinique) et une valeur qui représente le nombre des individus. Un compartiment peut être structuré par les attributs des individus, donc sa valeur peut également être un vecteur d'entiers.

Équation

Une équation est une instance de la classe `KEEquation` représentant une équation différentielle ordinaire dont la variable correspond à un compartiment. Les équations sont spécifiées sous forme de chaînes de caractères et traduites par un analyseur syntaxe (*parser*) que nous avons développé pour la plateforme que nous utilisons. Dans le cas où les compartiments sont des vecteurs, les équations sont indexées. L'équation se constitue de deux autres concepts : *variable et expression*. La valeur d'une variable à un instant t représente le nombre d'individus du compartiment correspondant.

Une équation est définie de la façon suivante :

```
equation := '[variable] : t = [expression]' parseAsAnEquation.
```

comme par exemple :

```
'S:t = -beta*S*I' parseAsAnEquation.
'S[i]:t = mu[i]*N[i] - sum_j(beta[i][j]*I[j])S[i] - mu[i]*S[i]'
parseAsAnEquation.
```

Paramètre

Un modèle peut nécessiter plusieurs paramètres. Par exemple, le modèle SIR (voir la Section 2.2.1) a deux paramètres β - le taux de transmission - et γ - le taux de guérison. Chaque paramètre est potentiellement une fonction temporelle pour pouvoir considérer le cas où un paramètre fluctue au cours du temps (la fonction peut rester constante si besoin). Les paramètres sont des instances de la classe `KEParameter`.

Un paramètre est défini de la façon suivante :

```
parameter := KEParameter new.
```

```
parameter name: [name] value: [a constant].
parameter name: [name] value: [an expression].
```

par exemple :

```
parameter name: #beta value: `cos(t)' parseAsAnExpression.
```

3.1.2 Syntaxe concrète

Il existe plusieurs façons de réaliser un langage métier (DSL) [51]. Généralement, on distingue 3 approches : langage interne, langage externe et langage embarqué.

Les DSLs internes partagent la même syntaxe que leur langage hôte et en général il s'agit d'un sous-ensemble. C'est l'approche la plus simple car elle permet de réutiliser en grande partie les bibliothèques et les outils du langage hôte, mais le principal inconvénient est qu'il n'est pas possible de changer la syntaxe ou la sémantique du DSL dans la plupart des langages de programmation.

Les DSLs externes, par contre, ont leur propre syntaxe et sont typiquement développés de façon indépendante de leur langage hôte. Le développement d'un langage externe n'est pas différent du développement d'un nouveau langage de programmation. Cela passe par la construction d'une grammaire, d'un analyseur syntaxique et d'un compilateur.

L'expressivité des langages internes est limitée par la syntaxe de leur langage hôte alors que les langages externes offrent une plus grande liberté à la fois dans la syntaxe et leur sémantique.

Une approche intermédiaire entre ses deux extrêmes consiste à utiliser un langage embarqué, qui étend un langage hôte en fournissant une nouvelle syntaxe et sémantique.

Pour décrire plus précisément, la nature du langage que nous avons défini, nous utilisons la terminologie introduite par Lukas Renggli dans sa thèse au chapitre 2 [101]. L. Renggli définit 3 niveau d'enchâssement pour un langage embarqué : pidgin, créole et argot. Un pidgin détourne la syntaxe d'un langage hôte et introduit un nouveau vocabulaire et sémantique. On peut par exemple utiliser des chaînes de caractères pour cela [5]. Un créole introduit une syntaxe complètement nouvelle et des transformations vers le langage hôte qui permet de définir la sémantique. Un argot utilise la syntaxe du langage hôte mais change sa sémantique.

Le langage de modélisation Kendrick combine à la fois une approche de langage interne (en s'appuyant sur le langage hôte `Smalltalk`) et en partie une approche de type langage embarqué de type pidgin, notamment pour décrire les équations différentielles. Il a été réalisé au moyen d'un dialecte moderne de Smalltalk appelé Pharo [39, 90] et utilise de nombreux outils de la plateforme de méta-modélisation libre (MOOSE [6]). MOOSE fournit un environnement intégré permettant de définir facilement des méta-modèles et leurs transformations, des analyseurs syntaxiques (PetitParser) et des outils

5. C'est le cas pour la chaîne de caractère de la fonction printf dans la bibliothèque standard de C.
6. `http://www.moosetechnology.org/`

de visualisation (Roassal).

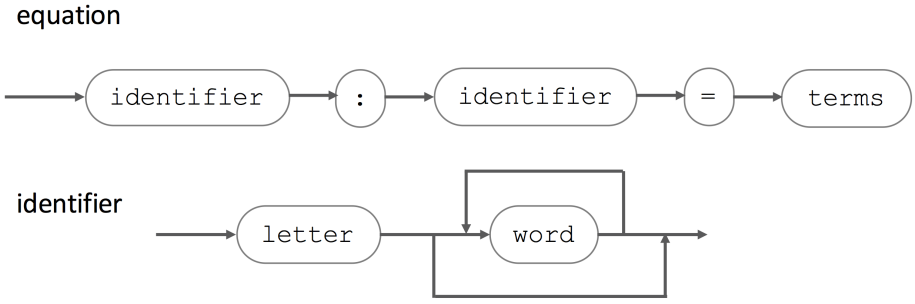


Figure 3.2 – La syntaxe des équations différentielles en Kendrick

Nous avons construit un analyseur syntaxique afin de pouvoir d'écrire des équations différentielles ordinaires sous une forme naturelle pour le modélisateur. On utilise pour cela PetitParser [7], un cadre d'application (framework) qui permet de construire facilement des analyseurs syntaxiques par réutilisation/composition/transformation de grammaires [102].

3.1.3 Les modules sémantiques du langage

Une module sémantique du langage va prendre en entrée des instances du méta-modèle pour effecter des calculs ou traitement. Comme nous l'avons abordé précédemment, les modules sémantiques de Kendrick mettent en œuvre deux tâches : l'exécution des simulations (avec trois modes : déterministe, stochastique et multi-agents) et la traduction des modèles exprimés en Kendrick vers le langage de programmation généraliste C/C++ (Figure 3.3).

La figure 3.4 représente le méta-modèle et les modules sémantiques de Kendrick. On remarquera qu'il n'y a pas de dépendances entre les modules sémantiques et le cœur du méta-modèle.

Les modélisateurs doivent indiquer l'algorithme utilisé par la simulation. On utilise des algorithmes de résolution d'EDO pour résoudre de façon déterministe le système d'équations, comme par exemple l'algorithme Runge-Kutta d'ordre 4 [56].

Le moteur de simulation stochastique extrait les événements à partir des équations différentielles et utilise la méthode directe de Gillespie ou la méthode τ-leap (voir la section 2.4) pour la génération des modèles stochastiques. Il est également possible de formuler des modèles individus-centrés à partir du système d'équations différentielles afin de pouvoir capturer la stochasticité au niveau individuel.

Enfin il est possible de générer du code exécutable C/C++. Ce générateur effectue une visite du modèle exprimé à parti d'équations différentielles, interprète les éléments et les traduit en C/C++. Par exemple, le modèle SEIR de la rougeole (que l'on reverra dans la section 3.1.4) peut être interprété (en utilisant la méthode directe de Gillepsie) et traduit en C/C++ comme suivant :

7. PetitParser est un mélange original de plusieurs techniques d'analyses syntaxiques : scannerless, parser combinators, packrat parsers, etc ... qui autorise la définition de grammaires et d'analyseurs qui peuvent être recombinés dynamiquement. PetitParser définit pour cela un DSL interne à Pharo.

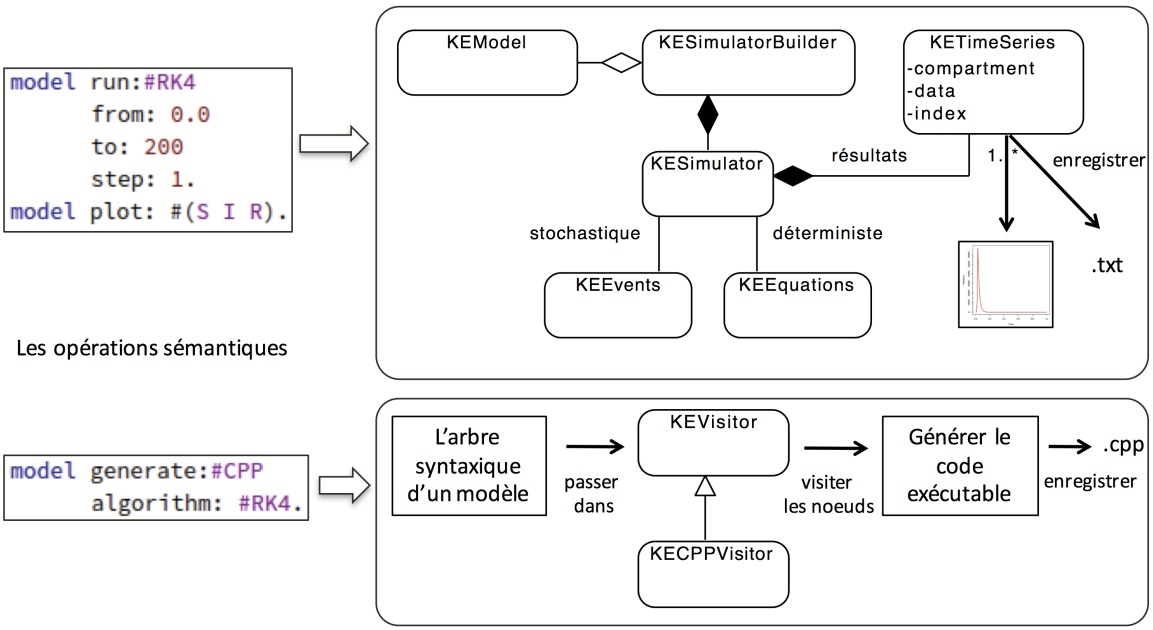


Figure 3.3 – Vue d'ensemble des modules sémantiques du langage Kendrick

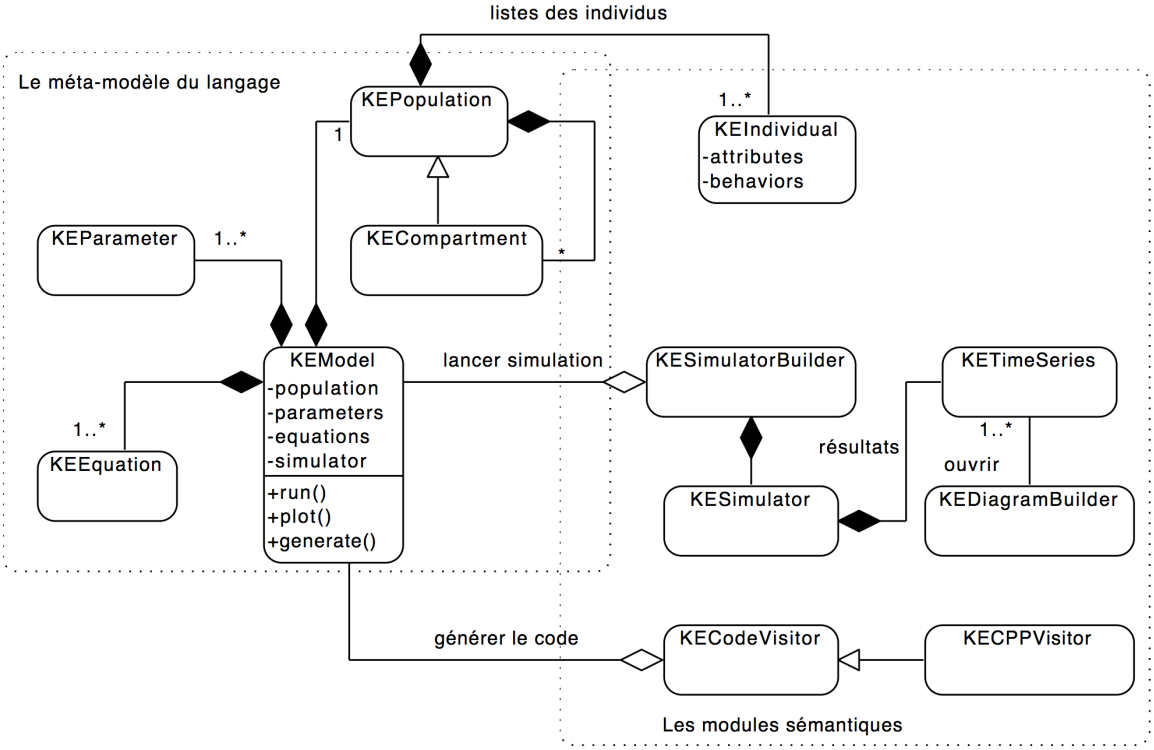


Figure 3.4 – Le schéma UML représente le méta-modèle et les modules sémantiques de Kendrick

```
model generate: #CPP algorithm: #Gillespie from: 0 to: 100 step: 0.001.
```

Voici un extrait du code C/C++ généré qui représente la fonction qui calcule les taux des événements du modèle SEIR générés à partir de ses équations différentielles spécifiées en Kendrick [8] :

```
void calculateRate()
{
    rates[0] = (-(gamma*x[0]));
    rates[1] = (sigma*x[2]);
    rates[2] = (-((beta*x[1])*x[0]));
    rates[3] = (-(mu*x[1]));
    rates[4] = (-(mu*x[3]));
    rates[5] = (-(mu*x[2]));
    rates[6] = (-(mu*x[0]));
    rates[7] = (mu*N);
}
```

3.1.4 Quelques études de cas en épidémiologie

Dans cette partie, nous détaillons deux exemples montrant comment utiliser le langage Kendrick. Le premier exemple vise à étudier une épidémie de rougeole. Le deuxième a pour but de montrer un modèle plus complexe dans lequel on considère la transmission du virus entre plusieurs espèces d'hôtes.

La modélisation de la rougeole

Le modèle le plus approprié représentant l'épidémie de la rougeole [116] est un modèle de type $SEIR$ avec démographie (voir figure 2.4) dans lequel les individus sont catégorisés en quatre classes : d'abord, les individus nouveau-nés entrent dans la classe S avec le taux de naissance μ. Puis les susceptibles passent dans la classe E des infectés mais pas encore infectieux avec le taux de transmission β. Ils deviennent alors infectieux (classe I) après une période moyenne de latence $1/\sigma$ et finalement, entrent dans la classe R (et sont donc Rétablis) après une période moyenne d'infectiosité de $1/\gamma$. Tous les individus meurt au taux μ afin de garder la taille de population constante $N = S + E + I + R$

Ce modèle $SEIR$ est représenté par l'ensemble d'EDO suivant :

8. Le modèle complet en C/C++ se trouve dans l'annexe à la fin de ce rapport.

$$\begin{cases} \frac{dS}{dt} &= \mu N - \beta SI - \mu S \\ \frac{dE}{dt} &= \beta SI - \sigma E - \mu E \\ \frac{dI}{dt} &= \sigma E - \gamma I - \mu I \\ \frac{dR}{dt} &= \gamma I - \mu R \end{cases} \tag{3.1}$$

La figure 3.5 représente ce modèle en Kendrick. La figure 3.6 montre les résultats déterministes du modèle généré par le moteur de simulation déterministe utilisant l'algorithme Runge-Kutta 4.

```
1  |model|
2  model := KEModel new.
3  model population compartments: #(S E I R).
4  model population
5        at: #S put: 99999;
6        at: #E put: 0;
7        at: #I put: 1;
8        at: #R put: 0.
9  model addParameter: #beta value: 0.0000214.
10 model addParameter: #gamma value: 0.143.
11 model addParameter: #sigma value: 0.125.
12 model addParameter: #mu value: 0.0000351.
13 model addParameter: #N value: 100000.
14 model addEquation:
         ('S:t=mu*N-beta*S*I-mu*S'
         parseAsAnEquation).
15 model addEquation:
         ('E:t=beta*S*I-sigma*E-mu*E'
         parseAsAnEquation).
16 model addEquation:
         ('I:t=sigma*E-gamma*I-mu*I'
         parseAsAnEquation).
17 model addEquation:
         ('R:t=gamma*I-mu*R'
         parseAsAnEquation).
18 model run: #RK4 from: 0 to: 100 step: 0.01.
19 model plot: #(I S E R)
```

Figure 3.5 – Le script représente le modèle de la rougeole en Kendrick

Supposons que l'on veut étudier l'impact de la vaccination sur la rougeole. La vaccination a pour effet de rendre un individu susceptible immunisé à la maladie, ce qui correspond donc au statut Rétabli (R). Il nous faut changer les équations de S et R pour ajouter des événements supplémentaires liées à la vaccination. Plus précisément, on suppose que la vaccination est réalisée à la naissance, c'est-à-dire qu'une partie des nouveaux-nés sont vaccinés (avec le taux de vaccination p) et vont donc entrer directement dans la classe R au lieu de la classe S. Ainsi, les nouveaux-nés qui entrent dans la classe S sont exprimés de façon mathématique par $\mu N(1-p)$ et ceux qui entrent dans la classe R sont formulés par μNp.

Le paramètre p qui décrit le taux de vaccination devrait être ajouté dans le script et

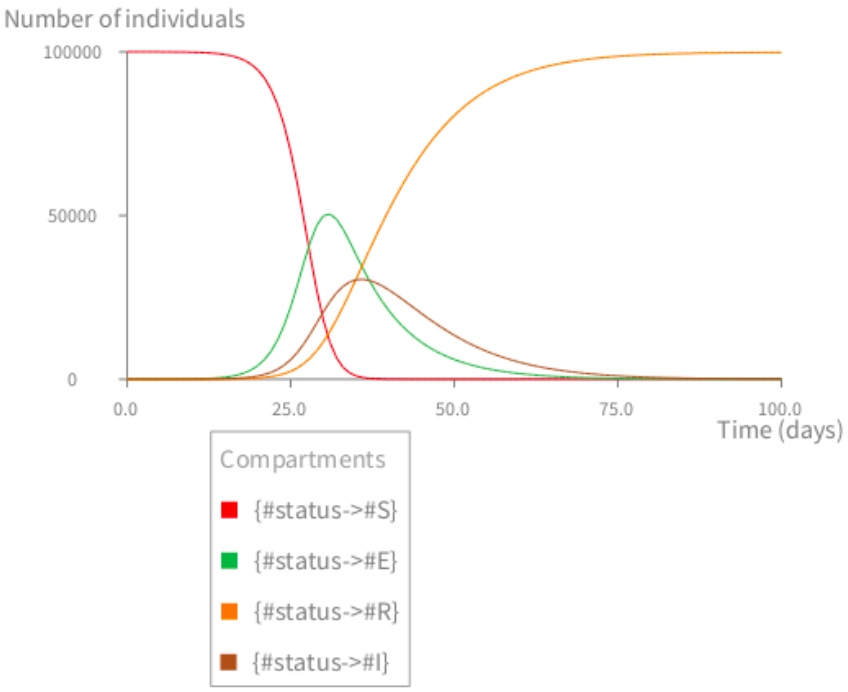


Figure 3.6 – La dynamique déterministe du modèle de la rougeole étudiée durant 100 jours, générée par le constructeur de diagramme de Kendrick

les lignes de code 14,17 devraient être remplacées par :

```
14  model addEquation:
          ('S:t=mu*N*(1-p)-beta*S*I-mu*S'
          parseAsAnEquation).
17  model addEquation:
          ('R:t=mu*N*p+gamma*I-mu*R'
          parseAsAnEquation).
    model addParameter: #p value: 0.7.
```

Figure 3.7 représente l'impact de la vaccination en comparant les valeurs différentes du taux de vaccination ($p = 0$ et $p = 0.7$). Les diagrammes représent les courbes générées par deux simulations déterministes du modèle (en changeant la valeur du paramètre p). On vérifie bien que grâce à la vaccination, le nombre de personnes infectées se réduit considérablement.

La modélisation d'une maladie vectorielle

Dans cette partie, on va considérer une maladie vectorielle dans laquelle la transmission de l'infection est indirecte par l'intermédiaire d'espèces vecteurs, comme les moustiques ou les tiques. Ces espèces sont nécessaires à la dispersion de l'infection en transportant l'agent pathogène d'un hôte à un autre. On s'intéresse ici à une maladie vectorielle dont l'infection est transmise par un moustique entre deux espèces appelées réservoirs [9]

9. En épidémiologie animale, on nomme espèce-réservoir toute espèce dans l'organisme de laquelle au moins un agent pathogène prolifère de manière prépondérante. Par exemple, le virus de la rage utilise des chauve-souris comme espèce-réservoir.

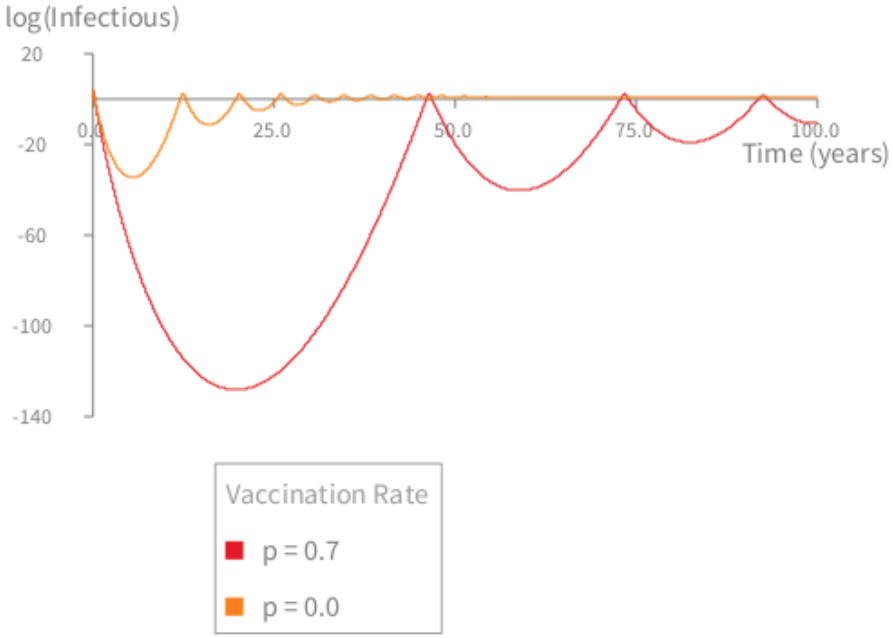


Figure 3.7 – Les dynamiques déterministes du modèle de la rougeole avec l'effet de la vaccination. L'axe Y est sur une échelle logarithmique pour des raisons de lisibilité. Les paramètres du modèle sont les mêmes que ceux du modèle précédent, mais l'unité de temps est exprimé en années.

Le modèle résultant est donc un modèle multi-espèces SIR dans lequel nous considérons la démographie des espèces vecteurs et réservoirs. Chaque compartiment S, I, R du modèle SIR est divisé en trois groupes correspondants aux trois espèces. La population est donc décomposée en 3×3 compartiments dont la dynamique est représentée par les équations suivantes :

$$\begin{cases} \frac{dS_i}{dt} &= \mu_i N_i - \sum_j^n \beta_{ij} I_j S_i - \mu_i S_i \\ \frac{dI_i}{dt} &= \sum_j^n \beta_{ij} I_j S_i - \gamma_i I_i - \mu_i I_i \\ \frac{dR_i}{dt} &= \gamma_i I_i - \mu_i R_i \end{cases} \tag{3.2}$$

La transmission du pathogène s'effectue via un contact entre des espèces différentes, c'est pourquoi le taux de transmission β est une matrice dont la diagonale principale est zéro. Nous utilisons une syntaxe compacte pour représenter ces équations et initialiser les paramètres ainsi que les valeurs initiales des compartiments. La figure 3.8 représente ce modèle en Kendrick.

En raison de l'hétérogénéité de la population qui est divisée en plusieurs espèces différentes, les compartiments S, I, R et les paramètres sont donc des vecteurs.

La figure 3.9 représente les dynamiques d'infection de chaque espèce générées par la simulation stochastique utilisant la méthode directe de Gillespie. POur utiliser une simulation multi-agents, il suffit de changer l'algorithme, de `#Gillespie` à `#IBM` [10].

10. L'approche que nous avons utilisée pour formuler le modèle multi-agents dans le cadre de cette thèse

```
1  |model|
2  model := KEModel new.
3  model population
4    compartments: #(S I R).
5  model population
6    at: #S put: #(9999 1000 2000);
7    at: #I put: #(1 0 0);
8    at: #R put: #(0 0 0).
9  model
     atParameter: #beta
     put: #(#(0 0.02 0.02) #(0.02 0.0 0.0)  #(0.02 0.0 0.0)).
10 model
     atParameter: #N
     put: #(10000 1000 2000).
11 model
     atParameter: #mu
     put: {365/30 . 1/20 . 1/20}.
12 model atParameter: #v put: 52.
13 model atParameter: #j put: (1 to: 3).
14 model atParameter: #i put: (1 to: 3).
15 model addEquation:
     'S[i]:t=mu[i]*N[i]-sum(j, beta[i, j]*I[j])*S[i]-mu[i]*S[i]'
     parseAsAnEquation.
16 model addEquation:
     'I[i]:t=sum(j, beta[i,j]*I[j])*S[i]-(mu[i]+v)*I[i]'
     parseAsAnEquation.
17 model run: #Gillespie from: 0 to: 100 step: 0.01.
18 model plot: #I do: #sqrt.
```

Figure 3.8 – Modèle multi-espèces exprimé avec Kendrick

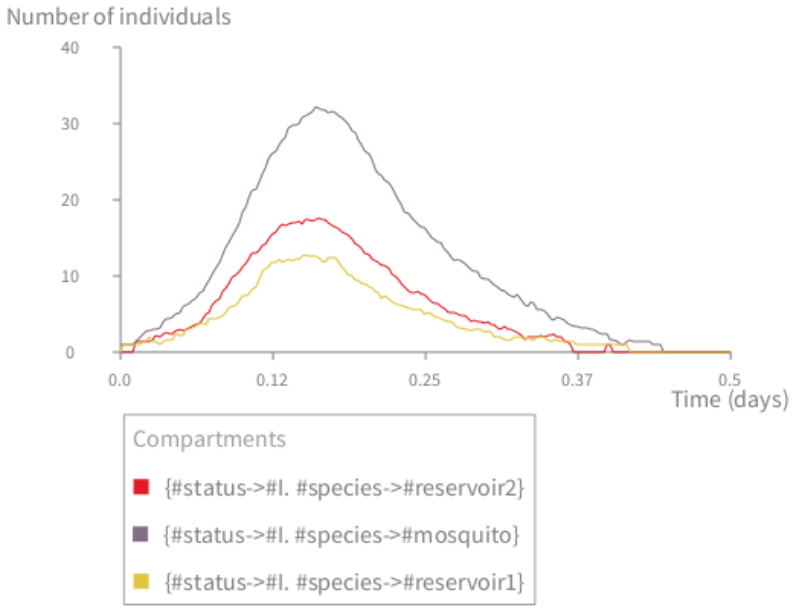


Figure 3.9 – Les dynamiques des infections du modèle stochastique multi-espèces

3.2 Retour d'expérience

3.2.1 Avantages du langage Kendrick

Au début de ce chapitre, nous avons dit que la conception du langage Kendrick devrait répondre à deux objectifs : un langage permettant aux utilisateurs de pouvoir facilement spécifier des modèles mathématiques en utilisant une syntaxe de haut niveau qui se base sur les concepts de l'épidémiologie, et de produire ensuite différentes sortes de simulation.

```
31 function [step, new_value] = Iterate(old, beta,gamma,mu,N)
32 n = 3;
33 nbOfEvents = 6;
34 S = old(1, :);
35 I = old(2, :);
36 R = old(3, :);
37 lambda = zeros(1,3);
38 for i=1:n
39    lambda(i)=sum(beta(i,:).*I);
40 end
41 rate = zeros(nbOfEvents,n);
42 change = zeros(n,nbOfEvents);
43 rate(1,:) = mu.*N;change(:,1)=[+1; 0; 0];
44 rate(2,:) = mu.*S;change(:,2)=[-1; 0; 0];
45 rate(3,:) = mu.*I;change(:,3)=[0; -1; 0];
46 rate(4,:) = mu.*R;change(:,4)=[0; 0; -1];
47 rate(5,:) = gamma.*I;change(:,5)=[0; -1; +1];
48 rate(6,:) = lambda.*S;change(:,6)=[-1; +1; 0];
49 rate = reshape(rate, [1 n*nbOfEvents]);
50 rand1 = rand(1,1);
51 rand2 = rand(1,1);
52 step = -log(rand2)/(sum(rate));
53 m=min(find(cumsum(rate)>=rand1*sum(rate)));
54 row = mod(m,nbOfEvents);
55 if (row == 0)
56     row = 6;
57 end
58 col = ceil(m/nbOfEvents);
59 new_value = old;
60 new_value(:,col) = old(:,col)+change(:,row);
```

Figure 3.10 – Partie du modèle SIR multi-espèces implémenté en MATLAB

Kendrick permet aux utilisateurs de pouvoir facilement spécifier les modèles équationnels grâce à la syntaxe compacte et déclarative. En évitant d'avoir des dépendances entre le méta-modèle et les modules sémantiques, nous avons séparé ce qui concerne le domaine de ce qui concerne la programmation. Cela permet aux modélisateurs de pouvoir formuler leurs modèles avec peu ou pas de compétences en programmation et facilite la modification des modèles sans avoir avoir à modifier trop de lignes.

Effectuons une étude comparative du même modèle stochastique multi-espèces exprimé en Kendrick (présenté dans la section précédente) et celui en MATLAB utilisant l'algorithme direct de Gillespie (voir la section 2.4.2). Comme on le voit dans le script

est souvent introduite dans la littérature comme Individual-Based-Model (IBM).

de la figure 3.10 [11], le modèle conceptuel (le formalisme à base d'équations) a tendance à être mélangé avec l'implémentation de l'algorithme Gillespie en MATLAB (les lignes de code de 8 à 18 représentent les événements extraits à partir des équations 3.2). Il sera vraisemblablement très difficile de pouvoir réutiliser ce modèle.

Chaque modification du modèle pour introduire de nouvelles fonctionnalités sont très coûteuses car nécessitant de très nombreuses modifications au niveau du script. Par exemple, dans le cas où l'on veut étudier un modèle multi-hôtes $SEIR$ au lieu d'un modèle SIR, il faut changer le code où les variables sont déclarées et utilisées ainsi que les événements et leurs actions qui affectent le changement sur les compartiments. L'ensemble du code est donc impacté !

Même si on utilise la bibliothèque de fonctions *GillespieSSA* mentionnée dans le Chapitre 2, il faut également spécifier tous les événements possibles et leurs actions. Avec Kendrick, il suffit de modifier la spécification du modèle (par exemple, modifier les équations du modèle), sans avoir à s'intéresser aux détails d'implémentation des simulations.

3.2.2 Difficultés liées à ce premier méta-modèle

La construction du langage Kendrick a été menée avec l'objectif de rendre cet outil plus accessible aux épidémiologistes en utilisant un formalisme principalement équationnel.

Comme nous avons abordé dans la section 2.3.2, un modèle peut se composer de plusieurs aspects de l'épidémiologie. Chacun de ses aspects vont diviser la population en plus petits groupes en fonction des caractéristiques des individus telles que âge, sexe, espèce, souches virales ou régions spatiales, etc. ou changer le cycle de transmission de la maladie étudiée en ajoutant de nouveaux états ou transitions.

Dans un premier temps, la conception du méta-modèle (présentée dans la section 3.1.1) a débutée par l'analyse des modèles épidémiologiques à base d'équations. On a donc considéré chaque variable d'équation comme un compartiment - un ensemble des individus de même état infectieux. Dans le cas où il y a plus d'un degré de division affectant la population, les variables sont indexées et vues comme des vecteurs (par exemple dans le cas du modèle multi-espèces) ou des matrices (quand il y a plus de deux degrés de décomposition).

L'implémentation des modèles dépend donc fortement de la structure des compartiments ou bien de la façon dont la population se decompose en sous-populations. Afin que la spécification des modèles soit clairement séparée de la programmation, il faut donc éviter cette dépendance.

Autr problème du méta-modèle existant : le concept de compartiment souffre d'une sémantique peu précise. Les individus dans un compartiment doivent non seulement avoir le même état infectieux mais également avoir les mêmes caractéristiques comme sexe, espèce etc. Plus précisément, un compartiment se compose donc d'un ensemble d'individus équivalents. Du point de vue mathématique, un compartiment peut être vue comme une classe d'équivalence résultant de l'application d'une relation d'équivalence

11. Le script complet du modèle se trouve dans les annexes de ce rapport

sur l'ensemble des individus (la population). Par exemple, la relation d'équivalence qui permet de diviser la population en quatre compartiments S, E, I, R est « ayant le même état infectieux ». L'état infectieux est donc un attribut des individus, qui peut prendre l'une de quatre valeurs S, E, I, R. En conséquence, le méta-modèle doit pouvoir exprimer quels sont les attributs d'un individu et quelle est la relation d'équivalence qui permet de partitionner la population en compartiments.

En observant l'exemple du modèle multi-espèces (voir la section 3.1.4), nous constatons que les deux aspects (SIR et multi-espèces) se mélangent entre eux au niveau des équations. En effet, à part des variables S, I, R qui deviennent vecteurs en raison de la décomposition de la population en plusieurs groupes d'espèces, les paramètres du modèle SIR (β, γ, μ, N) deviennent hétérogènes pour prendre en compte les hétérogénéités causées par l'aspect multi-espèces (c'est-à-dire que les espèces différentes vont avoir des taux différents pour contracter la maladie ainsi que pour être guéries).

Si on a besoin d'ajouter un autre aspect (par exemple, la distribution spatiale) ou de changer les détailles d'un aspect, il faut changer tout le modèle (équations, paramètres, compartiments). Il est donc difficile de pouvoir réutiliser les parties du modèle (les aspects constituant le modèle). De plus, les équations peuvent devenir très complexes dans le cas où le modèle se compose de plusieurs aspects, ce qui rend difficile l'écriture du modèle. Pour faciliter l'écriture, la réutilisation ainsi que l'évolution des modèles, il serait souhautable d'exprimer les aspects séparément et de construire les modèles en recombinant des aspects modulaires.

« Quelles sont les questions que l'on poser au méta-modèle ? » Puisque l'on souhaite étudier l'évolution au cours du temps de la population d'individus infectés, la première question qui nous intéresse est *[Q1] Quelle est la taille (la cardinalité) d'un compartiment donné à un instant donné ?* Le méta-modèle se basant sur la description des équations différentielles que nous avons proposé a pu répondre à cette question. Cependant, la formulation des équations différentielles du méta-modèle ne capture que la dynamique déterministe de la population. Pour interpréter les modèles du point de vue stochastique, il faut les implémenter au travers d'algorithme dédiés (par exemple, en utilisant l'algorithme de Gillespie). Le méta-modèle, par contre, ne permet pas de représenter les modèles comme des processus stochastiques. Sachant que les modèles déterministes (se formulant par un ensemble d'équations différentielles) peuvent être considérés comme une approximation de la moyenne du modèle stochastique correspondant (typiquement se formule par des chaînes de Markov) quand la taille de la population s'approche à l'infinité [89], capturer la dynamique stochastique permettrait donc d'être plus général que la dynamique déterministe. Pour cela, le méta-modèle devrait être capable de répondre à la question : *[Q2] Quelle est la probabilité pour passer d'un compartiment à un autre ?*.

Dans le chapitre suivant, nous allons présenter notre solution pour concevoir un nouveau méta-modèle, qui permet d'adresser tous ces problèmes.

3.3 Conclusion

Nous avons construit le premier prototype d'un langage de modélisation permettant de représenter des modèles épidémiologiques. Le fait qu'il s'agisse d'un langage, traduit

notre volonté d'offrir une grande capacité d'expression aux modélisateurs. L'aspect métier se concrétise par l'introduction de concepts de base issus des équations différentielles :

1. Population
2. Compartiments
3. Equations - qui représentent comment passer d'un compartiment à un autre
4. Paramètres - Fonctions temporelles qui sont évaluées à chaque itération lors des simulations

Lors de la conception du méta-modèle, nous avons séparé l'aspect métier de celui de programmation en évitant toutes les dépendances du méta-modèle sur les modules sémantiques. L'idée est de pouvoir développer autant de modules sémantiques que possible pour un seul méta-modèle. Un modèle spécifié en Kendrick peut être interprété en utilisant les simulations déterministes, stochastiques et multi-agents implémentés dans la plate-forme. Les modélisateurs peuvent également générer la version exécutable en langage de programmation général C/C++ pour ce modèle (trois versions déterministe, stochastique et multi-agents peuvent être générées en C/C++).

Nous avons fait un retour d'expérience sur la conception du langage pour voir s'il peut s'adapter à différents cas spécifiques fréquemment rencontrés en épidémiologie.

Dans le chapitre suivant, nous allons nous intéresser à séparer les préoccupations de l'épidémiologie ainsi que comment les recombiner pour construire un modèle de façon modulaire.

Chapitre 4

Séparation des Préoccupations en Epidémiologie

Sommaire

Ce chapitre est le cœur de la thèse et présente les bases formelles de la solution proposée. Nous commençons par analyser les difficultés de la modélisation en épidémiologie causées par le mélange des préoccupations. Un exemple sera utilisé comme support pour motiver notre proposition sur la séparation des préoccupations. Nous nous concentrons ensuite sur la présentation de notre solution qui vise à faciliter l'écriture et l'évolution des modèles en séparant des préoccupations.

Dans un premier temps, nous présentons le méta-modèle mathématique qui exprime la dynamique d'une population à base de compartiments. Les questions de modélisation

et les hypothèses de recherche seront également abordées. Ce méta-modèle sera utilisé pour représenter des modèles et des préoccupations de l'épidémiologie.

Dans un deuxième temps, nous donnons une définition formelle des préoccupations et décrivons certains types de dépendances entre elles. Nous proposons ensuite un moyen de les composer pour la composition des préoccupations et montrons comment elles interagissent dans un modèle. Quelques exemples à la fin du chapitre illustrerons notre approche.

Publication :

Thi Mai Anh Bui, Mikal Ziane, Serge Stinckwich, Tuong Vinh Ho, Benjamin Roche, and Nick Papoulias. 2016. Separation of concerns in epidemiological modelling. In *Companion Proceedings of the 15th International Conference on Modularity* (MODULARITY Companion 2016). ACM, New York, NY, USA, 196-200.

4.1 Problèmes de la modélisation en épidémiologie

4.1.1 Exemple pour motiver notre approche

La plupart des modèles épidémiologiaue reposent sur une structure SIR ou des variantes de celle-ci (voir section 2.2). À partir de cette structure de base, de nombreuses modifications biologiques peuvent être ajoutées pour étudier des structures plus complexe concernant les individus hôtes et leurs effets sur la propagation des maladies infectieuses (voir section **??**).

Après avoir étudié l'implémentation des modèles publiés que nous avons trouvés dans la littérature, le plus souvent écrit en MATLAB, nous avons constaté que les préoccupations se mélangent les unes avec les autres dans ces modèles et sont dispersées dans l'ensemble du programme.

Pour illustrer ce fait, nous allons prendre un exemple qui porte sur la grippe aviaire [13]. Dans cet exemple, on représente le cycle de transmission par le modèle SEIR avec démographie dans lequel l'immunité n'est pas permanente (c.a.d après un certain temps, les individus immunisés redeviennent susceptibles). La population se décompose en quatre classes : tout d'abord, des nouveaux-nés entrent dans la classe *Susceptible* (statut S) au taux μ, puis deviennent *Exposé* (statut S), c.a.d infectés mais pas encore infectieux, au taux $\lambda = \beta I/N$. Les *Exposés* deviennent *Infectieux* (I) après une période de latence donné par $1/\sigma$ et ensuite, entrent dans la classe *Rétabli* (R) après $1/\gamma$ unités de temps. Finalement, après $1/\nu$ unités de temps, les *Rétabli* redeviennent *Susceptible* (la perte de l'immunité). Les individus dans toutes les classes sont supposés mourir au taux μ. Le modèle $SEIRS$ s'exprime de façon mathématique par le système d'EDO :

$$\begin{cases} \frac{dS}{dt} &= \mu N + \nu R - \lambda S - \mu S \\ \frac{dE}{dt} &= \lambda S - \sigma E - \mu E \\ \frac{dI}{dt} &= \sigma E - \gamma I - \mu I \\ \frac{dR}{dt} &= \gamma I - \mu R - \nu R \end{cases} \tag{4.1}$$

Supposons que l'on veuille étudier la transmission de la maladie des oiseaux aux humains, entre n patchs (i.e. zones géographiques telles que pays, cités etc.). On va considérer donc deux préoccupations : multi-espèces et spatiale. Les quatre compartiments *S, E, I, R* vont être divisé chacun en deux *pour chaque espèce* et en n zones [1] résultant en $4 \times 2 \times n$ compartiments $S_{ps}, E_{ps}, I_{ps}, R_{ps}$ avec $s \in \{humains, oiseaux\}$ et $p \in [1..n]$.

La migration des individus a lieu entre les patchs voisins. L'équation de mobilité pour chaque zone p est :

$$\frac{dN_p}{dt} = \sum_{q=1}^{n} \rho_{pq} N_q - \sum_{q=1}^{n} \rho_{qp} N_p \tag{4.2}$$

où N_p est la taille de la population dans la zone p et ρ_{pq} désigne le taux d'immigration des individus de la zone q à la zone p. Le premier terme dans la partie droite de l'équation 4.2 représente l'immigration dans la zone p, tandis que le deuxième décrit l'émigration à partir de p.

1. également appelé patch

En raison des hétérogénéités qui sont issues des deux préoccupations multi-espèces et spatiale, **différents taux de transition** sont souvent considérés pour **chaque espèce dans chaque zone**. Cela peut être modélisé en remplaçant les paramètres $\mu, \sigma, \gamma, \nu, \lambda$ dans l'équation 4.1 par les matrices : $\mu_{sp}, \sigma_{sp}, \gamma_{ps}, \nu_{ps}, \lambda_{ps}$. En outre, dans une population hétérogène, la transmission de l'infection entre les différents sous-groupes doit être prise en compte, contrairement à une population considérée comme homogène où tous les individus ont un risque identique de contracter l'infection ce qui se traduit par $(\lambda = \beta I/N)$. Dans cet exemple, le taux auquel un individu susceptible de l'espèce s dans le patch p devient infecté est modélisée par $\lambda_{ps} = \sum_{i}^{humains,oiseaux} \beta_{isp} I_{pi}/N_{pi}$ avec $s \in \{humains, oiseaux\}$.

Cette formule a été construite en supposant que, l'infection peut être transmise lorsqu'un individu susceptible de l'espèce s contacte les infectieux de deux espèces dans le même patch. L'infection entre les patchs est modélisée par la mobilité des individus d'un patch à un autre. D'autres formules peuvent être considérées en modifiant la façon de faire le contact entre susceptibles et infectieux pour transmettre l'infection [69]. Le seul paramètre β est donc remplacé par une matrice de trois dimensions. Les équations 4.1 sont remplacées par le système d'EDO 4.3 pour décrire le modèle multi-espèces spatial de la grippe aviaire. L'équation 4.2 est prise en compte pour chaque compartiment S_{ps}, E_{ps}, I_{ps}, R_{ps}, la matrice à deux dimensions ρ_{pq} est remplacée par une à trois dimensions pour décrire la différence entre les groupes d'espèces dans chaque patch.

$$\begin{cases} \frac{dS_{ps}}{dt} &= \mu_{ps}N_{ps} + \nu_{ps}R_{ps} - \lambda_{ps}S_{ps} - \mu_{ps}S_{ps} \\ & \quad + \sum_{q=1}^{n} \rho_{pqs}S_{qs} - \sum_{q=1}^{n} \rho_{qps}S_{ps} \\ \frac{dE_{ps}}{dt} &= \lambda_{ps}S_{ps} - \sigma_{ps}E_{ps} - \mu_{ps}E_{ps} \\ & \quad + \sum_{q=1}^{n} \rho_{pqs}E_{qs} - \sum_{q=1}^{n} \rho_{qps}E_{ps} \\ \frac{dI_{ps}}{dt} &= \sigma_{ps}E_{ps} - \gamma_{ps}I_{ps} - \mu_{ps}I_{ps} \\ & \quad + \sum_{q=1}^{n} \rho_{pqs}I_{qs} - \sum_{q=1}^{n} \rho_{qps}I_{ps} \\ \frac{dR_{ps}}{dt} &= \gamma_{ps}I_{ps} - \mu_{ps}R_{ps} - \nu_{ps}R_{ps} \\ & \quad + \sum_{q=1}^{n} \rho_{pqs}R_{qs} - \sum_{q=1}^{n} \rho_{qps}R_{ps} \\ \lambda_{ps} &= \sum_{i}^{humans,birds} \beta_{isp} I_{pi}/N_{pi} \end{cases} \tag{4.3}$$

Le système d'équations 4.3 montre que les préoccupations SEIRS, multi-espèces, spatiales se mélangent les unes avec les autres même dans la définition des paramètres et des variables. Ceci est sans doute acceptable [2] dans un modèle mathématique de petite taille mais dans une implémentation dans un langage de programmation c'est nettement plus gênant comme nous allons le voir, ne serait que parce ces préoccupations sont de plus mêlées à de nombreux détails de bas niveau.

Pour illustrer et valider notre démarche nous utiliserons un exemple écrit en MATLAB qui est l'un des langages les plus utilisés en modélisation épidémiologique (cf. http ://www.modelinginfectiousdiseases.org).

2. Même pour un modélisateur mathématicien, il faudra un peu temps pour qu'il comprenne un modèle aussi complexe !

```
19 I=zeros(5,2);I(1,2)=10;
20 E=zeros(5,2);R=E; %...
23 S = reshape(S,[1 ns*np]); E = reshape(E,[1 ns*np]);
24 I = reshape(I,[1 ns*np]);R = reshape(R,[1 ns*np]);
25 [T,Y]=ode45(@rightSideAIModel,[0 tMax],[S E I R],options); %...
35 function res=rightSideAIModel(t, pop)
40 I=reshape(pop(2*np*ns+1:3*np*ns),[np ns]); %...
43 lambda=zeros(np,ns);
44 for p=1:np
45     for s=1:ns
46         lambda(p,s)=sum(beta(s,:,p).*I(p,:)./N(p,:));
47     end
48 end %...
51 deltaI = zeros(np,ns); deltaR = zeros(np,ns);%...
52 for s=1:ns
55    deltaI(:,s)=rho(:,:,s)*I(:,s)-sum(rho(:,:,s))'.*I(:,s); %...
57 end
58 dSdt = mu.*N + nu.*R - lambda.*S - mu.*S + deltaS;
59 dEdt = lambda.*S - sigma.*E - mu.*E + deltaE;
60 dIdt = sigma.*E - gamma.*I - mu.*I + deltaI;
61 dRdt = gamma.*I - mu.*R - nu.*R + deltaR;%...
```

Figure 4.1 – Partie du modèle de grippe aviaire en Matlab

La figure 4.1 montre une partie du modèle écrit en MATLAB [3]. Bien que MATLAB permette de spécifier les EDO en utilisant une syntaxe compacte (voir les lignes de 58 à 61), on peut constater deux problèmes : (1) La *dispersion du code* traitant une préoccupation dans différent emplacement du programme (2) L'*enchevêtrement du code* des différentes préoccupations : les préoccupations spatiales et multi-espèces sont croisées avec la préoccupation SEIRS dans certains fragments de code. C'est le cas notamment pour les déclarations de paramètres et de variables (voir les lignes 19, 20, les variables S, E, I, R sont des matrices de deux dimensions, une dimension représente l'espèce et l'autre la zone géographique). En outre, dans la fonction `rightSideAIModel` qui calcule les parties à droite des EDOs, ces préoccupations sont également entremêlées lors du calcul du paramètre λ (la préoccupation SEIRS) (lignes 44-48) et les expressions de la mobilité (la préoccupation spatiale) (lignes 52-57). Le traitement de la préoccupation SEIRS, par ailleurs, est dispersée dans l'ensemble du programme.

La dispersion et l'entrecroisement des préoccupations rend plus difficile leur réutilisation mais de façon encore plus manifeste, sans doute, modifier certains détails concernant une préoccupation, induit des modifications à plusieurs endroits où cette préoccupation est mêlée avec d'autres.

Impact d'un changement

Supposons, par exemple, que l'on veuille distinguer différentes souches de pathogène de la grippe aviaire parce que certaines sont plus virulentes ou résistent mieux à tel ou tel traitement. Nous allons donc introduire une préoccupation multi-souches dans le modèle ci-dessus. Cette préoccupation va diviser la classe I en plusieurs petites classes pour représenter l'infection causée par différentes souches de pathogène.

Distinguons donc deux souches du pathogène : I_1 et I_2. On doit alors remplacer I par

3. Voir le modèle complet en annexe à la fin de ce rapport

```
19  I1=zeros(5,2) ;I2=zeros(5,2) ;I1(1,2)=10 ;I2(1,2)=10
20  E=zeros(5,2);R=E; %...
23  S = reshape(S,[1 ns*np]); E = reshape(E,[1 ns*np]);
24  I1 = reshape(I1,[1 ns*np]) ;
    I2 = reshape(I2,[1 ns*np]) ;
    R = reshape(R,[1 ns*np]);
25  [T,Y]=ode45(@rightSideAIModel,[0 tMax], [S E I1 I2 R] ,options); %...
35  function res=rightSideAIModel(t, pop)
40  I1=reshape(pop(2*np*ns+1 :3*np*ns),[np ns]) ;
    I2=reshape(pop(3*np*ns+1 :4*np*ns),[np ns]) ;  %...
43  lambda=zeros(np,ns);
44  for p=1:np
45      for s=1:ns
46          lambda(p,s)=sum( ((beta1(s, :,p).*I1(p, :))
            +(beta2(s, :,p).*I2(p, :))) ./N(p,:));
47      end
48  end %...
51  deltaI1 = zeros(np,ns) ;
    deltaI2 = zeros(np,ns) ;
    deltaR = zeros(np,ns); %...
52  for s=1:ns
55      deltaI1( :,s)=rho( :, :,s)*I1( :,s) - sum(rho( :, :,s))'.*I1( :,s) ;
        deltaI2( :,s)=rho( :, :,s)*I2( :,s) - sum(rho( :, :,s))'.*I2( :,s) ;
57  end
58  dSdt = mu.*N + nu.*R - lambda.*S - mu.*S + deltaS;
59  dEdt = lambda.*S - sigma1.*E - sigma2.*E - mu.*E + deltaE;
60  dI1dt = sigma1.*E - mu.*I1 - gamma1.*I1 + deltaI1 ;
    dI2dt = sigma2.*E - gamma2.*I2 - mu.*I2 + deltaI2 ;
61  dRdt = gamma1.*I1 + gamma2.*I2 - mu.*R - nu.*R + deltaR;%...
```

Figure 4.2 – Les changements affectant le code du modèle lors de la modification du cycle de transmission.

I_1 et I_2 dans le code du modèle. Les lignes $19, 24, 25, 40, 46, 51, 55, 59, 60et61$ dans la figure 4.1 sont modifiées comme indiqué (en jaune) sur la figure 4.2.

Dans cet exemple, nous avons considéré le modèle déterministe de la grippe aviaire. Le problème serait similaire pour un modèle stochastique.

Pour réduire l'impact des changements et faciliter la réutilisation, nous voulons donc séparer les préoccupations autant que possible. Pour ce faire on peut tout d'abord, lorsque le code est monolithique comme dans notre exemple, tenter d'effectuer une découpe fonctionnelle. On essaye donc de le diviser en modules, représentant chacun une fonctionnalité particulière. Toutefois une préoccupation est justement, par hypothèse, un aspect de la fonctionnalité qu'il n'est pas facile de découper en modules (fonctions ou classes).

Dans le modèle MATLAB de notre exemple, après consultation d'experts, une seule fonction a pu être définie : `function rightSideAIModel`. Cependant, on n'a pas pu localiser chaque préoccupation du modèle dans une fonction. Il aurait été possible de séparer la visualisation mais selon les experts consultés ce n'est pas habituel. De même, bien que MATLAB permette de programmer avec des objets, ce n'est pas utilisées par les modélisateurs.

4.1.2 Paradigme de séparation des préoccupations

Nous nous trouvons dans la même situation que dans le développement des logiciels en général, où les préoccupations entremêlées et dispersées gênent l'évolution et la maintenance [72, 64].

Hürch et Lopes [64] donnent une liste de problèmes [4] concernant un logiciel dans lequel l'algorithme principal se mélange avec le code de synchronisation :

1. Programmer du code dans lequel les préoccupations sont entrelacées est difficile et complexe car elles doivent alors être gérées en même temps et au même niveau.
2. Le code entrelacé est difficile à comprendre parce que l'implémentation des préoccupations manque d'abstractions.
3. Le code entrelacé est difficile à maintenir et modifier parce que toutes les préoccupations sont fortement couplées.

Hürch et Lopes proposent alors une approche générale qui distingue entre deux niveaux de la séparation des préoccupations : le niveau conceptuel et celui de l'implémentation. Au niveau conceptuel, ils distinguent deux enjeux :

1. Produire une définition claire d'une préoccupation.
2. Assurer que les concepts individuels sont primitifs.

Au niveau de l'implémentation, des blocs du code distincts doivent gérer des préoccupations différentes et permettre un faible couplage entre elles [64].

Dans l'ensemble, mis à part l'exigence que les préoccupations soient primitives ce qui nous parait secondaire, nous sommes d'accord avec Hürch et Lopes notamment lorsqu'ils plaident en faveur de la séparation des préoccupations aux deux niveaux, conceptuel et d'implémentation.

4. Ces problèmes ont d'ailleurs été reconnus par beaucoup d'autres chercheurs (cf. par exemple [4, 71, 78]).

Il nous semble cependant que Hürch et Lopes ont sous-estimé la difficulté au niveau conceptuel en se focalisant sur le niveau d'implémentation : « separation of concerns is often practiced at the conceptual level but not at the implementation level... » [64]. Au moins, dans la modélisation de l'épidémiologie, la séparation des préoccupations n'est pas réalisée au niveau des EDO comme nous l'avons vu sur le système d'équations (4.3).

Diverses solutions ont été développées pour réaliser la séparation des préoccupations dans les systèmes informatiques mais comme nous allons le voir elle se focalisent sur le niveau de l'implémentation. Hürch et Lopes ont groupé les techniques de la séparation des préoccupations dans trois catégories principales : (1) La méta-programmation (2) La programmation orientée-patron et (3) La programmation avec les filtres de composition [64]. Ces techniques sont différentes de la découpe fonctionnelle (qui divise le système en modules (procédures, fonctions, classes/objets, packages etc.), représentant chacun une fonctionnalité particulière) des langages de programmation généraux. La découpe fonctionnelle ne peut pas résoudre les problèmes des préoccupations qui se recoupent (comme les exigences non-fonctionnelles des systèmes informatiques). Les techniques mentionnées, par contre, proposent des mécanismes plus génériques qui permettent de séparer des préoccupations au niveau plus abstrait.

La méta-programmation [71] manipule le comportement et l'implémentation d'un langage de programmation en ajoutant des méta-objets qui contrôlent les messages d'envoie et reçus des objets. La séparation des préoccupations est atteinte au méta-niveau en réifiant messages d'envoie et reçus des objets et en utilisant méta-objets pour définir ces préoccupations. La composition des préoccupations consiste à composer des méta-objets. Par contre, il n'y a pas de méthodologies générales pour la composition des méta-objets. Celle-là va dépendre ensuite des programmeurs lorsqu'ils font face à un problème particulier.

La programmation orientée-patron [78] propose un modèle de programmation se basant sur l'ensemble de patrons qui est classifié dans certaines catégories pour capturer les différentes abstractions de la programmation (par exemple, le patron de synchronisation pour définir des schèmes de synchronisation entre les objets etc.), dont chacune adresse une certaine sorte de préoccupations. Un graphe de classes est exigé pour définir la structure détaillée des objets auxquels appliquent les patrons et pour générer les programmes orienté-objet. La composition des patrons (pour composer des préoccupations) est réalisée par la définition d'un autre graphe de classe pour la structure composée.

La programmation avec des filtres de composition [4] étend le modèle orienté-objet par l'ajout de filtres de composition des objets. Un filtre (une instance d'une classe de filtre) prend en charge le contrôle et la manipulation des messages envoyés et reçus par les objets. La séparation des préoccupation est réalisée par la définition d'une classe de filtre pour chaque préoccupation, qui va contrôler et manipuler des messages des objets associés à la préoccupation. La composition des filtres (pour composer des préoccupations) est réalisée par un autre filtre qui définit l'ordre séquentiel des filtres à composer.

La programmation orientée aspects, qui a été présentée par Kiczales [72], vise à séparer dans un module les préoccupations qui se recoupent avec d'autres. D'après Kiczales, une préoccupation est soit : (1) un composant si elle peut être clairement encapsulée

dans un objet ou un module, ce qui est donc, par définition, une unité fonctionnelle d'un système (2) un aspect si elle ne peut pas être clairement encapsulée dans un composant, ce qui correspond donc aux exigences non-fonctionnelles d'un système. Par rapport à l'orienté objet qui ne permet que de séparer les composants, la programmation orientée aspect aide à séparer clairement les aspects et les composants les uns des autres en offrant des mécanismes qui permettent de les abstraire et de les composer pour obtenir le système général.

Ces techniques visent donc la séparation des préoccupations des systèmes informatiques au niveau de l'implémentation. Elles ne dispensent aucunement de la séparation des préoccupation au niveau conceptuel.

Il faut aussi distinguer les préoccupations qu'on pourrait qualifier de techniques comme la synchronisation ou la journalisation qui sont applicables à de nombreux domaines applicatifs, qu'elles croisent (*cross-cut*), des préoccupations propre à un domaine applicatif donné. Les solutions au niveau de l'implémentation peuvent éventuellement convenir pour des préoccupations techniques qui seront de toute façon prises en charge a priori par des développeurs.

Au contraire, quand on vise un domaine applicatif précis, en s'appuyant sur la sémantique de ce domaine il devient envisageable de définir rigoureusement les préoccupations de ce domaine ainsi qu'un ou plusieurs opérateurs pour les combiner.

On pourrait certes, pour implémenter ces opérateurs, envisager de s'appuyer sur les techniques de séparation au niveau de l'implémentation mais ceci nous parait constituer une fausse piste. D'une part les mécanismes pour combiner les préoccupations au niveau de l'implémentation n'ont rien à voir avec la sémantique du domaine visé. De plus, ces mécanismes demandent des compétences de programmation spécialisées aussi bien pour définir les préoccupations que pour les combiner.

La séparation des préoccupations au niveau d'un modèle. La tendance actuelle en génie logiciel consiste à gérer des programmes au niveau de leurs concepts, en focalisant sur le niveau d'abstraction où les programmes sont conçus et développés. L'ingénierie du logiciel s'oriente donc aujourd'hui vers l'ingénierie dirigée par les modèles (IDM) dont le principle est « tout est modèle » [15]. L'IDM vise de manière plus radicale que pouvaient l'être les approches des objets et des aspects, à fournir un grand nombre de modèles pour exprimer séparément chacune des préoccupations du système informatique. Un modèle dans l'IDM est une abstraction d'un système, modélisé sous forme d'un ensemble de faits qui permettent de répondre à certaines questions posées au système qu'il représente [21]. Un modèle doit être suffisant pour pouvoir répondre aux questions exactement de la même façon que le système aurait répondu lui-même. La notion du modèle de l'IDM fait explicitement référence à la notion de langage bien défini. Autrement dit, tous les modèles qui représentent un même système sont exprimés par un langage de modélisation qui doit être clairement défini. De manière naturelle, la définition d'un langage a pris la forme d'un modèle que l'on appelle *méta-modèle*. L'UML [5] est un méta-modèle la plus utilisé dans la scène du développement des logiciels, mais pas le seul. Pour éviter le risque d'avoir une variété de différents méta-modèles non-compatibles, on a besoin donc un langage qui permet de définir des méta-modèles, i.e. un méta-méta-modèle.

5. Unified Modeling Language : http ://www.omg.org/spec/UML

C'est sur ses principes de base que s'appuie l'OMG [6] pour définir l'ensemble de ses standards, en particulier l'approche MDA (Model Driven Architecture) [85] pour promulguer de bonnes pratiques de modélisation et explorer pleinement les avantages des modèles. Le principe du MDA consiste à s'appuyer sur certains standards tels que UML et MOF [7] pour décrire séparément des modèles pour les différentes phases du cycle de développement d'une application. Cette approche prévilégière l'élaboration de trois modèles : (1) modèle d'exigence (CIM - *Computation Independent Model*) dans lequel aucune considération informatique n'apparaît (2) modèle d'analyse et de conception (PIM - *Platform Independent Model*) et (3) modèle de codage (PSM - *Platform Specific Model*). Le PIM permet de se focaliser sur la spécification de la fonctionnalité du système, en se cachant tous les détailles spécifiques de la plateforme dans laquelle sera implémenté le système. Le passage de PIM à PSM fait intervenir des mécanismes de transformation de modèle.

L'approche MDA propose donc de modulariser un système informatique sous forme de plusieurs modèles dont chacun représente un niveau d'abstraction. Dans chaque niveau du MDA, on désire faire la séparation des préoccupations au niveau de modèle pour faciliter la capacité de réutilisation, l'évolution ainsi qu'augmenter la productivité des systèmes informatiques. Il exige donc des mécanismes qui permettent de diviser un modèle d'un système en plusieurs sous-modèles dont chacun représente une préoccupation spécifique et puis de les composer pour construire le système complète.

Il existe plusieurs approches qui permettent de réaliser la séparation et la composition des modèles. L'approche la plus utilisée se base sur les profils de l'UML qui permettent de personnaliser/étendre l'UML pour modéliser tout domaine spécifique [45, 76, 110]. Un système est représenté par un ensemble de domaines conceptuelles dont chacun définit une famille de modèles (tous ces modèles sont conformés à un même méta-modèle). Le standard de modélisation UML permet de définir des méta-modèles, par contre, il manque des mécanismes qui permettent de composer des méta-modèles différents (donc composer des domaines différentes). On propose donc des extensions d'UML (qui inclut de nouveaux opérateurs et de nouvelles règles) pour faire la composition des méta-modèles.

Une autre approche vise à appliquer des techniques de l'AOSD [8] pour faire la séparation des préoccupations au niveau de modèle [52, 73, 99, 112]. Typiquement, la modélisation orientée aspects est appliqué au niveau de PIM et PSM. Au niveau de PIM, on divise le système en plusieurs modèles : un modèle principal qui encapsule les fonctionnalités de l'application et un ensemble des modèles d'aspect dont chacun décrit un aspect. L'intégration des modèles d'aspect avec le modèle principal se fait par la définition d'un ensemble des actions de composition [115]. Le passage de PIM à PSM se fait par transformer séparément chaque modèle.

Certaines approches visent à étendre des méta-modèles existants, telles que les facettes EMF [114], KerMeta [66], pour faire la composition des modèles. Les facettes EMF fournissent un mécanisme d'extension non-intrusive pour ajouter des attributs, des références ou des opérations supplémentaires à un modèle sans modifier ce modèle ou son méta-modèle. Alors que KerMeta fournit des fonctionnalités pour inclure

6. The Object Management Group (OMG) : http ://www.omg.org
7. Meta-Object Facility : http ://www.omg.org/spec/MOF/
8. Aspect Oriented Software Development - C'est une approche qui étend l'AOP pour pouvoir être appliquée à toutes les étapes du développement des logiciels

des préoccupations dans des méta-modèles existants en utilisant la méta-modélisation orientée aspects. Il permet également de définir des opérations d'exécution au méta-niveau pour les éléments d'un méta-modèle, ce qui offre une flexibilité pour manipuler tout modèle qui conforme à ce méta-modèle. Il existe d'autres approches qui proposent des moyens pour combiner plusieurs méta-modèles dans un espace de modélisation unique. Certaines de ces techniques utilisent plusieurs opérateurs (par exemple, fusion, équivalence etc.) pour assembler de différents éléments de plusieurs méta-modèles dans un seul méta-modèle composite [49].

Dans le cadre de cette thèse, nous, par contre, n'avons pas utilisé l'approche de l'IDM. D'une part, nous n'avons utilisé ni MOF ni EMF sauf UML pour la première version du méta-modèle (voir le chapitre 3). Il paraît donc difficile de pouvoir appliquer les techniques du MDA. D'autre part, le niveau conceptuel auquel nous visons est plus abstrait que le niveau de modèle ou de méta-modèle de l'IDM. En effet, notre objectif est de pouvoir définir des modèles ainsi que des préoccupations épidémiologiques de façon mathématique. L'utilisation d'une technique de séparation des préoccupations de type IDM ne dispense donc pas de trouver un opérateur au niveau mathématique.

4.1.3 Les problèmes que nous voulons résoudre

Cette thèse vise à séparer les préoccupations épidémiologiques des autres aspects (la simulation, la visualisation ...) mais surtout de les séparer les uns des autres. En fait les deux enjeux se rejoignent car pour séparer les préoccupations épidémiologiques les unes des autres nous les formaliseront ce qui prépare naturellement leur séparation des autres aspects.

Notre problème principal consistera donc à déterminer si on peut donner une structure aux modèles épidémiologiques. Au minimum on cherchera un opérateur qui soit une loi de composition interne sur l'ensemble des aspects. De préférence on aimerait que l'opérateur soit associatif et commutatif pour ne pas avoir à se préoccuper des parenthèses ni de l'ordre des aspects dans un modèle. Un problème secondaire consistera à séparer l'expression de cette structure des détails d'implémentation.

Une difficulté nous attend : les aspects de la modélisation épidémiologique paraissent a priori difficiles à séparer du statut infectieux. Considérons par exemple un modèle incluant la distribution spatiale de la population dans certaines régions. Si cette distribution était complètement indépendante du statut infectieux des individus, elle n'aurait guère d'intérêt : autant étudier la population dans son ensemble plutôt que de la découper inutilement en régions. Si un modèle inclut un aspect, c'est parce que ce dernier présente, au moins potentiellement, des hétérogénéités par rapport au statut infectieux. C'est donc qu'il y a des interdépendances entre le statut infectieux et cet aspect, ce qui a priori rend difficile leur séparation.

Notre troisième problème consistera donc à permettre l'expression de ces hétérogénéités tout en introduisant un minimum de dépendances entres les préoccupations.

4.2 Le méta-modèle mathématique de la dynamique des populations en compartiments

Dans cette section, nous allons expliquer les relations entre les concepts de notre méta-modèle mathématique qui va permettre de définir ce qui est un modèle, ce qui est une préoccupation et comment les préoccupations peuvent être combinées ensemble pour construire des modèles. Cette définition mathématique est censée être aussi générale que possible, contrairement à sa mise en œuvre en Kendrick qui peut introduire certaines limitations (voir le chapitre 5). Par exemple, le méta-modèle mathématique se repose sur le concept mathématique de la relation d'équivalence pour décomposer la population étudiée (un ensemble d'individus). En revanche, dans l'implémentation actuelle de Kendrick, ce concept est limité au cas le plus typique : une expression booléenne sur les attributs des individus en utilisant un ensemble limité d'opérateurs booléens. La focalisation sur les concepts mathématiques nous permet de séparer plus facilement les détails de l'implémentation des modèles et de définir ensuite plus facilement des langages métiers comme Kendrick.

Tout d'abord, nous avons défini des modèles avec aussi peu d'hypothèses que possible en faisant référence aux approches les plus utilisées en épidémiologie : modèles en compartiments et processus stochastique (typiquement les chaînes de Markov à temps continu - CTMCs). Ainsi, ces modèles peuvent être décrits comme des automates stochastiques dont les états sont les compartiments et les transitions sont étiquetées par les taux dans la matrice des taux de transition des chaînes de Markov à temps continu (CTMC par la suite). Nous allons rappeler ce qui est une chaîne de Markov et comment on peut formaliser un modèle compartimental de l'épidémiologie sous forme d'une CTMC.

Chaînes de Markov à temps continu Étant donné un processus stochastique représenté sous forme d'un ensemble de variables aléatoires :

$$\{X(t)\}_{t=0}^{\infty}$$

Une chaîne de Markov est tout d'abord un processus stochastique, dont $X(t)$ est une variable aléatoire discrète [9] : $X(t) \in \{0, 1, 2, ...\}$. La fonction de probabilité associée au processus :

$$p_i(t) = Prob\{X(t) = i\}$$

Supposez que l'espace d'état de X est limitée, $i = 0, 1, 2, ...N$ et $\sum_{i=0}^{N} p_i(t) = 1$. $p(t) = (p_0(t), p_1(t), ...p_N(t))^T$ dénote le vecteur de probabilité associé à $X(t)$.

Le processus est ensuite supposé être satisfait par la propriété de Markov :

$$Prob\{X(t_{n+1})|X(t_n), X(t_{n-1}), ..., X(t_1), X(t_0)\} = Prob\{X(t_{n+1})|X(t_n)\}$$

pour toute suite de nombres réels satisfaisant $0 \leq t_0 < t_1 < ... < t_n < t_{n+1}$. C'est-à-dire que l'état suivant du processus ne dépend que de l'état courant, et ne dépend pas du passé.

9. le terme "chaîne" (C) veut dire que les variables aléatoires du processus sont discrètes

Le processus est supposé être homogène dans le temps si la probabilité d'une transition entre deux états à deux instants du temps donnés ne dépend que de la différence entre deux instants :

$$Prob\{X(t+\Delta t)|X(t)\} = Prob\{X(t'+\Delta t)|X(t')\}, \forall \Delta t > 0$$

La formulation des modèles CTMC en épidémiologie Etant donné le modèle SIR de l'épidémiologie qui est représenté par le système d'EDO (2.1). On suppose que $\{\mathcal{S}(t), \mathcal{I}(t), \mathcal{R}(t)\}$ est l'ensemble des variables aléatoires discrètes qui représentent le nombre de susceptibles, infectieux et rétablis à un temps t, respectivement. Car $\mathcal{R}(t) = N - \mathcal{S}(t) - \mathcal{I}(t)$, N est la taille totale de la population, il est possible de considérer le processus stochastique du modèle SIR comme le processus bivarié $\{\mathcal{S}(t), \mathcal{I}(t)\}_{t=0}^{\infty}$ dont la fonction de probabilité conjointe donnée par :

$$p_{s,i}(t) = Prob\{\mathcal{S}(t) = s, \mathcal{I}(t) = i\}$$

Le processus stochastique du modèle SIR satisfait à la propriété de Markov (**première hypothèse**).

Nous allons ensuite définir les probabilités de transition du modèle. Ces probabilités sont définies pour un intervalle de temps Δt suffisamment petit. La probabilité d'une transition de l'état $(\mathcal{S}(t) = s, \mathcal{I}(t) = i)$ à l'état $(\mathcal{S}(t+\Delta t) = s+k, \mathcal{I}(t+\Delta t) = i+j)$ est :

$$p_{(s+k,i+j),(s,i)}(t+\Delta t, t) = Prob\{(\mathcal{S}(t+\Delta t), \mathcal{I}(t+\Delta t)) = (s+k, i+j)|(\mathcal{S}(t), \mathcal{I}(t)) = (s,i)\}$$

On suppose que le processus est homogène dans le temps (**deuxième hypothèse**), les probabilités de transition donc ne dépend pas du temps. On peut écrire :

$$p_{(s+k,i+j),(s,i)}(\Delta t) = Prob\{(\Delta\mathcal{S}, \Delta\mathcal{I}) = (k,j)|(\mathcal{S}(t), \mathcal{I}(t)) = (s,i)\}$$

où $\Delta\mathcal{S} = \mathcal{S}(t+\Delta t) - \mathcal{S}(t)$. C'est pareil pour $\Delta\mathcal{I}$.

Pour réduire le nombre de transitions dans une intervalle de temps Δt, on fait la **troisième hypothèse** : l'intervalle de temps Δt est supposée être suffisamment petite de sorte qu'il y ait au plus un changement d'état qui se produit pendant cette durée. Par conséquent, il y a seulement trois transitions suivantes :

$$(s,i) \to (s-1, i+1); (s,i) \to (s, i-1); (s,i) \to (s,i)$$

Ces deux premières transitions sont correspondantes aux deux événements : $S \to I$ et $I \to R$. Les probabilités de transition sont définies utilisant les taux ($\times\Delta t$) dans le modèle SIR déterministe [10] :

$$p_{(s+k,i+j),(s,i)}(\Delta t) = \begin{cases} \beta si/N\Delta t + o(\Delta t), & (k,j) = (-1,1) \\ \gamma i\Delta t + o(\Delta t), & (k,j) = (0,-1) \\ 1 - \beta si/N\Delta t - \gamma i\Delta t + o(\Delta t), & (k,j) = (0,0) \\ o(\Delta t), & tout\, autre\, cas \end{cases} \tag{4.4}$$

L'équation (4.4) explique la relation entre les taux de transfert dans le modèle déterministe et les probabilités de transition dans le modèle CTMC. À partir de la matrice de

10. Le terme $o(\Delta t)$ est ajouté pour expliquer le temps continu de CTMC $\lim_{t\to\infty}(o(\Delta t)/\Delta t) = 0$

probabilité de transition (formulée par l'équation (4.4)), on peut déterminer la matrice des taux de transition, dénotée Q (ou la matrice infinitésimale de la CTMC) à l'aide des équations différentielles de Kolmogorov [7] :

$$q_{(s+k,i+j),(s,i)}(\Delta t) = \begin{cases} \beta si/N, & (k,j) = (-1,1) \\ \gamma i, & (k,j) = (0,-1) \\ -\beta si/N - \gamma i, & (k,j) = (0,0) \\ 0, & tout\, autre\, cas \end{cases} \tag{4.5}$$

Q est une matrice de $2N \times 2N$. On peut interpréter le modèle stochastique formulé comme une CTMC en se basant sur le fait que le temps pour passer d'un état à un autre a une distribution exponentielle, ce qui résulte de la propriété de Markov. Cela nous conduit à l'algorithme de Gillespie qui a été présenté dans le chapitre 2.

On a formulé le modèle SIR comme une CTMC, en considérant l'état du modèle est un ensemble $\{\mathcal{S}(t), \mathcal{I}(t), \mathcal{R}(t)\}$. Il en résulte la matrice Q de taille $2N \times 2N$. La chaîne de Markov du modèle SIR peut être formulée de manière différente en considérant un autre ensemble de variables $X = \{\mathcal{X}_1, \mathcal{X}_2, ..., \mathcal{X}_N\}$ dont $\mathcal{X}_i \in \{S, I, R\}$, qui représente l'état de chaque individu dans la population de taille N. Il est facile de trouver la relation entre deux ensembles de variables, comme $\mathcal{S}(t) = \sum_{i=1}^{N} (\mathcal{X}_i = S)$. La fonction de probabilité associé au processus :

$$p_X(t) = Prob\{X = \{\mathcal{X}_1, \mathcal{X}_2, ..., \mathcal{X}_N\}\}$$

Le taux de transfert entre les différents états du processus est décrit par la matrice Q dont la dimension est $3^N \times 3^N$. Etant donné $q_{X,Y}$ un élément de la matrice Q qui représente le taux auquel le processus passe de l'état X à l'état Y :

- $q_{X,Y} = \gamma$ lorsqu'il y a un individu dans l'état X qui passe de I à R dans l'état Y.
- $q_{X,Y} = \beta I/N = \lambda$ lorsqu'il y a un individu dans l'état X qui passe de S à I dans l'état Y, I est le nombre des individus qui ont l'état I dans X.
- $q_{X,X} = -\sum_{Y \neq X} q_{X,Y}$.
- $q_{X,Y} = 0$, tout autre cas.

Ce processus prend les mêmes hypothèses que nous avons abordées ci-dessus. Le changement d'état d'un individu i peut être vu comme une CTMC dont la matrice Q_i est :

$$Q_i = \begin{pmatrix} -\lambda & \lambda & 0 \\ 0 & -\gamma & \gamma \\ 0 & 0 & 0 \end{pmatrix}$$

En vue de la troisième hypothèse, dans un intervalle suffisamment petit, il n'y a qu'un individu qui change son état. La CTMC $X = \{\mathcal{X}_1, \mathcal{X}_2, ..., \mathcal{X}_N\}$ peut être vue comme une combinaison de N CTMCs indépendantes $\mathcal{X}_i$. D'après le travail de Plateau et al. (que nous allons continuer à utiliser dans la suite de ce chapitre) [96], la matrice Q est une somme tensorielle des Q_i :

$$Q = \bigoplus_{i=1}^{N} Q_i$$

Pour interpréter cette CTMC, on utilise également le fait que le temps pour passer d'un état à un autre a une distribution exponentielle. La différence est que dans ce cas-là, on fait au niveau des individus, ce qui nous conduit à l'algorithme IBM qui a été introduit

dans le chapitre 2. Ces deux versions de CTMC nous donnent des résultats qui ne sont pas statistiquement différents ce que nous allons montrer dans le chapitre 6.

La matrice des taux de transition individuelle Q_i peut être utilisée pour spécifier le modèle. Un modèle en épidémiologie va être, donc, défini comme un *automate stochastique* dont les états sont les compartiments (étiquetés par l'ensemble d'état d'un individu) et les transitions sont étiquetées par les taux dans la matrice des taux de transition individuelle Q_i (qui représente le taux de changement de l'état d'un individu). Par exemple, l'automate stochastique qui représente le modèle SIR est dans la figure 4.3.

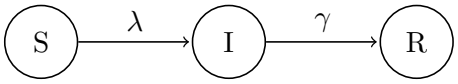


Figure 4.3 – L'automate stochastique du modèle SIR

Nous avons postulé ensuite qu'une préoccupation peut être combinée avec d'autres ou au moins avec des modèles. Il est apparu que de nombreuses préoccupations peuvent être également décrites, comme des modèles, par des automates stochastiques. La raison est que la plupart des préoccupations font décomposer la population étudiée afin d'introduire une certaine hétérogénéité chez les individus hôtes, comme on verra dans l'exemple ci-dessous.

Considérez une distribution spatiale des individus. La population se décompose en deux villes : Paris et Lyon. La figure 4.4 représente deux automates stochastiques séparés : l'automate du modèle SIR à gauche et celui de la préoccupation spatiale à droite dont les transitions sont étiquetées par les taux auxquels un individu se déplace d'une ville à une autre.

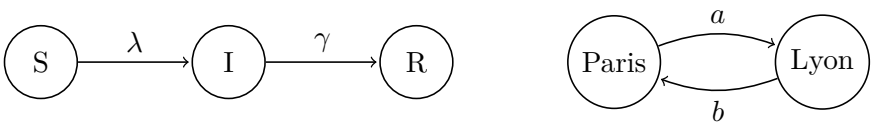


Figure 4.4 – Deux automates stochastiques : SIR (à gauche) et une préoccupation spatiale (à droite).

Lorsque la préoccupation spatiale est combinée au modèle SIR, elle divise chacun des compartiments S, I, R en fonction de la structure spatiale (Figure 4.5). Par exemple, le compartiment S se divise en S_{Pr} et S_{Ly}, qui dénotent les individus susceptibles qui se trouvent à $Paris$ et $Lyon$, respectivement. C'est pareil pour I et R.

Les préoccupations sont des structures mathématiques qui incluent des variables. Il faut noter que ces variables peuvent dénoter des fonctions. Il est parfois nécessaire de distinguer les préoccupations dont les variables sont libres, de celles partiellement instanciées dont certaines variables sont liées aux valeurs, et de celles dont toutes les variables sont liées aux valeurs.

Les modèles complètement instanciés (« ground models ») et donc préoccupations complètement instanciées (« ground concerns »), doivent donner deux sortes d'information en fonction du point de vue. Du point de vue déterministe, ils doivent donner la cardinalité d'un compartiment donné à un instant donné. Du point de vue stochastique, ils

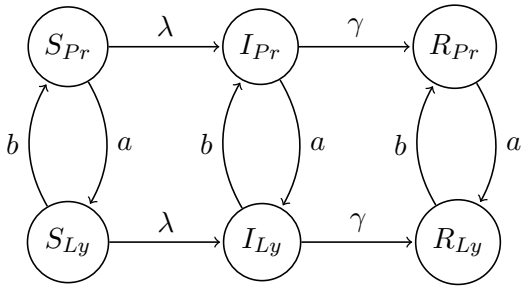


Figure 4.5 – La préoccupation spatiale Paris-Lyon a été combinée avec le modèle SIR, résultant d'un automate stochastique de 3×2 états.

doivent donner la probabilité à laquelle un certain individu passe d'un compartiment donné à un autre.

Parce que les compartiments sont considérés comme une partition de la population (un ensemble d'individus), il était naturel de les définir comme des classes d'équivalences résultantes de l'application d'une relation d'équivalence sur la population. La décomposition ultérieure de la population peut être capturée par une relation d'équivalence additionnelle qui peut être combinée avec celle d'un autre modèle ou d'une autre préoccupation en utilisant un opérateur logique `et` (`and`). L'état d'un individu dans une certaine préoccupation peut être la région où se trouve l'individu, son sexe, son espèce etc.

Ayant observé que les préoccupations sont essentiellement des automates stochastiques et que les modèles qui les composent sont aussi des automates nous avons cherché si un opérateur n'avaient pas déjà été défini pour combiner ces automates. Or c'était bien le cas quoi que que dans un domaine complètement différent de l'épidémiologie à savoir les performances des systèmes distribués. En effet Brigitte Plateau a en effet proposé de combiner ces automates via un opérateur tensoriel (la somme tensorielle $\oplus$ dans le cas des CTMC) [95, 96].

Un opérateur tensoriel donne une structure de monoïde aux automates : il est associatif et possède un élément neutre, l'automate vide. Nous pouvons donc proposer une définition très générale des aspects et des modèles (des combinaisons d'aspects) en leur donnant les propriétés voulues pour la séparation des préoccupations et ce d'autant plus que l'opérateur tensoriel peut être rendu commutatif.

Le méta-modèle mathématique, qui donne une définition formelle d'un modèle en épidémiologie, se compose donc des abstractions suivantes : *Population, Attribut, Relation d'equivalence, Compartiment, Paramètre, Transition.*

La population P , l'ensemble des individus, est implicite et non mentionnée dans les modèles et les préoccupations. Un modèle est alors défini comme :

Définition 4.1. $Modèle = \{\mathcal{A}, \mathcal{R}, Prms, Tr\}$

$\mathcal{A}$ est l'ensemble des attributs du modèle, un attribut étant une application de P vers un domaine quelconque. $\mathcal{R}$ est une relation d'équivalence qui est réalisée sur la population P. $\mathcal{C} = P/\mathcal{R}$ est l'ensemble quotient c'est-à-dire l'ensemble des classes d'équivalences ou encore des *compartiments*. $Prms$ est un ensemble de paramètres (fonctions temporelles)

du modèle.

$Tr \in \mathcal{C} \times \mathcal{C} \to \mathbb{R}^+$ indique la matrice des taux de transition de la chaîne de Markov individuelle dont chaque élément dans Tr (sauf ceux qui sont sur la diagonale principale) représente le taux auquel un individu peut passer d'un compartiment à un autre.

$$Tr = (q_{ij})_{n \times n}$$

où n est le nombre de transitions. Les éléments sur la diagonale principale sont définis de façon suivante :

$$q_{ii} = -\sum_{j \neq i} q_{ij}$$

Le modèle est complètement instancié dès que l'on assigne les valeurs initiales à tous ses compartiments et paramètres.

Nous définissons plus bas **l'opérateur** $\bowtie$ qui combine les modèles (et de façon plus générale les préoccupations) et applique notamment la somme tensorielle sur les automates c'est-à-dire sur leurs matrices de transition.

Le modèle vide Θ : L'élément neutre du monoïde est le modèle vide (noté Θ) qui est définit comme suit :

$$\Theta = \{\mathcal{A}_o, \mathcal{R}_o, Prms_o, Tr_o\} \tag{4.6}$$

où : $\mathcal{A}_o = \emptyset$, $\mathcal{R}_o = true$, $Prms_o = \{N\}$ où N représente la taille de la population P, $Tr_o = 0$.

Exemple : Définition du modèle SIR de l'épidémiologie Le modèle SIR de l'épidémiologie est défini en utilisant le méta-modèle proposé au-dessus comme suit :

$$SIR = \{\mathcal{A}_{SIR}, \mathcal{R}_{SIR}, Prms_{SIR}, Tr_{SIR}\}$$

$\mathcal{A}_{SIR} = \{status\}$. Chaque individu de la population P va avoir un *status* dont la valeur est dans $\mathcal{D} = \{S, I, R\}$. $Prms_{SIR} = \{\beta, \gamma, \lambda\}$

La relation d'équivalence $\mathcal{R}_{SIR}$ est typiquement définie en utilisant l'attribut *status*, i.e, *ayant le même statut* :

$$\mathcal{R}_{SIR} = \{\forall a, b \in P : a.status = b.status\}$$

Appliquer cette relation sur la population va décomposer la population P en trois compartiments et nous donne $\mathcal{C} = P/\mathcal{R}_{SIR} = \{C_S, C_I, C_R\}$. Pour faire plus simple et court, nous utilisons les symboles S, I, R pour indiquer les compartiments du modèle SIR au lieu de C_S, C_I, C_R.

La matrice des taux de transition est :

$$Tr = \begin{pmatrix} -\lambda & \lambda & 0 \\ 0 & -\gamma & \gamma \\ 0 & 0 & 0 \end{pmatrix}$$

4.3 La séparation des préoccupations en épidémiologie

La façon la plus simple de définir les préoccupations, une fois l'opérateur de somme tensorielle $\oplus$ identifié, consiste à les définir comme des modèles atomiques, les modèles étant des combinaisons de préoccupations. Il faut seulement préciser, ce qui est trivial, comment se combinent les ensembles d'attributs (par une union ensembliste), les relations d'équivalence (par un ET logique), et les ensembles de paramètres (par l'union ensembliste aussi). L'opérateur $\oplus$ peut donc à lui seul parachever quasiment la séparation des préoccupations : on voudrait juste qu'il soit commutatif et il faut encore trouver comment exprimer les corrélations entre chaque préoccupation et le statut infectieux puisque si elles étaient totalement orthogonales à ce dernier elles n'apporteraient rien à un modèle sinon de la complexité inutile.

Une telle définition des préoccupations serait toutefois relativement restrictive. Intuitivement une préoccupation modifie un modèle et s'apparente plus à une transformation de modèles qu'à un simple modèle. Si on cherche en particulier à représenter les statuts infectieux eux-mêmes (SIR, SEIR, ...) comme des préoccupations la simple somme tensorielle parait trop limitée. Toutefois la notion de transformation de modèle est vague aussi préférons nous, pour préserver la structure de monoïde induite par l'opérateur de somme tensorielle, imposer que les transformations soient des morphismes de modèles, c'est-à-dire des transformations de modèles qui préservent leur structure. On sait en effet que l'ensemble des morphismes d'un monoïde est aussi un monoïde.

Définition 4.2. $C = \{\mathcal{A}_C, \mathcal{R}_C, Prms_C, \mathcal{F}_C\}$

Comme on peut le voir la définition d'une préoccupation est similaire à celle d'un modèle. La seule différence est $\mathcal{F}_C$ qui généralise la matrice des taux de transition des modèles. $\mathcal{F}_C$ est une fonction qui modifie la matrice des taux de transition. Grâce à la somme tensorielle (dénotée $\oplus$), un automate (donc un modèle) peut-être vue comme une préoccupation. En effet, soit A un automate. Alors en posant $\mathcal{F}_C(X) = X \oplus A$ on obtient une préoccupation.

Nous avons même élargi encore la notion de préoccupation en identifiant les préoccupations aux morphismes de l'ensemble des modèles. En particulier, au lieu de dire que l'automate stochastique A est une préoccupation, on considère alors que $M = M \oplus A$ est une préoccupation. Ceci permet de conserver une structure de monoïde tout en laissant la possibilité de définir librement une préoccupation du moment que la garantie est donnée qu'il s'agissent bien d'un morphisme [11], de préférence commutatif.

Pour que C soit un morphisme, il faut que deux conditions soient réalisées. Tout d'abord $C(\Theta)$ doit être l'élément neutre du co-domaine de C et on doit avoir $\mathrm{C}(P_1 \oplus P_2) = C(P_1) \oplus C(P_2)$.

11. Un morphisme f sur un monoïde $(M, \oplus, \theta)$ doit être tel que $\forall a, b \in M : f(a \oplus b) = f(a) \oplus' f(b)$ et $f(\theta) = \theta'$ où $\oplus'$ et θ' sont l'opérateur et l'élément neutre du co-domaine de f. $m = m \oplus A$ est, par exemple, un morphisme si on s'arrange pour que f soit idempotent, c'est-à-dire pour que $m \oplus A \oplus A = m \oplus A$. Notez que l'élément neutre du co-domaine est alors $\theta \oplus A$ et non plus θ.

4.4 Combiner des préoccupations

Une préoccupation peut être combinée avec une autre pour générer une nouvelle préoccupation. Nous définissons la combinaison de deux préoccupations en utilisant un opérateur binaire, dénoté $\bowtie$. Il faudra distinguer le cas des préoccupations ad hoc où $\mathcal{F}_C$ est définie librement, des préoccupations additives telles que $\mathcal{F}(X) = X \oplus Tr$ où Tr et X sont des matrices des taux de transition.

Définition 4.3. Etant donnée deux préoccupations C_1 and $C_2 : C_1 = \{\mathcal{A}_1, \mathcal{R}_1, Prms_1, \mathcal{F}_1\}$ and $C_2 = \{\mathcal{A}_2, \mathcal{R}_2, Prms_2, \mathcal{F}_2\}$

$$C_1 \bowtie C_2 \rightarrow C = \{\mathcal{A}_C, \mathcal{R}_C, Prms_C, \mathcal{F}_C\}$$

où :

$$\mathcal{R}_C = \mathcal{R}_1 \wedge \mathcal{R}_2, Prms_C = Prms_1 \cup Prms_2, \mathcal{F}_C = \mathcal{F}_2 \circ \mathcal{F}_1$$

et $\mathcal{A}_C = \mathcal{A}_1 \uplus \mathcal{A}_2 = \{a, \mathcal{D}_a^C = \mathcal{D}_a^{C_1} \cup \mathcal{D}_a^{C_2} | a \in \mathcal{A}_1 \wedge a \in \mathcal{A}_2\} \cup \{a | a \in \mathcal{A}_1 \wedge a \notin \mathcal{A}_2\} \cup \{a | a \notin \mathcal{A}_1 \wedge a \in \mathcal{A}_2\}$ avec $\mathcal{D}_a^C$, $\mathcal{D}_a^{C_1}$, $\mathcal{D}_a^{C_2}$ étant le domaine d'attribut a de la préoccupation C, C_1, C_2. $\mathcal{F}_2 \circ \mathcal{F}_1$ est la fonction composite qui donne, $\mathcal{F}_C(Tr) = \mathcal{F}_2(\mathcal{F}_1(Tr))$ étant donné une matrice des taux de transition Tr.

4.4.1 Combiner deux préoccupations additives

La fonction $\mathcal{F}_1, \mathcal{F}_2$ génèrent typiquement une somme tensorielle (dénotée par $\oplus$) de deux matrices des taux de transition. Par exemple :

$$\mathcal{F}_1(Tr) = Tr_1 \oplus Tr$$

où Tr_1 est la matrice des taux de transition de l'automate qui représente la préoccupation C_1. Il faut noter que la somme tensorielle est définie en termes de la somme de matrices de deux produits tensoriels [96]. La somme tensorielle de deux matrices Q_1 et Q_2 peut être écrit comme :

$$Q_1 \oplus Q_2 = Q_1 \otimes I_2 + I_1 \otimes Q_2 \tag{4.7}$$

où I_1 et I_2 sont les matrices d'identité correspondantes aux Q_1 et Q_2. Généralement, le produit tensoriel de deux matrices $Q_1 \otimes Q_2$ est calculé en remplaçant chaque élément q_{ij} de Q_1 par un bloc $q_{ij}Q_2$ [96].

Exemple : Produit Tensoriel de deux matrices. Etant donné deux automates stochastiques A_1 et A_2 dont les matrices de taux de transition sont :

$$Q_1 = \begin{pmatrix} -\lambda_1 & \lambda_1 \\ \lambda_2 & -\lambda_2 \end{pmatrix} \quad et \quad Q_2 = \begin{pmatrix} -\mu_1 & \mu_1 & 0 \\ 0 & -\mu_2 & \mu_2 \\ \mu_3 & 0 & -\mu_3 \end{pmatrix}$$

En appliquant l'expression de la somme tensorielle : $Q_1 \oplus Q_2 = Q_1 \otimes I_2 + I_1 \otimes Q_2$, nous allons obtenir :

$$\begin{aligned} Q_1 \oplus Q_2 &= Q_1 \otimes I_2 + I_1 \otimes Q_2 \\ &= \begin{pmatrix} -\lambda_1 & \lambda_1 \\ \lambda_2 & -\lambda_2 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} -\mu_1 & \mu_1 & 0 \\ 0 & -\mu_2 & \mu_2 \\ \mu_3 & 0 & -\mu_3 \end{pmatrix} \end{aligned}$$

$$= \left(\begin{array}{ccc|ccc} -\lambda_1 & 0 & 0 & \lambda_1 & 0 & 0 \\ 0 & -\lambda_1 & 0 & 0 & \lambda_1 & 0 \\ 0 & 0 & -\lambda_1 & 0 & 0 & \lambda_1 \\ \hline \lambda_2 & 0 & 0 & -\lambda_2 & 0 & 0 \\ 0 & \lambda_2 & 0 & 0 & -\lambda_2 & 0 \\ 0 & 0 & \lambda_2 & 0 & 0 & -\lambda_2 \end{array}\right) + \left(\begin{array}{ccc|ccc} -\mu_1 & \mu_1 & 0 & 0 & 0 & 0 \\ 0 & -\mu_2 & \mu_2 & 0 & 0 & 0 \\ \mu_3 & 0 & -\mu_3 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & -\mu_1 & \mu_1 & 0 \\ 0 & 0 & 0 & 0 & -\mu_2 & \mu_2 \\ 0 & 0 & 0 & \mu_3 & 0 & -\mu_3 \end{array}\right)$$

$$= \left(\begin{array}{ccc|ccc} -(\lambda_1+\mu_1) & \mu_1 & 0 & \lambda_1 & 0 & 0 \\ 0 & -(\lambda_1+\mu_2) & \mu_2 & 0 & \lambda_1 & 0 \\ \mu_3 & 0 & -(\lambda_1+\mu_3) & 0 & 0 & \lambda_1 \\ \hline \lambda_2 & 0 & 0 & -(\lambda_2+\mu_1) & \mu_1 & 0 \\ 0 & \lambda_2 & 0 & 0 & -(\lambda_2+\mu_2) & \mu_2 \\ 0 & 0 & \lambda_2 & \mu_3 & 0 & -(\lambda_2+\mu_3) \end{array}\right)$$

Théorème 4.4. *Soient C_1, C_2 et C_3 trois préoccupations additives dont les fonctions $\mathcal{F}_1, \mathcal{F}_2$ et $\mathcal{F}_3$ sont donc des sommes tensorielles. L'opérateur de combinaison $\bowtie$ est dit associatif :*

$$C_1 \bowtie C_2 \bowtie C_3 = (C_1 \bowtie C_2) \bowtie C_3 = C_1 \bowtie (C_2 \bowtie C_3) \tag{4.8}$$

Démonstration. Vu la définition 4.3 et grâce à l'associativité des opérateurs $\cup$ et $\wedge$ nous pouvons obtenir :

$$(\mathcal{A}_1 \uplus \mathcal{A}_2) \uplus \mathcal{A}_3 = \mathcal{A}_1 \uplus (\mathcal{A}_2 \uplus \mathcal{A}_3)$$

$$(\mathcal{R}_1 \wedge \mathcal{R}_2) \wedge \mathcal{R}_3 = \mathcal{R}_1 \wedge (\mathcal{R}_2 \wedge \mathcal{R}_3)$$

$$(Prms_1 \cup Prms_2) \cup Prms_3 = Prms_1 \cup (Prms_2 \cup Prms_3)$$

$\mathcal{F}_3 \circ \mathcal{F}_2 \circ \mathcal{F}_1$ donne $Tr_3 \oplus Tr_2 \oplus Tr_1$. Grâce à l'assotiativité de la somme tensorielle, on peut écrire :

$$(Tr_3 \oplus Tr_2) \oplus Tr_1 = Tr_3 \oplus (Tr_2 \oplus Tr_1)$$

L'associativité de l'opérateur $\bowtie$ a été prouvée. □

Théorème 4.5. *Soient C_1 et C_2 deux préoccupations additives dont les fonctions $\mathcal{F}_1$ et $\mathcal{F}_2$ sont donc des sommes tensorielles. Ces deux préoccupations peuvent alors être combinées entre elles dans un ordre quelconque :*

$$C_1 \bowtie C_2 = C_2 \bowtie C_1 \tag{4.9}$$

L'opérateur de combinaison $\bowtie$ est dit commutatif.

Démonstration. Vu la Définition 4.3 et grâce à la commutativité des opérateurs $\cup$ and $\wedge$ nous pouvons obtenir :

$$\mathcal{A}_1 \uplus \mathcal{A}_2 = \mathcal{A}_2 \uplus \mathcal{A}_1, \mathcal{R}_1 \wedge \mathcal{R}_2 = \mathcal{R}_2 \wedge \mathcal{R}_1, Prms_1 \cup Prms_2 = Prms_2 \cup Prms_1$$

Afin de pouvoir démontrer l'Équation 4.9, il nous faut démontrer :

$$\mathcal{F}_1(\mathcal{F}_2(Tr)) = \mathcal{F}_2(\mathcal{F}_1(Tr)) \tag{4.10}$$

L'équation 4.10 est équivalente à l'équation suivante :

$$Tr_1 \oplus (Tr_2 \oplus Tr) = Tr_2 \oplus (Tr_1 \oplus Tr) \tag{4.11}$$

En raison de l'associativité de la somme tensorielle, nous pouvons réécrire l'équation 4.11 comme :

$$(Tr_1 \oplus Tr_2) \oplus Tr = (Tr_2 \oplus Tr_1) \oplus Tr$$

Donc, nous devons démontrer :

$$Tr_1 \oplus Tr_2 = Tr_2 \oplus Tr_1 \tag{4.12}$$

En fait, la somme tensorielle ($\oplus$) ainsi que le produit tensoriel ($\otimes$) ne sont commutatif. Cependant, si l'ordre lexicographique est prévu avant de faire le produit de deux automates, les deux parties de l'équation 4.12 vont donner les mêmes résultats en utilisant des matrices de permutation [95]. En effet, lors que l'on fait joindre deux automates C_1 et C_2, il y a deux permutations (notée σ qui représente les ordres lexicographiques de ces automates), soit on considère les états de C_1 avant ceux de C_2, soit à l'inverse : $\sigma = [C_1, C_2]$ or $\sigma = [C_2, C_1]$. P_σ est la matrice de permutation correspondante au choix σ. En raison de l'ordre lexicographique fixé, les lignes et les colonnes de la matrice de produit seront réarrangées en faisant la multiplication avec la matrice de permutation. Par exemple, en choisissant $\sigma = [C_1, C_2]$, la partie droite de l'équation 4.12 est obligatoirement multiplié avec la matrice de permutation correspondante P_σ :

$$Tr_1 \oplus Tr_2 = P_\sigma^T (Tr_2 \oplus Tr_1) P_\sigma \tag{4.13}$$

L'équation 4.13 est appelée *pseudo-commutativité*[12] des opérateurs tensoriels [96]. Grâce à cette propriété, le théorème a été démontré. □

4.4.2 Appliquer une préoccupation à un modèle

Une préoccupation peut être appliquée à un modèle pour créer un nouveau modèle. L'application d'une préoccupation C sur le modèle M est dénotée $C(M)$.

Définition 4.6. Etant donné une préoccupation additive $C = \{\mathcal{A}_C, \mathcal{R}_C, Prms_C, \mathcal{F}_C = \oplus\}$ et un modèle M. Appliquer C à M nous donne un nouveau modèle M' :

$$C(M) = M' = \{\mathcal{A}_{M'}, \mathcal{R}_{M'}, Prms_{M'}, Tr_{M'}\}$$

où : $\mathcal{A}_{M'} = \mathcal{A}_C \uplus \mathcal{A}_M$, $\mathcal{R}_{M'} = \mathcal{R}_C \wedge \mathcal{R}_M$, $Prms_{M'} = Prms_C \cup Prms_M$, $Tr_{M'} = Tr_C \oplus Tr_M$

En utilisant cette définition, si l'on applique la préoccupation C sur le modèle vide Θ on va obtenir :

$$C(\Theta) = \{\mathcal{A}_C, \mathcal{R}_C, Prms_C, Tr_C\}$$

Théorème 4.7. *Etant donné un modèle $M = \{\mathcal{A}_M, \mathcal{R}_M, Prms_M, Tr_M\}$. Il est donc toujours possible d'extraire la préoccupation additive C_M à partir de M tellement que :*

$$M = C_M(\Theta)$$

C_M est donc appelée la préoccupation correspondante du modèle M.

12. Un exemple détaillé qui montre comment marche la pseudo-commutativité est présenté dans les annexes

Démonstration. Ce théorème peut être facilement prouvé en utilisant la définition 4.6. En effet, soit $C_M = \{\mathcal{A}_M, \mathcal{R}_M, Prms_M, \mathcal{F}_{C_M} = \oplus\}$ avec $Tr_{C_M} = Tr_M$ et $\Theta = \{\mathcal{A}_o, \mathcal{R}_o, Prms_o, Tr_o\}$. En utilisant la définition 4.6, nous pouvons obtenir le modèle initial M. □

Théorème 4.8. *Étant donné une préoccupation additive C et un modèle M :*

$$C(M) = (C \bowtie C_M)(\Theta)$$

Démonstration. Ce théorème peut être facilement démontré en utilisant les définitions 4.3 et 4.6 sachant que $\mathcal{F}_C$ et $\mathcal{F}_{C_M}$ sont $\oplus$. □

Définition 4.9. Soit deux modèles M_1, M_2. La combinaison de deux modèles $M_1 \bowtie M_2$ est :

$$M_1 \bowtie M_2 = C_{M_1}(\Theta) \bowtie C_{M_2}(\Theta) = (C_{M_1} \bowtie C_{M_2})(\Theta) = C_M(\Theta) = M$$

Théorème 4.10. *Soit $\mathcal{M}$ un ensemble des modèles. $(\mathcal{M}, \bowtie, \Theta)$ est un monoïde.*

Démonstration. $\forall M_1, M_2 \in \mathcal{M} : M_1 \bowtie M_2$ donne un modèle $M \in \mathcal{M}$.

De plus, grâce à l'associativité et la commutativité de $\bowtie$ quand on faire la combinaison de deux préoccupations addictives (voir les théorèmes 4.4 4.5), on peut facilement démontrer :

$$M_1 \bowtie (M_2 \bowtie M_3) = (M_1 \bowtie M_2) \bowtie M_3$$

et

$$M_1 \bowtie M_2 = M_2 \bowtie M_1$$

En outre, il faut montrer que Θ est l'élément neutre du monoïde. En effet : $M \bowtie \Theta$ donne un modèle $M' = \{\mathcal{A}_{M'}, \mathcal{R}_{M'}, Prms_{M'}, Tr_{M'}\}$:

$$\mathcal{A}_{M'} = \mathcal{A}_M \uplus \mathcal{A}_o = \mathcal{A}_M \uplus \emptyset = \mathcal{A}_M$$

$$\mathcal{R}_{M'} = \mathcal{R}_M \wedge \mathcal{R}_o = \mathcal{R}_M \wedge true = \mathcal{R}_M$$

$$Prms_{M'} = Prms_M \cup Prms_o = Prms_M, Tr_{M'} = Tr_M \oplus Tr_o = Tr_M \oplus 0 = Tr_M$$

Ainsi, $M' = M$. Grâce à la commutativité de $\bowtie$, on a donc : $M \bowtie \Theta = \Theta \bowtie M = M$. Θ est donc l'élément neutre du monoïde $(\mathcal{M}, \bowtie, \Theta)$. □

Théorème 4.11. *Soit C une préoccupation addictive. On suppose que C soit idempotent : $C(C(M)) = C(M)$. C est donc un morphisme sur le monoïde $(\mathcal{M}, \bowtie, \Theta)$.*

Démonstration. Il faut prouver : $\forall M_1, M_2 \in \mathcal{M} : C(M_1 \bowtie M_2) = C(M_1) \bowtie C(M_2)$. En effet :

$$C(M_1 \bowtie M_2) = C(C_{M_1}(\Theta) \bowtie C_{M_2}(\Theta)) = C((C_{M_1} \bowtie C_{M_2})(\Theta)) = (C \bowtie C_{M_1} \bowtie C_{M_2})(\Theta)$$

En outre :

$$C(M_1) \bowtie C(M_2) = (C \bowtie C_{M_1})(\Theta) \bowtie (C \bowtie C_{M_2})(\Theta) = (C \bowtie C_{M_1} \bowtie C \bowtie C_{M_2})(\Theta)$$

Grâce à la commutativité de $\bowtie$ et l'idempotence de C, on peut écrire :

$$C \bowtie C_{M_1} \bowtie C \bowtie C_{M_2} = C \bowtie C \bowtie C_{M_1} \bowtie C_{M_2} = C \bowtie C_{M_1} \bowtie C_{M_2}$$

Ainsi, $C(M_1 \bowtie M_2) = C(M_1) \bowtie C(M_2)$. Ensuite, on doit montrer $C(\Theta)$ est l'élément neutre du nouveau monoïde $(C(M), \bowtie, C(\Theta))$. En effet :

$$C(M) \bowtie C(\Theta) = (C \bowtie C_M)(\Theta) \bowtie C(\Theta) = (C \bowtie C_M \bowtie C)(\Theta) = (C \bowtie C_M)(\Theta) = C(M)$$

C est donc un morphisme sur le monoïde $(\mathcal{M}, \bowtie, \Theta)$. □

4.4.3 Combiner deux préoccupations ad hoc

Nous distinguons deux cas de $\mathcal{F}_C$: soit $\mathcal{F}_C$ génère une somme tensorielle de deux matrices des taux de transition, soit elle est librement définie et la préoccupation est dite *ad hoc*. Puisque $\mathcal{F}_C$ donne un moyen pour transformer une matrice des taux de transition d'une autre préoccupation. Elle peut donc :

- Introduire de nouveaux états, ce qui entraîne à la mise à jour du domaine de l'attribut correspondant à l'état,
- Ajouter de nouvelles transitions,
- Supprimer/mettre à jour une transition existante.

La préoccupation ad hoc introduit donc une certaine dépendance structurelle sur la préoccupation qu'elle transforme. Le concept de dépendance structurelle est proche de la notion de dépendance qui est appliquée souvent au code source. Ce concept, néanmoins, est appliqué aux définitions de préoccupations.

Définition 4.12. Une préoccupation C_2 dépend structurellement d'une autre C_1 si la définition de C_2 mentionne certaines(s) entité(s) qui ont été introduites dans C_1.

Dans le cas où la préoccupation C_2 dépend structurellement de C_1, C_2 ne peut pas être utilisée sans C_1. La façon la plus simple pour assurer cela est que C_1 doit être définie avant la définition de C_2.

Au contraire, une préoccupation est dite indépendante si elle n'a aucune dépendance structurelle sur les autres. Les préoccupations additives (dont la fonction F_C est la somme tensorielle) sont généralement définie de façon indépendante des autres.

Nous proposons ensuite des opérateurs pour exprimer des combinaisons qui dépendent des autres préoccupations. Ces opérateurs permettent de modifier/transformer la matrice des taux de transition de ces préoccupations.

Définition 4.13. Etant donné une préoccupation $C_0 = \{\mathcal{A}_{C_0}, \mathcal{R}_{C_0}, Prms_{C_0}, \mathcal{F}_{C_0}\}$. L'opérateur *addStatus* qui ajoute un nouvel état X dans C_0 est formulé de façon suivante :

(a) $addStatus_X(C_0) \rightarrow C = \{\mathcal{A}_C, \mathcal{R}_{C_0}, Prms_{C_0}, \mathcal{F}_C\}$

(b) $addStatus_X(C_0)$ est idempotent, c'est-à-dire : $addStatus_X(C) = addStatus_X(C_0)$.

(c) $\forall M \in \mathcal{M} : addStatus_X(C_0(M)) = (addStatus_X(C_0))(M) = C(M)$.

Cet opérateur va mettre à jour le domaine de l'attribut correspondant dans $\mathcal{A}_{C_0}$ en ajoutant X (dénoté $\mathcal{A}_C$). Il va ensuite créer Tr_C en ajoutant une ligne de 0 et un colonne de 0 dans la matrice des taux de transition Tr_{C_0}.

Théorème 4.14. *$addStatus_X$ est un morphisme sur le monoïde $(C_0(\mathcal{M}), \bowtie, C_0(\theta))$.*

Démonstration. Il faut d'abord prouver que $\forall M_1, M_2 \in \mathcal{M}$:

$$addStatus_X(C_0(M_1) \bowtie C_0(M_2)) = addStatus_X(C_0(M_1)) \bowtie addStatus_X(C_0(M_2)) \tag{4.14}$$

En effet :

$$addStatus_X(C_0(M_1) \bowtie C_0(M_2)) = addStatus_X(C_0(M_1 \bowtie M_2)) = C(M_1 \bowtie M_2)$$

En outre, $addStatus_X(C_0(M_1)) = C(M_1)$ et $addStatus_X(C_0(M_2)) = C(M_2)$. Car $C(M_1) \bowtie C(M_2) = C(M_1 \bowtie M_2)$. L'équation 4.14 a été prouvée.

Ensuite, il faut prouver que $addStatus_X(C_0(\theta))$ est l'élément neutre du nouveau monoïde $(addStatus_X(C_0(\mathcal{M})), \bowtie, addStatus_X(C_0(\Theta)))$. En effet : $\forall M \in \mathcal{M}$

$$addStatus_X(C_0(M)) = (addStatus_X(C_0))(M) = C(M), addStatus_X(C_0(\Theta)) = C(\Theta)$$

De plus :

$$addStatus_X(C_0(M)) \bowtie addStatus_X(C_0(\Theta)) = C(M) \bowtie C(\Theta) = C(M \bowtie \Theta) = C(M)$$

Ainsi, $addStatus_X(C_0(M)) \bowtie addStatus_X(C_0(\Theta)) = addStatus_X(C_0(M))$. Grâce à la commutativité de $\bowtie$, on a démontré que $addStatus_X(C_0(M))$ est l'élément neutre.

Puisque $addStatus_X$ est idempotent, il est également idempotent sur ce monoïde, c'est-à-dire que :

$$addStatus_X(addStatus_X(C_0(M))) = addStatus_X(C_0(M))$$

$addStatus_X$ est donc un morphisme sur le monoïde $(C_0((M), \bowtie, C_0(\theta))$.

Si on suppose que $\forall C' \neq C_0 : addStatus_X(C') \rightarrow C'$, $addStatus_X$ est un morphisme sur le monoïde $(\mathcal{M}, \bowtie, \theta)$. □

Définition 4.15. Etant donné une préoccupation $C_0 = \{\mathcal{A}_{C_0}, \mathcal{R}_{C_0}, Prms_{C_0}, \mathcal{F}_{C_0}\}$ et un triplet $T = \{X_1, taux, X_2\}$ représente la transition $X_1 \xrightarrow{taux} X_2$. L'opérateur $addTransition_T$ qui ajoute T dans C_0 est défini de façon suivante :

(a) $addTransition_T(C_0) \rightarrow C = \{\mathcal{A}_C, \mathcal{R}_{C_0}, Prms_{C_0}, \mathcal{F}_C\}$

(b) Si T existe dans C_0, on va mettre à jour le taux de T (en considérant que la mise $taux = 0$ enlève cette transition de C_0).

(c) $addTransition_T(C_0)$ est idempotent : $addTransition_T(C) = addTransition_T(C_0)$.

(d) $\forall M \in \mathcal{M} : addTransition_T(C_0(M)) = (addTransition_T(C_0))(M) = C(M)$.

Cet opérateur va modifier l'élément correspondant à T dans la matrice des taux de transition par $taux$ et mettre à jour l'élément correspondant sur la diagonale principale pour assurer : $q_{ii} = -\sum q_{ij}$.

Théorème 4.16. *$addTransition_T$ est un morphisme sur le monoïde $(C_0(\mathcal{M}), \bowtie, C_0(\theta))$.*

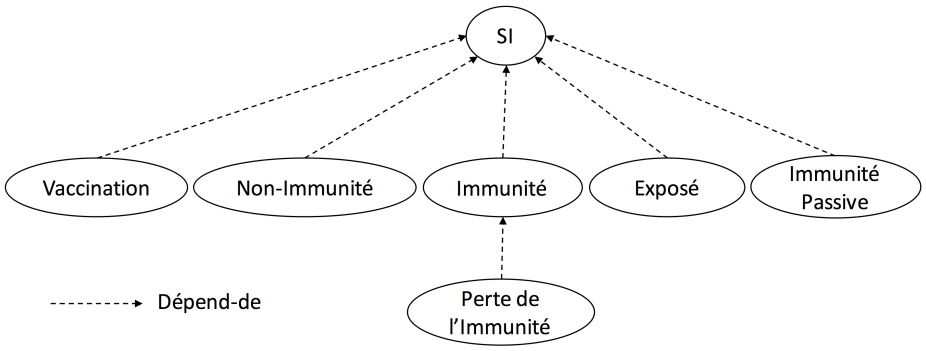
Démonstration. Grâce à la définition de $addTransition_T(C_0)$ on peut prouver le théorème comme le cas de $addStatus_X$.

Si on suppose que $\forall C' \neq C_0 : addTransition_T(C') \rightarrow C'$, $addTransition_T$ est un morphisme sur le monoïde $(\mathcal{M}, \bowtie, \theta)$. □

4.5 Gérer les dépendances entre les préoccupations

Dans cette partie, nous nous intéressons à l'intérêt des opérateurs vis à vis de l'expression des dépendances entre des préoccupations. Il faut distinguer le cas des dépendances structurelles (voir la définition 4.12), des dépendances non-structurelles qui ne sont pas liées aux définitions des préoccupations mais à leur instances.

4.5.1 Gérer les dépendances structurelles



$SIR = SI \bowtie Immunité$	$SIRS = SIR \bowtie PerteImmunité$
$SIS = SI \bowtie NonImmunité$	$MSIR = SIR \bowtie ImmunitéPassive$
$SISV = SIS \bowtie Vaccination$	$SEIRS = SEIR \bowtie PerteImmunit$
$SEIR = SIR \bowtie Exposé$	$MSEIR = SEIR \bowtie ImmunitéPassive$

Figure 4.6 – Certaines préoccupations qui dépendent de SI

Une préoccupation P dépend parfois logiquement d'une préoccupation Q ce qui peut souvent impliquer une dépendance structurelle : un identificateur définie dans Q est utilisé dans P. Ces dépendances sont inévitables. Ce que nous voulions éviter ce sont les dépendances artificielles. Pour cela nous avions besoin d'un moyen pour exprimer indépendamment les préoccupations logiquement indépendantes et de les combiner, ce que nous avons fait avec le formalisme que nous avons proposé et avec l'opérateur de somme tensorielle.

Les combinaisons de préoccupations, qui sont donc en fait des modèles, dépendent structurellement des préoccupations qui les composent. C'est pourquoi en Kendrick, la composition de préoccupations est dans une section "composition" à part du code, de façon à ne pas affecter les définitions. De même les taux fonctionnels que nous allons aborder maintenant, sont définis dans une section "instanciation" en Kendrick de façon à ne pas impacter le reste du code.

Comme nous avons présenté dans le chapitre 2, les modèles à base des compartiments de l'épidémiologie se reposent en grande partie sur le modèle SIR. À partir de cette structure de base, on peut introduire des variations du cycle de transmission des maladies infectieuses en ajoutant différents statuts infectieux et/ou ajoutant/modifiant des transitions. Il paraît que tous les modèles épidémiologiques doivent inclure la préoccupation SI. D'une part, certaines maladies conduisent à une infection permanente, la préoccupation SI est donc utilisée au lieu de SIR. D'autre part, le modèle SIR peut être vu

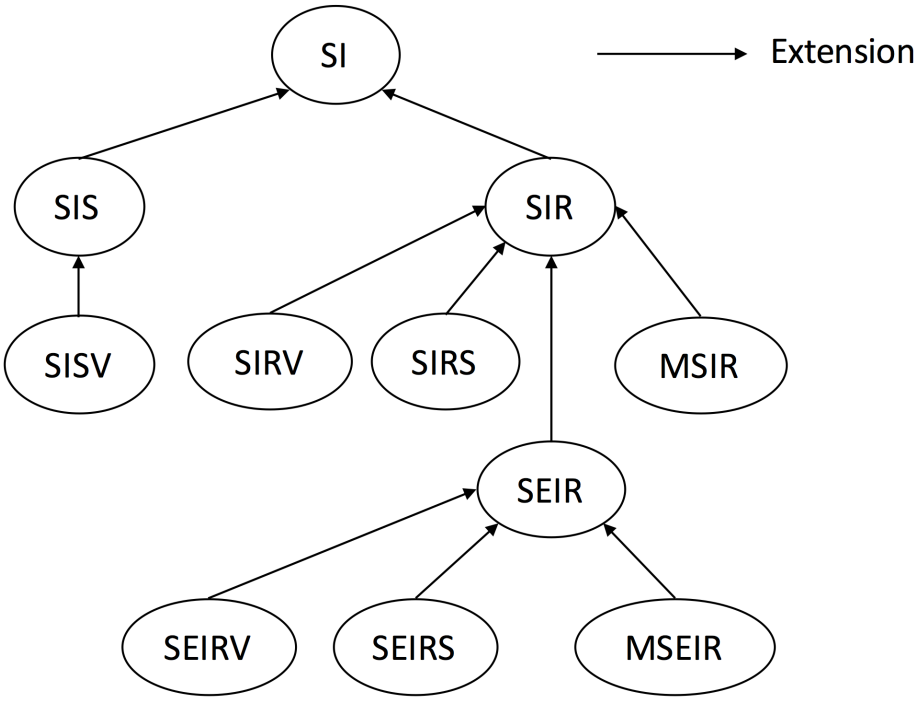


Figure 4.7 – L'arbre d'extension des modèles à base de SI

comme une combinaison de SI et la préoccupation d'immunité qui introduit le statut R. La figure 4.6 représente la structure de dépendances de certaines préoccupations sur la préoccupation SI. Les préoccupations (ad hoc), comme *Vaccination, Non-Immunité, Immunité, Exposé, Perte de l'immunité, Immunité passive*[13] etc., peuvent être exprimées en utilisant les opérateurs $addStatus_X$ et $addTransition_T$ que nous avons présentés dans la section précédente. Comme ces opérateurs sont des morphismes, ces préoccupations sont également des morsphismes qui nous donnent des structures de monoïdes. Par exemple, l'ensemble des modèles SIR, dénoté $SIR(\mathcal{M}) = Immunité(SI(\mathcal{M}))$. La figure 4.7 représente l'arbre extensions des modèles à base de SI en épidémiologie, en combinant des préoccupations abordées ci-dessus. Un exemple sur les dépendances structurelles entre les préoccupations sera présenté dans la section 4.6.2.

4.5.2 Gérer des dépendances non-structurelles : le taux de transition fonctionnel

Comme nous avons déjà abordé, le fait que les préoccupations soient définies de façon indépendante permet de pouvoir les faire varier indépendamment les unes avec les autres. Cependant cela ne signifie pas que les instances de ces préoccupations (par exemple, leur application à un tel modèle avec certains paramètres concrets) ne dépendent pas de l'autre. Il est donc très importante de pouvoir, **dans une seconde étape** (tel que la phase de composition des préoccupations dans un modèle), exprimer certaines interactions entre leurs instances. On va reprendre l'exemple des préoccupations spatiale et multi-espèces. Il est clair que la définition de la distribution spatiale des individus est totalement indépendante de celle de la multi-espèces. Cependant, dans

13. C'est un type d'immunité acquise transférée naturellement de la mère au foetus par le placenta ou de la mère à l'enfant par le colostrum [87].

un modèle multi-espèce spatial, il est nécessaire de pouvoir exprimer la distribution hétérogène de telles ou telles espèces sur l'espace.

Ces interactions peuvent être considérées comme des **dépendances non-structurelles**. Du point de vue des réseaux d'automates stochastiques, les interactions des automates se représentent par des *taux de transition fonctionnels* ou des déclenchement de transitions (c.a.d que certaines transitions dans un automate permettent de déclencher d'autres transitions dans un autre automate) [96]. Puisque dans le domaine de l'épidémiologie, on suppose souvent qu'il y ait au plus un changement d'état qui se produit pendant un intervalle de temps suffisamment petit. Le déclenchement des transitions conduit au changements d'état d'au moins deux automates, donc n'est pas considéré dans la modélisation épidémiologique. Nous nous intéressons donc seulement aux taux de transition fonctionnels pour représenter les interactions entre préoccupations de l'épidémiologie.

Un taux de transition fonctionnel est défini ci-dessous :

Définition 4.17. Ètant donné N préoccupations C_1, C_2,..., C_N dont les matrices de taux de transition sont Tr_1, Tr_2,..., Tr_N. Supposons que $\mathcal{S}_1$, $\mathcal{S}_2$,..., $\mathcal{S}_N$ ($\mathcal{S}_i = P/\mathcal{R}_i$) sont les ensembles des états de N automates stochastiques correspondantes A_1, A_2,..., A_N. Le taux de transition de l'état $s_i^{(k)}$ à l'état $s_j^{(k)}$ de l'automate A_k, qui dépend des états de d'autres automates est formulé comme :

$$Tr_k(s_i^{(k)}, s_j^{(k)}) = f : \mathcal{S}_1 \times ... \times \mathcal{S}_{k-1} \times \mathcal{S}_{k+1} \times ... \times \mathcal{S}_N \mapsto \mathbb{R}^+$$

On va reprendre l'exemple de deux automates stochastiques mentionnées dans les sections précédentes. Etant donné deux automates stochastiques A_1 et A_2 dont les matrices de taux de transition sont les suivantes :

$$Q_1 = \begin{pmatrix} -\lambda_1 & \lambda_1 \\ \lambda_2 & -\lambda_2 \end{pmatrix} \quad et \quad Q_2 = \begin{pmatrix} -\mu_1 & \mu_1 & 0 \\ 0 & -\mu_2 & \mu_2 \\ \mu_3 & 0 & -\mu_3 \end{pmatrix}$$

En faisant la composition des deux automates, on va obtenir un nouvel automaten dont la matrice des taux de transition (comme on l'a déjà montré) est :

$$\left(\begin{array}{ccc|ccc} -(\lambda_1+\mu_1) & \mu_1 & 0 & \lambda_1 & 0 & 0 \\ 0 & -(\lambda_1+\mu_2) & \mu_2 & 0 & \lambda_1 & 0 \\ \mu_3 & 0 & -(\lambda_1+\mu_3) & 0 & 0 & \lambda_1 \\ \hline \lambda_2 & 0 & 0 & -(\lambda_2+\mu_1) & \mu_1 & 0 \\ 0 & \lambda_2 & 0 & 0 & -(\lambda_2+\mu_2) & \mu_2 \\ 0 & 0 & \lambda_2 & \mu_3 & 0 & -(\lambda_2+\mu_3) \end{array}\right)$$

Maintenant, on va étudier l'interaction entre deux automates. Supposons que le taux de transition de l'état 1 à l'état 2 dans le deuxième automate A_2 est $\hat{\mu}_1$ lorsque le premier automate est dans l'état 1, et $\tilde{\mu}_1$ lorsque il est dans l'état 2. La matrice des

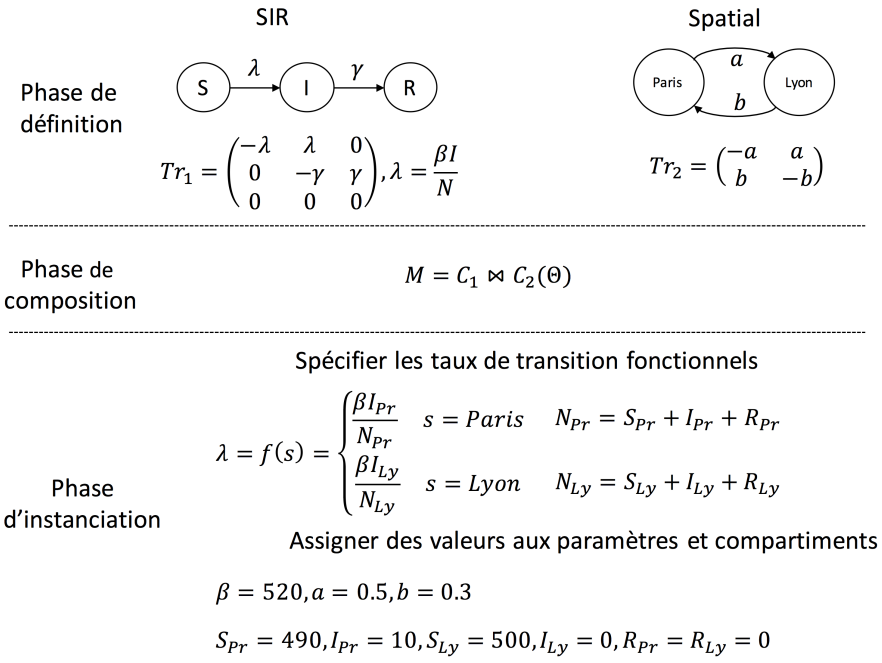


Figure 4.8 – La définition des taux fonctionnels est séparée de celle des préoccupations. A la phase de composition, les instances des préoccupations vont être “composées" en spécifiant les taux fonctionnels. Typiquement, on le fait en redéfinissant certains paramètres des préoccupations (ils deviendront des fonctions qui prennent les états courants des autres préoccupations comme arguments).

taux de transition devient :

$$\left(\begin{array}{ccc|ccc} -(\lambda_1+\hat{\mu}_1) & \hat{\mu}_1 & 0 & \lambda_1 & 0 & 0 \\ 0 & -(\lambda_1+\mu_2) & \mu_2 & 0 & \lambda_1 & 0 \\ \mu_3 & 0 & -(\lambda_1+\mu_3) & 0 & 0 & \lambda_1 \\ \hline \lambda_2 & 0 & 0 & -(\lambda_2+\tilde{\mu}_1) & \tilde{\mu}_1 & 0 \\ 0 & \lambda_2 & 0 & 0 & -(\lambda_2+\mu_2) & \mu_2 \\ 0 & 0 & \lambda_2 & \mu_3 & 0 & -(\lambda_2+\mu_3) \end{array}\right)$$

En outre, si on suppose que le taux auquel le premier automate produit la transition de l'état 1 à l'état 2 est $\bar{\lambda}_1, \hat{\lambda}_1, \tilde{\lambda}_1$ dépendant si le deuxième automate est dans l'état 1, 2 ou 3. La matrice des taux de transition est :

$$\left(\begin{array}{ccc|ccc} -(\bar{\lambda}_1+\hat{\mu}_1) & \hat{\mu}_1 & 0 & \bar{\lambda}_1 & 0 & 0 \\ 0 & -(\hat{\lambda}_1+\mu_2) & \mu_2 & 0 & \hat{\lambda}_1 & 0 \\ \mu_3 & 0 & -(\tilde{\lambda}_1+\mu_3) & 0 & 0 & \tilde{\lambda}_1 \\ \hline \lambda_2 & 0 & 0 & -(\lambda_2+\tilde{\mu}_1) & \tilde{\mu}_1 & 0 \\ 0 & \lambda_2 & 0 & 0 & -(\lambda_2+\mu_2) & \mu_2 \\ 0 & 0 & \lambda_2 & \mu_3 & 0 & -(\lambda_2+\mu_3) \end{array}\right)$$

4.5.3 Intérêts du taux fonctionnel

L'introduction des *taux de transition fonctionnels* présente certains avantages :

— **Les taux fonctionnels permettent d'exprimer les dépendances entre les préoccupations séparément de leurs définitions.** Comme nous avons indiqué dans la figure 4.8, les taux fonctionnels sont spécifiés dans la phase de composition des préoccupations. Qvec Kendrick que nous allons présenter dans le chapitre 5, nous séparons la composition des préoccupations (la phase de composition) de leur instanciations (la phase d'instanciation) où les taux fonctionnels sont spécifiés. Ils expriment donc les interactions entre les instances des préoccupations. Cela permet de pouvoir réutiliser les définitions de préoccupation dans un autre contexte et spécifier d'autres sortes d'interaction entre elles en fonction du modèle que l'on veut construire.

— **Les taux fonctionnels n'ont aucun impact sur d'autres préoccupations, sauf la préoccupation à laquelle ces taux fonctionnels appartiennent.** Puisque les taux fonctionnels sont spécifiés après l'application des préoccupations au modèle, ils n'influencent pas la structure de la matrice des taux de transition globale du modèle (la matrice obtenue après la phase de composition des préoccupations).
Nous nous intéressons maintenant à comment les taux fonctionnels influencent les préoccupations qui ont été intégrées dans le modèle.
Théorème 4.18. *Les taux de transition fonctionnels n'affectent que l'état des préoccupations auxquelles ils appartiennent.*

Démonstration. Supposons que l'on applique m préoccupations à un modèle en partant du principe que chaque préoccupation ne provoque pas de conflict avec d'autres. Selon le théorème 4.8 on va obtenir le nouveau modèle :

$$M' = (C_1 \bowtie C_2 \bowtie ... \bowtie C_m)(M) = (C_1 \bowtie C_2 \bowtie ... \bowtie C_N)(\Theta)$$

où N est le nombre de toutes les préoccupations, y compris la préoccupation correspondante au modèle initial et celles qui viennent d'être ajoutées. La matrice des taux de transition globale est :

$$\begin{aligned} Tr_M &= Tr_1 \oplus Tr_2 \oplus ... \oplus Tr_N \\ &= Tr_1 \otimes I_2 \otimes ... \otimes I_N + I_1 \otimes Tr_2 \otimes ... \otimes I_N + ... + I_1 \otimes I_2 \otimes ... \otimes Tr_N \end{aligned} \tag{4.15}$$

où $I_1, I_2, ..., I_N$ sont les matrices identité dont les dimensions sont $n_1, n_2, ..., n_N$. Le k^e terme dans la partie droite de l'équation 4.15 est la contribution de la préoccupation C_k à la matrice des taux de transition globale. Noter que $I_1 \otimes I_2 \otimes ... \otimes I_N = I_{n_1 \times n_2 \times ... \times n_N} = I_{1:N}$ et grâce à l'associativité du produit tensoriel, on peut réécrire l'équation 4.15 comme :

$$Tr_M = \sum_{i=1}^{N} I_{1:k-1} \otimes Tr_k \otimes I_{k+1:N} \tag{4.16}$$

Le terme $I_{1:k-1} \otimes Tr_k \otimes I_{k+1:N}$ signifie que lorsque le k^e automate change son état, tous les autres automates restent dans leur état courant (en raison de la

multiplication de chaque élément de Tr_k avec la matrice identité).

$$I_{1:k-1} \otimes Tr_k \otimes I_{k+1:N} = \begin{pmatrix} Tr_k & 0 & \cdots & 0 \\ 0 & Tr_k & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & Tr_k \end{pmatrix} \otimes I_{k+1:N}$$

$$= \begin{pmatrix} Tr_k \otimes I_{k+1:N} & 0 & \cdots & 0 \\ 0 & Tr_k \otimes I_{k+1:N} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & Tr_k \otimes I_{k+1:N} \end{pmatrix}$$

Dans la matrice des taux de transition globale, le taux de la transition :

$$[s_{i_1}^{(1)}, s_{i_2}^{(2)}, ..., s_{i_k}^{(k)}, ..., s_{i_N}^{(N)}] \rightarrow [s_{i_1}^{(1)}, s_{i_2}^{(2)}, ..., s_{j_k}^{(k)}, ..., s_{i_N}^{(N)}]$$

est donc égal au taux de la transition $s_{i_k}^{(k)} \rightarrow s_{j_k}^{(k)}$ de Tr_k ($s_{i_k}^{(k)}$ est l'état courant du k^e automate). Supposons que ce taux qui correspond à l'élément $s_{ij}^{(k)}$ of Tr_k est une fonction des états des autres préoccupations, noté $s_{ij}^{(k)}[C_1, ..., C_{k-1}, C_{k+1}, ..., C_N]$. Dès que l'état courant du modèle occasionne la transition composite ci-dessus, son taux sera évalué en fonction des états courants de toutes les autres préoccupations ($[s_{i_1}^{(1)}, s_{i_2}^{(2)}, ..., s_{i_k}^{(k)}, ..., s_{i_N}^{(N)}]$. La valeur du taux va décider si l'état de la k^e automate sera mis à jour. Les taux de transition fonctionnels n'affectent donc que la préoccupation à laquelle ces transitions appartiennent. □

— **Les taux fonctionnelles permettent de pouvoir exprimer les hétérogénéités de l'épidémiologie** Les hétérogénéités en épidémiologie sont causées par la différence entre les caractéristiques et/ou les comportements des individus (un individu donc peut avoir un taux pour contracter une maladie plus élevé que d'autres). Elles expriment donc des dépendances dynamiques entre les préoccupations (c.a.d les dépendances apparentes lorsque deux préoccupations se présentent en même temps dans un modèle). Les taux de transition fonctionnels nous permettent de facilement les représenter sans modifier la définition des préoccupations.

4.6 Exemples

Nous finissons ce chapitre en montrant la construction de modèles par la composition des préoccupations indépendantes (Exemple 1) et dépendantes (Exemple 2) en utilisant l'approche que nous avons présenté.

4.6.1 Exemple 1 - Le modèle SIR spatial

Considérons deux préoccupations : SIR représente la transmission d'une maladie infectieuse dans laquelle chaque individu est supposé être susceptible (avec le status S)

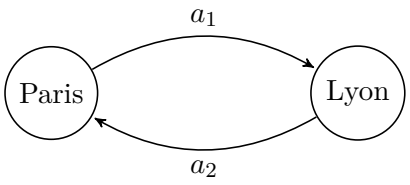


Figure 4.9 – L'automate qui représente la préoccupation spatiale avec deux régions

ou infectieux (avec le status I) ou rétabli (avec le status R) (Figure 4.3).

$$C_{SIR} = \{\mathcal{A}_{SIR}, \mathcal{R}_{SIR}, Prms_{SIR}, \mathcal{F}_{SIR}\}$$

où $\mathcal{A}_{SIR} = \{status\}$ avec $\mathcal{D}_{status} = \{S, I, R\}$, $\mathcal{R}_{SIR} : \forall a, b,\ a\mathcal{R}_{SIR} b \ ssi\ a.status = b.status$, $Prms_{SIR} = \{\lambda, \gamma\}$, $\mathcal{F}_{SIR} = \oplus$.

La préoccupation spatiale décompose la population en deux zones géographiques *Paris, Lyon*. Les individus se déplace d'une zone à une autre (Figure 4.9).

$$C_{sp} = \{\mathcal{A}_{sp}, \mathcal{R}_{sp}, Prms_{sp}, \mathcal{F}_{sp}\}$$

où $\mathcal{A}_{sp} = \{city\}$ avec $\mathcal{D}_{city} = \{Pr, Ly\}$, $\mathcal{R}_{sp} : \forall a, b,\ a\mathcal{R}_{sp} b \ ssi\ a.city = b.city$, $Prms_{sp} = \{a_1, a_2\}$, $\mathcal{F}_{sp} = \oplus$.

Les matrices de taux de transition des automates stochastiques correspondantes à ces deux préoccupations sont :

$$Tr_{SIR} = \begin{pmatrix} -\lambda & \lambda & 0 \\ 0 & -\gamma & \gamma \\ 0 & 0 & 0 \end{pmatrix} \quad et \quad Tr_{sp} = \begin{pmatrix} -a_1 & a_1 \\ a_2 & -a_2 \end{pmatrix}$$

Appliquons ces deux préoccupations à Θ :

$$M = (C_{SIR} \bowtie C_{sp})(\Theta)$$

$\mathcal{A}_M = \{status, city\}$, $\mathcal{R}_M = \mathcal{R}_{SIR} \wedge \mathcal{R}_{sp} : \forall a, b, a\mathcal{R}_M b \ ssi\ (a.status = b.status) \wedge (a.city = b.city)$, $Prms_M = \{\lambda, \gamma, a_1, a_2, N\}$.

$$Tr_M = Tr_{SIR} \oplus Tr_{sp} = Tr_{SIR} \otimes I_{sp} + I_{SIR} \otimes Tr_{sp}$$

$$Tr_M = \left(\begin{array}{c|cccccc} & S_{Pr} & S_{Ly} & I_{Pr} & I_{Ly} & R_{Pr} & R_{Ly} \\ \hline S_{Pr} & -(a_1+\lambda) & a_1 & \lambda & 0 & 0 & 0 \\ S_{Ly} & a_2 & -(a_2+\lambda) & 0 & \lambda & 0 & 0 \\ I_{Pr} & 0 & 0 & -(a_1+\gamma) & a_1 & \gamma & 0 \\ I_{Ly} & 0 & 0 & a_2 & -(a_2+\gamma) & 0 & \gamma \\ R_{Pr} & 0 & 0 & 0 & 0 & -a_1 & a_1 \\ R_{Ly} & 0 & 0 & 0 & 0 & a_2 & -a_2 \end{array}\right)$$

Dans une modélisation épidémiologique, quand la population est distribuée de façon homogène [14], la transition $S \xrightarrow{\lambda} I$ a le taux $\lambda = \beta I/N$ (*beta* est le taux de transmission, N est la taille de la population et I dénote la cardinalité du compartiment I) [69].

14. Homogeneous population

Cependant, dû à l'hétérogénéité spatiale, la transmission d'infection dans chaque zone peut être différente. Le taux λ de la préoccupation SIR sera donc spécifié comme une fonction dépendant de l'état de la préoccupation spatiale :

$$Tr_{SIR}(S,I) = \lambda = f(s) = \begin{cases} \lambda_{Pr} = \beta_{Pr} I_{Pr}/N_{Pr} & s = Paris \\ \lambda_{Ly} = \beta_{Ly} I_{Ly}/N_{Ly} & s = Lyon \end{cases}$$

où β_{Pr}, β_{Ly} sont les paramètres éventuels, $N_{Pr} = S_{Pr} + I_{Pr} + R_{Pr}$, $N_{Ly} = S_{Ly} + I_{Ly} + R_{Ly}$ et S_{Pr}, S_{Ly}, I_{Pr}, I_{Ly}, R_{Pr}, R_{Ly} sont la cardinalité des compartiments. La matrice des taux de transition globale devient donc :

$$Tr_M = \left(\begin{array}{c|cccccc} & S_{Pr} & S_{Ly} & I_{Pr} & I_{Ly} & R_{Pr} & R_{Ly} \\ \hline S_{Pr} & -(a_1 + \lambda_{Pr}) & a_1 & \lambda_{Pr} & 0 & 0 & 0 \\ S_{Ly} & a_2 & -(a_2 + \lambda_{Ly}) & 0 & \lambda_{Ly} & 0 & 0 \\ I_{Pr} & 0 & 0 & -(a_1 + \gamma) & a_1 & \gamma & 0 \\ I_{Ly} & 0 & 0 & a_2 & -(a_2 + \gamma) & 0 & \gamma \\ R_{Pr} & 0 & 0 & 0 & 0 & -a_1 & a_1 \\ R_{Ly} & 0 & 0 & 0 & 0 & a_2 & -a_2 \end{array} \right)$$

Dans cet exemple, l'hétérogénéité de la transmission d'infection due à la distribution spatiale est capturée par la transition fonctionnelle $S \xrightarrow{\lambda} I$. Nous pouvons également varier l'interaction entre ces deux préoccupations en supposant que le taux de passage d'une zone à une autre dépend aussi des status cliniques des individus. Autrement dit, a_1 et a_2 deviennent les fonctions dépendant des états S, I, R de la préoccupation SIR.

4.6.2 Exemple 2 - Le modèle SIS avec vaccination

Cet exemple a pour but de démontrer comment construire un modèle par la combinaison de préoccupations structurellement dépendantes. Supposons le modèle SIS avec vaccination. On combine trois préoccupations : SI, NonImmunité et $Vaccination$. La figure 4.10 montre les deux automates SIS et $SISV$.

$$SIS = SI \bowtie NonImmunité, SISV = SI \bowtie NonImmunité \bowtie Vaccination$$

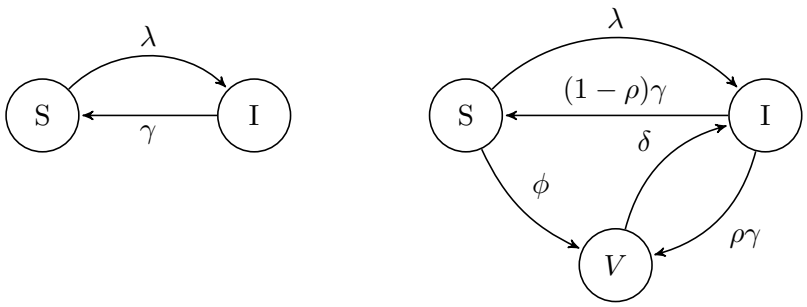


Figure 4.10 – Automates qui représentent les modèle SIS et SISV (SIS avec vaccination)

La préoccupation SI est :

$$SI = \{\mathcal{A}_{SI}, \mathcal{R}_{SI}, Prms_{SI}, \mathcal{F}_{SI}\}$$

où : $\mathcal{A}_{SI} = \{status\}$ avec $\mathcal{D}_{status} = \{S, I\}$, $\mathcal{R}_{SI} = \mathcal{R} : \forall a, b\ a\mathcal{R}b$ ssi $a.status = b.status$, $Prms_{SI} = \{\lambda\}$, $\mathcal{F}_{SI} = \oplus$. La matrice des taux de transition de la préoccupation SI est $Tr_{SI} = \begin{pmatrix} -\lambda & \lambda \\ 0 & 0 \end{pmatrix}$.

La préoccupation $NonImmunité$ qui dépend de SI est définie comme suivant :

$$NonImmunité = \{\mathcal{A}_{NoImm} = \emptyset, \mathcal{R}_{NoImm} = true, Prms_{NoImm} = \{\gamma\}, \mathcal{F}_{NoImm}\}$$

$\mathcal{F}_{NoImm} = addTransition_{I,\gamma,S}(SI)$. La préoccupation SIS est la combinaison de SI et $NoImmunity$:

$$SIS = SI \bowtie NonImmunité = \{\mathcal{A}_{SIS}, \mathcal{R}_{SIS}, Prms_{SIS}, \mathcal{F}_{SIS}\}$$

où : $\mathcal{A}_{SIS} = \{status\}$ with $\mathcal{D}_{status} = \{S, I\}$, $\mathcal{R}_{SIS} = \mathcal{R}_{SI}$, $Prms_{SIS} = \{\lambda, \gamma\}$, $\mathcal{F}_{SIS} = \oplus$. La matrice des taux de transition de la préoccupation SIS est $Tr_{SIS} = \mathcal{F}_{NoImm}(Tr_{SI}) = \begin{pmatrix} -\lambda & \lambda \\ \gamma & -\gamma \end{pmatrix}$.

Maintenant, nous allons définir la préoccupation pour la vaccination qui dépend de SIS. La vaccination est appliquée uniquement à des personnes en bonne santé, donc seulement les individus susceptibles sont vaccinés. Dans cet exemple, nous prenons également en compte le fait que les vaccinations sont rarement parfaites. Il arrive que des individus vaccinés puissent être infectés et infectieux (Figure 4.10).

$$Vaccination = \{\mathcal{A}_V = \{status\}, \mathcal{R}_V = true, Prms_V = \{\delta, \rho, \phi\}, \mathcal{F}_V\}$$

$$\text{où } \mathcal{F}_V = \begin{cases} addStatus_V(SIS) \\ addTransition_{\{S,\phi,V\}}(SIS) \\ addTransition_{\{V,\delta,I\}}(SIS) \\ addTransition_{\{I,(1-\rho)\gamma,S\}}(SIS) \\ addTransition_{\{I,\rho\gamma,V\}}(SIS) \end{cases}$$

$$SISV = SIS \bowtie Vaccination = \{\mathcal{A}_{SISV}, \mathcal{R}_{SISV}, Prms_{SISV}, \mathcal{F}_{SISV}\}$$

$\mathcal{A}_{SISV} = \{status\}$ avec $\mathcal{D}_{status} = \{S, I, V\}$, $\mathcal{R}_{SISV} = \mathcal{R}_{SIS}$, $Prms_{SISV} = \{\lambda, \gamma, \delta, \rho, \phi\}$, $\mathcal{F}_{SISV} = \oplus$.

La matrice des taux de transition de l'automate $SISV$ est $Tr_{SISV} = \mathcal{F}_V(Tr_{SIS}) = \begin{pmatrix} -\lambda - \phi & \lambda & \phi \\ (1-\rho)\gamma & -\gamma & \rho\gamma \\ 0 & \delta & -\delta \end{pmatrix}$.

4.7 Préoccupations de l'épidémiologie

Dans cette section, nous allons éatblir une liste des préoccupations les communes en épidémiologie qui ont été introduites dans la littérature [69, 81]. Notre objectif est de voir comment ces préoccupations peuvent être exprimées utilisant notre méta-modèle mathématique. Nous les présenterons en deux sous-sections : la première se focalise sur les préoccupations de base et la deuxième se porte sur les préoccupations spatiales de l'épidémiologie.

4.7.1 Les préoccupations de base de l'épidémiologie

Ce sont les préoccupations les plus utilisées pour formaliser les modèles infectieux :

- Les préoccupations épidémiques,
- Les stratégies de contrôle,
- Les pathogène multi-souches/multi-maladies,
- Les structures hétérogènes chez les individus hôtes comme la structure d'age, sexuelle/sociale, multi-espèces,
- Les préoccupations spatiales.

Les préoccupations épidémiques

Ces préoccupations ont pour but de représenter les différents cycles de transmission des maladies infectieuses comme SI, SIS, SIR, $SEIR$, $SEIRS$ etc. dans lesquels SI peut être considéré comme le cycle de transmission de base (les susceptibles seront infectés via contacts avec les infectieux). Les autres préoccupations épidémiques peuvent être définies comme extensions de SI (voir figure 4.7).

Les stratégies de contrôle

On peut ajouter certains nouveaux compartiments dans le cycle de transmission des maladies, associés aux différentes stratégies de contrôle y compris (1) Quarantaine/isolation (2) Vaccination (3) Traitement [81]. Par exemple, le nouveau status Q est ajouté pour représenter les individus qui sont en *Isolation* ou *Quarantaine*. Une quarantaine est appliquée aux individus qui ont été contact avec un individu infectieux et peuvent ou ne peuvent pas être infectés. Tandis que l'isolation a pour objectif de restreindre le contact entre un infectieux et ceux qui sont susceptibles. L'isolation quant à elle est appliquée seulement aux individus infectieux.

C'est pareil pour le cas de compartiments comme *Vaccination* et *Traitement*, qui représent respectivement des individues vaccinés ou qui suivent un traitement médical. Les transitions d'une préoccupation épidémique peuvent être modifiées ou de nouvelles transitions peuvent être ajoutées en fonction de chaque stratégie de contrôle. Ces stratégies peuvent donc être définies comme des extensions des préoccupations épidémiques.

Multi-souches/Multi-maladies

Les préoccupations distinguent différentes souches de pathogènes qui provoquent soit la même maladie (multi-souches) soit différentes maladies (multi-maladies). Ces préoccupations dépendent des préoccupations épidémiques en divisant leurs compartiments par souche de pathogène. Par exemple, on suppose un modèle SIR avec deux souches de pathogène en considérant l'immunité complètement croisée (c.a.d que l'infection causée par l'une de deux souches peut conférer une immunité permanente pour toutes les deux). Dans ce cas-là, les quatre compartiments du modèle sont : susceptible aux deux souches, infectieux causé par la première souche, infectieux causé par la deuxième, immune à deux souches (S, I_1, I_2, R). Il est possible de supposer que la susceptibilité

peut être distinguée par souche (où on doit diviser également S en S_1, S_2).

On a alors besoin d'opérateurs pour faire la décomposition des compartiments des préoccupations épidémiques par des souches de pathogène. Ces opérateurs dépendent des mécanismes de réaction immunitaire des hôtes contre les pathogènes (immunité complètement croisée, pas de l'immunité croisée qui est très souvent considéré en cas des différentes maladies). À cause des interactions entre lsouches, la transmission de ou la susceptibilité à une souche peuvent être modifiées à cause de la résistance à une autre souche, ce qui est peut être capturé par des taux fonctionnels. Un exemple de modèle multi-souches sera donné dans le chapitre 6.

Les préoccupations concernant les hétérogénéités des hôtes

Ces préoccupations décomposent la population par âge, sexe, espèces, ce qui causent les hôtes hétérogènes. Les individus peuvent *se déplacer* d'un groupe à un autre, par exemple, les enfants peuvent grandir et *entrer* vers le groupe *adolescent* ou *adultes*, ce qui est peut être capturé par des transitions adéquates. Ces préoccupations sont définies indépendantes des autres. Les hétérogénéités causées par les interactions entre différents groupes sont exprimées par des taux fonctionnels.

4.7.2 Les préoccupations spatiales de l'épidémiologie

La modélisation spatiale des maladies infectieuses est l'un des sujets les plus intéressants en épidémiologie. La variabilité des effets spatiaux sur la transmission de l'infection est toujours un défi auquel on doit faire face lors de la modélisation spatiale des maladies infectieuses.

On distingue quatre approches pour définir une préoccupation spatiale : (a) transmission spatiale entre *patches* (b) transmission spatiale dans un réseau (c) transmission spatiale entre ménages (*households*) et (d) transmission spatiale à distance [103]. Chaque approche modélise comment les individus sont en contact dans une population structurée par l'espace. Une préoccupation spatiale peut s'interagir avec d'autres préoccupations (comme multi-espèces, structure d'âge etc.) afin de faire varier les hétérogénéités entre individus.

Le méta-modèle mathématique que nous avons proposé est capable de pouvoir exprimer ces différentes approches et addresse la variabilité des mécanismes de transmission en espace grâce à l'utilisation de taux fonctionnels. Par exemple, la probabilité d'infection peut être spécifié comme une fonction temporelle dépendant des positions des individus en espace.

- **La première approche** suppose que les individus de la population se regroupent en différentes *patches*, où chaque patch est une zone géographique (un pays, une région, une cité, un village etc.). La transmission de l'infection entre les patches se représente par la mobilité des individus d'une patch à une autre. Un exemple de cette préoccupation spatiale va être présenté dans le chapitre 6.
- **La deuxième approche** suppose un réseau de contacts entre individus. Cette approche reflète le fait que le nombre de contacts appartenant à un individu est considérablement plus petit que la taille totale de la population. Cela veut dire

que les contacts entre les individus dans un modèle de réseau ne se produisent pas au hasard, ce qui est en opposition aux modèles à base de compartiments dans lesquels la population est supposée être *homogènes* et où les contacts entre individus se produisent au hasard [17, 68].
Néanmoins, un modèle à base de compartiments peut s'approcher d'un modèle de réseau en considérant un réseau aléatoire régulier qui dit que chaque individu va avoir un nombre de contacts précis. La différence entre un réseau aléatoire régulier et un réseau réel est que le premier ne prend pas en compte la structure évolutive où les contacts entre individus peuvent être changés au cours du temps. La formalisation d'un modèle de réseau réel, par contre, exige des techniques particuliers pour recueillir des informations sur le réseau par exemple, le traçage des infections, la recherche des contacts [68], ce qui peut s'avérer être très compliqués en place, même si cela est fait pour une petite population.
En épidémiologie, on utilise essentiellement des modèles de réseau aléatoire régulier. Notre méta-modèle mathématique permet de représenter ce type de modèle. La structure de réseau attribue à chaque individu un ensemble fini de contacts à qui ils peuvent transmettre l'infection et de qui ils peuvent être infecté. Les modèles à base de compartiments sont donc modifiés pour capturer un réseau où la probabilité d'infection dépend du nombre de contacts infectieux (contacts avec des individus infectieux) d'un individu [15]. Le méta-modèle va traiter l'ensemble de contacts d'un individu comme un attribut. La probabilité d'infection d'un individu en réseau est capturé par des taux de transition fonctionnel.

— **La troisième approche** considère une population structurée par *foyers* ou ménages (ou *households* en anglais). Cette approche tient compte du fait qu'il y a plus de contacts entre les membres d'un même foyer familial qu'avec d'autres personnes. Le méta-modèle exprime cette approche en réifiant les foyers comme des *compartiments*. La variabilité des taux de contact est capturé en utilisant des taux de transition fonctionnels.
Des modèles hybriques peuvent considérer des contacts dans un foyer couplés avec des contacts extérieurs. Chaque individu possède un comportement pour aller au travail, à l'école etc. On peut imaginer alors décomposer un foyer (compartiment) sous forme d'individus. Notre méta-modèle mathématique peut théoriquement décrire ces modèles mais il est en général difficile d'avoir des infomrations sur les activités précises des individus.

— **La dernière approche** définit des groupes spatiaux en fonction de leur proximité spatiale, par exemple, chaque groupe possède une position précisée (considérée alors comme un *attribut* des individus). Les taux de contact entre groupes dépendent de leur distance spatiale entre eux. Chaque individu est en contact avec tous les autres et peut être directement infecté par eux, dans ce cas, les taux de contact peuvent varier en fonction de la distance entre eux et pour certains paires d'individu, peuvent être très petits.

15. La force d'infection d'un individu i, $\lambda_i = \tau \sum_j G_{ij} I_j$. τ est le taux de transmission à travers un contact, G_i est l'ensemble de nœuds ayant contact avec nœud i dans le réseau, $I_j = 1$ si l'individu à nœud j est infectieux

4.8 Conclusion

Nous avons proposé dans ce chapitre une définition formelles d'un modèle et des préoccupations dans un contexte de modélisation épidémiologiques. En génie logiciel, une préoccupation représente une unité de réutilisation qui regroupe des artéfacts de logiciel décrivant certaines propriétés et comportements lié à un domaine d'intérêt [5]. La séparation des préoccupations est un principe clé du développement des logiciels, qui permet d'améliorer la modularité, la modifiabilité et la maintenabilité des applications. Nous avons appliqué ce principe dans le développement d'un logiciel pour la modélisation de l'épidémiologie. Un modèle va être construit en faisant une composition des préoccupations réutilisables de l'épidémiologie. Les préoccupations considérées dans le cadre de la thèse sont des morphismes de modèles.

Dans le cas où les préoccupations peuvent être définies de façon additives, les modèles qui les composent sont alors des réseaux d'automates stochastiques (Stochastic Automata Networks - SANs). Les réseaux d'automates stochastiaues sont un moyen puissant pour formuler un modèle/système par composition de composants interactifs. Ils permettent aux modélisateurs de focaliser sur le comportement des composants individuels et les interactions entre eux plutôt que de faire face à la complexité de l'ensemble du système.

Il est aussi possible de définir des préoccupations ad hoc du moment que ce sont des morphismes de modèles. Nous avons proposé certaines opérateurs qui relèvent de ce cas et peuvent servir à définir des préoccupation plus complexes.

Les interactions entre les préoccupations dans le modèle sont capturées par l'introduction de taux fonctionnels. Ces interactions peuvent être facilement modifiées par les modélisateurs pour exprimer ce qu'ils veulent modéliser en spécifiant différentes formules de fonctions sans avoir aucun impact sur les définitions de préoccupation. Nous avons donc résolu le problème de la modélisation en épidémiologie causé par le mélange des préoccupations.

La base mathématique présentée dans ce chapitre est un préalable à la construction d'une nouvelle version du langage métier Kendrick que nous allons décrire dans le chapitre suivant.

Chapitre 5

KENDRICK - Langage de Modélisation en Epidémiologie

Sommaire

Ce chapitre décrit l'implémentation de l'outil de modélisation KENDRICK en se basant sur l'approche présentée dans le chapitre précédent. Sont présentés dans ce chapitre les trois principaux composants construits pour KENDRICK : la syntaxe abstraite, la syntaxe concrète conjointement avec les entités du langage et les modules sémantiques.

Publication : Thi Mai Anh Bui, Nick Papoulias, Mikal Ziane and Serge Stinckwich. Explicit Composition Constructs in DSLs - The case of the epidemiological language KENDRICK. In *Proceedings of the International Workshop on Smalltalk Technologies, IWST 2016, Prague, Czech Republic, August 22-26, 2016.*

5.1 Vue d'ensemble du langage de modélisation Kendrick

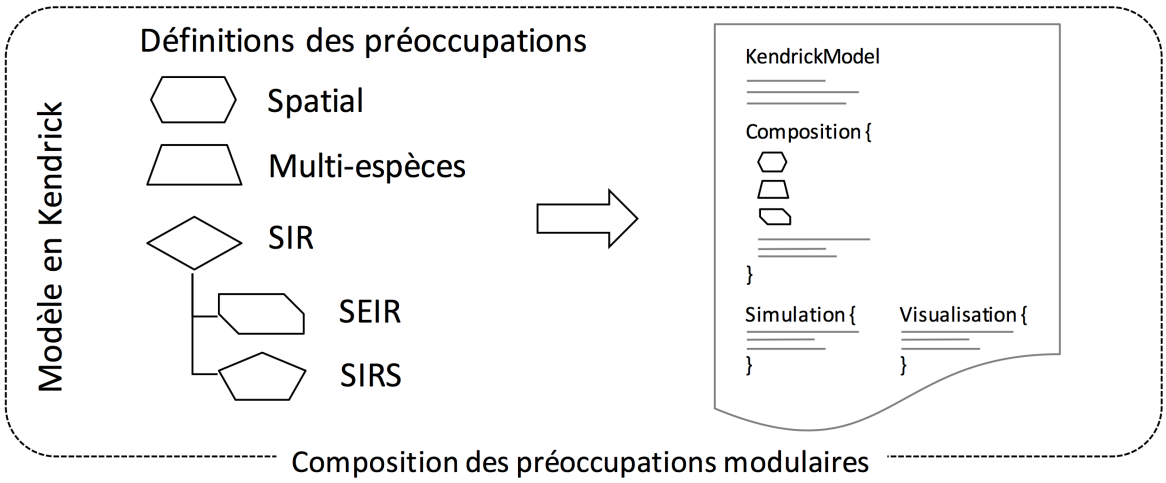


Figure 5.1 – Construction d'un modèle en Kendrick

Nous rappelons que la construction du langage Kendrick a été menée avec les objectifs suivants :

- Faciliter la construction des modèles en compartiments de l'épidémiologie en offrant aux utilisateurs une syntaxe descriptive qui se base sur des concepts du domaine.
- Produire certains types de simulation comme déterministe, stochastique, multi-agents ainsi qu'une possibilité de générer des versions avec des langages de programmation généralistes.
- Fournir des moyens pour visualiser les résultats de simulation.
- Pouvoir exprimer les préoccupations de l'épidémiologie, séparément les unes des autres, et permettre ensuite de construire les modèles par la composition de préoccupations modulaires.

À partir du premier prototype du langage qui a été présenté dans le chapitre 3, nous avons construit une deuxième version qui adresse tous ces objectifs en respectant les principes et la sémantique définies dans le chapitre 4.

Avec la deuxième version de Kendrick, un modèle épidémiologique est construit en définissant des préoccupations indépendamment des modèles (voir la figure 5.1). Ces préoccupations sont ensuite composées les unes avec les autres pour construire des modèles. En fonction de leur objectifs, les modélisateurs spécifient dans un deuxième temps les simulations/visualisations appropriées pour étudier leur modèles. Le processus de la construction d'un modèle Kendrick, y compris la spécification et la simulation du modèle, se décompose en 4 phases (la figure 5.2) :

(a) Définition
(b) Composition
(c) Instantiation
(d) Expérimentation

Phase de définition. Dans cette phase, les modèles et les préoccupations sont définis de façon séparée les uns des autres. On spécifie leur attributs, paramètres, équations

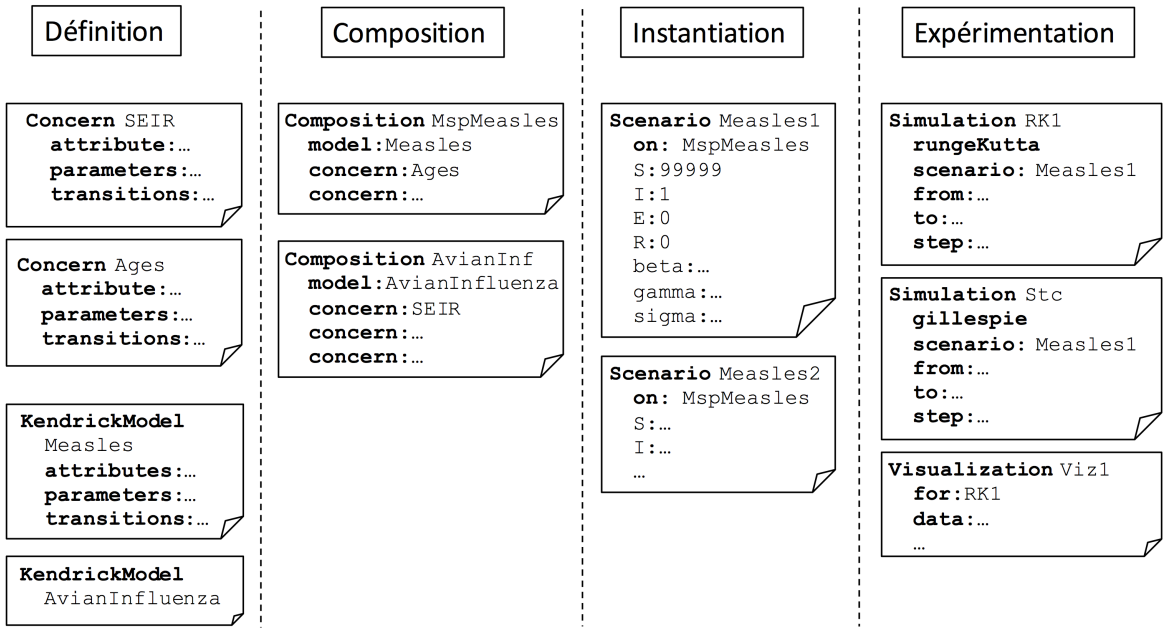


Figure 5.2 – Les phases du processus pour construire un modèle en Kendrick

différentielles ou transitions. S'il n'y a pas de transitions ou équations, la matrice de taux de transition, par défaut, se compose de zéros. Dans le cas où on veut définir un modèle, il est possible de spécifier la taille totale de la population (c'est un paramètre prédéfini pour tous les modèles). Dans le cas où une préoccupation est définie en dépendant d'une autre, il faut spécifier les dépendances, par exemple, par ajouter/modifier des transitions. La préoccupation dont elle dépend doit être définie auparavant.

Lors de la définition, on peut également initialiser les paramètres et les compartiments en leur donnant des valeurs initiales. Après la définition, une préoccupation/un modèle peut être :

- **abstrait(e)** : C'est-à-dire que leur paramètres et compartiments (dans le cas des modèles) ne sont pas liés à des valeurs.
- **partiellement instancié(e)** : certains éléments sont instanciés.
- **complètement instancié(e)** : tous les paramètres et compartiments ont été des valeurs.

Cela signifie que le langage permet de combiner non seulement les définitions des préoccupations mais également leurs instances. C'est pareil avec les modèles. On peut composer une préoccupation avec un modèle qui est complètement instancié. Cela rend les modèles plus réutilisables.

Phase de composition. Dans cette phase, on compose des préoccupations avec des modèles pour créer de nouveaux modèles dont les compartiments, les ensembles d'attributs, de paramètres et de transitions sont mis à jour en raison de la présence de nouvelles préoccupations.

Phase d'initialisation. L'initialisation des compartiments, la spécification des taux fonctionnels et/ou la spécification des valeurs pour les paramètres sont réalisées dans la phase (c).

Phase d'expérimentation. Une fois que le modèle est complètement instancié, il est prêt à être simuler. Une simulation est définie en indiquant l'algorithme utilisé, la durée de temps et le pas de temps. Il est possible de spécifier plusieurs simulations pour un même modèle.

Pour visualiser les résultats de simulation, on spécifie ensuite des visualisations. Il est possible de définir plusieurs sortes de visualisation pour une simulation, par exemple : dynamique des compartiments au cours du temps, graphe des compartiments d'un modèle, réseaux de contacts entre individus, cartes géographiques pour les modèles spatiaux.

Sont présentés dans ce chapitre les trois principaux composants de Kendrick : le méta-modèle (ou syntaxe abstraite), les modules sémantiques et un langage métier construit au dessus de la syntaxe abstraite.

5.2 Le méta-modèle du langage

5.2.1 L'implémentation du méta-modèle

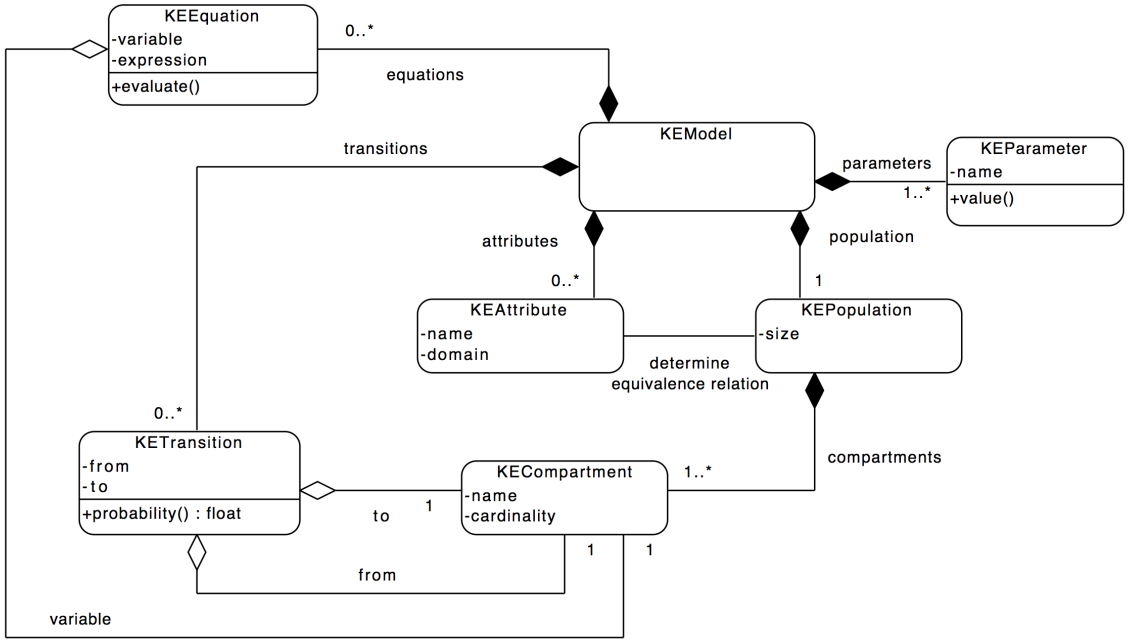


Figure 5.3 – Méta-modèle de Kendrick défini au moyen d'UML

La syntaxe abstraite (ou le méta-modèle) du langage est construite en se basant sur le méta-modèle mathématique générique introduit dans la Section 4.2. Ce méta-modèle est implémenté comme un modèle orienté objet classique (la figure 5.3).

Modèle épidémiologique

Chaque modèle de l'épidémiologie est une instance de la classe `KEModel`, qui se compose d'une population, des paramètres, des attributes et des transitions ou équations. Dans

la version actuelle , une relation d'équivalence est implémenté comme une expression d'égalité entre les attributs. Les compartiments du modèle, résultant de l'application de la relation d'équivalence sur la population, sont donc nommé par l'ensemble des valeurs des attributs. Par exemple, la relation d'équivalence « ayant le même status infectieux » du modèle SIR de l'épidémiologie permet de déterminer trois compartiments dont le nom est S, I, R, respectivement. Chaque compartiment se compose donc un ensemble des individus équivalents qui ont les mêmes valeurs pour un ensemble d'attributs.

Le méta-modèle peut capturer à la fois les modèles déterministes qui sont représentés par un ensemble de EDO et les modèles stochastiques dont la matrice de taux de transition représente une chaîne de Markov à temps continu.

L'ensemble de transitions du modèle permet de déterminer sa matrice de taux de transition : chaque élément non-zéro de la matrice $s_{ij}(i \neq j)$ représente le taux de la transition correspondante ; les éléments sur la diagonale principale satisfont à la condition : $s_{ii} = -\sum_{j \neq i} s_{ij}$. Au lieu de spécifier l'ensemble de transitions, un modèle peut être représenté par un ensemble d'équations différentielles. Ces équations peuvent être transformées vers les transitions et vice-versa [88] lors de l'interprétation du modèle (par exemple, du point de vue déterministe, l'ensemble d'équations sera considéré).

Préoccupations de l'épidémiologie

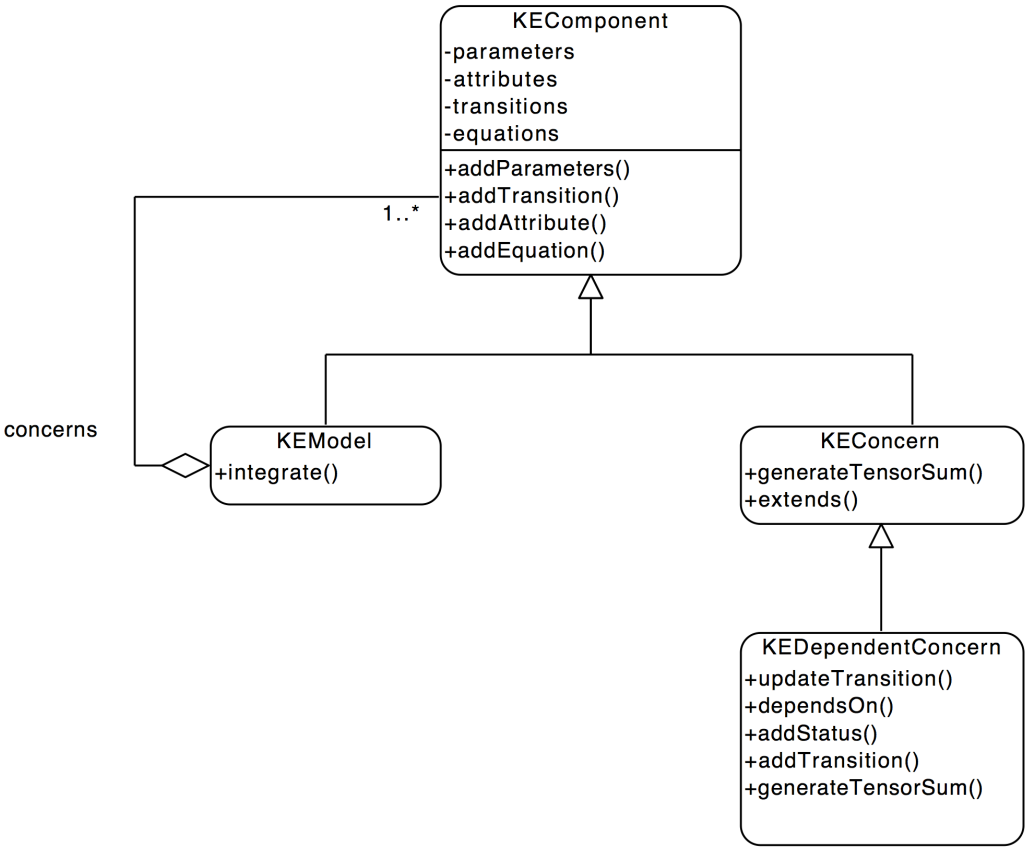


Figure 5.4 – Diagramme de classes relatif à l'intégration des préoccupations dans un modèle Kendrick

Les préoccupations indépendantes sont des instances de la classe `KEConcern`.

Les préoccupations qui sont définies en dépendant d'une autre sont des instances de la classe `KEDependentConcern`. Les dépendances peuvent être ajoutées en utilisant les méthodes `addStatus`, `addTransition`, `updateTransition`.

Ces instances peuvent être intégrées dans un modèle par la méthode `integrate` de la classe `KEModel` (figure 5.4). Dans l'objectif de pouvoir manipuler les objets des classes `KEModel` et `KEConcern` de manière uniforme, nous utilisons le patron de conception *Objet Composite*[1]. L'intention de ce patron de conception est de composer des objets en des structures arborescentes pour représenter des hiérarchies composants/composés [8]. Car on construit des modèles en composant des préoccupations individuelles, la classe `KEModel` joue donc le rôle de la classe *composite* (voir figure 5.4).

En fonction de la préoccupation à composer, la méthode `integrate` suit le déroulement suivant :

(a) Si la préoccupation est indépendante :
   1. Mettre à jour l'ensemble des attributs du modèle et donc la relation d'équivalence (l'expression booléenne sur le nouvel ensemble d'attributs).
   2. Re-décomposer la population en nouveaux compartiments en fonction de la nouvelle relation d'équivalence.
   3. Mettre à jour l'ensemble des paramètres du modèle.
   4. Invoquer la méthode `generateTensorSum` de `KEConcern` pour générer la somme tensorielle de deux matrices de taux de transition (l'une du modèle et l'autre de la préoccupation).
   5. Mettre à jour la matrice de taux de transition du modèle.

(b) Si la préoccupation (nommée Y) dépend d'une autre (nommée X) :
   1. Vérifier si X est déjà inclue dans le modèle.
   2. S'il n'en y a pas, retourner un message pour en informer et ne faire rien.
   3. Si non, faire (a).1., (a).2., (a).3.
   4. Invoquer la méthode `generateTensorSum` de Y
      - À partir de la matrice de taux du modèle (Tr_M), extraire la matrice de taux de X (Tr_X). Mettre à jour la matrice de taux du modèle après avoir extrait Tr_X.
      - Transformer Tr_X en prenant en compte les dépendances provenant de Y.
      - $Tr_M := Tr_M \oplus Tr_X$

Les classes `KEComponent, KEModel` et `KEConcern` fournissent quelques méthodes permettant de pouvoir définir des préoccupations et des modèles épidémiologiques (par exemple, pour définir l'attribut, on utilise `addAttribute` etc.).

5.2.2 Utilisation de l'API pour créer des modèles et des préoccupations

La définition du modèle SIR. La figure 5.5 représente le script dans lequel on définit le modèle SIR en utilisant directement l'API fournie par le méta-modèle orienté-objet.

1. En anglais, composite design pattern

```
1  |model sirConcern|
2  sirConcern := KEConcern new.
3  sirConcern addAttribute: #status value: #(S I R).
4  sirConcern addParameters: { #beta. #gamma }.
5  sirConcern addEquations: {
     'S:t = - beta*S*I'.
     'I:t = beta*S*I - gamma*I'.
     'R:t = gamma*I'}.
6
7  model := KEModel new population: (KEPopulation size: 100000).
8  model integrate: sirConcern.
9  model atParameter: #beta assignValue: 0.0052.
10 model atParameter: #gamma assignValue: 52.
11 model atCompartment: { #status->#S } put: 99999 atOthersPut: 0.
12 model atCompartment: { #status->#I } put: 1.
```

Figure 5.5 – Modèle SIR défini en utilisant l'API de base de Kendrick

On a défini séparément la préoccupation SIR (lignes 2-5) [2], que l'on intègre dans un modèle (ligne 8) puis le modèles est initialisé en donnant des valeurs aux paramètres et compartiments (lignes 9-12). Cette préoccupation peut alors être réutilisée dans d'autres modèles.

Les préoccupations dépendantes. Comme nous avons l'abordé déja abordé dans la section 4.4.3, les dépendances structurelles sont généralement à l'origine des modifications des transitions, de l'ajout de nouveaux états ou de nouvelles transitions. Elles sont spécifiées en utilisant les opérateurs de graphe comme : `addStatus`, `addTransition`. La modification d'une transition est considérée comme l'addition d'une transition existante.

Par exemple, la préoccupation *Immunity*, qui introduit le nouveau statut R pour exprimer ceux qui sont guéris après une certaine période d'infection, dépend de la préoccupation SI. Il est possible d'appliquer *Immunity* sur un ensemble de modèles SI (c'est-à-dire l'ensemble de modèles dans lesquels a été intégrée la préoccupation SI). Par exemple, $SIR = SI \bowtie Immunity$, $SIER = SIE \bowtie Immunity$, SIR spatial $= SI \bowtie spatial \bowtie Immunity$ etc.

Le script suivant représente la définition du modèle SIR en composant deux préoccupations SI et *Immunity* :

```
1  siConcern := KEConcern new.
2  siConcern addAttribute: #status value: #(S I).
3  siConcern addParameter: #beta.
4  siConcern addParameter: #lambda value: 'beta*I/N'.
5  siConcern
     addTransitionFrom: '{#status: #S}'
     to: '{#status: #I}'
     probability: 'lambda'.
6
7  immunity := KEConcern dependsOn: siConcern.
```

2. Rappel : les équations et/ou expressions mathématiques sont enchassés en utilisant des chaînes de caractères en Kendrick

```
8  immunity addStatus: #R.
9  immunity addParameter: #gamma.
10 immunity
     addTransitionFrom: '{#status: #I}'
     to: '{#status: #R}'
     probability: 'gamma'.
11
12 sirModel := KEModel new.
13 sirModel integrate: siConcern.
14 sirModel integrate: immunity.
```

La préoccupation *Immunity* peut être vue comme un opérateur de transformation qui peut s'appliquer sur des modèles.

Noter qu'une préoccupation peut être instanciée en donnant des valeurs spécifiques aux paramètres. Par exemple, dans le cas où on veut indiquer les valeurs par défaut pour certains paramètres, on peut les spécifier au sein de la définition d'une préoccupation. Sinon, les paramètres d'une préoccupation vont être spécifiés lors de l'instanciation des modèles (dont la préoccupation fait partie).

L'extensibilité des préoccupations. On peut également définir une nouvelle préoccupation en copiant une préoccupation existante pour en construire une nouvelle en utilisant la méthode « `copy` » (voir figure 4.7). La nouvelle préoccupation fonctionne ensuite comme une préoccupation indépendante.

Par exemple, on définit $SEIR$ comme une extension de SIR en ajoutant le nouvel état E, les nouvelles transitions $S \rightarrow E$ et $E \rightarrow I$ (les lignes de 11, 12) ; la transition $S \rightarrow I$ est supprimée (ligne13). Dans ce contexte, on a donc réutilisé la préoccupation SIR.

```
1  sirConcern := KEConcern new.
2  sirConcern addAttribute: #status value: #(S I R).
3  sirConcern addParameters: #(beta gamma).
4  sirConcern addParameter: #lambda value: 'beta*I/N'.
5  sirConcern
     addTransitionFrom: '{#status: #S}'
     to: '{#status: #I}'
     probability: 'lambda'.
6  sirConcern
     addTransitionFrom: '{#status: #I}'
     to: '{#status: #R}'
     probability: 'gamma'.
7
8  seirConcern := KEConcern copy: sirConcern.
9  seirConcern addStatus: #E.
10 seirConcern addParameter: #(sigma).
11 seirConcern
     addTransitionFrom: '{#status: #E}'
     to: '{#status: #I}'
     probability: 'sigma'.
12 seirConcern
     addTransitionFrom: '{#status: #S}'
     to: '{#status: #E}'
     probability: 'lambda'.
13 seirConcern
     updateTransitionFrom: '{#status: #S}'
     to: '{#status: #I}'
     probability: '0'.
```

5.3 Les modules sémantiques du langage Kendrick

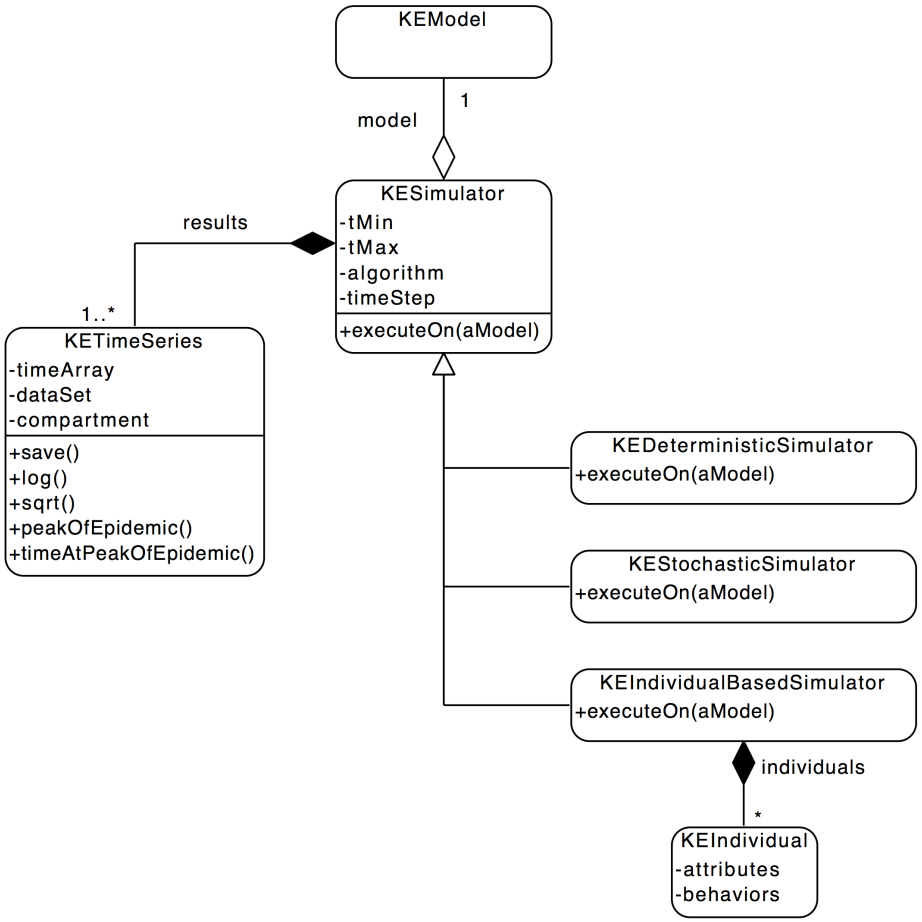


Figure 5.6 – La hiérarchie des classes des simulateurs en Kendrick

Une fois que les modèles sont formulés, ils sont prêts à être étudiés à l'aide des outils mathématiques. Comme nous avons abordé dans le chapitre 3, l'objectif des modules sémantiques du langage Kendrick est de traduire les instances du méta-modèle (les modèles en compartiments de l'épidémiologie) vers des modèles mathématiques. Le choix d'un modèle mathématique dépend de la question que l'on pose aux modèles. Selon le processus de modélisation en épidémiologie présenté dans le chapitre 2, on peut élaborer deux questions de base :

- Quelle est l'évolution globale de l'épidémie ? (la dynamique au cours du temps des compartiments du modèle)
- Étant donné des observations, quelles sont les valeurs des paramètres du modèle qui correspondent le mieux à ces observations ?

À partir de ces questions, on peut poser ensuite des requêtes pour étudier les modèles, par exemple analyser la sensibilité des paramètres, évaluer l'efficacité des stratégies de contrôle ou donner des prédictions etc. Dans le cadre de la thèse, nous visons principalement à répondre à la première question : *Quelles sont les dynamiques au cours du temps des compartiments du modèle ?* La deuxième question conduit à la construction

d'autres modules sémantiques d'estimation des paramètres [34].

Nous réutilisons les modules sémantiques développés pour le premier méta-modèle (voir le chapitre 3) que nous avons adapté pour le nouveau méta-modèle (voir figure 5.6).

5.3.1 Simulation déterministe

De façon déterministe, un modèle en Kendrick est interprété sous forme d'un ensemble d'EDO. À partir de l'ensemble des transitions d'un modèle, son système d'équation est généré en se basant sur le fait que du point de vue déterministe, les taux de transfert entre les compartiment sont exprimés de façon mathématique sous forme de dérivés de la taille des compartiments par rapport au temps. Les équations sont ensuite résolues en utilisant des méthodes numériques d'intégration telles que Euler, Runge-Kutta etc. [56]. Ces méthods d'intégration sont disponibles dans la librarie Pharo de calcul mathématique `PolyMath`[3]. Le modèle SEIR représenté dans la figure 5.5 est interprété de façon déterministe par le script suivant :

```
simulator := KESimulator new: #RungeKutta from: 0.0 to: 0.3 step: 0.1.
simulator executeOn: model.
```

Le simulateur est donc complètement spécifié en donnant l'algorithme utilisé (par exemple, `RungeKutta`), les bornes de temps ainsi que le pas de temps pour la résolution.

5.3.2 Simulation stochastique

Du point de vue stochastique, un modèle en Kendrick est interprété comme une chaîne de Markov à temps continu (CTMC). Pour générer des résultats numériques d'un modèle stochastique, les modélisateurs peuvent utiliser la méthode directe de Gillespie ou la méthode τ-leap qui ont été présentées dans la section 2.4.2.

Le script suivant montre comment spécifier un simulateur en utilisant l'algorithme Gillespie[4] :

```
simulator := KESimulator new: #Gillespie from: 0.0 to: 0.3 step: 0.1.
simulator executeOn: model.
```

5.3.3 Simulation multi-agents

Comme nous avons déjà présenté dans le chapitre 2, on considère ici une vision minimale des modèles multi-agents en les considérant comme des processus de Markov (typiquement, à temps continu) (voir la section 2.4.3). Un ensemble d'individus est initialisé en donnant des valeurs à leurs attributs en fonction de la taille initiale des

3. `https://github.com/PolyMathOrg/PolyMath`
4. Remplacer `#Gillespie` par `#TauLeap` pour utiliser l'algorithme τ-leap.

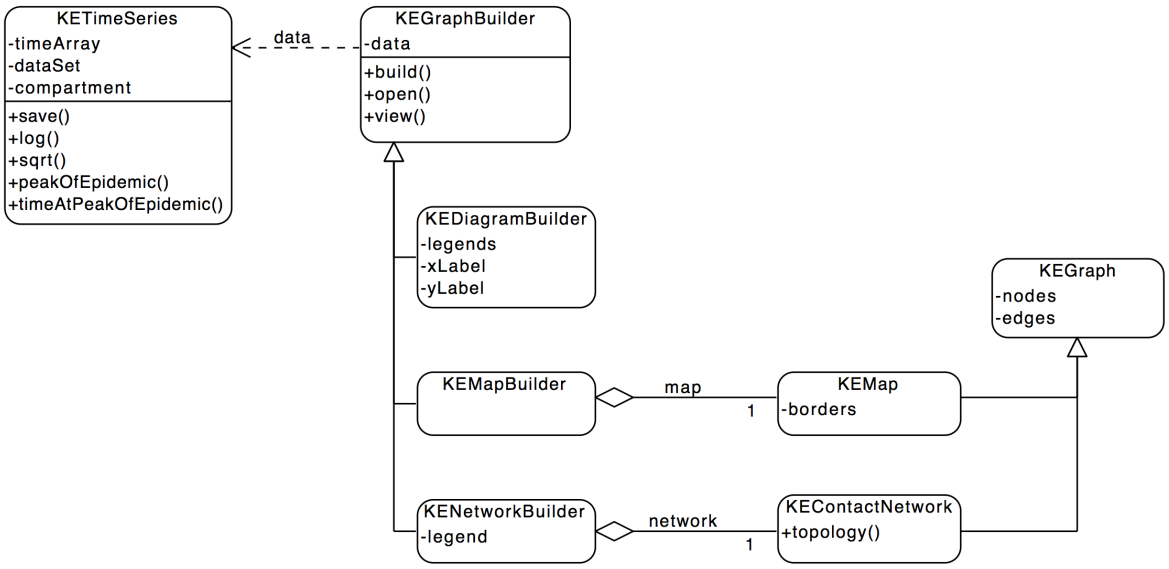


Figure 5.7 – Hiérarchie des classes des outils de visualisations de Kendrick

compartiments. En fonction de l'état courant, chaque individu a plusieurs comportements, par exemple, un susceptible peut contacter des infectieux pour contracter la maladie. À chaque itération, la probabilité pour passer à l'état suivant est calculée pour tous les individus, en supposant que le temps d'attente dans chaque état est distribué de façon exponentielle (cette hypothèse résulte de la propriété de Markov) (voir section 2.4.3). Par exemple, le taux d'infection d'un susceptible est $\lambda = \beta I/N$, la probabilité pour que cet individu reste dans la classe S est : $P = e^{-\lambda \delta t}$ étant donné que δt est choisi suffisamment petit. La probabilité pour que cet individu change son état de S à I est donc $P = 1 - e^{-\lambda \delta t}$. À la fin de l'itération, on met à jour l'état de tous les individus en fonction des valeurs des probabilités.

Le script suivant montre comment spécifier un simulateur multi-agents pour les modèles en Kendrick :

```
simulator := KESimulator new: #IBM from: 0.0 to: 0.3 step: 0.01.
simulator executeOn: model.
```

5.3.4 Visualisation des résultats

L'exécution d'un simulateur sur un modèle produit comme résultat un ensemble de séries temporelles. Chaque série temporelle capture la dynamique d'un compartiment du modèle. Il est possible d'extraire ces valeurs sous forme de fichiers CSV [5] afin d'utiliser les valeurs obtenues avec des outils externes écrits en Python ou R.

Kendrick fournit également quelques visualisations standards pour le modélisateur. Les classes du module de visualisation ont pour but d'encapsuler les résultats des simulations puis nous utilisons ensuite Roassal - un DSL interne pour Pharo pour la construction de visualisation intégré dans la plateforme Moose/Pharo pour afficher des diagrammes, cartes géographiques [12], etc ...

5. Comma-separated values

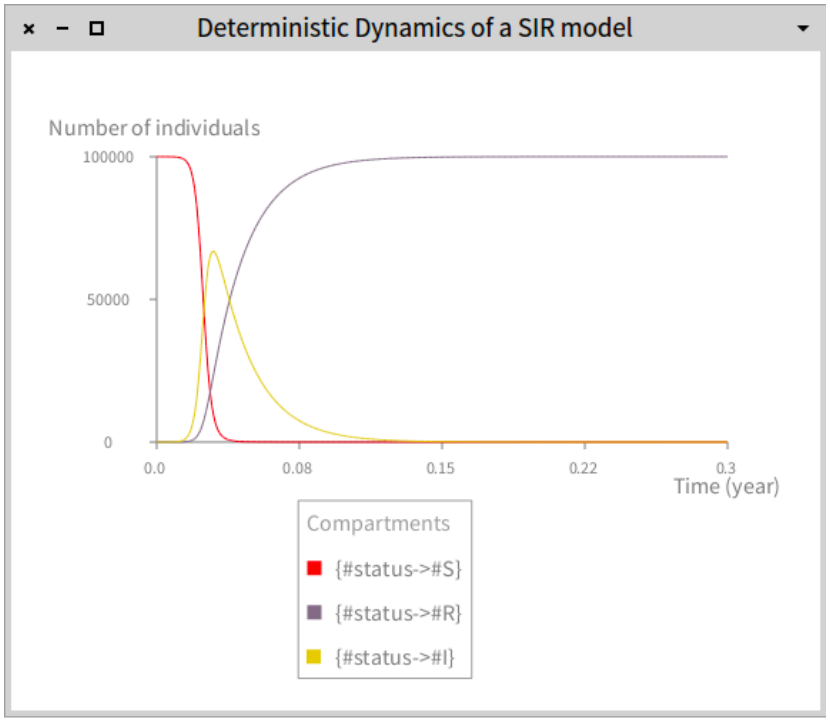


Figure 5.8 – Visualisation de la dynamique des compartiments du modèle SIR

Visualisation de séries temporelles La visualisation la plus répandue consiste à représenter un graphe de séries temporelles comme on peut le voir à la figure. Le script suivant (figure 5.8) visualise les dynamiques déterministes (figure 5.7) des compartiments S, I, R du modèle SIR (figure 5.5) :

```
diag := (KEDiagramBuilder new) data: simulator allTimeSeries.
diag xLabel: 'Time (year)'.
diag yLabel: 'Number of individuals'.
diag open
```

Visualisation de cartes géographiques Une préoccupation spatiale de l'épidémiologie peut être encapsulée en utilisant des données spatiales (par exemple, les bordures d'une carte, les données de SIG [6], les connexions aériennes etc.). Une carte géographique est une instance de la classe `KEMap`, qui est ensuite peut-être visualisée en utilisant `KEMapBuilder`.

Supposons un modèle SIR spatial dont la préoccupation spatiale est spécifiée pour six pays africains. On suppose également que ces pays sont reliés par un certain nombre de routes aériennes entre eux. Le modèle décrit ici est celui correspondant a une épidémie de la maladie Ebola en Afrique de l'Ouest décrivant une dynamique similaire à celle observée pendant l'épidémie de décembre 2013 à Novembre 2015 [7]. Ce modèle à été construit lors d'un hackathon sur Ebola auquel nous avons participé en 2014.

```
1 map := KEMap new.
```

6. Système d'Information Géographique
7. https://en.wikipedia.org/wiki/West_African_Ebola_virus_epidemic

```
map countries: #(#Liberia #Guinea #SierraLeone #Nigeria #Senegal #Niger).
map routesFrom: #Liberia toAll: #(#Guinea #SierraLeone).
map routesFrom: #Guinea toAll: #(#Nigeria).
map routesFrom: #SierraLeone toAll: #(#Senegal).
map routesFrom: #Senegal toAll: #(#Niger).
spatialConcern := KEConcern new.
spatialConcern addAttribute: #country value: map countries.
spatialConcern addParameter: #rho value: 0.05.
spatialConcern transitions: (map routesToTransitions: 'rho').

sirConcern := KEConcern new.
sirConcern addAttribute: #status value: #(S I R).
sirConcern addParameters: #(#lambda #beta #gamma).
sirConcern
    addTransitionFrom: { #status->#S }
    to: { #status->#I }
    probability: 'lambda'.
sirConcern
    addTransitionFrom: { #status->#I }
    to: { #status->#R }
    probability: 'gamma'.

model := KEModel new.
model integrate: spatialConcern.
model integrate: sirConcern.
```

Le script suivant affiche la carte (voir figure 5.9). Les pays sont colorés en fonction du pic de l'épidémie dans chaque pays après avoir fait la simulation déterministe sur le modèle pendant la durée 100 jours. Le script complète du modèle se trouve dans les annexes à la fin du manuscrit.

```
simulator := KESimulator new: #RungeKutta from: 0 to: 100 step: 0.1.
simulator executeOn: model.
mapBuilder := KEMapBuilder africa.
mapBuilder data: (mapBuilder countries collect: [:c|
  ((model atAttribute: #country) includes: c)
  ifTrue: [ (simulator timeSeriesAt:
      {#status->#I. #country->c}) first peakOfEpidemic ]
  ifFalse: [ 0 ]
  ]).
mapBuilder open.
```

Visualisation de réseaux de contacts Comme mentionné dans la section 4.7.2, on peut formuler des modèles épidémiologiques en réseau en spécifiant un réseau de contact entre les individus de la population étudiée. Dans un modèle où l'on considère un réseau de contacts entre les individus, chaque individu est assigné à un nombre limité de contacts. Le réseau se représente sous forme d'un graphe dont les noeuds sont les individus et les arêtes représentent les contacts entre eux.

Il y a plusieurs algorithmes pour générer des réseaux aléatoires[68, 80]. Dans la version actuelle de l'implémentation, nous avons implémenté trois générateurs :

- Le modèle Erdös Rényi [44] pour générer un graphe aléatoire $G(n, p)$ dont n est le nombre de noeuds et chacune des $n(n-1)/2$ arêtes est présente avec la probabilité p, absente avec probabilité $1-p$, indépendamment du statut des autres arêtes.

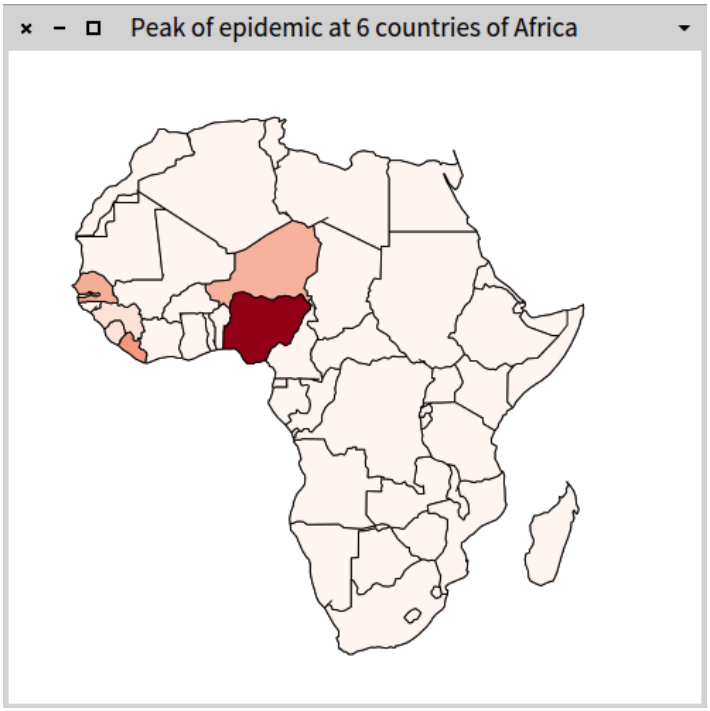


Figure 5.9 – La carte géographique affiche le pic de l'épidémie dans 6 pays en Afrique.

- Le modèle Barabási Albert [6] pour générer un réseau sans échelle [8] dont la proportion de nœuds de degré k suit une loi en puissance.
- Le modèle Watts Strogatz [119] pour générer un réseau de type « petit monde » [9] [84]. Pour modéliser ce type de réseau, on introduit une procédure de recablage aléatoire : en partant d'un treillis [10] régulier, on modifie au hasard la destination de chaque arête avec une probabilité p assez faible.

Le modèle d'Erdös-Rényi est intéressant pour commencer quand l'on veut formuler un modèle de réseau mais il n'est en fait pas très réaliste. Parce que le nombre moyen de liens que possède un nœud est égal à $n \times p$ et la plupart des nœuds ont un nombre de connexions proches de cette valeur moyenne. Ce sont les raisons pour lesquelles ce modèle est dit homogène. Par contre, les modèles sans échelle et petit-monde offrent une bonne modélisation des réseaux sociaux, surtout dans le cadre de réseaux de partenaires sexuels, et présentent un grand intérêt pour l'étude de la diffusion d'une épidémie au sein d'une population [74, 97, 100].

Le script suivant représente un modèle SIR dans lequel on formule un réseau de contacts entre les individus :

```
1 model sirConcern spatialConcern network
2 model := KEModel new population: (KEPopulation size: 100).
3
4 sirConcern := KEConcern new.
5 sirConcern addAttribute: #status value: #(S I R).
6 sirConcern addParameters: { #beta. #gamma. #lambda }.
7 sirConcern
```

8. En anglais, scale-free network

9. En anglais, on parle de small-world network. D'un point de vue mathématique, l'effet petit monde a deux particularités : (1) la distance moyenne entre deux nœuds es proportionnelle au logarithme du nombre de nœuds (2) les voisins d'un sommet donné seront souvent connectés entre eux.

10. En anglais, lattice

```
8    addTransitionFrom: { #status->#S }
9    to: { #status->#I }
10   probability: 'lambda'.
11 sirConcern
12   addTransitionFrom: { #status->#I }
13   to: { #status->#R }
14   probability: 'gamma'.
15
16 network := KEContactNetwork nodes: 100 topology: { #random. #p->0.02 }.
17 spatialConcern := KEConcern new.
18 spatialConcern addParameter: #network value: network.
19 spatialConcern addAttribute: #node value: network allContacts.
20
21 model integrate: sirConcern.
22 model integrate: spatialConcern.
```

On a défini une préoccupation spatiale qui prend en compte les nœuds du réseau comme attribut (lignes 16-19). On peut passer à un autre type de réseau en changeant le mot-clé `#random` en `#scale-free` ou `#small-world`.

La visualisation des réseaux de contacts entre les individus est faite par l'intermédiaire de la classe `KENetworkBuilder` (voir résultat figure 5.10).

```
1  simulator := KESimulator new: #IBM from: 0.0 to: 50 step: 0.1.
2  simulator executeOn: model.
3
4  nb := KENetworkBuilder new
5    data: simulator allTimeSeries;
6    network: (model atParameter: #network);
7    status: #(#S #I #R);
8    colors: #(#green #red #blue);
9    viewDataAtTime: 10;
10   legend: 'random network, p = 0.02'.
11 nb open
```

Le script complet de ce modèle est disponible à la fin de ce rapport.

Visualisation des compartiments Le dernier type de visualisation est celui liée à la représentation du graphe de compartiments d'un modèle (voir figure5.11). Nous avons intégré cette visualisation en utilisant les fonctionnalités de Glamourous Toolkit [11] un framework intégré à la plateforme de dévelopement Pharo qui permet de concevoir des inspecteurs et des visualisations spécifiques en fonction de l'objet examiné à moindre coût [12].

5.4 Le langage métier Kendrick

Dans les sections précédentes, nous avons présenté comment construire les modèles de l'épidémiologie en utilisant directement les classes et les méthodes du méta-modèle

11. `http://gtoolkit.org/`
12. On trouvera plus d'informations sur le Glamourous Toolkit et la notion de "Moldable tools" qui permettent de construire des visualisations et des outils spéficiques à un domaine en minimisant le coût de développement dans la thèse de Andrei Chis[29]

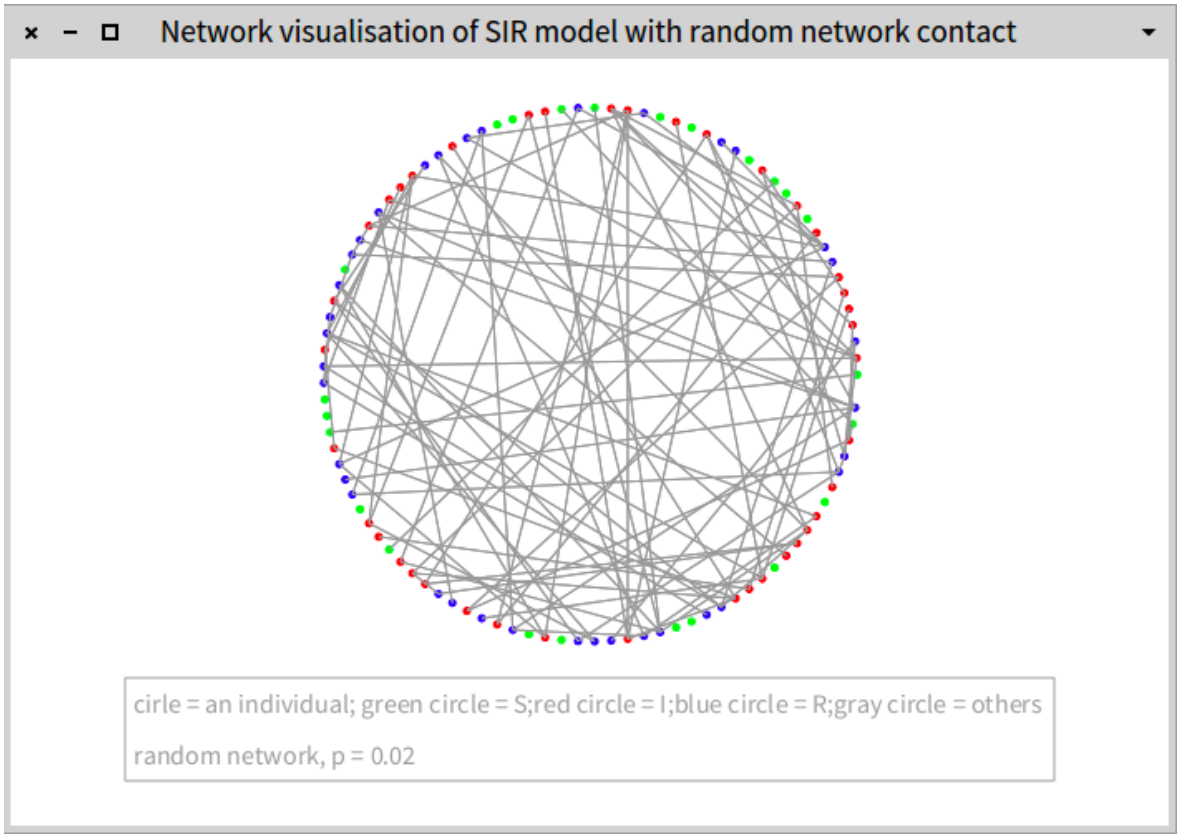


Figure 5.10 – La visualisation d'un réseau de contacts entre individus

orienté-objet. Pour rendre le langage plus accessible aux experts du domaine, nous avons construit au dessus de l'API de Kendrick, un langage métier. Ce nouveau DSL est toujours un DSL embarqué dont la syntaxe est celle de Smalltalk, mais la sémantique a été modifiée. Il s'agit donc maintenant en reprenant la terminologie de L. Renggli d'un *argot* [102].

Cette section est donc consacrée à présenter les entités du DSL qui permettent de simplifier la définition des modèles (par rapport à l'API précédente). Ces entités sont également organisées autour d'une variation du patron de conception `Composite` [8]. La raison pour laquelle nous choisissons cet architecture de conception est montrée dans la figure 5.12 où nous décrivons la sémantique de la composition des préoccupations dans un modèle Kendrick comme un arbre d'interdépendances. Puisque les dépendances proviennent seulement des nœuds parents vers leurs enfants, chaque sous-arbre est donc réutilisable. Cela signifie que chaque modèle ou chaque préoccupation (avec celles qui dépendent d'elle-même) qui est représenté comme des entités syntaxiques peut être réutilisé dans de nombreux projets de modélisation.

Les entités du langage sont représentées dans la figure 5.13, y compris `KendrickModel`, `Concern`, `Composition`, `Scenario`, `Simulation` and `Visualization`. À l'exception de deux entités, `Composition` et `Scenario`, les autres correspondent aux classes de base de Kendrick (voir les figures 5.4, 5.6, 5.7). Toutes les entités du langage possèdent une identité unique indiquée par l'utilisateur. L'accès aux entités se fait par l'utilisation de leur identité.

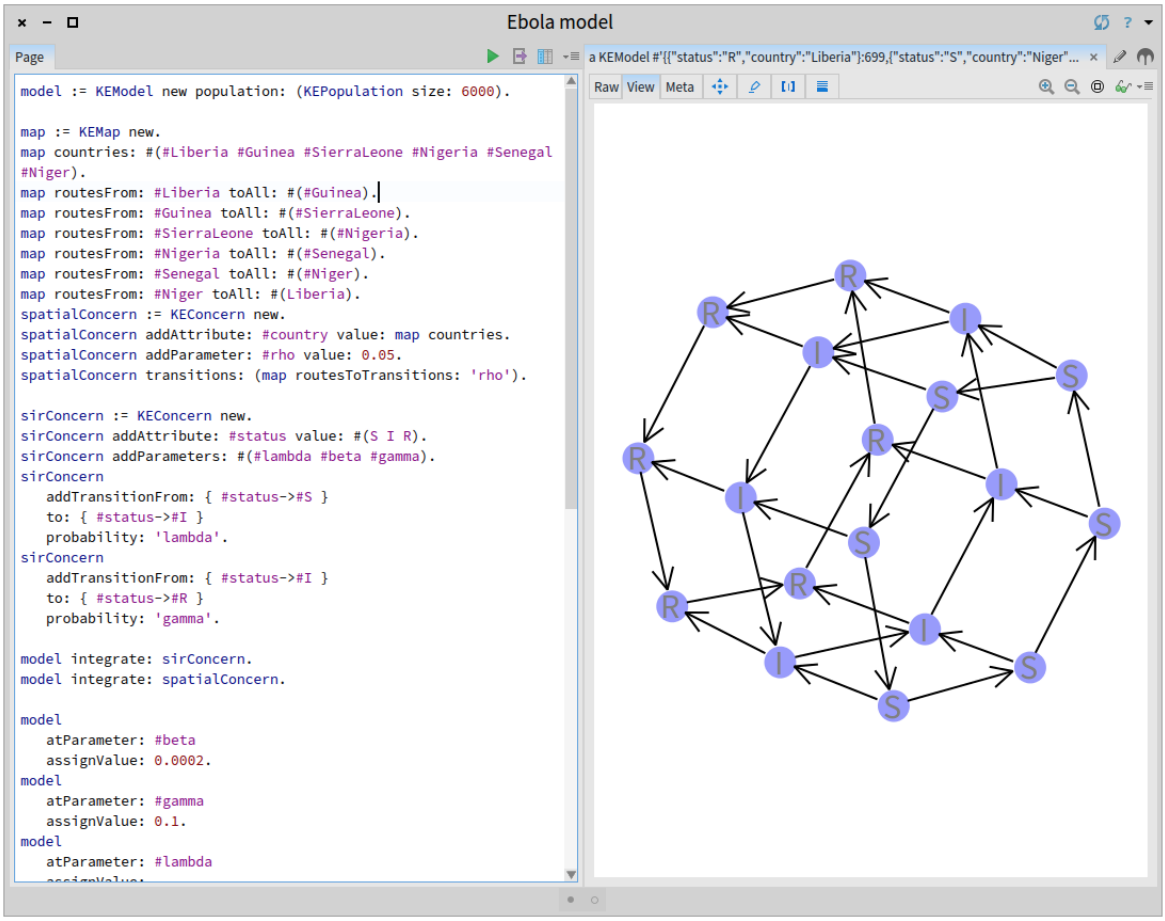


Figure 5.11 – La visualisation du graphe de compartiments du modèle Ebola

`KendrickModel`. Une entité de type `KendrickModel` représente la racine d'un modèle. Ce modèle peut être un *void modèle* dont la population n'a pas été décomposée, c'est-à-dire qui se compose d'un seul compartiment (la relation d'équivalence $\mathcal{R} = true$) ou un modèle où la population a été décomposée en plusieurs compartiments.

Le script suivant reprend la la figure 5.5 pour utiliser la nouvelle syntaxe. Dans cet exemple, on a spécifié le modèle en indiquant l'ensemble de ses éléments : paramètres, attributs et équations. On peut assigner les valeurs aux paramètres et aux compartiments directement lors de la définition du modèle (lignes 8-12). Il est également possible de séparer tout ce qui concerne l'instantiation dans une autre entité (l'entité de scénario que nous allons présenter par la suite).

```
1  KendrickModel  SIRModel
2    attribute: #(status -> S I R);
3    parameters: #(beta gamma);
4    equations: #(
5      'S:t = - beta*S*I'.
6      'I:t = beta*S*I - gamma*I'.
7      'R:t = gamma*I');
8    populationSize: 100000;
9    beta: 0.0052;
10   gamma: 52;
11   S: 99999;
12   I: 1.
```

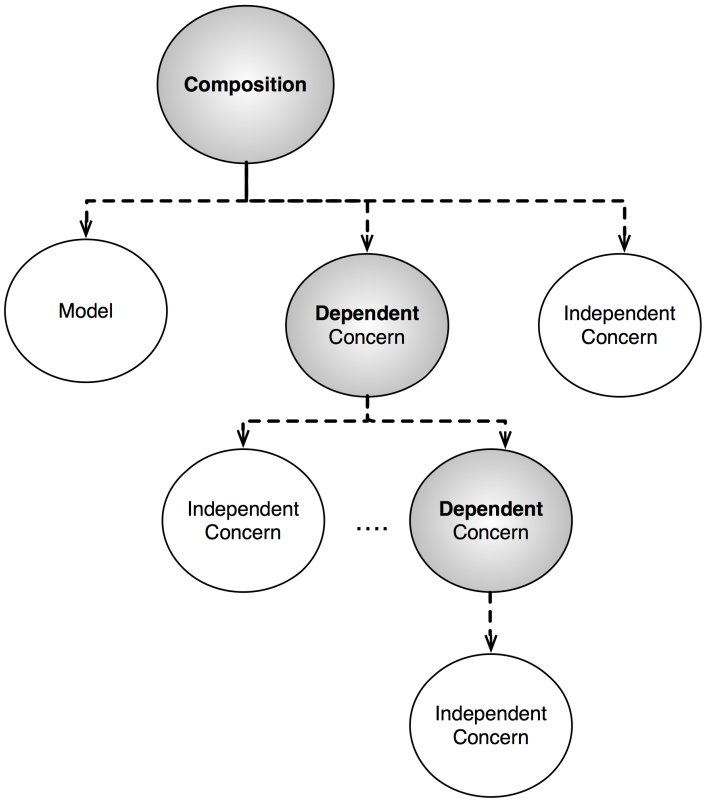


Figure 5.12 – L'arbre de composition en Kendrick : chaque sous-arbre peut être réutilisé

Concern. Une entité de type `Concern` représente une préoccupation qui est constituée d'un certain nombre d'éléments comme : paramètres, attributs, transitions (ou équations le cas échéant, par exemple dans le cas des préoccupations SIR, SEIR, etc.). Ci-dessous, c'est la définition des préoccupations `SI`, `SIR` :

```
Concern SI
  attribute: #(status -> S I R);
  parameters: #(beta lambda gamma);
  lambda:#(beta*I/N);
  transitions: #(
    S -- lambda --> I).

Concern SIR
  extends: 'SI';
  parameters: #(gamma);
  addStatus: #(R);
  addTransition: #(I -- gamma --> R).
```

La définition des dépendances structurelles entre deux préoccupation est mise en œuvre en utilisant les opérateurs de graphes comme : `addStatus`, `addTransition`. Afin de pouvoir exprimer la sémantique de certaines dépendances et simplifier leur définition, nous proposons des opérateurs de haut niveau comme :

- `delay` : Cet opérateur a pour but d'ajouter un nouvel état pour exprimer une période intermédiaire avant une telle étape d'infection d'une maladie. Par exemple, le script suivant représente la définition de SEIR en utilisant les opérateurs de base (`addTransition`, `addStatus`) :

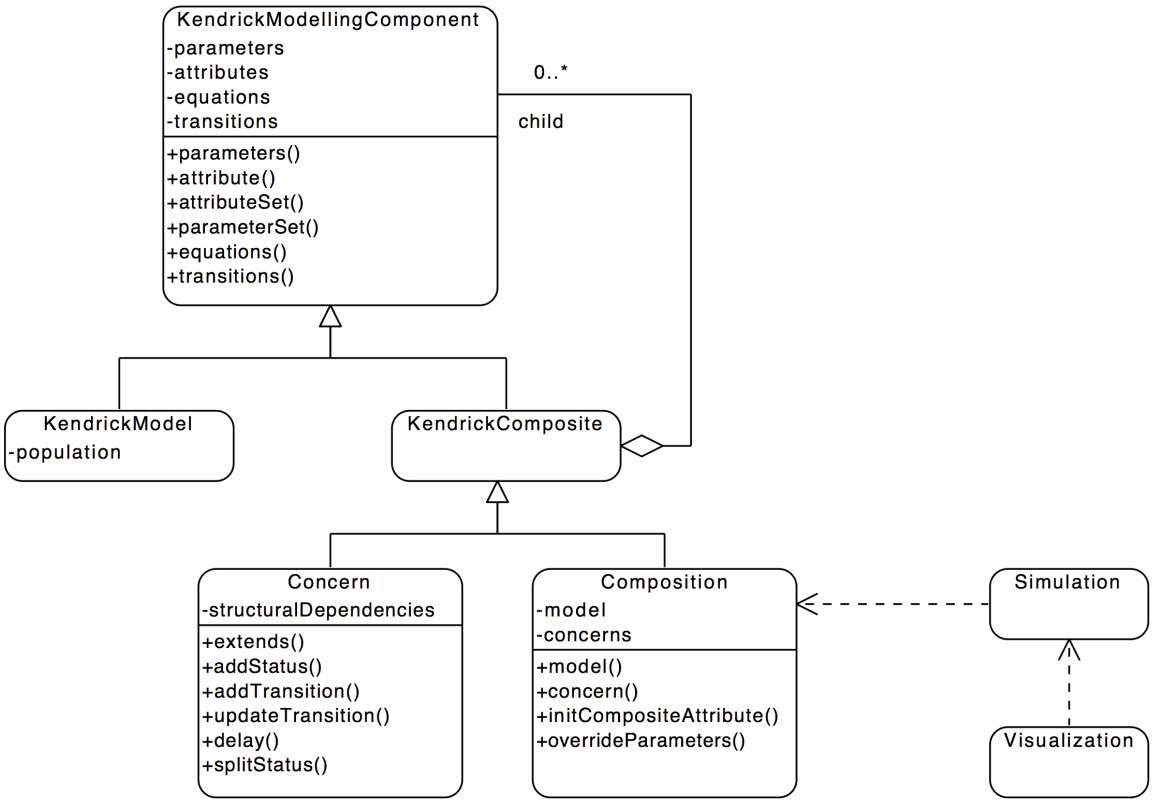


Figure 5.13 – Les entités de Kendrick organisées autour d'un patron composite

```
Concern SEIR
  extends: 'SIR';
  parameters: #(sigma);
  addStatus: #(E);
  addTransition:#(
    S -- lambda --> E.
    S -- 0 --> I.
    E -- sigma --> I.).
```

On peut simplifier cette définition en utilisant `delay` :

```
Concern SEIR
  extends: 'SIR';
  parameters: #(sigma);
  delay: #(sigma, S -- lambda --> I, E).
```

La sémantique de l'introduction du status intermédiaire E se représente de façon plus descriptive : une période d'incubation (déterminer par le paramètre `sigma`) a été ajoutée avant que les individus soient infectieux.

- `splitStatus` : Cet opérateur a pour but de diviser un statut en plusieurs statuts intermédiaires et de reproduire les mêmes transitions de ce statut pour chacun des nouveaux statuts introduits.

  Cet opérateur permet de simplifier la définition d'une préoccupation multi-souches dans laquelle le cycle de transmission est reproduit pour chaque souche de pathogène. Par exemple, on peut définir une préoccupation deux-souches en

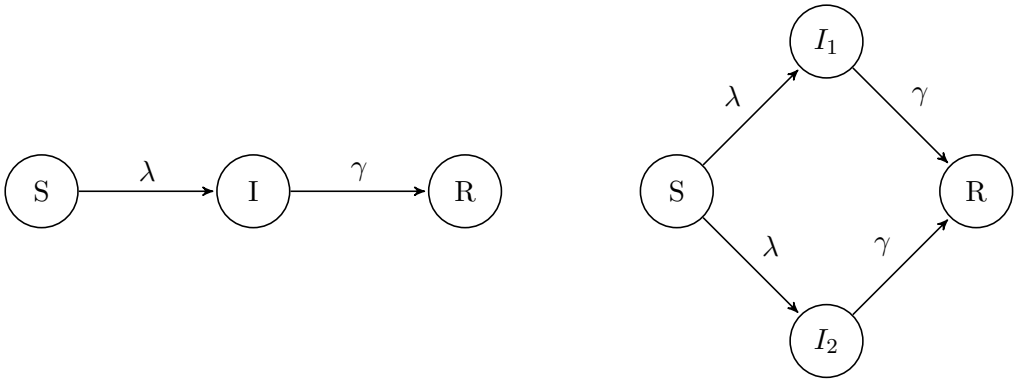


Figure 5.14 – Les deux automates qui représentent la préoccupation SIR (à gauche) et une préoccupation deux-souches (à droite).

appliquant `splitStatus` sur la préoccupation SIR (voir la figure 5.14) :

```
Concern SIRMultiStrains
  extends: 'SIR';
  splitStatus: {I. index. 1 to: 2}.
```

L'opérateur **`splitStatus`** va ajouter deux nouveaux statuts $I1$ et $I2$ et remplacer les deux transitions $S \xrightarrow{\lambda} I$ et $I \xrightarrow{\sigma} R$ par quatre nouvelles transitions : $S \xrightarrow{\lambda} I1$, $S \xrightarrow{\lambda} I2$, $I1 \xrightarrow{\sigma} R$ et $I2 \xrightarrow{\sigma} R$. En raison des hétérogénéités causées par les différentes souches de pathogène, les taux de transitions dépendent fonctionnellement de $I1$ ou $I2$. Ces transitions sont définies dans la phase d'instantiation après la composition de la préoccupation dans un modèle. Par exemple :

$$S \xrightarrow{\lambda=\beta_1 I_1/N} I_1; I_1 \xrightarrow{\gamma=\gamma_1} R; S \xrightarrow{\lambda=\beta_2 I_2/N} I_2; I_2 \xrightarrow{\gamma=\gamma_2} R$$

Il est possible d'appliquer cet opérateur sur un ensemble de statuts. Par exemple, si on veut distinguer deux souches de pathogène pour la classe S :

```
1 Concern SIRMultiStrains
2   extends: 'SIR';
3   splitStatus: {#(S I). index. 1 to: 2}.
```

Dans ce cas-là, quatre nouveaux statuts $S1, S2, I1, I2$ sont ajoutés et deux transitions $S \to I$ et $I \to R$ sont remplacées par $S1 \to I1$, $S_2 \to I2$, $I1 \to R$, $I2 \to R$. On applique cet opérateur pour chaque statut concerné et on clone les transitions du modèle pour chaque groupe de nouveaux statuts (dans l'exemple ci-dessus, on a deux groupes : $(S1, I1, R)$ et $(S2, I2, R)$)

Ces opérateurs fournissent des transformations souvent utilisées dans les modèles épidémiologiques. La possibilité d'autoriser les utilisateurs à spécifier leurs propres opérateurs est laissé pour des travaux futurs. Dans ce cas-là, certaines contraintes doivent être respectées, par exemple, la préoccupation après avoir été transformée doit rester toujours une chaîne de Markov.

`Composition.` Une entité de type `Composition` permet de prendre en charge la composition des préoccupations avec des modèles. Les préoccupations sont ajoutées explicitement par les utilisateurs. Comme présenté dans la figure 5.13, les entités sont

organisées autour du patron de conception `Composite` dans lequel l'entité `Concern` joue le rôle de feuille terminale. L'entité `Composition` permet de formuler des modèles composites en ajoutant des préoccupations modulaires. Par exemple, le modèle SIR peut être créé en composant la préoccupation SIR avec un void-modèle (dont le nom est ici *VoidModel*) :

```
1  KendrickModel VoidModel.
2
3  Concern SIR
4    attribute: #(status -> S I R);
5    parameters: #(beta lambda gamma);
6    lambda:#(beta*I/N);
7    transitions: #(
8      S -- lambda --> I.
9      I -- gamma --> R).
10
11 Composition SIRModel
12   model: 'VoidModel';
13   concern: 'SIR'.
```

Après avoir créé des modèles composites, on définit ensuite des scénarios dans lesquels on initialise les compartiments et on donne des valeurs aux paramètres.

`Scenario`. Une entité de type `Scenario` est définie pour représenter un cas d'usage du modèle dans lequel on spécifie les valeurs des paramètres et n initialise les compartiments du modèle. Il faut noter que certains paramètres peuvent être déjà instanciés lors de la définition des préoccupations ou du modèle. Ils seront réutilisés si l'utilisateur ne les rédéfinit pas dans le scénario.

Le script suivant représente un scénario (nommé *Scr1*) qui est défini pour expérimenter le modèle SIR ci-dessus. Le ligne numéro 2 indique le scénario `Scr1` est défini pour l'entité `SIRModel`.

```
1  Scenario Scr1
2    on: 'SIRModel';
3    populationSize: 100000;
4    beta: 0.0052;
5    gamma: 52;
6    S: 99999;
7    I: 1.
```

`Simulation` et `Visualization`. Une fois que le modèle est complètement instancié (tous les paramètres ont des valeurs et les compartiments sont initialisés), il peut être analysé en définissant des simulations par l'utilisation des entités de type `Simulation`. Les résultats des simulations sont ensuite visualisés en définissant des entités `Visualization`. Les scripts suivants donnent des exemples d'utilisation des entités `Simulation` et `Visualization` dans le cadre d'un modèle SIR.

```
1  Simulation SIRRK rungeKutta
2      scenario: 'Scr1';
3      from: 0;
```

```
    to: 365;
    step: 1.

Visualization SIRViz1 diagram
    for: 'SIRRK';
    data: #(I);
    xLabel: 'Time (days)'.

Simulation SIRSct gillespie
    scenario: 'Scr1';
    from: 0;
    to: 365;
    step: 1.

Visualization SIRViz2 diagram
    for: 'SIRSct';
    data: #(I);
    xLabel: 'Time (days)'.
```

En plus des entités représentées dans cette section, nous avons implémenté certaines autres entités qui permettent d'encapsuler des données spatiales. Par exemple, l'entité `Map` permet de définir une carte géographique qui va être utilisée pour définir une préoccupation spatiale.

Map On peut représenter l'aspect spatial par des graphes orientés dans lesquels les nœuds correspondent aux éléments spatiaux comme pays, villes, etc. et les arêtes représentent les frontières entre les éléments. Un tel graphe est souvent représenté en utilisant une matrice d'adjacence dont les lignes et les colonnes sont étiquetées par les sommets du graphes $\{v_i\}$. L'élément (v_i, v_j) de la matrice peut être 1 ou 0 s'il y a respectivement une arête entre v_i et v_j ou non.

Par exemple, la carte géographique de cinq pays du Sud-Est asiatique est définie comme suit :

```
Map IndoChina
    for: #(country -> Thailand Laos Vietnam Cambodia MyanmarBurma);
    borders: #(
        #(0 1 0 1 1)
        #(1 0 1 1 1)
        #(0 1 0 1 0)
        #(1 1 1 0 0)
        #(1 1 0 0 0)
).
```

Dans cet exemple, nous avons importé la matrice d'adjacence qui représente la structure spatiale (dans ce cas, les frontières géographiques) de cinq pays' :

```
Concern Spatial
    maps: 'IndoChina'.
```

Les transitions de la préoccupation spatiale sont déterminées à partir de la structure spatiale de la carte (par exemple, les éléments non-zéro de la matrice d'adjacence).

5.5 Limitations de l'implémentation actuelle

Le méta-modèle mathématique présenté dans le chapitre 4 a été implémenté comme un modèle orienté-object dans lequel nous avons simplifié certains concepts. Tout d'abord, le méta-modèle mathématique repose sur le concept de *relation d'équivalence* pour décomposer la population étudiée en petits groupes d'individus. Dans la version courante de l'implémentation de Kendrick, ce concept est restreint au cas le plus commun : une expression booléenne sur les attributs des individus. Par conséquent, un compartiment est une classe d'équivalence qui se compose des individus ayant les mêmes valeurs pour un ensemble d'attributs. Une relation d'équivalence plus complexe peut être introduite à condition qu'un moyen soit également donné pour nommer chacun des compartiments sans ambiguïté. En effet, l'application d'une relation d'équivalence sur une population nécessite denommer ces compartiments de façon explicite. Pour l'instant, nous avons nommé les compartiments en assignant les valeurs à un ensemble d'attributs. Dans le cas où une relation d'équivalence est plus complexe que l'expression booléenne sur les attributs, nommer correctement les compartiments est un défi.

Dans les sections 4.7.1, 4.7.2, nous avons montré que le méta-modèle mathématique est capable d'exprimer les préoccupations les plus communes de l'épidémiologie. Il est possible de porter ces préoccupations vers la version d'implémentation courante de Kendrick, puisque la relation d'équivalence introduite par chacune de ces préoccupations est typiquement une expression booléenne sur les attributs. Par contre, pour simplifier la spécification de certains modèles, surtout ceux dont les individus possèdent un grand ensemble d'attributs (par exemple, des modèles spatiaux avec la structure de foyers (*household structure*)), il nous faut développer des entités pour encapsuler les données qui viennent des sources extérieures ainsi que proposer les DSLs pour rendre plus agréable l'écriture de ces modèles.

Dans le premier prototype du langage (voir le chapitre 3), nous avions développé un générateur du code C/C++ qui permet de générer les trois versions (déterministe, stochastique en utilisant la méthode directe de Gillespie et multi-agents) à partir d'un modèle en Kendrick. Ce module sémantique par contre n'a pas encore été porté faute de temps dans la version courante de l'outil mais cela ne pose pas de difficultés conceptuelles.

5.6 Conclusion

Notre objectif est de produire un outil de modélisation couplé avec un langage métier, que les chercheurs en sciences de l'épidémiologie peuvent utiliser. Cet outil a été développé pour favoriser la construction modulaire des modèles en épidémiologie en se basant sur l'approche proposée dans le chapitre 4. Pour rendre plus agréable l'écriture des préoccupations ainsi que des modèles, nous avons développé un DSL au-dessus de la syntaxe abstraite et des modules sémantique du langage.

Cependant, la syntaxe abstraite, les modules sémantiques et le DSL que nous avons implémentés constituent une première étape du développement de l'outil. Les étapes suivantes (qui sont hors du cadre de cette thèse) vont consister à faire évoluer les modules sémantiques du langage (faire fonctionner le générateur du code C/C++, tra-

duire les modèles en Kendrick vers d'autres langages comme R, mettre en œuvre des modules pour estimer les valeurs des paramètres d'un modèle étant donné des données d'observation). Il sera également nécessaire d'enrichir le DSL pour faciliter l'écriture des modèles complexes qui exigent d'encapsuler des données des sources extérieures (par exemple, les données SIG pour construire les modèles spatiaux).

Chapitre 6

Études de Cas et Validation

Sommaire

Pour montrer l'intérêt de l'outil de modélisation Kendrick et vérifier notre approche, nous avons construit une base d'exemples en se basant sur les modèles qui ont été publiés dans [69, 1].

Dans le but d'illustrer comment construire des modèles en composant des préoccupations modulaires :

- Nous commençons par présenter un modèle de la rougeole SEIR et un modèle multi-espèces qui décrit une maladie vectorielle à plusieurs réservoirs dont nous avons déjà parlé dans le chapitre 3 en utilisant le langage métier Kendrick que nous avons présenté dans le chapitre 5.
- Ensuite, nous nous focalisons sur des modèles pour la grippe aviaire en considérant l'ajout de plusieurs préoccupations de l'épidémiologie comme spatiale, multi-espèces, multi-souches ou mise en quarantaine.

Dans un premier temps, nous allons valider l'implémentation de notre plateforme en comparant les résultats déterministes d'un modèle en Kendrick avec ceux donnés par MATLAB. Puis, nous comparons les dynamiques d'une simulation déterministe avec celles des simulations stochastiques (utilisant Gillespie et une modélisation multi-agents) d'un même modèle en Kendrick.

Dans une deuxième partie, nous allons valider l'approche proposée dans le chapitre 4 en répondant à deux questions. D'un côté, il nous faut évaluer la qualité des abstractions qui constituent le méta-modèle mathématique. L'objectif est de vérifier si elles sont capable de pouvoir exprimer les préoccupations du domaine. De l'autre côté, il nous faut évaluer la séparation des préoccupations dans les modèles pour vérifier si les problèmes qui sont issus de l'enchevêtrement et de la dispersion des préoccupations ont été résolus à l'aide de l'approche proposée.

6.1 Études de cas

6.1.1 Modèle de la rougeole

Comme nous l'avons vu dans le chapitre 3, la préoccupation SEIR est souvent utilisée pour représenter le cycle de transmission de la rougeole. Le script qui suit utilise les quatre phases de Kendrick : définition - composition - instantiation - expérimentation.

```
1  Concern SIR
2    attribute: #(status -> S I R);
3    parameters: #(beta lambda gamma mu);
4    lambda:#(beta*I);
5    transitions: #(
6      S -- lambda --> I.
7      I -- gamma --> R.
8      Empty -- mu --> S.
9      status -- mu --> Empty).
10
11 Concern SEIR
12   extends: 'SIR';
13   parameters: #(sigma);
14   addTransition: #(Empty -- mu --> S);
15   addTransition: #(status -- mu --> Empty);
16   delay: #(sigma , S -- lambda --> I   , E).
17
18 KendrickModel Measles.
19
20 Composition SEIRMeasles
21   model: 'Measles';
22   concern: 'SEIR'.
23
24 Scenario MeaslesScr1
25   on: 'SEIRMeasles';
26   populationSize: 100000;
27   beta: 0.0000214;
28   gamma: 0.143;
29   sigma: 0.125;
30   mu: 0.0000351;
31   S: 99999;
32   I: 1.
33
34 Simulation MeaslesRK rungeKutta
35     scenario: 'MeaslesScr1';
36     from: 0;
37     to: 150;
38     step: 1.
39
40 Visualization MeaslesViz1 diagram
41     for: 'MeaslesRK';
42     xLabel: 'Time (days)'.
```

Dans la phase de définition, on définit les préoccupations `SIR` et `SEIR` (lignes 1-16) et on déclare un *void-modèle* pour la rougeole (ligne 18).

La préoccupation `SEIR` est définie comme une extension de `SIR` avec démographie (tenant en compte de la mortalité et de la natalité des individus (voir Section 2.3.1).

Le modèle SEIR de la rougeole est créé dans la phase de composition par la spéci-

fication d'une entité de type `Composition` (nommée `SEIRMeasles`) qui compose la préoccupation `SEIR` avec le void modèle `Measles` (lignes 20-22).

La définition du modèle SEIR de la rougeole est alors complète. Pour étudier ce modèle, les entités `Scenario` (`MeaslesScr1`), `Simulation` (`MeaslesRK`) et `Visualization` (`MeaslesViz1`) sont ensuit rajoutées (lignes 24-42). Pour afficher les diagrammes, il faut utiliser le message « open » sur l'entité visualisation :

```
40 (Visualization MeaslesViz1 diagram
41     for: 'MeaslesRK';
42     xLabel: 'Time (days)') open.
```

La visualisation nous donne les mêmes résultats que ceux présenté dans le chapitre 3 (voir figure 3.6). Il est possible de spécifier d'autres simulations et visualisations pour ce même scénario :

```
1 Simulation MeaslesStc gillespie
2     scenario: 'MeaslesScr1';
3     from: 0;
4     to: 150;
5     step: 1.
6
7 Visualization MeaslesViz2 diagram
8     for: 'MeaslesStc';
9     xLabel: 'Time (days)').
```

Pour étudier l'impact de la vaccination dès la naissance sur la rougeole, on va définir une autre préoccupation qui étend la préoccupation SEIR en ajoutant une nouvelle transition pour représenter une partie des nouveaux nés qui sont vaccinés (avec le taux de vaccination p) qui vont entrer directement dans la classe R au lieu de la classe S.

À la suite de cette addition, la transition `Empty - mu -> S` va être également modifiée pour prendre en compte la vaccination dès la naissance.

```
1  Concern SEIRVacc
2    copy: 'SEIR';
3    parameters: #(p);
4    addTransition: #(Empty -- 'mu*p' --> R);
5    addTransition: #(Empty -- 'mu*(1-p)' --> S).
6
7  Composition SEIRMeaslesVacc
8    model: 'Measles';
9    concern: 'SEIRVacc'.
10
11 Scenario MeaslesScr2
12   on: 'SEIRMeaslesVacc';
13   populationSize: 100000;
14   beta: 0.00782;
15   gamma: 52.14;
16   sigma: 45.625;
17   mu: 0.0128;
18   S: 99999;
19   I: 1;
20   p: 0.7.
21
```

```
Simulation MeaslesRKVacc rungeKutta
    scenario: 'MeaslesScr2';
    from: 0;
    to: 100;
    step: 1/365.

Visualization MeaslesViz2 diagram
    for: 'MeaslesRKVacc';
    data: #(I log);
    xLabel: 'Time (years)'.
```

Comme l'effet de la vaccination est souvent étudié sur le long terme, on peut définir un nouveau scénario où l'on spécifie quelles valeurs des paramètres du modèle `SEIRMeasles` ont été modifiées pour les mettre à l'échelle d'une année. Le modèle de vaccination de la rougeole donne alors les mêmes résultats par rapport au modèle représenté dans le chapitre 3 (voir Figure 3.7).

Nous avons donc réussi à reproduire les mêmes résultats avec notre nouveau DSL que ceux que nous avions obtenu avec le premier méta-modèle du chapitre 3.

6.1.2 Un modèle multi-espèces

Dans le chapitre 3, on a construit un modèle monolithique multi-espèces SIR avec démographie pour étudier une maladie vectorielle à plusieurs réservoirs. La population se compose de trois espèces (le moustique et deux autres espèces réservoir). On va redéfinir ce modèle en le décomposant maintenant en différentes préoccupations. Nous considérons deux préoccupations indépendantes : SIR avec démographie et multi-espèces. Il est possible de réutiliser la préoccupation SIR qui a été définie ci-dessus. Il nous faut juste définir la préoccupation multi-espèces et la combiner avec `SIR` pour créer le modèle qui correspond aux équations 3.2.

```
Concern ThreeSpecies
  attribute: #(species -> mosquito reservoir1 reservoir2).

KendrickModel VectorBorne.

Composition VB3Species
  model: 'VectorBorne';
  concern: 'SIR';
  concern: 'ThreeSpecies'.

Scenario VBScr1
  on: 'VB3Species';
  populationSize: 13000;
  beta_species: #(
    #(0 0.02 0.02)
    #(0.02 0 0)
    #(0.02 0 0)
  );
  mu_species: #(12.17 0.05 0.05);
  N: #(species);
  gamma: 52;
  lambda: #(beta*I sum);
  S_species: #(9999 1000 2000);
  I_species: #(1 0 0).
```

```
25
26 Simulation VBStc gillespie
27   scenario: 'VBScr1';
28   from: 0.0;
29   to: 0.5;
30   step: 1/365.
31
32 Visualization VBViz diagram
33   for: 'VBStc';
34   data: #(I sqrt);
35   xLabel: 'Time (years)'.
```

La définition de la préoccupation multi-espèces `ThreeSpecies` est très simple car il n'y a pas de transitions (car l'espèce d'un hôte ne change pas). On crée le modèle en combinant deux préoccupations (lignes 6-9). Il faut ensuite spécifier le scénario sur lequel on veut étudier le modèle (lignes 11-24). Dans ce scénario, on spécifie les interactions (ou autrement dit, les dépendances dynamiques) entre deux préoccupations (`SIR` et `ThreeSpecies`).

En raison des hétérogénéités causées par les espèces, certains paramètres de `SIR` deviennent hétérogènes. Par exemple, le taux de transmission β est une matrice de 3×3 pour capturer la transmission de l'infection entre les différents groupes d'espèces.

```
  beta_species: #(
      #(0 0.02 0.02)
      #(0.02 0 0)
      #(0.02 0 0))
```

Le suffixe `_species` indique que le paramètre `beta` est une fonction dépendante de la valeur de l'attribut `species` dans l'état courant des individus. De même façon, chaque espèce a un taux différent de mortalité et natalité. Le paramètre `mu` est également une fonction dépendant de la valeur de l'attribut `species` dans l'état courant des individus. Le paramètre `lambda` de SIR est modifié pour prendre en compte les infections causées par différentes espèces, `lambda: #(beta*I sum)` ($\lambda_i = \sum_j \beta_{ij} I_j$). Pour initialiser les valeurs des compartiments du modèle, on utilise également le suffixe `_species` (lignes 23-24). Les valeurs des paramètres et compartiments correspondent aux espèces en fonction des valeurs de l'attribut `species`. Par exemple, `mu_species = [12.17 0.05 0.05]` signifie que la valeur du paramètre `mu` correspondant à trois espèces *mosquito, reservoir1, reservoir2* est respectivement $12.17, 0.05, 0.05$.

Dans les formules mathématiques des modèles en épidémiologie, N représente toujours la taille totale de la population. Ce paramètre est défini de façon implicite comme la taille de la population étudiée. La ligne 20 indique que N n'est plus la taille totale de la population mais la taille d'une sous-population (d'une espèce).

On a défini les préoccupations indépendamment, elles sont réutilisables. Par exemple, on a réutilisé la préoccupation SIR dans les deux exemples ci-dessus. On a également séparé les dépendances éventuelles entre les préoccupations de leur définition. Cela rend le modèle plus facile à changer et à faire évoluer maintenant.

Il est possible de réutiliser non seulement les préoccupations mais également d'autres entités comme scénarios. Supposons par exemple que l'on veut changer le cycle de transmission du modèle, de `SIR` à `SEIR`. On assume également que les valeurs des

paramètres concernant `SIR` ne changent pas. On peut alors réutiliser le scénario dans lequel on a spécifié les dépendances éventuelles entre `SIR` et `ThreeSpecies`. Il suffit de modifier l'entité de composition (ligne 8) et d'ajouter un nouveau scénario pour spécifier les valeurs des nouveaux paramètres qui viennent de la préoccupation `SEIR` (ligne 27-29). Ces deux scénarios sont définis pour le même modèle `VB3Species`. Notons que la simulation doit prendre en compte les deux scénarios (ligne 32).

```
6  Composition VB3Species
7    model: 'VectorBorne';
8    concern: 'SEIR';
9    concern: 'ThreeSpecies'.
26
27 Scenario VBScr2
28   on: 'VB3Species';
29   sigma_species: #(52 121 121).
30
31 Simulation VBStc gillespie
32   scenarios: #(VBScr1 VBScr2);
33   from: 0.0;
34   to: 0.5;
35   step: 1/365.
36
37 Visualization VBViz diagram
38   for: 'VBStc';
39   data: #(I sqrt);
40   xLabel: 'Time (years)'.
```

Dans la section suivante, nous allons présenter d'autres exemples modèles qui se composent de plusieurs préoccupations, notamment les modèles de la grippe aviaire qui ont été abordés dans le chapitre 4.

6.1.3 Modèles de la Grippe Aviaire (GA)

On va commencer par le modèle spatial multi-espèces de GA. Ce modèle a été abordé dans la section 4.1.1 comme un exemple qui motive notre démarche. Ensuite, pour montrer l'intérêt de la séparation des préoccupations auprès de la réduction de l'impact des changements, nous créons certains scénarios pour changer le modèle spatial multi-espèces de GA : (1) Considérons deux souches virales qui cause la GA (2) Considérons la stratégie de quarantaine. Dans ces deux scénarios, nous changeons le modèle spatial multi-espèces en ajoutant la nouvelle préoccupation : (1) Ajouter une préoccupation multi-souches virales (2) Ajouter une préoccupation pour la quarantaine.

Modèle spatial multi-espèces

Dans cette partie, on va définir le modèle spatial multi-espèces de la grippe aviaire qui se compose de trois préoccupations : SEIRS, spatial, multi-espèces. Dans cet exemple, on va définir la préoccupation SEIRS comme une préoccupation indépendante (il faut noter qu'il est possible de définir SEIRS comme une extension de SIR ou SEIR) :

```
1  Concern SEIRS
2      attribute: #(status -> S E I R);
```

```
3       parameters: #(beta lambda gamma mu sigma nu);
4       lambda: #(beta*I/N);
5       transitions: #(
6           S -- lambda --> E.
7           E -- sigma --> I.
8           I -- gamma --> R.
9           R -- nu --> S.
10          status -- mu --> Empty.
11          Empty -- mu --> S).
```

La population étudiée se décompose en deux espèces : *humains* et *oiseaux*. La préoccupation multi-espèces est donc définie de la façon suivante :

```
1  Concern TwoSpecies
2    attribute: #(species -> humans birds).
```

On va formuler ensuite un modèle spatial où les individus se trouvent dans des zones géographiques différentes. L'infection peut être transmise entre les zones par la mobilité des individus. On suppose que l'effet spatial est étudié sur cinq pays du Sud-Est asiatique : Thaïlande, Vietnam, Cambodge, Laos et Birmanie.

Dans notre exemple, la préoccupation spatiale est formulée par la matrice d'adjacence qui représente les frontières géographiques de ces cinq pays. Nous utilisons une entité `Map` pour le faire :

```
1  Map IndoChina
2       for: #(country -> Thailand Laos Vietnam Cambodia MyanmarBurma);
3       borders: #(
4            #(0 1 0 1 1)
5            #(1 0 1 1 1)
6            #(0 1 0 1 0)
7            #(1 1 1 0 0)
8            #(1 1 0 0 0)
9  ).
10 Concern Spatial
11      maps: 'IndoChina';
12      withTransitionRate: #(rho).
```

Les transitions de la préoccupation spatiale sont formulées à partir des éléments non-zéro de la matrice d'adjacence. Dans cet exemple, on suppose que toutes les transitions ont le même taux `rho`, taux de déplacement des individus d'un pays à un autre. Il est cependant possible de spécifier explicitement le taux pour chaque transition.

On compose enuite les préoccupations. Pour étudier ce modèle, on définit une entité scénario dans laquelle les dépendances éventuelles entre les préoccupations sont établies.

```
1  KendrickModel Influenza.
2
3  Composition AvianInfluenza
4       model: 'Influenza';
5       concern: 'SEIRS';
6       concern: 'MultiSpecies';
7       concern: 'Spatial'.
```

```
8
9  Scenario AIScr1
10     on: 'AvianInfluenza';
11     populationSize: 27500;
12     mu_species: #(0.000365 0.00137);
13     beta_species: #(
            #(0 0.21)
            #(0 0.42));
14     gamma_species: #(0.25 0.233);
15     sigma_species: #(0.5 0.67);
16     nu: 0.00274;
17     rho_species: #(0.03 0.1);
18     lambda: #(beta*I_country/N sum);
19     N: #(species_country);
20     S_species_country: #(
            #(500 500 500 500 500)
            #(4990 5000 5000 5000 5000));
21     I_species_country: #(
            #(0 0 0 0 0)
            #(10 0 0 0 0)).
```

Comme le modèle multi-espèces qui a été présenté dans la section précédente, les paramètres du modèle sont également suivis par des suffixes (tels que `_species`, `_country`, `_species_country`) afin de représenter les taux de transition fonctionnels. Dans cet exemple, pour simplifier le modèle, on suppose que les paramètres ont les mêmes valeurs pour tous les pays. Par exemple, la ligne 12 signifie que la valeur du paramètre `mu` est une fonction dépendante de la valeur de l'attribut `species` dans l'état courant des individus (`mu` a les mêmes valeurs pour les cinq pays). Comme nous avons présenté dans Section 4.1.1, la probabilité d'infection d'un individu susceptible de l'espèce s dans le pays p est $\lambda_{ps} = \sum_{i}^{species} \beta_{isp} I_{pi}/N_{pi}$. Cette formule est donnée dans la ligne 20. Le paramètre N est défini comme la taille des individus d'une même espèce d'un même pays (ligne 21). Enfin, on initialise les valeurs pour les compartiments du modèle (lignes 22-27). Le suffixe `_species_country` signifie qu'il faut donner une valeur pour chaque espèce dans chaque pays (donc une matrice de taille 2x5).

Pour interpréter le modèle, on définit une entité de simulation déterministe, puis on visualise les résultats obtenus.

```
1  Simulation AIRK rungeKutta
2    scenario: 'AIScr1';
3    from: 0;
4    to: 500;
5    step: 1.
6
7  Visualization AIViz1 diagram
8    for: 'AIRK';
9    data: #(I_species);
10   legendTitle: 'Total of Infectious';
11   legends: #('humans' 'birds');
12   xLabel: 'Time (days)'.
13
14 Visualization AIViz2 map
15     for: 'AIRK';
16     data: #(country I_bird_country peakOfEpidemic).
```

Deux entités de visualisation ont été définies : la première pour visualiser la dynamique

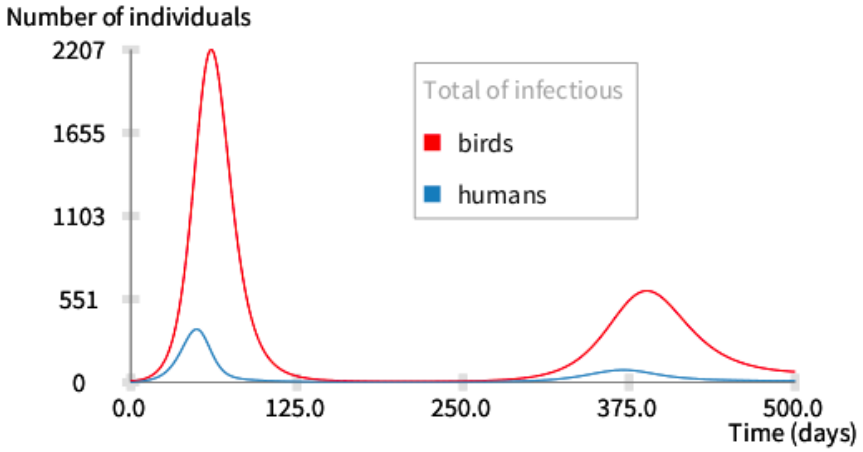


Figure 6.1 – Les dynamiques déterministes du nombre total des infectieux comptés pour chaque espèce dans tous les pays. $S_{b_1} = 4990, I_{b_1} = 10, S_{b_p} = 5000, I_{b_p} = 0, S_{h_p} = 500, I_{h_p} = 0,\ \beta_b = [00.42], \beta_h = [00.21], 1/\mu_b = 2\,years, 1/\mu_h = 75\,years, 1/\sigma_b = 1.5\,days, 1/\sigma_h = 2\,days, 1/\gamma_b = 4.3\,days, 1/\gamma_h = 4\,days,\ \rho_b = 0.1,\ \rho_h = 0.03,\ \forall p \in [1..5]$.

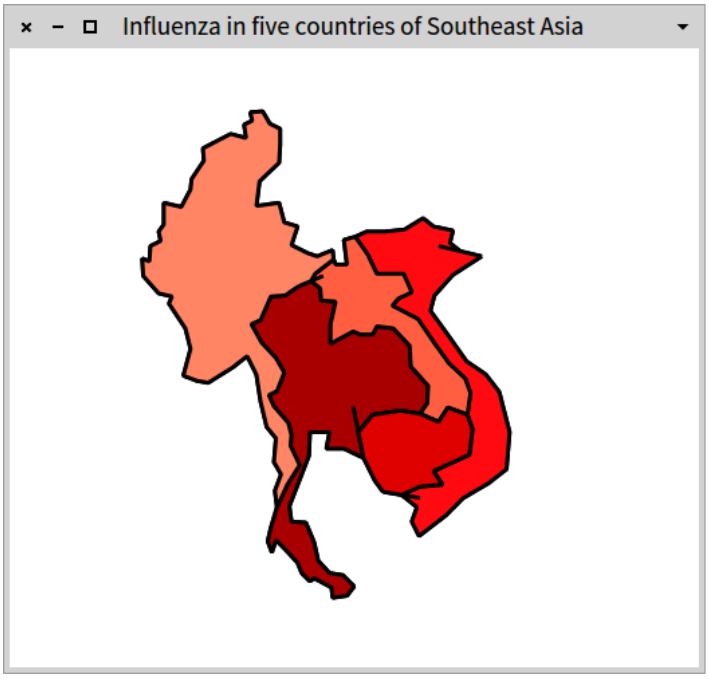


Figure 6.2 – L'affichage d'une carte des cinq pays d'Asie du Sud-Est concernés. Les couleurs de chaque pays représente le nombre maximal des oiseaux infectés

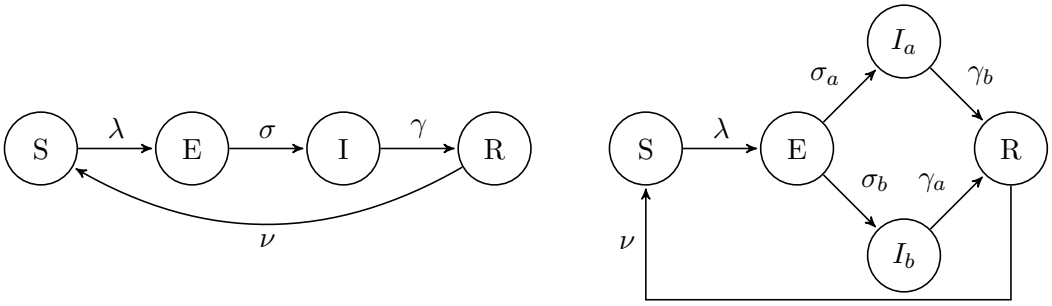


Figure 6.3 – Les deux automates qui représentent la préoccupation SEIRS (à gauche) et une préoccupation deux-souches (à droite).

des infectieux (Figure 6.1), la deuxième pour afficher la propagation spatiale de la grippe aviaire sur les cinq pays d'Asie du Sud-Est (Figure 6.2).

Modèle spatial multi-espèces en considérant deux souches virales

Supposons que nous voulons faire la distinction entre différentes souches de pathogène parce que l'une souche est plus virulente ou résiste mieux à tel ou tel traitement.

On construit une préoccupation qui introduit deux souches pathogènes pour le modèle ci-dessus, où l'infection causée par l'une souche confère une immunité permanente pour les deux, que l'on appelle aussi *une immunité complètement croisée* [69].

Cette préoccupation divise la classe infectieuse I en deux sous-classes I_a et I_b pour indiquer deux status infectieux (causés par deux souches pathogènes).

Elle va être définie comme une extension d'une préoccupation épidémique (dans cet exemple, c'est la préoccupation SEIRS) en appliquant l'opérateur `splitStatus`. Cet opérateur va transformer `SEIRS` en `SEIaIbRS` (Figure 6.3) en ajoutant deux statuts I_a, I_b et les transitions $E \to I_a$, $E \to I_b$, $I_a \to R$ et $I_b \to R$ (les transitions $E \to I$ et $I \to R$ sont enlevées).

```
1  Concern SEIRSTwoStrains
2    extends: 'SEIRS';
3    parameters: #(betaA betaB);
4    splitStatus: #(I index #(a b)).
```

Le modèle GA que l'on a défini dans la section précédente devient :

```
1  KendrickModel Influenza.
2
3  Composition 'AvianInfluenza'
4    model: 'Influenza';
5    concern: 'SEIRSTwoStrains';
6    concern: 'TwoSpecies';
7    concern: 'Spatial'.
8
9  Scenario AIScr1
10   on: 'AvianInfluenza';
```

```
11    populationSize: 27500;
12    mu_species: #(0.000365 0.00137);
13    betaA_species: #(
        #(0 0.21)
        #(0 0.42));
14    betaB_species: #(
        #(0 0.021)
        #(0 0.042));
15    gamma_species_Ia: #(0.25 0.233);
16    gamma_species_Ib: #(0.0025 0.0023);
17    sigma_species_Ia: #(0.5 0.67);
18    sigma_species_Ib: #(0.005 0.0067);
19    nu: 0.00274;
20    rho_species: #(0.03 0.1);
21    lambda: #((betaA*(Ia_country/N))+(betaB*(Ib_country/N)) sum);
22    N: #(species_country);
23    S_species_country: #(
        #(500 500 500 500 500)
        #(4990 5000 5000 5000 5000));
24    Ia_species_country: #(
        #(0 0 0 0 0)
        #(9 0 0 0 0));
25    Ib_species_country: #(
        #(0 0 0 0 0)
        #(1 0 0 0 0)).
```

L'entité de composition du modèle est modifiée pour ajouter la nouvelle préoccupation (ligne 5). On modifie ensuite l'entité de scénario pour prendre en compte les changements provenant de la nouvelle préoccupation.

En raison des hétérogénéités causées par les souches différentes, les paramètres σ, γ de la préoccupation SEIRS ont une valeur différente en fonction de I_a et I_b (Figure 6.3). Dans le scénario, ces paramètres ont été spécifiés pour I_a (lignes 15-17) et I_b (lignes 16-18) en utilisant les suffixes `_Ia` et `_Ib`. La probabilité d'infection doit prendre en compte l'infection causée par deux souches $\lambda_{ps} = \sum_{i}^{species} \beta_{a_{isp}} I_{api}/N_{pi} + \beta_{b_{isp}} I_{bpi}/N_{pi}$ (ligne 21). On doit spécifier également les valeurs initiales pour les nouveaux compartiments (lignes 24-25).

Dans ce contexte, l'ajout de la préoccupation multi-souches entraîne des changements dans deux entités du modèle, l'entité de composition (`AvianInfluenza`) et l'entité de scénario (`AIScr1`). En comparant avec le même modèle en MATLAB (voir la figure 4.2), on peut constater qu'au lieu d'avoir à effectuer des changements dans tout le code à un niveau d'abstraction très bas, dans le modèle exprimé en Kendrick, la modularité du modèle permet de localiser les changements dans certaines entités. De plus, les préoccupations du modèle spatial multi-espèces sont réutilisées dans le nouveau modèle, elles ne sont pas donc affectées par les changements.

Modèle spatial multi-espèces en considérant la stratégie de quarantaine

Dans cette partie, on ajoute une stratégie de contrôle, la quarantaine, pour isoler des individus qui peuvent être infectés parce qu'ils ont eu un contact avec un individu infectieux. Les individus en quarantaine sont représentés par le statut Q. La figure 6.4 représente la préoccupation SEIRS avec quarantaine.

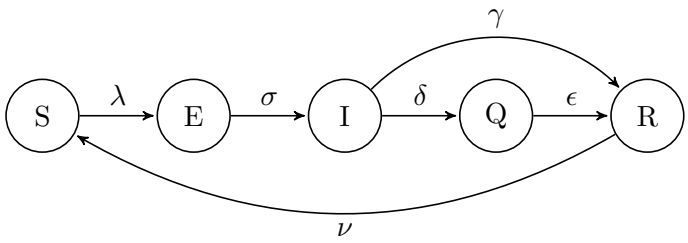


Figure 6.4 – L'automate de la préoccupation SEIRS avec quarantaine (SEIRSQ)

Cette préoccupation va être définie comme une extension de SEIRS :

```
1  Concern SEIRSQ
2    extends: 'SEIRS';
3    parameters: #(delta epsilon);
4    addStatus: #(Q);
5    addTransition: #(I -- delta --> Q);
6    addTransition: #(Q -- epsilon --> R);
7    addTransition: #(Q -- mu --> Empty);
8    lambda: #(beta*I/(N-Q)).
```

L'entité de composition du modèle spatial multi-espèces (`AvianInfluenza`) est modifiée pour ajouter la préoccupation `SEIRSQ`. Dans cet exemple, on réutilise le scénario définit pour le modèle spatial multi-espèces (`AIScr1`). Tout ce qui concerne les paramètres et les dépendances éventuelles entre la préoccupation `SEIRSQ` et les autres sont définis dans un autre scénario (`AIScr2`, lignes 23-27).

```
1  KendrickModel Influenza.
2
3  Composition AvianInfluenza
4    model: 'Influenza';
5    concern: 'SEIRSQ';
6    concern: 'TwoSpecies';
7    concern: 'Spatial'.
8
9  Scenario AIScr1
10 ...
22
23 Scenario AIScr2
24   on: 'AvianInfluenza';
25   delta_species: #(0.068 0.055);
26   epsilon_species: #(0.096 0.082);
27   rho_species_Q: #(0.03 0.1 0);
28   lambda: #(beta*(I_country/(N-Q_country)) sum).
```

Supposons que les individus qui sont dans la classe Q sont interdit de voyage. Le taux de déplacement ρ des individus quarantaines a donc la valeur 0 (ligne 27).

L'entité de simulation de ce modèle doit être également modifiée pour prendre en compte le scénario `AIScr2` :

```
1  Simulation AIRK rungeKutta
2    scenarios: #(AIScr1 AIScr2);
```

```
3    from: 0;
4    to: 500;
5    step: 1.
6
7  Visualization AIViz1 diagram
8    for: 'AIRK';
9    data: #(I_species);
10   legendTitle: 'Total of Infectious';
11   legends: #('humans' 'birds');
12   xLabel: 'Time (days)'.
```

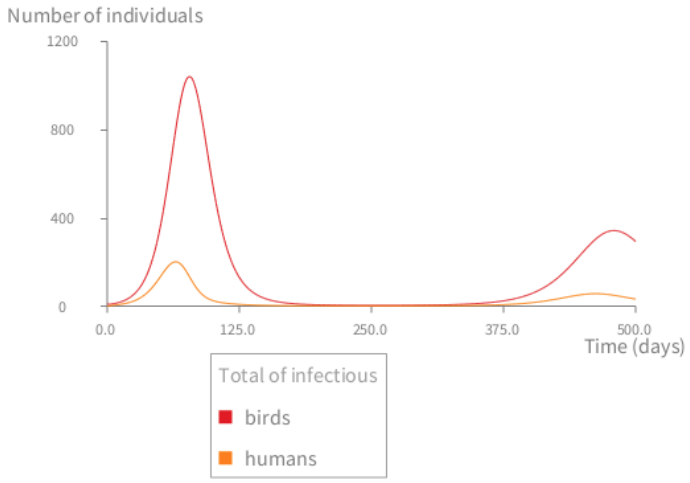


FIGURE 6.5 – Dynamiques déterministes du nombre total des infectieux comptés pour chaque espèce dans tous les pays en considérant la quarantaine.

En observant les figures 6.1, 6.5, on peut voir que grâce à la quarantaine mise en place, le nombre total des infectieux dans chaque espèces se réduit de manière importante.

6.2 Validation

Notre validation s'effectue à deux niveaux.

La premier niveau, le plus simple est de l'implémentation des simulations déterministe, stochastique et multi-agents dans KENDRICK. L'objectif est pour montrer que les résultats donnés par les simulations de KENDRICK sont identiques à ceux donnés par d'autres outils (i.e, MATLAB). La comparaison avec d'autres outils se fait pour le cas de la simulation déterministe. Nous comparons ensuite la simulation déterministe à celle stochastique dans le but de montrer que le résultat donné par la simulation déterministe s'approche au moyenne de ceux donnés par la simulation stochastique. Pour comparer les simulations stochastique et multi-agents, nous faisons un test statistique pour montrer que les résultats donnés par deux sortes de simulation en KENDRICK ne sont pas différents du point de vue statistique.

La deuxième validation, plus complexe, vise à réaliser une validation de notre démarche en la comparant à la démarche classique en MATLAB sur une série des modèles de plus en plus complexes de façon à mesurer un certain nombre de critères liés à la séparation des préoccupations ou à leur réutilisabilité.

6.2.1 La validation de l'implémentation de Kendrick

Pour valider l'implémentation du langage et de la plateforme de simulation, nous allons comparer les résultats déterministes d'un modèle en Kendrick avec ceux du même modèle en MATLAB. Il suffit de le faire sur un ou deux modèles pour la comparaison car tous les modèles Kendrick sont interprétés en utilisant les mêmes modules sémantiques (qui implémentent les trois paradigmes de simulation). On choisit donc le modèle SEIR de la rougeole (qui se compose d'une seule préoccupation) et le modèle multi-espèces (qui se compose de deux préoccupations).

Tout d'abord, on va valider l'implémentation de la simulation déterministe en comparant le résultat déterministe d'un modèle en Kendrick avec ceux du même modèle en MATLAB. Considérons le modèle SEIR de la rougeole, les résultats déterministes (Figure 3.6, Figure A.1) montrent que Kendrick produit des résultats identiques à ceux produits par le même modèle en MATLAB. Les mêmes résultats sont donnés par les deux versions (en Kendrick et MATLAB) du modèle multi-espèces. Les scripts MATLAB de deux modèles (modèle SEIR de la rougeole et modèle multi-espèces) sont donnés en annexes à la fin du manuscrit.

Pour valider les simulation stochastique et multi-agents, on va comparer les dynamiques déterministes avec celles obtenues par les simulations stochastiques (en utilisant l'algorithme direct de Gillespie) et multi-agents d'un même modèle en Kendrick (voir figure 6.6). Le résultat de la comparaison du modèle de la rougeole (dans deux cas, sans et avec la vaccination) est vu dans les diagrammes en haut, celles en bas affichent les résultats du modèle multi-espèces pour chaque espèce. Les dynamiques sont similaires dans tous les cas, suggérant que l'implémentation des simulations stochastiques et multi-agents est correcte.

Modèles		Propriétés épidémiologiques		
		Pic de l'épidémie	Moment du pic	Durée de l'épidémie
Rougeole	Sans Vaccination	0.8762	0.2471	0.5246
	Avec Vaccination	0.8928	0.2467	0.6262
Maladie vectorielle	Moustiques	0.7920	0.2700	0.7112
	Réservoir 1	0.6272	0.3927	0.8643
	Réservoir 2	0.7920	0.1122	0.2202

Table 6.1 – *Valeurs-p* du test de Kolmogorov-Smirnov effectué sur les résultats de simulation de deux modèles en utilisant des propriétés clés.

De plus, on va comparer les résultats obtenu par la simulation stochastique avec ceux obtenu par la simulation multi-agents du même modèle en Kendrick. L'objectif est pour montrer que ces deux simulations donnent les résultats similaires (comme souhaité car ces deux simulations reposent sur les mêmes principes - les chaînes de Markov à temps continu (voir la section 4.2)).

À partir des résultats, on extrait certaines propriétés spécifiques de l'épidémie : le point culminant de l'épidémie (ou le pic de l'épidémie), le temps auquel l'épidémie a atteint son point culminant, la durée de l'épidémie. Pour chaque propriété, on obtient donc deux distributions (l'une de la simulation stochastique et l'autre de la simulation multi-agents) contenant 200 éléments. Afin de comparer les deux distributions, un test

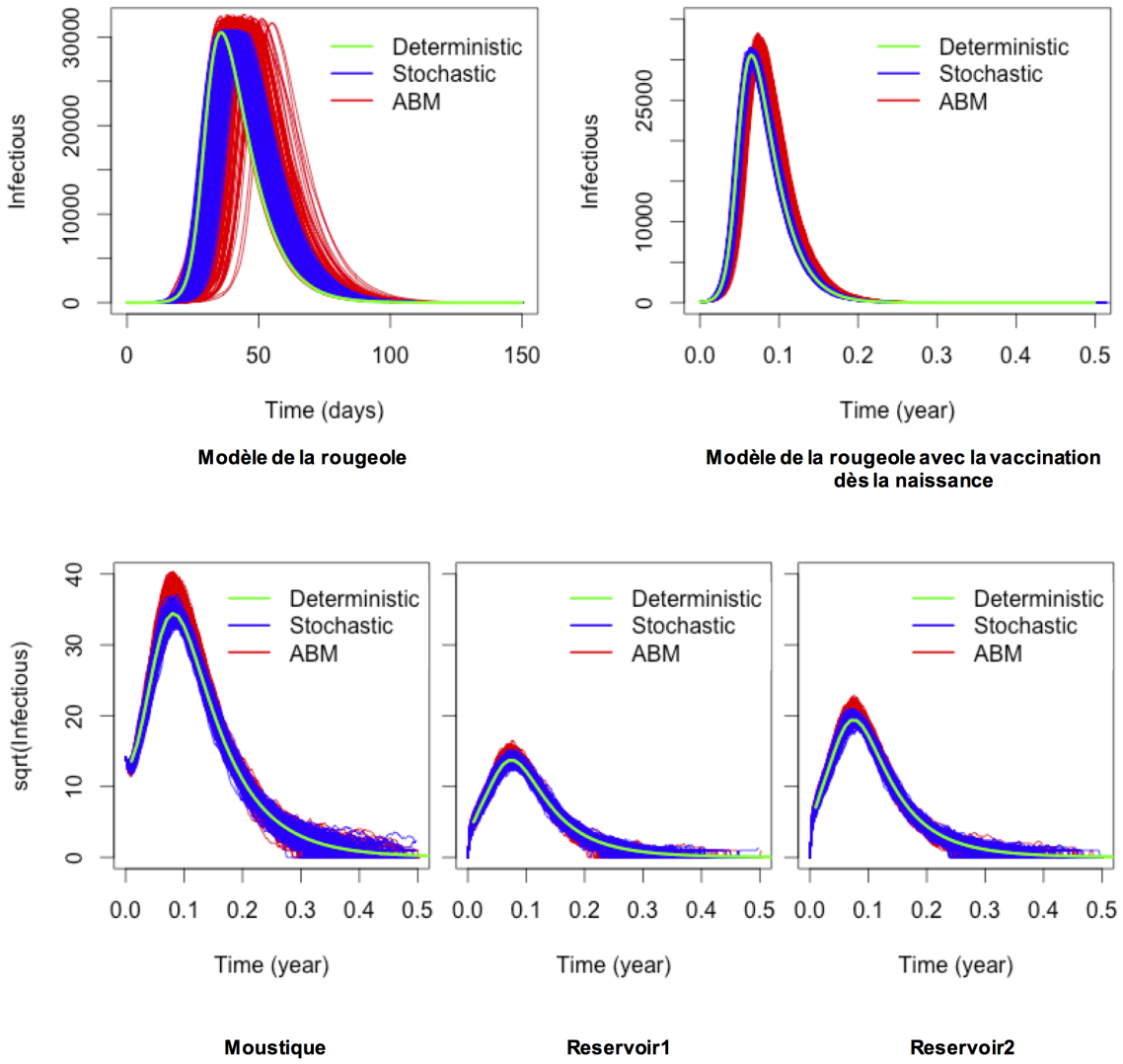


FIGURE 6.6 – Comparaison des dynamiques déterministes (courbe verte), stochastiques (courbe bleu) et multi-agents (courbe rouge) des modèles en KENDRICK. Les deux diagrammes en haut représentent les résultats du modèle de la rougeole (sans la vaccination à gauche et avec la vaccination à droite). Les diagrammes en bas représentent les résultats du modèle multi-espèces.

statistique de Kolmogorov-Smirnov est utilisé.

Rappel sur le test statistique de Kolmogorov-Smirnov Un test de Kolmogorov-Smirnov (KS) est un test d'hypothèse utilisé pour comparer la distribution de deux échantillons statistiques. Il détermine si deux échantillons suivent une même loi en se basant sur la fonction de répartition empirique [1] des échantillons [111].

La distribution (ou la loi) d'une variable aléatoire X est décrite par sa fonction de répartition :

$$F(x) = \mathbb{P}(X \leq x)$$

La fonction de répartition empirique associée au n échantillon $X_1, X_2, ..., X_n$ est :

$$F_n(x) = \frac{1}{n} \sum_{i=1}^{n} 1_{X_i \leq x}$$

Pour n observations d'une variable aléatoire X, et m observations d'une variable aléatoire Y, on teste l'hypothèse nulle : « Les fonctions de répartition F_X de X et F_Y de Y sont égales avec un risque d'erreur α ».

Le test statistique est basé sur un écart en valeur absolue entre les deux fonctions de répartition empiriques de deux suites d'observations.

$$D_{n,m} = sup_{x \in \mathbb{R}} |F_X(x) - F_Y(x)|$$

Il consiste à rejeter l'hypothèse si $D_{n,m} \geq d_{n,m,1-\alpha}$ ($D_{n,m}$ est appelé *valeur critique*). La loi de cette statistique de test est donnée dans la table de Smirnov. À côté de la valeur critique, on peut interpréter la valeur-p du test statistique, c'est le plus petit seuil de significativité pour lequel l'hypothèse nulle est acceptée. L'utilisation d'une valeur-p est généralement plus simple à utiliser que la valeur critique (par simple consultation de la table de Smirnov). La valeur-p peut être calculée en utilisant la valeur critique [111].

Dans le contexte de la validation, nous avons utilisé le test statistique KS intégré dans le logiciel R [2] pour comparer les deux observations correspondantes à chaque propriété mentionnée ci-dessus. Le logiciel donne à la fois la valeur critique et la valeur-p. Nous allons examiner la valeur-p pour interpréter la sortie du test (si $p \geq \alpha$, on va accepter l'hypothèse nulle).

Les valeur-p sont toutes supérieures à 0.05 ($\alpha = 0.05$) (voir table 6.1) ce qui montrent qu'il n'y a pas de différences statistiques entre les résultats stochastiques et multi-agents. Cela suggère que les deux algorithmes de simulation ont été correctement implémentés.

6.2.2 Validation de l'intérêt de la séparation des préoccupations en épidémiologie

Dans cette section, nous allons valider l'approche proposée dans le chapitre 4 à savoir : la séparation et la composition des préoccupations afin de construire des modèles

1. En anglais : Cumulative Distribution Function (CDF)
2. `https://www.r-project.org/`

épidémiologiques de façon plus modulaire.

Nous nous intéressons aux deux problèmes de validation qui suivent : (1) Évaluer la qualité du méta-modèle mathématique afin d'exprimer les principales préoccupations de l'épidémiologie ; (2) Évaluer si les difficultés liés à l'enchevêtrement et de la dispersion du code, ont été adressées à l'aide de l'approche proposée.

Évaluation de la qualité du méta-modèle mathématique

Dans le chapitre 4, nous avons proposé un méta-modèle mathématique qui exprime la dynamique d'évolution d'une population divisée en compartiments infectieux. Ce méta-modèle fournit un ensemble de concepts mathématiques pour décrire des modèles et des préoccupations en épidémiologie.

Les concepts mathématiques introduits sont très généraux et n'imposent aucune implémentation particulière. Par exemple, pour capturer la décomposition de la population étudiée, nous avons utilisé le concept de la relation d'équivalence.

La définition mathématique des modèles les rend beaucoup plus facile à être interprétés dans différentes approches de simulation (par exemple, déterministe sous forme des équations différentielles ou stochastique - typiquement comme des processus de Markov à temps continu (CTMC)).

En outre, comme nous avons montré dans les sections 4.7.1, 4.7.2, le méta-modèle proposé est capable d'exprimer les préoccupations les plus communes de l'épidémiologie. Ces préoccupations transforment les modèles par raffinement et/ou modification de leurs éléments. Par exemple, certaines préoccupations raffinent la relation d'équivalence du modèle pour diviser la population en plus petite population, d'autres modifient la matrice de taux de transition du modèle etc. Le méta-modèle permet d'exprimer toutes ces transformations.

Une autre question qui se pose, porte sur la description des dépendances entre les préoccupations. Comme on l'a dit dans le chapitre 4, on distingue deux types de dépendances entre préoccupations : dépendances non-structurelles (ou dépendances éventuelles qui apparaissent lorsque deux préoccupations se présentent dans un modèle) et dépendances structurelles. Les travaux de Plateau et *al.*[95, 96] ont montré que les dépendances entre deux automates stochastiques (qui représentent des processus de Markov) peut être exprimées soit par des taux fonctionnels pour les transitions, soit par des transitions de déclenchement (une transition dans un automate fait se déclencher une transition dans un autre automate). Nous avons utilisé seulement les taux fonctionnels des transitions pour exprimer des dépendances éventuelles entre les préoccupations. Ce choix est essentiellement dû à la façon dont les modèles épidémiologiques sont formulés. En effet, on suppose que pendant un intervalle de temps suffisamment petit, au maximum un seul changement dans l'état se produit, donc seulement un automate change son état. L'ensemble de ces considérations suggère que le méta-modèle est suffisamment exhaustif par rapport au domaine d'étude.

Validation de l'approche proposée pour la séparation/composition des préoccupations

L'objectif de cette validation est d'examiner si les problèmes issus de l'enchevêtrement et de la dispersion des préoccupations en épidémiologie sont adressés à l'aide de l'approche proposée.

On va mesurer la dépendance entre les préoccupations dans un modèle. Si il y a réduction des dépendances peut suggérer que les préoccupations sont bien séparées. Ces difficultés se produisent le plus souvent lorsque l'on fait évoluer les modèles, surtout par l'ajout de nouvelles préoccupations. On va évaluer également l'effort nécessaire aux changements lors du passage d'un modèle à un autre dans des scénarios d'évolution. Afin de pouvoir montrer l'efficacité de l'approche proposée, on va comparer les résultats obtenus à ceux en utilisant MATLAB.

On a choisi MATLAB comme référentiel de comparaison car il s'agit d'un langage utilisé très fréquemment par les épidémiologistes dans leurs publications comme par exemple dans le livre de Keeling dont nous avons utilisé beaucoup de modèles dans le cadre de cette thèse [1].

Procédures de validation Nous avons préparé une série de cinq modèles de plus en plus complexes se focalisant sur la grippe aviaire par l'ajout au fur et à mesure de préoccupations :

(1) Modèle 1 est le modèle SEIRS où la maladie est étudiée au sein d'une population d'oiseaux. Il y a seulement une préoccupation - SEIRS.

(2) Modèle 2 est créé à partir du modèle 1 en introduisant une population avec ddeux espèces (humains et oiseaux). Modèle 2 contient donc deux préoccupations : SEIRS et multi-espèces.

(3) Modèle 3 ajoute la préoccupation spatiale et on a alors 3 préoccupations.

(4) Modèle 4 étend le modèle 3 en introduisant deux souches de pathogènes. Le modèle 4 contient donc quatre préoccupations : SEIRS, multi-espèces, spatiale et multi-souches.

(5) Modèle 5 introduit la stratégie de quarantaine au modèle 3 et on a également quatre préoccupations : SEIRS, multi-espèces, spatiale et quarantaine.

Les trois derniers modèles ont été présentés dans la section 6.1.3. Afin d'évaluer l'effort des changements, nous créons quatre scénarios d'évolution. Pour chaque scénario, un nouveau modèle est obtenu en effectuant des changements sur le précédent :

(S1) Du Modèle 1 au Modèle 2

(S2) Du Modèle 2 au Modèle 3

(S3) Du Modèle 3 au Modèle 4

(S4) Du Modèle 3 au Modèle 5

Organisation des modèles Comme ce sont les modèles le plus simples à exprimer en MATLAB, on se focalisera ici uniquement sur des modèles déterministes. Cinq modèles mentionnés ci-dessus ont été implémentés en Kendrick et MATLAB. Nous

avons repris les modèles du livre de Keeling et al. [1] pour MATLAB. On peut donc considérer que ce code est proche de ce que pourrait faire un chercheur du domaine.

Chaque modèle MATLAB contient deux scripts (appelés entités par la suite) qui se trouve dans des fichiers séparés : un script principal *main* et un script *fonction* qui contient les fonctions utilisés dans le premier script. Le script *main* d'un modèle comprend les déclarations et les instantiations des variables (une variable exprime un paramètre ou un compartiment). Le deuxième script joue le rôle d'un moteur de simulation qui calcule les EDO du modèle pour produire des résultats numériques et est utilisé par le premier script. MATLAB offre également la possibilité de programmer en orienté objet, mais ce n'est pas ce font les épidémiologistes la plupart du temps et donc nous n'en feront pas usage ici.

Nous avons également implémentés ces 5 modèles en KENDRICK. Chaque modèle comprend des entités telles que : `Concerns` (une ou plusieurs plusieurs entités liées à des préoccupations), `Composition`, `Scenario`, `Simulation` et `Visualization`.

Métriques On définit des métriques pour mesurer la dépendance entre les préoccupations dans un modèle et pour estimer l'effort nécessaire pour faire évoluer un modèle (i.e coût en terme de changements dans le code pour le modélisateur).

(a) Métriques de dépendance.

Les métriques suivantes sont proposées pour évaluer la dépendance entre les entités dans un modèle :

- `Afferent Coupling` (`Ca`)
- `Efferent Coupling` (`Ce`)
- `Instability` (`I`)

`Afferent Coupling`, `Efferent Coupling` et `Instability` sont des métriques souvent utilisées pour mesurer la stabilité des modules d'un programme[82]. Dans ce travail, nous utilisons ces métriques pour mesurer le nombre de dépendances entrantes et sortantes d'une entité. Nous considérons qu'une dépendance à un élément nommé X (X peut être le noms ou les valeurs des attributs, le nom des paramètres, etc.) si il y a une occurrence de son nom dans d'autres entités.

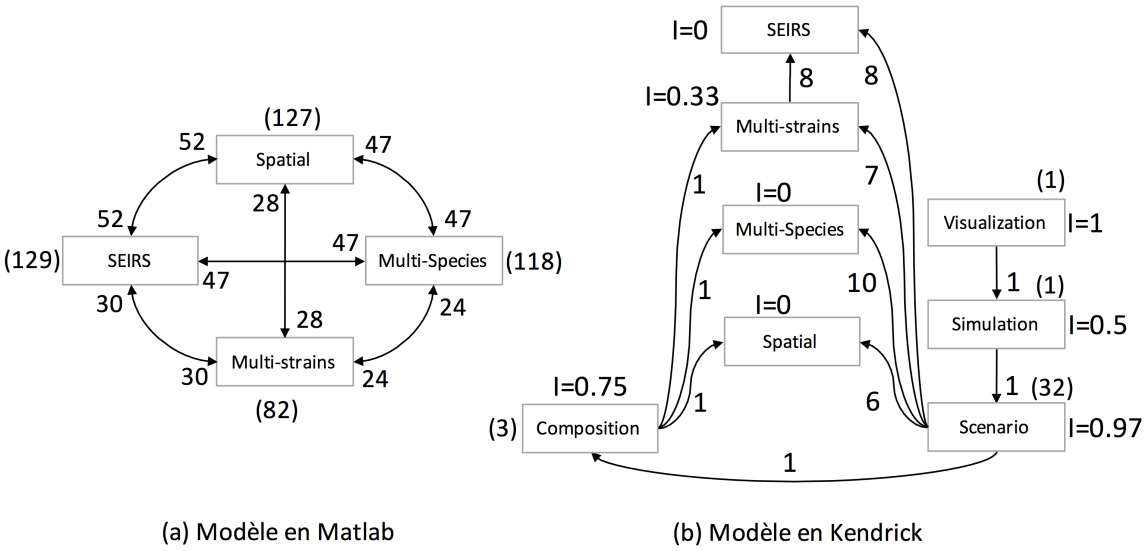
Les métriques sont calculées de la façon suivante :

- `Ca` : Le nombre d'occurrences qui dépendent des éléments de l'entité étudiée dans d'autres entités
- `Ce` : Le nombre d'occurrences qui dépendent des éléments des autres entités dans l'entité étudiée
- `I = Ce / (Ca + Ce)`. Cette métrique a une valeur entre 0 et 1. L'entité est stable lorsque $I = 0$, instable si $I = 1$.

(b) Les métriques pour estimer l'effort lié aux changements.

Pour évaluer l'effort lié aux changements, on va calculer des métriques telles que : la taille des modèles (en nombre de lignes de code `#LoCs` et en nombre de caractères `#characters`) et la taille moyenne des entités en nombre de caractères `#characters`.

Le nombre d'entités (`#entités`) dans chaque modèle est également utile comme un indicateur de la modularité des modèles. Lorsque les modèles sont bien modulaires, les modifications d'une entité peuvent être localisées dans un seul endroit plutôt que d'être dispersées à plusieurs endroits.



(a) Modèle en Matlab (b) Modèle en Kendrick

Figure 6.7 – Graphes de dépendances du Modèle 4 écrit en (a) MATLAB (b) Kendrick. Les arêtes qui se dirigent vers un nœud représentent ses dépendances entrantes (Ca), alors que ceux en sortant représentent ses dépendances sortantes (Ce). Le nombre total de dépendances sortantes de chaque nœud est entre parenthèses.

On calcule l'effort de changements pour chaque scénario d'évolution pour évaluer la facilité de faire des changements sur ces modèles. Précisément, on va compter le nombre de lignes de code qui ont été changées/ajoutées, le nombre d'entités affectées/réutilisées/ajoutées après avoir fait évoluer le modèle.

Résultats de la validation

Mesurer la dépendance entre les préoccupations dans un modèle Puisque les cinq modèles dans la série ont été créés en ajoutant au fur et à mesure les préoccupations, nous choisissons un modèle avec autant de préoccupations que possible pour mesurer les dépendances entre elles. On va présenter les résultats pour le modèle 4, le modèle 5 donnant des résultats similaires.

Les métriques `Ca`, `Ce` et `I` ont été calculées pour les deux versions du modèle 4 (en Kendrick et en MATLAB). Nous avons visualisé les résultats obtenus sous forme de graphes de dépendances (voir figure 6.7). Les arêtes qui se dirigent vers un nœud représentent ses dépendances entrantes (Ca), alors que ceux en sortant représentent ses dépendances sortantes (Ce). Dans le cas de Kendrick , chaque nœud du graphe correspond à une entité du modèle. Tandis que pour MATLAB, un nœud représente une préoccupation qui a été implicitement introduites dans le modèle. Les `Ca` et `Ce` ont été calculés pour chaque préoccupation du modèle MATLAB de telle sorte qu'il y a une dépendance entre deux préoccupations si elles ont été introduites dans le même identifiant ou fonction (`Ca`, `Ce` de chaque préoccupation augmentent de 1).

La figure 6.7 (a) montre que les préoccupations du modèle MATLAB dépendent fortement les unes des autres. En effet, la préoccupation *Multi-species* a un total de 118 dépendances sortantes qui proviennent de trois autres préoccupations. Le nombre total de dépendances des préoccupations *SEIRS, Spatial, Multi-strains* est respectivement

de 129, 127 et 82. Toutes les préoccupations du modèle sont mélangées, ce qui suggère que le modèle MATLAB est monolithique et que le couplage est fort.

Comme on peut voir dans dans la figure 6.7 (b), toutes les entités comme *SEIRS, Spatial, Multi-species*, à l'exception de *Multi-trains*, ont une valeur de `I = 0` (parce que `Ce = 0`). Elle sont donc complètement stables et ne dépendent d'aucune autre entité. L'entité *Multi-strains* a une valeur de `Ce = 8` car elle a des dépendances structurelles avec l'entité *SEIRS*.

On peut également constater que les entités qui changent plus rapidement dépendent de celles qui sont plus stables. En effet, l'ordre de dépendance entre les entités est *Visualisation* → *Simulation* → *Scenario* → *Composition* → *Concern* (→ signifie *dépend de*). Par conséquent, il est possible de définir différentes entités de visualisation pour une simulation (par exemple diagrammes, cartes, etc.), différentes entités de simulations pour un scénario (par exemple déterministe, stochastique) ainsi que différents scénarios pour un seul modèle. En outre, l'entité de scénario dépend également des entités liées aux préoccupations. Elle a un total de 32 dépendances sortantes, qui proviennent principalement des préoccupations : 6 dépendances à `Spatial`, 10 dépendances à `Multi-species`, 7 dépendances à `Multi-strains` et 8 dépendances à `SEIRS`. Ces dépendances sont dues à l'initialisation des paramètres et des compartiments du modèle, ce qui expriment les dépendances non-structurelles entre les préoccupations.

Il est possible de conclure que les dépendances entre les entités liées aux préoccupation ont été évitées, ce qui suggère que les préoccupations ne sont plus aussi entremêlées et dispersées que dans le modèle MATLAB.

(b) Mesurer l'effort pour changer un modèle

		Taille du Modèle `#LoCs`	Taille du Modèle `#Characters`	T. moye. des Entités `#Characters`	`#Entités`
Modèle 1	M	28	477	238.5	2
	K	33	551	110.2	5
Modèle 2	M	34	640	320	2
	K	41	823	137.1	6
Modèle 3	M	62	1602	801	2
	K	53	1087	155.2	7
Modèle 4	M	75	2195	1097.5	2
	K	59	1400	175	8
Modèle 5	M	69	1863	931.5	2
	K	63	1351	168.8	8
Moye. de 5 Modèles	M	53.6	1355.4	677.7	2
	K	49.8	1042.4	153.3	6.8

Table 6.2 – Comparaison entre modèles MATLAB et Kendrick

La table 6.2 montre la taille des cinq modèles en Kendrick et en MATLAB. Les modèles simples (i.e comportant peu de préoccupations) sont de taille plus faible en MATLAB. En effet, Modèle 1 (*SEIRS*) et Modèle 2 (*Multi-espèces SEIRS*) en MATLAB sont plus petits que ceux en Kendrick (en terme de `#LoCs` et `#characters`).

Néanmoins quand le nombre de préoccupations augmentent, les modèles Kendrick ont une taille inférieure à ceux de MATLAB (voir Modèles 3, 4 et 5).

On voit bien que le passage de Modèle 1 à Modèle 4 (avec l'ajout de trois préoccupations *multi-espèces, spatiales* et *multi-souches*) a nécessité l'ajout de 1718 caractères (47 lignes) avec MATLAB et seulement 849 caractères (26 lignes) pour Kendrick.

En outre, puisque les modèles en Kendrick sont plus modulaires que ceux en MATLAB, la taille moyenne des entités des modèles en Kendrick est plus faible que celle des modèles en MATLAB, ce qui suggère qu'il est peut-être plus facile d'écrire et de modifier des entités Kendrick.

Scénarios	Entités changées		Entités ajoutées		Entités réutilisées		Lignes changées		Lignes ajoutées	
	M	K	M	K	M	K	M	K	M	K
Du Modèle 1 au Modèle 2	2	3	0	1	0	2	20	8	6	8
Du Modèle 2 au Modèle 3	2	2	0	1	0	4	24	6	28	12
Du Modèle 3 au Modèle 4	2	3	0	1	0	4	22	8	13	6
Du Modèle 3 au Modèle 5	2	2	0	1	0	5	8	3	7	10
Du Modèle 1 au Modèle 4	2	3	0	3	0	1	24	8	49	26
Du Modèle 1 au Modèle 5	2	3	0	3	0	1	24	8	41	32

Table 6.3 – Comparaison de l'effort pour les changements

On calcule ensuite l'effort qu'il faut pour changer un modèle pour les quatre scénarios d'évolution (S1, S2, S3, S4) et on compare les résultats de Kendrick avec ceux de MATLAB (voir table 6.3). Notez que les modifications apportées aux modèles dans tous ces scénarios sont issues à chaque fois de l'ajout d'une nouvelle préoccupation.

Les résultats dans la table 6.3 montrent que les changements s'effectuent dans deux entités (les scripts *main* et *function*) d'un modèle en MATLAB. Cela est dû à l'introduction d'une nouvelle préoccupation ce qui entraîne certains changements dans le système des EDO du modèle (par exemple, l'ajout de nouvelles variables). Ces changements se propagent dans les deux entités du modèle.

Avec Kendrick, pour chaque scénario , une seule nouvelle entité est ajoutée (i.e la nouvelle préoccupation). Toutes les entités liées aux préoccupations appartenant au

premier modèle du scénario d'évolution ont été réutilisées. Les entités changées sont *Composition* (pour importer la nouvelle préoccupation), *Scénario* (pour modifier les valeurs des paramètres et des compartiments) et *Visualisation* (pour configurer la visualisation, comme dans le cas de S1 et S3 où nous avons visualisé la dynamique des infectieux par espèces et souches).

Kendrick permet de définir des entitées qui ont à chaque qu'une seule responsabilité. Les changements associées à une entité sont effectués à un seul emplacement et ne se propagent pas à d'autres entités.

On a calculé également l'effort pour passer du modèle 1 au modèle 4 et du modèle 1 au modèle 5 (voir les deux dernières lignes dans la table 6.3). Dans ces cas-là, seulement l'entité SEIRS est réutilisée en cas de Kendrick. Trois nouvelles préoccupations ont été rajoutées et les entités *Composition, Scénario* et *Visualisation* ont été modifiées. Cependant dans les deux cas, moins de changements affectent le modèle 1 en Kendrick que le même modèle en MATLAB. En effet, pour aller du modèle 1 au modèle 4, avec Kendrick on change 8 lignes et on ajoute 26 lignes, alors qu'avec MATLAB, on change 24 lignes et on ajoute 49 lignes.

Pour conclure, nous pouvons dire que les modèles exprimés avec Kendrick peuvent être modifiés plus facilement que ceux exprimés en MATLAB, ce qui suggère que les problèmes liés à la dispersion et de l'enchevêtrement des préoccupations en épidémiologie ont été en partie résolus et, par la même occasion on réduit l'impact des changements sur les modèles.

6.3 Conclusion

Les expériences que nous avons réalisées avec Kendrick ont permis de tester notre approche de modélisation, les concepts de base du langage et les modules de simulation. Nous avons pu constater, en travaillant sur ces modèles, que leur expression modulaire a rendu l'écriture des modèles moins complexes et plus flexibles. En raison de la séparation des définitions de préoccupation, on a pu exprimer des préoccupations individuelles indépendamment les unes des autres et donc augmenter la flexibilité et la possibilité de réutilisation.

Certaines préoccupations les plus utilisées en épidémiologie (par exemple, préoccupations épidémiques, spatiales...) ont été réutilisées à travers plusieurs modèles. En outre, la séparation de l'instantiation des préoccupations de leur définition a permis de pouvoir produire plusieurs scénarios pour étudier les modèles.

Nous avons également validé notre approche de modélisation. La validation a été effectuée sur certains modèles par l'évaluation de certaines métriques. En comparant avec les résultats donnés par les mêmes modèles en MATLAB, nous avons pu constater que, avec Kendrick, en appliquant l'approche proposée, les préoccupations ont été bien séparées. Tandis que dans les modèles MATLAB, les préoccupations sont été fortement entremêlées avec les détails de l'implémentation de bas niveau.

Chapitre 7

Conclusion

Sommaire

7.1 Rappel des problèmes abordés

Les modèles épidémiologiques deviennent de plus en plus complexes par l'ajout de statuts infectieux (latence, asymptomatique, porteurs sains, etc.) ainsi que par l'intégration de différents aspects de l'épidémiologie (les stratégies de contrôle, la distribution de la population selon le sexe, l'âge, la zone géographique etc.). De plus, ces modèles sont le plus souvent implantés pour résolution ou simulation numérique, typiquement en MATLAB.

Or, comme nous l'avons montré, les modèles écrits en MATLAB selon la démarche traditionnelle souffrent des défauts bien connus en séparation des préoccupations. Une préoccupation est un aspect qui ne se sépare pas facilement des autres en utilisant les abstractions de la programmation (classes ou fonctions). Les préoccupations sont dispersées dans tout le code et entremêlées. De plus les détails d'implémentation sont mêlés aux préoccupations épidémiologiques.

Les techniques classiques de séparation des préoccupations, comme la programmation par aspects, ne sont pas d'un grand secours dans notre cas car elles opèrent au niveau du code ce qui non seulement requièrent des connaissances pointues en programmation mais ne tire pas profit de la sémantique du domaine applicatif.

Nous avons donc pris le parti de poser notre problème à un niveau plus abstrait. Notre problème principal a donc consisté à déterminer si on pouvait donner une structure aux modèles épidémiologiques. Ce problème consiste d'une part à formaliser les notions de modèles et d'aspects puis à trouver un opérateur pour composer les aspects en modèles. De préférence nous cherchions un opérateur associatif et commutatif de façon à faciliter l'écriture des combinaisons d'aspects. Nous nous étions en outre fixé comme problème secondaire la séparation de l'expression des modèles des détails d'implémentation.

Nous savions qu'une difficulté nous attendait dans le mesure où les aspects épidémiologiques sont justement introduits dans les modèles parce qu'on suppose qu'ils présentent une hétérogénéité donc une certaine dépendance vis-à-vis du statut infectieux. Il n'était donc pas possible de les séparer complètement mais nous voulions limiter les interdépendances au minimum.

7.2 Formalisation et conception

Dans un premier temps nous avons toutefois conçu un méta-modèle assez classique en UML. Ceci nous a permis d'une part de commencer à dégager les premières abstractions pertinentes et d'autre part de préparer la construction d'un premier prototype supportant le langage Kendrick. Dès le départ un objectif important était de pouvoir basculer automatiquement du paradigme déterministe à base d'EDO au paradigme stochastique, le méta-modèle devant permettre de gérer les deux, voire à terme, des modèles multi-agents.

Ce premier méta-modèle était insuffisant car certains concepts (comme celui de compartiment) n'avaient une sémantique clairement définie, ce qui rend difficile de pouvoir exprimer des modèles complexes où il y a plusieurs préoccupations sans modifier la syntaxe abstraite.

Nous nous sommes donc tournés vers un méta-modèle mathématique où les compartiments sont des classes d'équivalences et où les modèles sont des automates stochastiques représentant une chaine de Markov à temps continu dont les états sont les compartiments et dont les transitions sont étiquetées par les taux dans la matrice de taux de transition. Modulo des hypothèses classiques (distribution de Poisson ...) il était possible de passer automatiquement d'EDO à un modèle stochastique ou de retrouver les EDO sous-jacentes à un modèle stochastique.

Il était même envisageable de basculer vers un modèle à agents, ou plutôt individu-centré, en réduisant drastiquement la taille d'une classe d'équivalence et en ajoutant des attributs donc les valeurs varient d'un individu à l'autre. Cette dernière piste n'a toutefois pas été suivie pour nous concentrer sur la séparation des préoccupations.

En cherchant dans la littérature, nous avons trouvé que les automates stochastiques avaient justement été dotés, bien que dans un domaine complètement différent, d'un opérateur tensoriel (une somme tensorielle $\oplus$ dans le cas des CTMC) [95, 96]. Un opérateur tensoriel est idéal parce qu'il donne à l'ensemble des aspects/préoccupations une structure de monoïde.

Nous avons donc pu proposer une définition très générale des préoccupations et des modèles (des combinaisons de préoccupations) en leur donnant les propriétés voulues et ce d'autant plus que l'opérateur tensoriel peut être rendu commutatif en triant les lignes et les colonnes des automates.

De façon à généraliser encore la notion de préoccupations, nous voulions permettre d'introduire des préoccupations ad hoc, sans utiliser nécessairement la somme tensorielle mais en les codant librement. Il fallait toutefois essayer de garder les propriétés d'associativité etc. Dans un premier temps nous avons identifié les préoccupations à des transformations de nos modèles. Cette notion est toutefois trop vague nous les avons donc finalement identifiées à des morphismes sur les modèles c'est-à-dire aux fonctions qui préservent leur structure. Rappelons que les morphismes d'un monoïde forment aussi un monoïde. Nous avons alors défini quelques opérateurs comme split-status ou add-state et add-transition qui sont des morphismes modulo quelques précautions. Ces opérateurs permettent notamment de considérer aussi les statuts infectieux tels que SEIR comme des préoccupations.

Enfin, il nous fallait encore permettre d'introduire des hétérogénéités par rapport au statut infectieux. Nous avons pour cela réutilisé la notion de taux fonctionnel de [96] qui consiste à permettre la définition de taux de transition dépendant de l'ensemble des états de l'automate. Les modèles proprement dits, c'est-à-dire la définition des aspects et leur composition, ne sont pas concernés par ces dépendances induites par le taux fonctionnel : elles sont différées à une phase d'instanciation.

7.3 Spécification du langage Kendrick

Nous avons spécifié les éléments du langage métier Kendrick dédié à la modélisation et la simulation des modèles compartimentaux épidémiologiques. Les modèles se construisent en composant des préoccupations modulaires de l'épidémiologie. Les principaux concepts que nous avons considérés pour construire et manipuler des préoccupa-

tions sont représentés sous la forme de mots clés du langage (`Concern, Composition, Scenario, Simulation, Visualisation, KendrickModel...`). La sémantique associée à ces concepts est donc intégrée au langage lui même. Les entités d'un modèle qui se représentent par ces concepts expriment la dynamique stochastique d'une population à base de compartiments. Il suffit d'une instruction en Kendrick pour appeler un algorithme de simulation. Le moteur de simulation correspondant va parcourir le modèle et l'interpréter (par exemple, reformuler les EDO pour étudier le modèle du point de vue déterministe) et donner des résultats numériques.

Un modèle écrit en Kendrick comporte en général quatre parties principales :

1. La définition des préoccupations qui font partie du modèle.
2. La composition des préoccupations modulaires pour construire le modèle.
3. La définition d'au moins un scénario dans lequel on initialise les paramètres et les compartiments du modèle ainsi que la définition des interactions entre les préoccupations qui sont considérés dans le modèle.
4. La définition des simulations et des visualisations pour afficher la dynamique de la population étudiée.

Les entités qui constituent un modèle peuvent être enregistrées dans des fichiers séparés, ce qui favorise la réutilisation de ces entités dans d'autres modèles. Nous avons conçu Kendrick à deux niveaux. D'un côté, nous avons implémenté le méta-modèle mathématique comme un modèle orienté-objet dans lequel une API a été fournie pour construire et manipuler des modèles. De l'autre côté, nous avons construit au dessus de cette API un DSL embarqué de type pidgin [101] pour simplifier l'écriture des modèles. Kendrick est le premier langage métier qui apporte la séparation des préoccupations dans la modélisation en épidémiologie. Chaque préoccupation définie en Kendrick a une sémantique propre qui fait partie du modèle et peut être manipulée pour représenter des situations épidémiologiques variées. Par exemple, une préoccupation spatiale peut être manipulée en combinaison avec une préoccupation multi-espèces et/ou multi-souches.

7.4 Expérimentations et validations

Dans le chapitre 6, nous avons fait des expérimentations dont l'objectif était de démontrer qu'il était possible de construire des modèles épidémiologiques valides en composant des préoccupations modulaires. Nous avons pour cela constitué une bibliothèque de modèles publiés dans la littérature, principalement ceux extraits du livre de Keeling[1]. Ces modèles déjà réalisés en Matlab par Keeling, ont été ré-implémentés par nos soin avec Kendrick.

Ces modèles ont été ensuite utilisés pour faire une validation de notre implémentation et notamment pour répondre aux questions liés aux problèmes d'enchevêtrement et de dispersion du code qui apparaissent lors du mélange de préoccupations épidémiologiques. Nous avons pour cela mesuré un certain nombre de critères objectifs.

Le résultat est très net. En MATLAB, les aspects sont éparpillés dans tout le code et mêlés les uns au autre et au reste du code. Chaque passage d'une version à la suivante des modèles induit des changements dans tout le code et à un niveau d'abstraction très bas. Au contraire, dans les modèles exprimés en Kendrick, les aspects sont exprimés

de façon modulaire, concise, dans un langage proche du domaine. Le couplage entre les aspects est minimal (limités à la phase d'instanciation). Enfin, l'impact des changements successifs (en passant d'un modèle à la version suivante) est clairement moins important avec Kendrick.

7.5 Perspectives

La formalisation que nous avons introduite dans cette thèse devrait pouvoir se généraliser à d'autres domaines que l'épidémiologie : la dynamique des populations, l'écologie, la pharmacie, la biochimie et de manière générale l'ensemble des disciplines qui s'intéressent aux systèmes biologiques utilisent des modèles compartimentaux. Par ailleurs, il pourrait être fécond de transposer notre démarche de séparation des préoccupation à d'autres domaines qui se formalisent différemment mais où un opérateur permet de donner une structure de semi-groupe ou de monoïde, idéalement commutatif, à l'ensemble des préoccupations.

L'approche que nous avons développée dans cette thèse est dédiée à la modélisation et la simulation des modèles compartimentaux qui peuvent se représenter sur la forme d'un système d'équation différentielles ordinaires ou d'une chaîne de Markov à temps continu. Une première perspective serait de permettre de modéliser et simuler d'autres sortes de modèles, notamment des modèles utilisant des équations aux dérivées partielles (EDP) ou des équations intégro-différentielles (EID). Ces formalismes permettent de représenter des processus spatiaux comme ceux de convection-diffusion notamment. Intuitivement, on peut imaginer qu'une modélisation utilisant des ODE peut être vu comme un passage limite pour une modélisation sous forme d'EDP.

Un autre possibilité d'extension est en direction des systèmes multi-agents qui intéressent de plus en plus les chercheurs en épidémiologie même si certains leur reproche leur manque de rigueur par rapport à la rigueur mathématique des modèles compartimentaux. Nous avons utilisé une forme très simplifiée de simulation multi-agents, où les individus possèdent certains comportements qui décrivent leurs passages d'un compartiment à un autre (par exemple, contracter l'infection, être guéri...) et on suppose que le temps d'attente dans chaque compartiment suit une distribution exponentielle. Néanmoins, un modèle multi-agents réaliste serait celui où les individus ont des comportements beaucoup plus complexes (par exemple en fonction des attributs de chaque agent ou des caractéristiques de l'environnement). Trouver une formulation de ces systèmes compatibles avec le style de modélisation que nous avons proposé est une perspective encore très éloignée pour l'instant.

La deuxième perspective possible de ce travail de thèse porte sur l'enrichissement du langage métier Kendrick afin de faciliter l'écriture de modèles complexes, en particulier les modèles spatiaux (comme les modèles structurés par foyers) où on a besoin d'encapsuler des données spatiales fournis par des sources externes (par exemple les données provenant d'un SIG). Kendrick nécessite pour l'instant de représenter l'ensemble des préoccupations de la même manière (i.e. un automate stochastique). On peut imaginer à terme utiliser un langage métier différent pour chaque préoccupation et de les coupler au sein d'une même plateforme. Cela nécessiterait d'une part de pouvoir projeter chaque langage vers le modèle d'automate stochastique que nous avons introduit mais également de disposer d'un atelier logiciel permettant de définir

et d'intégrer plusieurs DSL [1].

Finalement, une dernière perspective intéressante serait de vérifier si le méta-modèle de Kendrick est toujours valide lorsque le modélisateur utilise des scénarios de type *what-if* qui permettent d'analyser la sensibilité des paramètres ainsi que d'évaluer l'efficacité des stratégies de contrôle.

1. Ce concept d'atelier appelé Language Workbench a été introduit par Martin Fowler en 2005. Voir `http://martinfowler.com/bliki/LanguageWorkbench.html`. On pourrait utiliser dans notre cadre, Helvetia[101] également écrit en Pharo

Bibliographie

[1] Modeling infectious diseases in humans and animals - programs.

[2] Joel M Addawe and JE Lope. Analysis of an age-structured malaria transmission model. *Philippine Science Letters*, 5(2) :169–182, 2012.

[3] Z. Agur, L. Cojocaru, G. Mazor, R. M. Anderson, and Y. L. Danon. Pulse mass measles vaccination across age cohorts. In *Proceedings of the National Academy of Sciences of the United States of America*, volume 90 (24), pages 11698–11702, 1993.

[4] Mehmet Akşit, Lodewijk Bergmans, and Sinan Vural. An object-oriented language-database integration model : The composition-filters approach. In *European Conference on Object-Oriented Programming*, pages 372–395. Springer, 1992.

[5] Omar Alam, Jörg Kienzle, and Gunter Mussbacher. Concern-oriented software design. In *Model-Driven Engineering Languages and Systems*, pages 604–621. Springer, 2013.

[6] Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1) :47, 2002.

[7] Linda JS Allen. An introduction to stochastic epidemic models. In *Mathematical epidemiology*, pages 81–130. Springer, 2008.

[8] Sherman R. Alpert, Kyle Brown, and Bobby Woolf. *The Design Patterns Smalltalk Companion*. Addison Wesley, Boston, MA, USA, 1998.

[9] R. M. Anderson and R. M. May. *Infectious diseases of humans : Dynamics and control*. Oxford Science Publications, Oxford, 1991.

[10] Roy M Anderson and Robert M May. Directly transmitted infections diseases : control by vaccination. *Science*, 215(4536) :1053–1060, 1982.

[11] T. Antao, I. Hastings, and P. McBurney. Ronald : A domain-specific language to study the interactions between malaria infections and drug treatments. In *Proceedings of the 2nd ACM SIGHIT International Health Informatics Symposium*, pages 785–790, 2012.

[12] Vanessa Pena Araya, Alexandre Bergel, Damien Cassou, Stéphane Ducasse, and Jannik Laval. Agile visualization with roassal. *Deep Into Pharo*, pages 209–239, 2013.

[13] Julien Arino, Jonathan R. Davis, David Hartley, Richard Jordan, Joy M. Miller, and P. Van Den Driessche. A multi-species epidemic model with spatial dynamics. *Mathematical Medicine and Biology*, 22(2) :129–142, 2005.

[14] Julien Arino, Richard Jordan, and P. Van den Driessche. Quarantine in a multi-species epidemic model with spatial dynamics. *Mathematical biosciences*, 206(1) :46–60, 2007.

[15] Colin Atkinson and Thomas Kuhne. Model-driven development : a metamodeling foundation. *IEEE software*, 20(5) :36–41, 2003.

[16] D. Balcan, B. Gonçalves, H. Hu, J. J Ramasco, V. Colizza, and A. Vespignani. Modelling the spatial spread of infectious diseases : The global epidemic and mobility computational model. *Journal of Computational Science*, 1 :132–145, 2010.

[17] Shweta Bansal, Bryan T Grenfell, and Lauren Ancel Meyers. When individual behaviour matters : homogeneous and network models in epidemiology. *Journal of the Royal Society Interface*, 4(16) :879–891, 2007.

[18] C. T. Bauch, E Szusz, and L. P. Garrison. Scheduling of measles vaccination in low-income countries : Projections of a dynamic model. *Vaccine*, 27(31) :4090–4098, 2009.

[19] Soufiene Benkirane. *Process algebra for epidemiology : evaluating and enhancing the ability of PEPA to describe biological systems.* PhD thesis, University of Stirling, 2011.

[20] D Bernoulli. Reflexions sur les avantages de l'inoculation. *Mém. Paris*, pages 439–482, 1758.

[21] Jean Bézivin. From object composition to model transformation with the mda. In *TOOLS (39)*, pages 350–354, 2001.

[22] Keith R Bisset, Jiangzhuo Chen, Xizhou Feng, VS Kumar, and Madhav V Marathe. Epifast : a fast algorithm for large scale realistic epidemic simulations on distributed memory systems. In *Proceedings of the 23rd international conference on Supercomputing*, pages 430–439. ACM, 2009.

[23] Andrew J. Black and Alan J. McKane. Stochastic formulation of ecological models and their applications. *Trends in Ecology and Evolution*, 27 :6, 2012.

[24] Jean-Sébastien Bolduc and Hans Vangheluwe. Expressing ode models as devs : Quantization approaches. In *Proceedings of the AIS'2002 Conference (AI, Simulation and Planning in High Autonomy Systems), April 2002, Lisboa, Portugal/F. Barros and N. Giambiasi (eds.)*, pages 163–169, 2002.

[25] Fred Brauer. Compartmental models in epidemiology. In *Mathematical epidemiology*, pages 19–79. Springer, 2008.

[26] T. M. A. BUI, S. Stinckwich, M. Ziane, B. Roche, and T. V. HO. Kendrick : a domain specific language and platform for epidemiological modelling. In *11th IEEE-RIVF International Conference on Computing and Communication Technologies*, RIVF-2015, Cantho, Vietnam, 2015.

[27] D. L. Chao, M. E. Halloran, V. J. Obenchain, and I. M. Jr. Longini. Flute, a publicly available stochastic influenza epidemic simulation model. *PLoS Comput Biol*, 6, 2010.

[28] Claude Chastel. Eventual role of asymptomatic cases of dengue for the introduction and spread of dengue viruses in non-endemic regions. *Front physiol*, 3 :70, 2012.

[29] Andrei Chis. *Moldable Tools.* PhD thesis, SCG, 2016.

[30] Mikayla C Chubb and Kathryn H Jacobsen. Mathematical modeling and the epidemiological research process. *European journal of epidemiology*, 25(1) :13–19, 2010.

[31] Federica Ciocchetta and Maria Luisa Guerriero. Modelling biological compartments in bio-pepa. *Electronic Notes in Theoretical Computer Science*, 227 :77–95, 2009.

[32] Federica Ciocchetta and Jane Hillston. Bio-pepa : an extension of the process algebra pepa for biochemical networks. *Electronic Notes in Theoretical Computer Science*, 194(3) :103–117, 2008.

[33] Federica Ciocchetta and Jane Hillston. Bio-pepa for epidemiological models. *Electronic Notes in Theoretical Computer Science*, 261 :43–69, 2010.

[34] Patrícia Cortés de Zea Bermudez, M Antónia Amaral Turkman, and Kamil Feridun Turkman. Parameter estimation of bilinear processes using approximate bayesian computation. *Textos de Matemática-Mathematical Texts*, 47 :135–151, 2015.

[35] P Degenne, D Lo Seen, D Parigot, R Forax, A Tran, A Ait Lahcen, O Curé, and R Jeansoulin. Design of a domain specific language for modelling processes in landscapes. *Ecological Modelling*, 220 :3527–3535, 2009.

[36] Pascal Degenne, Ayoub Ait Lahcen, Olivier Curé, Rémi Forax, Didier Parigot, and Danny Lo Seen. Modelling the environment using graphs with behaviour : do you speak ocelet ? In *iEMSs 2010*, pages 8–pages, 2010.

[37] K. Dietz and J. Heesterbeek. Daniel bernoulli's epidemiological model revisited. *Journal Mathematical Biosciences*, 180 :1–21, 2002.

[38] Klaus Dietz. The estimation of the basic reproduction number for infectious diseases. *Statistical methods in medical research*, 2(1) :23–41, 1993.

[39] Stéphane Ducasse, Tudor Gîrba, Adrian Kuhn, and Lukas Renggli. Meta-environment and executable meta-language using smalltalk : an experience report. *Software & Systems Modeling*, 8(1) :5–19, 2009.

[40] H. P. Duerr, S. O. Brockmann, I. Piechotowski, M. Schwehm, and M. Eichner. Influenza pandemic intervention planning using influsim : pharmaceutical and non-pharmaceutical interventions. *BMC Infectious Diseases*, 7 :76, 2007.

[41] M. Eichner, M. Schwehm, H. P. Duerr, and S. O. Brockmann. The influenza pandemic preparedness planning tool influsim. *BMC Infectious Diseases*, 7 :17, 2007.

[42] Abdulrahman M El-Sayed, Peter Scarborough, Lars Seemann, and Sandro Galea. Social network analysis and agent-based modeling in social epidemiology. *Epidemiologic Perspectives & Innovations*, 9(1) :1, 2012.

[43] Joshua M Epstein, D Michael Goedecke, Feng Yu, Robert J Morris, Diane K Wagener, and Georgiy V Bobashev. Controlling pandemic flu : the value of international air travel restrictions. *PloS one*, 2(5) :e401, 2007.

[44] Paul Erdös and Alfréd Rényi. On random graphs, i. *Publicationes Mathematicae (Debrecen)*, 6 :290–297, 1959.

[45] Jacky Estublier and Anca Daniela Ionita. Extending uml for model composition. In *2005 Australian Software Engineering Conference*, pages 31–38. IEEE, 2005.

[46] Alexander Falenski, Matthias Filter, Christian Thöns, Armin A. Weiser, Jan-Frederik Wigger, Matthew Davis, Judith V. Douglas, Stefan Edlund, Kun Hu, James H. Kaufman, et al. A generic open-source software framework supporting scenario simulations in bioterrorist crises. *Biosecurity and bioterrorism : biodefense strategy, practice, and science*, 11(S1) :S134–S145, 2013.

[47] Zhilan Feng and Horst R Thieme. Recurrent outbreaks of childhood diseases revisited : the impact of isolation. *Mathematical biosciences*, 128(1) :93–130, 1995.

[48] N. M. Ferguson, D. A. T. Cummings, S. Cauchemez, C. Fraser, S. Riley, A. Meeyai, and D. S. ... Burke. Strategies for containing an emerging influenza pandemic in southeast asia. *Nature*, 437(7056) :209–214, 2005.

[49] Franck Fleurey, Benoit Baudry, Robert France, and Sudipto Ghosh. A generic approach for automatic model composition. In *International Conference on Model Driven Engineering Languages and Systems*, pages 7–15. Springer, 2007.

[50] D. A. Ford, J. H. Kaufman, and I. Eiron. An extensible spatial and temporal epidemiological modelling system. *International Journal of Health Geographics*, 5 :4, 2006.

[51] Martin Fowler. *Domain-specific languages*. Pearson Education, 2010.

[52] Robert France, Indrakshi Ray, Geri Georg, and Sudipto Ghosh. Aspect-oriented approach to early design modelling. *IEE Proceedings-Software*, 151(4) :173–185, 2004.

[53] Rebecca J Garten, C Todd Davis, Colin A Russell, Bo Shu, Stephen Lindstrom, Amanda Balish, Wendy M Sessions, Xiyan Xu, Eugene Skepner, Varough Deyde, et al. Antigenic and genetic characteristics of swine-origin 2009 a (h1n1) influenza viruses circulating in humans. *science*, 325(5937) :197–201, 2009.

[54] D. T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *The Journal of Physical Chemistry*, 81 :2340–2361, 1977.

[55] John J. Grefenstette, Shawn T. Brown, Roni Rosenfeld, Jay DePasse, Nathan T.B. Stone, Phillip C. Cooley, William D. Wheaton, Alona Fyshe, David D. Galloway, Anuroop Sriram, et al. Fred (a framework for reconstructing epidemic dynamics) : an open-source software system for modeling infectious diseases and control strategies using census-based populations. *BMC public health*, 13(1) :940, 2013.

[56] David F. Griffiths and Desmond J. Higham. *Numerical methods for ordinary differential equations*. Springer, Springer Undergraduate Mathematics Series, 2010.

[57] Volker Grimm. Ten years of individual-based modelling in ecology : what have we learned and what could we learn in the future ? *Ecological modelling*, 115(2) :129–148, 1999.

[58] Duane J Gubler. Epidemic dengue/dengue hemorrhagic fever as a public health, social and economic problem in the 21st century. *Trends in microbiology*, 10(2) :100–103, 2002.

[59] Hamish Harvey. Languages and metamodels for modelling frameworks. In *MODSIM 2005 International Congress on Modelling and Simulation. Modelling and Simulation Society of Australia and New Zealand*, pages 669–675, 2005.

[60] JAP Heesterbeek. A brief history of r 0 and a recipe for its calculation. *Acta biotheoretica*, 50(3) :189–204, 2002.

[61] Herbert Hethcote, Ma Zhien, and Liao Shengbing. Effects of quarantine in six endemic models for infectious diseases. *Mathematical Biosciences*, 180(1) :141–160, 2002.

[62] Herbert W Hethcote. The mathematics of infectious diseases. *SIAM review*, 42(4) :599–653, 2000.

[63] Niels Holst and Getachew F Belete. Domain-specific languages for ecological modelling. *Ecological Informatics*, 27 :26–38, 2015.

[64] Walter L. Hürsch and Cristina Videira Lopes. Separation of concerns. Technical Report NUCCS-95-03, College of Computer Science, Northeastern University, February 1995.

[65] Faheem Hussain, Arvind Ramanathan, Laura L Pullum, and Sumit Kumar Jha. Epispec : A formal specification language for parameterized agent-based models against epidemiological ground truth. In *Computational Advances in Bio and Medical Sciences (ICCABS), 2014 IEEE 4th International Conference on*, pages 1–6. IEEE, 2014.

[66] Jean-Marc Jézéquel, Olivier Barais, and Franck Fleurey. Model driven language engineering with kermeta. *Generative and Transformational Techniques in Software Engineering III : International Summer School, GTTSE 2009, Braga, Portugal, July 6-11, 2009, Revised Papers*, 6491 :201, 2011.

[67] Kate E Jones, Nikkita G Patel, Marc A Levy, Adam Storeygard, Deborah Balk, John L Gittleman, and Peter Daszak. Global trends in emerging infectious diseases. *Nature*, 451(7181) :990–993, 2008.

[68] Matt J Keeling and K. T. D. Eames. Networks and epidemic models. *J R Soc Interface*, 2(4) :295–307, 2005.

[69] Matthew J Keeling and P Rohani. *Modeling Infectious Diseases*. Princeton University Press, Princeton, 2008.

[70] W. O. Kermack and A. G. McKendrick. Contributions to the mathematical theory of epidemics. *Bulletin of mathematical biology*, 53 (1) :33–55, 1991.

[71] Gregor Kiczales. Towards a new model of abstraction in the engineering of software. In *International Workshop on Reflection and Meta-Level Architecture*, pages 67–76. Citeseer, 1992.

[72] Gregor Kiczales. Aspect-oriented programming. *ACM Computing Surveys (CSUR)*, 28(4es) :154, 1996.

[73] Vinay Kulkarni and Sreedhar Reddy. Separation of concerns in model-driven development. *IEEE software*, 20(5) :64, 2003.

[74] Marcelo Kuperman and Guillermo Abramson. Small world effect in an epidemiological model. *Physical Review Letters*, 86(13) :2909, 2001.

[75] Carl Latkin, Wallace Mandell, Maria Oziemkowska, David Celentano, David Vlahov, Margaret Ensminger, and Amy Knowlton. Using social network analysis to study patterns of drug use among urban drug users at high risk for hiv/aids. *Drug and alcohol dependence*, 38(1) :1–9, 1995.

[76] Ákos Lédeczi, Greg Nordstrom, Gabor Karsai, Peter Volgyesi, and Miklos Maroti. On metamodel composition. In *Control Applications, 2001.(CCA'01). Proceedings of the 2001 IEEE International Conference on*, pages 756–760. IEEE, 2001.

[77] A. Levin, C. Burgess, L. P. Garrison, C. Bauch, J. Babigumira, E. Simons, and A. Dabbagh. Global eradication of measles : an epidemiologic and economic evaluation. *The Journal of infectious diseases*, 204 Suppl (Suppl 1) :98–106, 2011.

[78] Karl J Lieberherr. The art of growing adaptive object-oriented software. *PWS Pub*, 1995.

[79] Karen R Lips, Forrest Brem, Roberto Brenes, John D Reeve, Ross A Alford, Jamie Voyles, Cynthia Carey, Lauren Livo, Allan P Pessier, and James P Collins. Emerging infectious disease and the loss of biodiversity in a neotropical amphibian community. *Proceedings of the national academy of sciences of the United States of America*, 103(9) :3165–3170, 2006.

[80] Alun L Lloyd and Steve Valeika. Network models in epidemiology : an overview. *Complex population dynamics : nonlinear modeling in ecology, epidemiology and genetics*, 2007.

[81] Maia Martcheva. *Introduction to Mathematical Epidemiology*, volume 61. Springer, 2015.

[82] Robert C. Martin. *Agile software development : principles, patterns, and practices*. Prentice Hall, 2002.

[83] RT Mikolajczyk, MK Akmatov, S Rastin, and Mirjam Kretzschmar. Social contacts of school children and the transmission of respiratory-spread pathogens. *Epidemiology and infection*, 136(06) :813–822, 2008.

[84] Stanley Milgram. The small world problem. *Psychology today*, 2(1) :60–67, 1967.

[85] Joaquin Miller and Jishnu Mukerji. Mda guide version 1.0. 1, object management group. *URL : http ://www. omg. org/docs/omg/03-06-01. pdf*, 2003.

[86] S. Mishra, D. Fisman, and M. Boily. The abc of terms used in mathematical models of infectious diseases. *Journal of Epidemiology and Community Health*, 54 :87–94, 2011.

[87] NA Mitchison. Passive transfer of transplantation immunity. *Proceedings of the Royal Society of London B : Biological Sciences*, 142(906) :72–87, 1954.

[88] MJ Morine and C Priami. Analysis of biological systems, 2013.

[89] Ingemar Nåsell. Stochastic models of some endemic infections. *Mathematical biosciences*, 179(1) :1–19, 2002.

[90] Oscar Nierstrasz, Stéphane Ducasse, and Tudor Gîrba. The story of moose : an agile reengineering environment. *ACM SIGSOFT Software Engineering Notes*, 30(5) :1–10, 2005.

[91] Tarence Parr. *Language implementation patterns : create your own domain-specific and general programming languages*. Pragmatic Bookshelf, 2009.

[92] Thomas Petzoldt and Karsten Rinke. Simecol : an object-oriented framework for ecological modeling in r. *Journal of Statistical Software*, 22(9) :1–31, 2007.

[93] M Pineda-Krch. Gillespiessa : Gillespie ?s stochastic simulation algorithm (ssa). *R package version*, pages 05–4, 2010.

[94] Mario Pineda-Krch. Gillespiessa : Implementing the stochastic simulation algorithm in r. *Journal of Statistical Software*, 25(12) :1–18, 2008.

[95] Brigitte Plateau and Jean-Michel Fourneau. A methodology for solving markov models of parallel systems. *Journal of parallel and distributed computing*, 12(4) :370–387, 1991.

[96] Brigitte Plateau and William J Stewart. Stochastic automata networks. In *Computational Probability*, pages 113–151. Springer, 2000.

[97] John J Potterat, SQ Muth, RB Rothenberg, H Zimmerman-Rogers, DL Green, JE Taylor, MS Bonney, and HA White. Sexual network structure as an indicator of epidemic phase. *Sexually transmitted infections*, 78(suppl 1) :i152–i158, 2002.

[98] Gauthier Quesnel, Raphaël Duboz, and Éric Ramat. The virtual laboratory environment–an operational framework for multi-modelling, simulation and analysis of complex dynamical systems. *Simulation Modelling Practice and Theory*, 17(4) :641–653, 2009.

[99] Antonia M Reina, J Torres, and M Toro. Towards developing generic solutions with aspects. In *proceedings of the Workshop in Aspect Oriented Modeling held in conjunction with UML*, 2004.

[100] Ben Y Reis, Isaac S Kohane, and Kenneth D Mandl. An epidemiological network model for disease outbreak detection. *PLoS Med*, 4(6) :e210, 2007.

[101] Lukas Renggli. *Dynamic Language Embedding with homogeneous tool support*. PhD thesis, Bern University, Bern University, 10 2010.

[102] Lukas Renggli, Stéphane Ducasse, Tudor Gîrba, and Oscar Nierstrasz. Practical dynamic grammars for dynamic languages. In *4th Workshop on Dynamic Languages and Applications (DYLA 2010)*, 2010.

[103] Steven Riley. Large-scale spatial-transmission models of infectious disease. *Science*, 316(5829) :1298–1301, 2007.

[104] Caitlin Rivers. Epipy : Python tools for epidemiology. 4 2014.

[105] Caitlin Rivers. Modeling emerging infectious diseases for public health decision support. 2015.

[106] Benjamin Roche, John M Drake, and Pejman Rohani. An agent-based model to study the epidemiological and evolutionary dynamics of influenza viruses. *BMC Bioinformatics*, 12 :87, 2011.

[107] Benjamin Roche, Jean-François Guégan, and François Bousquet. Multi-agent systems in epidemiology : a first step for computational biology in the study of vector-borne disease transmission. *BMC bioinformatics*, 9(1) :435, 2008.

[108] Oliver Schneider, Christopher Dutchyn, and Nathaniel Osgood. Towards frabjous : a two-level system for functional reactive agent-based epidemic simulation. In *Proceedings of the 2008 International Conference on Bioinformatics and Computational Biology, BIOCOMP*, pages 747–752, 2008.

[109] Jose L Segovia-Juarez, Suman Ganguli, and Denise Kirschner. Identifying control mechanisms of granuloma formation during m. tuberculosis infection using an agent-based model. *Journal of theoretical biology*, 231(3) :357–376, 2004.

[110] Bran Selic. A systematic approach to domain-specific language design using uml. In *10th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC'07)*, pages 2–9. IEEE, 2007.

[111] Galen R Shorack and Jon A Wellner. *Empirical processes with applications to statistics*, volume 59. Siam, 2009.

[112] Arnor Solberg, Devon Simmonds, Raghu Reddy, Sudipto Ghosh, and Robert France. Using aspect oriented techniques to support separation of concerns in model driven development. In *29th Annual International Computer Software and Applications Conference (COMPSAC'05)*, volume 1, pages 121–126. IEEE, 2005.

[113] J Erin Staples, Robert F Breiman, and Ann M Powers. Chikungunya fever : an epidemiological review of a re-emerging infectious disease. *Clinical Infectious Diseases*, 49(6) :942–948, 2009.

[114] Dave Steinberg, Frank Budinsky, Ed Merks, and Marcelo Paternostro. *EMF : eclipse modeling framework*. Pearson Education, 2008.

[115] Greg Straw, Geri Georg, Eunjee Song, Sudipto Ghosh, Robert France, and James M Bieman. Model composition directives. In *International Conference on the Unified Modeling Language*, pages 84–97. Springer, 2004.

[116] Ousmane Moussa Tessa. Mathematical model for control of measles by vaccination. In *Proceedings of Mali Symposium on Applied Sciences*. Niger : Abdou Moumouni University, 2006.

[117] Joseph H Tien and David JD Earn. Multiple transmission pathways and disease dynamics in a waterborne pathogen model. *Bulletin of mathematical biology*, 72(6) :1506–1533, 2010.

[118] W. Van den Broeck, C. Gioannini, B. Gonçalves, M. Quaggiotto, V. Colizza, and A. Vespignani. The gleamviz computational tool, a publicly available software to explore realistic epidemic spreading scenarios at the global scale. *BMC Infectious Diseases*, 11 :37, 2011.

[119] Duncan J Watts and Steven H Strogatz. Collective dynamics of ?small-world ?networks. *nature*, 393(6684) :440–442, 1998.

[120] Y. Xia, O. N. Bjornstad, and B. T. Grenfell. Measles metapopulation dynamics : a gravity model for epidemiological coupling and dynamics. *Am Nat*, 164(2) :267–281, 2004.

[121] Bernard P Zeigler, Yoonkeon Moon, Doohwan Kim, and George Ball. The devs environment for high-performance modeling and simulation. *IEEE Computational Science & Engineering*, 4(3) :61–71, 1997.

Annexe A

Scripts Matlab des modèles utilisés dans le mémoire de thèse

A.1 Modèle déterministe de la rougeole

```
global mu gamma beta N sigma;
N = 100000;
beta = 0.0000214;
gamma = 0.143;
sigma = 0.125;
mu = 0.0000351;
S = 99999;
I = 1;
E = R = 0;

tMax = 150;
options = [];
[T,Y]=ode45(@rightSideSEIR,[0 tMax],[S E I R],options);
plot(T,Y);
legend('Susceptible','Exposed', 'Infectious', 'Recovered');
xlabel('Time');
ylabel('Number of individuals');

function res=@rightSideSEIR(t,pop)
    global mu gamma beta N nu sigma;
    S = pop(1);
    E = pop(2);
    I =  pop(3);
    R = pop(4);
    dSdt = mu*N - beta*S*I/N - mu*S;
    dEdt = beta*S*I/N - sigma*E - mu*E;
    dIdt = sigma*E - gamma*I - mu*I;
    dRdt = gamma*I - mu*R;
    res = [dSdt dEdt dIdt dRdt]';
```

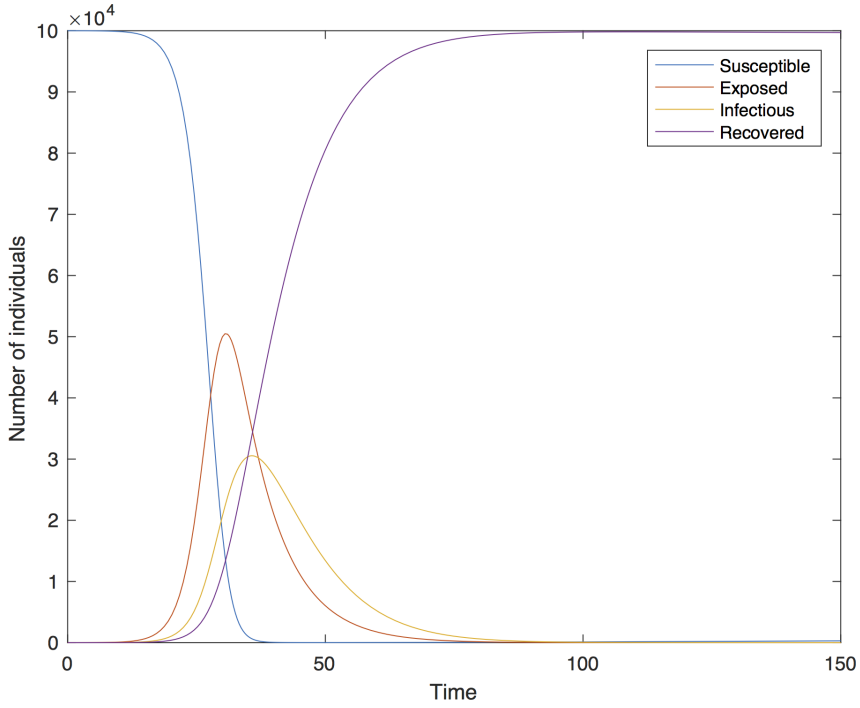


Figure A.1 – Simulation déterministe du modèle de la rougeole écrit en Matlab

A.2 Modèle multi-espèces d'une maladie vectorielle à plusieurs réservoirs

A.2.1 Modèle déterministe

```
global mu;
global beta1;
global v;
global d;
global m;
global sigma;
global N;

n=3;
N(1)=10000;
N(2)=1000;
N(3)=2000;
mu(1)=365/30;
mu(2:3)=1/20;
v=52;

S(1:n)=N;
I(1:n)=0;
S(1)=N(1)-1;
I(1)=1;


beta1=[0.02 0.02 0.02;0.02 0.02 0.02;0.02 0.02 0.02;];

tMax=2;
options=[];
```

```
[T,Y]=ode45(@rightSideMultiHost,[0 tMax],[S I],options);
plot(T,sqrt(Y(:,n+1:end)'));
legend('Mosquito','reservoir 1','reservoir 2');
xlabel('Time');
ylabel('sqrt(I)');

function res=rightSideMultiHost(t,pState)

global mu;
global beta1;
global v;
global sigma;
global N;

n=length(pState)/2;

pS=pState(1:n);
pI=pState(n+1:end);

for i=1:n
    lambda(i)=sum(beta1(:,i).*pI);
end

for i=1:n
    dsidt(i)=mu(i)*N(i)-lambda(i)*pS(i)-mu(i)*pS(i);
    diidt(i)=lambda(i)*pS(i)-(v+mu(i))*pI(i);
end

res=[dsidt diidt]';
```

A.2.2 Modèle stochastique utilisant l'algorithm direct de Gillespie

```
n = 3;
mu = [365/30 1/20 1/20];
N = [10000 1000 2000];
gamma = [52 52 52];
beta=[0 0.02 0.02;0.02 0 0;0.02 0 0;];
S(1:n)=N;
I(1:n)=0;
R(1:n)=0;
I(1)=1;
S(1)=N(1)-I(1);

[t, pop]=Stoch_Iteration([0 0.5],[S I R],beta,gamma,mu,N);
plot(t, sqrt(pop(:,4:6)));
legend('Mosquito','Reservoir1', 'Reservoir2');
xlabel('Time');
ylabel('sqrt(Number of Infectious)');

function [T,P]=Stoch_Iteration(Time,Initial,beta,gamma,mu,N)
loop = 1;
P(1,:) = Initial;
T(1) = 0;
old = reshape(Initial, [3 3])';
while (T(loop) < Time(2))
   [step,new]=Iterate(old,beta,gamma,mu,N);
   loop = loop + 1;
   T(loop)=T(loop-1) + step;
   P(loop,:)=reshape(new',[1 9]);
```

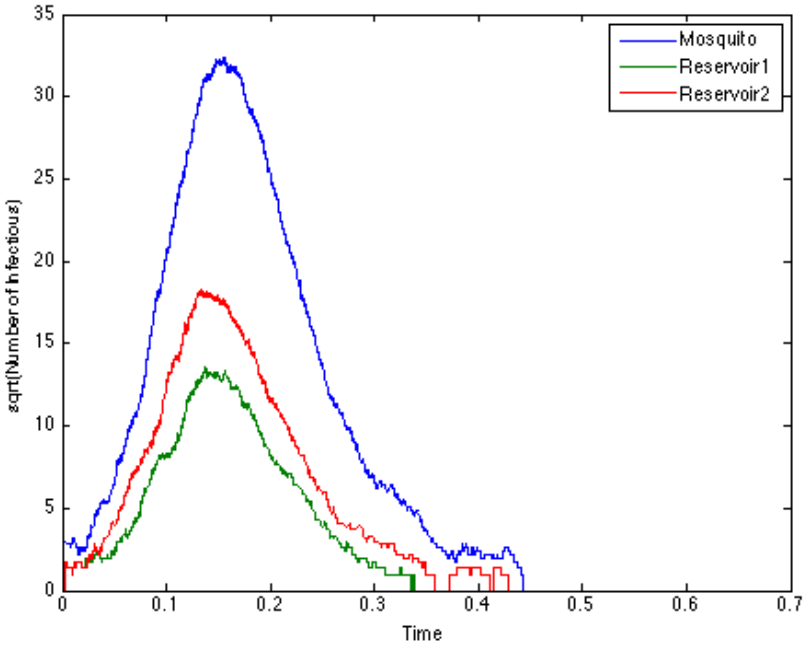


Figure A.2 – Simulation stochastique du modèle avec trois hôtes-espèces implémenté en Matlab

```
28      old = new;
29  end
30
31  function [step, new_value] = Iterate(old, beta,gamma,mu,N)
32  n = 3;
33  nbOfEvents = 6;
34  S = old(1, :);
35  I = old(2, :);
36  R = old(3, :);
37  lambda = zeros(1,3);
38  for i=1:n
39      lambda(i)=sum(beta(i,:).*I);
40  end
41  rate = zeros(nbOfEvents,n);
42  change = zeros(n,nbOfEvents);
43  rate(1,:) = mu.*N;change(:,1)=[+1; 0; 0];
44  rate(2,:) = mu.*S;change(:,2)=[-1; 0; 0];
45  rate(3,:) = mu.*I;change(:,3)=[0; -1; 0];
46  rate(4,:) = mu.*R;change(:,4)=[0; 0; -1];
47  rate(5,:) = gamma.*I;change(:,5)=[0; -1; +1];
48  rate(6,:) = lambda.*S;change(:,6)=[-1; +1; 0];
49  rate = reshape(rate, [1 n*nbOfEvents]);
50  rand1 = rand(1,1);
51  rand2 = rand(1,1);
52  step = -log(rand2)/(sum(rate));
53  m=min(find(cumsum(rate)>=rand1*sum(rate)));
54  row = mod(m,nbOfEvents);
55  if (row == 0)
56      row = 6;
57  end
58  col = ceil(m/nbOfEvents);
59  new_value = old;
60  new_value(:,col) = old(:,col)+change(:,row);
```

A.3 Modèle de la grippe aviaire

```
%global parameters
global mu N nu sigma gamma rho beta;
np=5;ns=2;
%repeat copies of a vector into a npx1 block arrangement
%which results in a npx2 matrix, each row is [500 5000]
N=repmat([500 5000],np,1);
mu=repmat([0.000365 0.00137],np,1);
sigma=repmat([0.5 0.67],np,1);
gamma=repmat([0.25 0.233],np,1);
%repeat copies of the matrix into a 1x1xnp block arrangement
beta=repmat([0 0.21;0 0.42],[1 1 np]);%matrix 2x2x5
nu=repmat(0.00274,np,ns);
%topology matrix between regions
top = [0 1 0 0 0;0 0 1 0 0;0 0 0 1 0;0 0 0 0 1;1 0 0 0 0];
%concatenate two matrices along the third dimension
rho=cat(3,top*0.03,top*0.1);%matrix 5x5x2
%initialize cardinality of compartments
S=N;S(1,2)=4990;
I=zeros(5,2);I(1,2)=10;
E=zeros(5,2);R=E;
tMax=500;
options=[];
S = reshape(S,[1 ns*np]); E = reshape(E,[1 ns*np]);
I = reshape(I,[1 ns*np]); R = reshape(R,[1 ns*np]);
[T,Y]=ode45(@rightSideAIModel,[0 tMax],[S E I R],options);
%plot total infectious humans and birds
res=zeros(length(T),2);
res(:,1)=sum(Y(:,2*np*ns+1:2*np*ns+5),2);
res(:,2)=sum(Y(:,2*np*ns+6:3*np*ns),2);
plot(T,res');
legend('Human','Bird');
xlabel('Time');
ylabel('Total Infectious');

function res=rightSideAIModel(t, pop)
global mu,N,nu,sigma,gamma,rho,beta;
np=5;ns=2;
S=reshape(pop(1:np*ns),[np ns]);
E=reshape(pop(np*ns+1:2*np*ns),[np ns]);
I=reshape(pop(2*np*ns+1:3*np*ns),[np ns]);
R=reshape(pop(3*np*ns+1:end),[np ns]);
%calculate lambda
lambda=zeros(np,ns);
for p=1:np
    for s=1:ns
        lambda(p,s)=sum(beta(s,:,p).*I(p,:)./N(p,:));
    end
end
%calculate migrations
deltaS = zeros(np,ns); deltaE = zeros(np,ns);
deltaI = zeros(np,ns); deltaR = zeros(np,ns);
for s=1:ns
   deltaS(:,s)=rho(:,:,s)*S(:,s)-sum(rho(:,:,s))'.*S(:,s);
   deltaE(:,s)=rho(:,:,s)*E(:,s)-sum(rho(:,:,s))'.*E(:,s);
   deltaI(:,s)=rho(:,:,s)*I(:,s)-sum(rho(:,:,s))'.*I(:,s);
   deltaR(:,s)=rho(:,:,s)*R(:,s)-sum(rho(:,:,s))'.*R(:,s);
end
dSdt = mu.*N + nu.*R - lambda.*S - mu.*S + deltaS;
dEdt = lambda.*S - sigma.*E - mu.*E + deltaE;
```

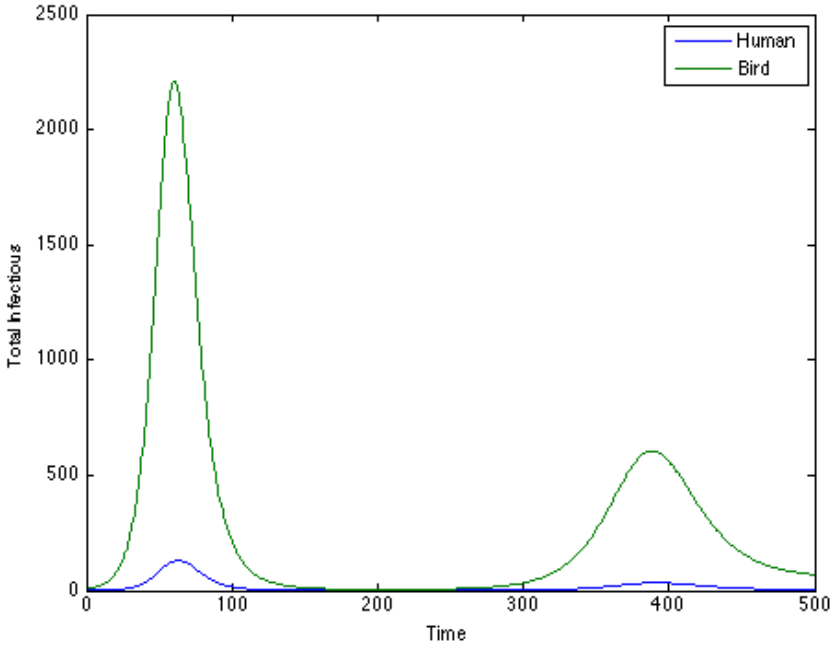


Figure A.3 – Simulation déterministe du modèle de la grippe aviaire implémenté en Matlab

```
60  dIdt = sigma.*E - gamma.*I - mu.*I + deltaI;
61  dRdt = gamma.*I - mu.*R - nu.*R + deltaR;
62  dSdt = reshape(dSdt,[1 ns*np]); dEdt = reshape(dEdt,[1 ns*np]);
63  dIdt = reshape(dIdt,[1 ns*np]); dRdt = reshape(dRdt,[1 ns*np]);
64  res = [dSdt dEdt dIdt dRdt]'
```

Annexe B

Illustration de la pseudo-commutativité de la somme tensorielle

Dans cette partie, on prend un exemple pour justifier la pseudo-commutativité de la somme tensorielle :

$$Tr_1 \oplus Tr_2 = P_\sigma^T (Tr_2 \oplus Tr_1) P_\sigma$$

où σ est un ordre lexicographique qu'on choisi et P_σ est la matrice de permutation correspondante au choix σ.

Supposons deux automates stochastiques (SI et Paris-Lyon) dont les matrices de taux de transitions sont :

$$Tr_{SI} = \begin{pmatrix} -\lambda & \lambda \\ 0 & 0 \end{pmatrix} \quad and \quad Tr_{sp} = \begin{pmatrix} -a_1 & a_1 \\ a_2 & -a_2 \end{pmatrix}$$

Les ensembles d'états de deux automates : $\mathcal{S}_{SI} = \{S, I\}$, $\mathcal{S}_{sp} = \{Pr, Ly\}$. En prédéfinissant l'ordre lexicographique comme $\sigma = [C_{SI}, C_{sp}]$, les ordres des lignes/colonnes dans la matrice de résultat sont $[S_{Pr}, S_{Ly}, I_{Pr}, I_{Ly}]$. La matrice de permutation correspondante qui permet de réarranger la matrice produit $Tr_{sp} \oplus Tr_{SI}$ est :

$$P_\sigma = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Le résultat de $Tr_{SI} \oplus Tr_{sp}$ est :

$$Tr = \left(\begin{array}{c|cccc} & S_{Pr} & S_{Ly} & I_{Pr} & I_{Ly} \\ \hline S_{Pr} & -(a_1+\lambda) & a_1 & \lambda & 0 \\ S_{Ly} & a_2 & -(a_2+\lambda) & 0 & \lambda \\ I_{Pr} & 0 & 0 & -a_1 & a_1 \\ I_{Ly} & 0 & 0 & a_2 & -a_2 \end{array}\right)$$

En appliquant l'équation 4.13, on obtient le résultat suivant :

$$Tr' = P_\sigma^T (Tr_{sp} \oplus Tr_{SI}) P_\sigma$$

$$= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \left(\begin{array}{c|cccc} & S_{Pr} & I_{Pr} & S_{Ly} & I_{Ly} \\ \hline S_{Pr} & -(a_1+\lambda) & \lambda & a_1 & 0 \\ I_{Pr} & 0 & -a_1 & 0 & a_1 \\ S_{Ly} & a_2 & 0 & -(a_2+\lambda) & \lambda \\ I_{Ly} & 0 & a_2 & 0 & -a_2 \end{array}\right) \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$= \left(\begin{array}{c|cccc} & S_{Pr} & S_{Ly} & I_{Pr} & I_{Ly} \\ \hline S_{Pr} & -(a_1+\lambda) & a_1 & \lambda & 0 \\ S_{Ly} & a_2 & -(a_2+\lambda) & 0 & \lambda \\ I_{Pr} & 0 & 0 & -a_1 & a_1 \\ I_{Ly} & 0 & 0 & a_2 & -a_2 \end{array}\right) = Tr$$

Annexe C

Pharo en bref

Cette annexe s'inspire fortement de la documentation de Pharo.

Le langage métier KENDRICK a été réalisé au moyen du langage de programmation Pharo [1].

Pharo est un langage moderne, open-source, dynamiquement typé supportant une approche de programmation "live" fortement inspiré de Smalltalk. Pharo et son écosystème sont basé sur les éléments fondamentaux suivants :

- un langage de programmation où tout est objet (y compris les classes), tout se passe par envoi de messages entre objets et où tout est réflexif (i.e. il est possible de modifier la structure et le comportement des objets lors de l'exécution).
- Un langage à typage dynamique avec une syntaxe si simple qu'elle peut tenir sur une carte postale et est lisible même pour quelqu'un qui ne serait pas familier avec elle.
- Un environnement de développement intégré "live" qui permet à la fois de naviguer dans le code mais également d'interagir directement avec les objets et de modifier le code à chaud lors de son exécution.
- Une bibliothèque de classes importantes qu permet de créer un environnement qu peut-être vu comme un OS virtuel et qui inclue une machine virtuelle JIT, l'accès aux fonctionnalités du système par l'intermédiaire de FFI.

La machine virtuelle de Pharo écrite en large partie en Pharo lui-même, ce qui lui permet d'être multi-plateforme (Mac OS X, Windows, Linux, iOS, Android et Raspberry Pi).

Nous avons tiré parti de l'ensemble des fonctionalités de Pharo pour prototyper les différentes versions de KENDRICK.

Moose [2] est une plateforme de réingénierie logicielle développée depuis 1996 par l'équipe SCG (Software Composition Group) de l'Université de Berne. Cette plateforme offre un certain nombres de fonctionnalités dont nous avons également tiré partie à l'occasion de cette thèse : analyseur syntaxique (PetitParseur), méta=modélisation (méta-méta-

1. `http://www.pharo.org/`
2. `http://moosetechnology.org/`

modèle FM3 [3]), métriques logicielles, visualisation agile et interactive (ROASSAL [4]).

C.1 Éléments de syntaxes

La syntaxe de Pharo s'inspire en grande partie de Smalltalk avec l'idée qu'il doit être possible de lire un programme comme on lirait un texte en anglais.

Les expressions Pharo sont formées à partir :
- 6 mots clés réservés ou pseudo-variables (qui ne peuvent pas être modifié directement par l'utilisateur) : **self**, **super**, **nil**, **true**, **false** et **thisContext**,
- des littéraux qui représentant des objets de base comme nombres, caractères, chaînes de caractères, symboles et tableaux,
- déclarations de variables (avec des barres verticales),
- affectations (**:=**),
- clotures lexicales,
- envois de messages.

La syntaxe des éléments de base de est résumée dans la table C.1.

Table C.1 – Résumé de la syntaxe de Pharo

Syntaxe	Ce qu'elle représente
1	entier décimal
2r101	entier avec notation binaire
1.5	nombre flottant
2.4e7	entier avec notation exponentielle
$a	caractère
'bonjour'	chaîne de caractères
#Bonjour	symbole #Bonjour
#(1 2 3)	tableau de littéraux
{1. 2. 1+2}	tableau dynamique
"c'est mon commentaire"	commentaire
|x y|	déclaration de deux variables x et y
x := 1	affectation de 1 à x
[x + y]	bloc qui évalue x+y
3 factorial	message unaire
3 + 4	message binaire
2 raisedTo : 6 modulo : 10	message à mots-clés

Tout programme Pharo s'exécute en envoyant des messages à des objets. Tout envoi de messages retourne un objet. Il y a trois type d'envois de messages dans Pharo :

3. qui s'inspire en grande partie de EMOF.
4. `http://agilevisualization.com/`

1. Les messages unaires : messages sans arguments. Par exemple : 3 `factorial` envoie le message `factorial` à l'objet 3 et retourne comme résultat 6.
2. Les messages binaires : message avec un seul argument. 1+2 envoie le message + avec l'argument 2 à 1. L'objet retourné est 3.
3. Les messages à mots-clés : message qui comporte un nombre arbitraire d'arguments. 2 `raisedTo:` 6 `modulo:` 10 envoie le message comprenant le sélecteur `raisedTo:modulo:` et les arguments 6 et 10 vers l'objet 2. On notera que les arguments sont insérés dans le message à mots-clés de telle sorte à pouvoir lire facilement ce genre de message.

Les messages unaires ont la plus haute priorité, puis viennent les message binaires et pour finir, les messages à mots-clés, ainsi : `2 raisedTo: 1 + 3 factorial` nous donne 128 (le message `factorial` est envoyé à 3, puis + 6 est envoyé à 1 et pour finir, `raisedTo: 7` est envoyé à 2). Les parenthèses permettent de changer l'ordre des priorités d'envois de messages.

On va brièvement introduire la syntaxe de blocs Pharo car les blocs constitue un élément important pour construire des langages internes à Pharo.

C.2 Syntaxe des blocs

Les blocs (également appelés fermetures lexicales) apportent un moyen de différer l'évaluation d'une expression. Un bloc est essentiellement une fonction anonyme, qui est évalué en lui envoyant le message `value`. Il va retourner la valeur de la dernière expression de son corps, à moins q'il y ait un retour explicite (avec ^). Par exemple : [1 + 2] value –> 3.

Les blocs peuvent prendre des paramètres, chacun doit être déclaré en le précédant d'un deux-points. Une barre verticale sépare les déclarations des paramètres et le corps du bloc. Pour évaluer un bloc avec un paramètre, il faut lui envoyer le message `value:` avec un argument. Un bloc à deux arguments doit recevoir le message `value:value:` et ainsi de suite jusqu'à 4 arguments.

```
[:x| 1 + x] value: 2.

 --> 3

[:x:y| x+y] value: 1 value: 2.

 --> 3
```

Des blocs peuvent aussi déclarer des variables locales, lesquelles seront entourées par des barres verticales, tout comme des déclaration de variables locales.

Les variables locales sont déclarées après les éventuels arguments :

```
[:x :y |
   |z| z := x + y. z]
   value: 1 value: 2.
```

```
--> 3
```

Les blocs sont en fait des fermetures lexicales, puisqu'ils font référence à des variables de leur environnement immédiat. Le bloc suivant fait référence à la variable x défini dans son environnement englobantdans.

```
|x|
x := 1.
[:y| x + y ] value: 2.

 --> 3
```

L'API de Kendrick introduit dans le chapitre 5 utilise fortement les blocs Pharo afin de définir les les taux de transition fonctionnels. Voir un exemple du modèle de réseau dans D.1.2 (les lignes 24-31) où l'on a défini la probabilité d'infection `lambda` comme un taux de transition fonctionnel, dépendant du compartiment où se trouve l'individu et ses contacts infectieux.

C.3 Conditions et Itérations

Pharo n'offre aucune syntaxe spécifique pour les structures de contrôle. Typiquement celles-ci sont obtenues par l'envoi de messages à des booléens, des nombres ou des collections, en prenant comme arguments des blocs. Les expressions conditionnelles sont obtenues par les envois de messages `ifTrue:`, `ifFalse:`, ou `ifTrue: ifFalse:` .

Par exemple :

```
(17 * 23) > 220
      ifTrue: ['plus grand]
      ifFalse: ['plus petit'] --> 'plus grand'
```

Les boucles ou itérations sont obtenues généralement par l'envoi de messages à des blocs, des entiers ou des collections. Comme la condition de sortie d'un boucle peut être évaluée de façon répétitive, elle se présentera sous la forme d'un bloc plutôt que de celle d'une valeur booléenne. Donnons quelques exemples d'une itération :

```
n := 1.
[n < 1000] whileTrue: [n := n * 2].
n --> 1024
```

`whileFalse:` inverse la condition de sortie :

```
n := 1.
[n > 1000] whileFalse: [n := n * 2].
n --> 1024
```

`timesRepeat:` offre un moyen simple pour implémenter la répétition d'une action un nombre donné d'itérations :

```
n := 1.
10 timesRepeat: [n := n * 2].
n --> 1024
```

Nous pouvons aussi envoyer le message `to:do:` à un entier qui devient alors la valeur initiale d'un compteur de boucle. Le premier argument est la borne supérieure, le second est un bloc qui prend la valeur courante du compteur de boucle comme argument :

```
n := 0.
1 to: 10 do: [:counter| n := n + counter].
n --> 55
```

Annexe D

Scripts KENDRICK de modèles épidémiologiques

Dans cette partie, nous présentons le code KENDRICK de modèles épidémiologiques du plus simple au plus complexe. On trouvera d'autres modèles sur le site de développement de KENDRICK [1].

D.1 Certains modèles utilisent les API de KENDRICK

Dans cette section, nous présentons les scripts complètes des modèles épidémiologiques qui ont été abordés dans le chapitre 5 en utilisant les API de KENDRICK.

D.1.1 Modèle SIR spatial dans six pays de l'Afrique

```
model map spatialConcern sirConcern simulator diagBuilder mapBuilder
model := KEModel new population: (KEPopulation size: 6000).

map := KEMap new.
map countries: #(#Liberia #Guinea #SierraLeone #Nigeria #Senegal #Niger).
map routesFrom: #Liberia toAll: #(#Guinea #SierraLeone #Senegal).
map routesFrom: #Guinea toAll: #(#Nigeria).
map routesFrom: #SierraLeone toAll: #(#Senegal).
map routesFrom: #Senegal toAll: #(#Niger).
map routesFrom: #Niger toAll: #(#Liberia).
spatialConcern := KEConcern new.
spatialConcern addAttribute: #country value: map countries.
spatialConcern addParameter: #rho value: 0.05.
spatialConcern transitions: (map routesToTransitions: 'rho').

sirConcern := KEConcern new.
sirConcern addAttribute: #status value: #(S I R).
sirConcern addParameters: #(#lambda #beta #gamma).
sirConcern
  addTransitionFrom: { #status->#S }
  to: { #status->#I }
```

1. https ://github.com/UMMISCO/kendrick/wiki

```
  probability: 'lambda'.
sirConcern
  addTransitionFrom: { #status->#I }
  to: { #status->#R }
  probability: 'gamma'.

model integrate: sirConcern.
model integrate: spatialConcern.

model atParameter: #beta assignValue: 0.0002.
model atParameter: #gamma assignValue: 0.1.
model atParameter: #lambda assignValue: [ :aModel| |c|
  c := aModel currentCompartment at: #country.
  (aModel atParameter: #beta)*(aModel atCompartment: {#status->#I. #country->c}) ].

#(#Guinea #SierraLeone #Nigeria #Senegal #Niger) do: [ :c|
  (model atCompartment: { #status->#S. #country->c } put: 1000)
  ].
model atCompartment: { #status->#S. #country->#Liberia } put: 700.
model atCompartment: { #status->#I. #country->#Liberia } put: 300.

simulator := KESimulator new: #IBM from: 0 to: 100 step: 0.1.
simulator executeOn: model.
mapBuilder := KEMapBuilder africa.
mapBuilder data: (mapBuilder countries collect: [:c|
  ((model atAttribute: #country) includes: c)
  ifTrue: [ (simulator timeSeriesAt: {#status->#I. #country->c}) first peakOfEpidemic ]
  ifFalse: [ 0 ]
  ]).
mapBuilder open.
diagBuilder := KEDiagramBuilder new data: (simulator timeSeriesAt: {#status->#I}).
diagBuilder open.
```

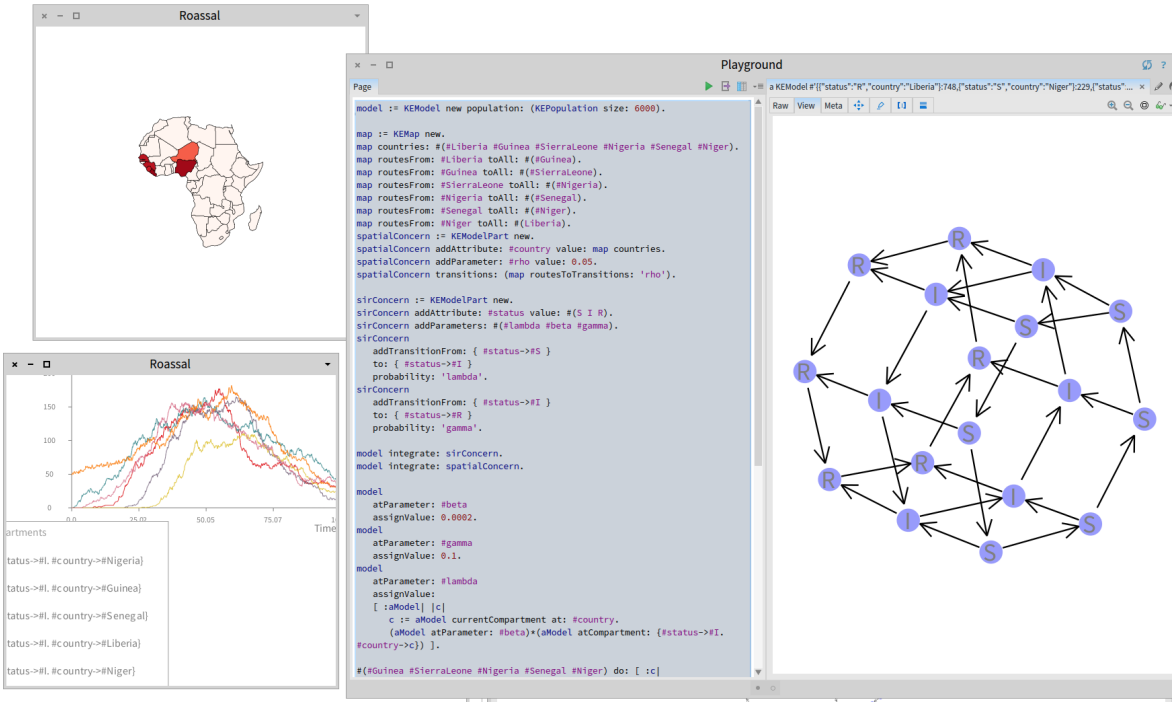


Figure D.1 – Vue d'ensemble des outils de la plateforme Kendrick

D.1.2 Un exemple du modèle réseau

```
model sirConcern spatialConcern network simulator nb
model := KEModel new population: (KEPopulation size: 100).

sirConcern := KEConcern new.
sirConcern addAttribute: #status value: #(S I R).
sirConcern addParameters: { #beta. #gamma. #lambda }.
sirConcern
  addTransitionFrom: { #status->#S }
  to: { #status->#I }
  probability: 'lambda'.
sirConcern
  addTransitionFrom: { #status->#I }
  to: { #status->#R }
  probability: 'gamma'.

network := KEContactNetwork nodes: 100 topology: { #random. #p->0.02 }.
spatialConcern := KEConcern new.
spatialConcern addParameter: #network value: network.
spatialConcern addAttribute: #node value: network allContacts.

model integrate: sirConcern.
model integrate: spatialConcern.

model
  atParameter: #lambda
  assignValue: [ :aModel
    node
    node := aModel currentCompartment at: #node.
    ((aModel atParameter: #network)
      contactsOf: {aModel. #node->node. #status->#I})
    * (aModel atParameter: #beta)/(aModel atParameter: #N) ].
model atParameter: #beta assignValue: 100.
model atParameter: #gamma assignValue: 0.1.

1 to: 99 do: [:i
  model atCompartment: {#status->#S. #node->i asString asSymbol} put: 1].
model atCompartment: { #status->#I. #node->#'100' } put: 1.

simulator := KESimulator new: #IBM from: 0.0 to: 50 step: 0.1.
simulator executeOn: model.
nb := KENetworkBuilder new
    data: simulator allTimeSeries;
    network: (model atParameter: #network);
    status: #(#S #I #R);
    colors: #(#green #red #blue);
    viewDataAtTime: 10;
    legend: 'random network, p = 0.02'.
nb open
```

D.2 Certains modèles utilisant le langage métier Kendrick

D.2.1 Modèle de la rougeole

```
KendrickModel  Measles
```

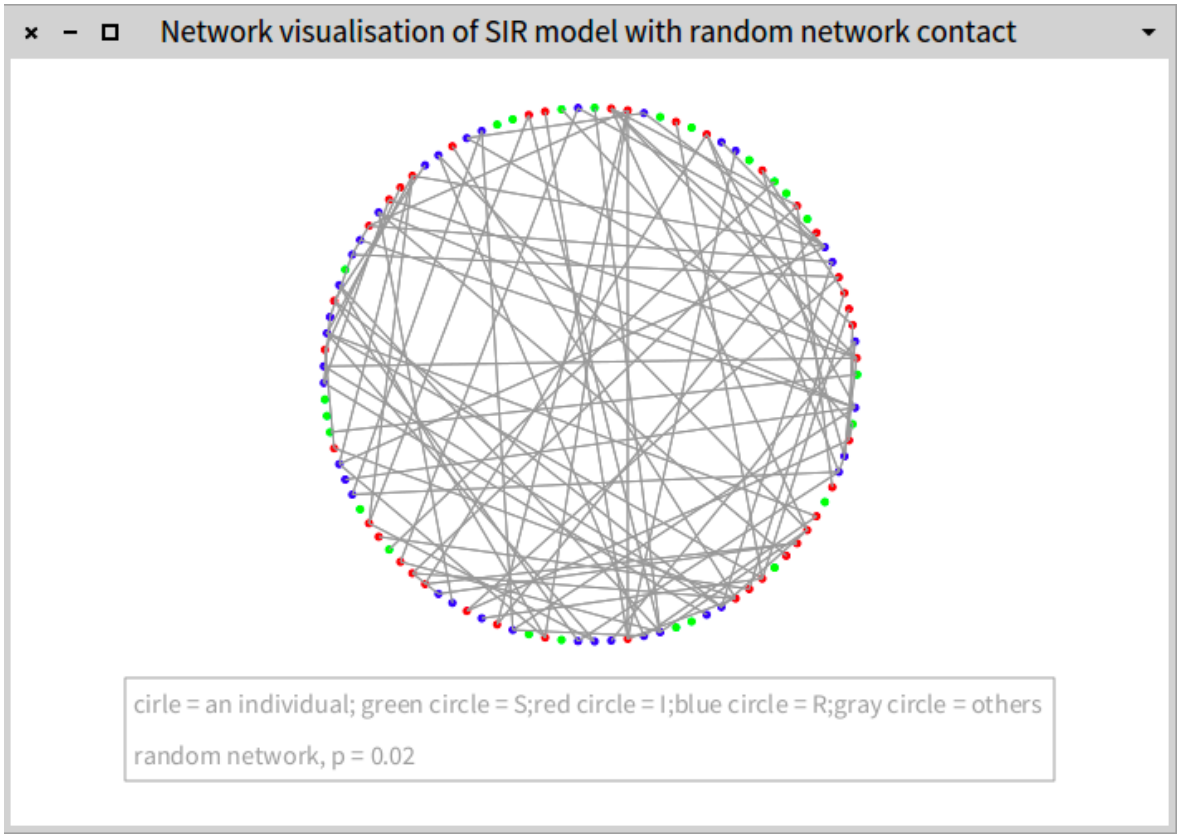


Figure D.2 – La visualisation du réseau de contacts entre individus avec ROASSAL

```
2    attribute: #(status -> S E I R);
3    parameters: #(beta gamma mu sigma);
4    equations: #(
5       S:t=mu*N - beta*S*I - mu*S.
6        E:t=beta*S*I - sigma*E - mu*E.
7        I:t=sigma*E - gamma*I - mu*I.
8        R:t=gamma*I - mu*R.
9    );
10   population: 100000;
11   S: 99999;
12   I: 1;
13   others: 0;
14   beta: 0.0000214;
15   gamma: 0.143;
16   mu: 0.0000351;
17   sigma: 0.125.
18
19 Simulation MeaslesRKSim rungeKutta
20   for: 'Measles';
21   from: 0.0;
22   to: 150;
23   step: 1.
24
25 Visualization MeaslesDiagramViz diagram
26   for: 'MeaslesRKSim';
27   xLabel: 'Time (days)'.
```

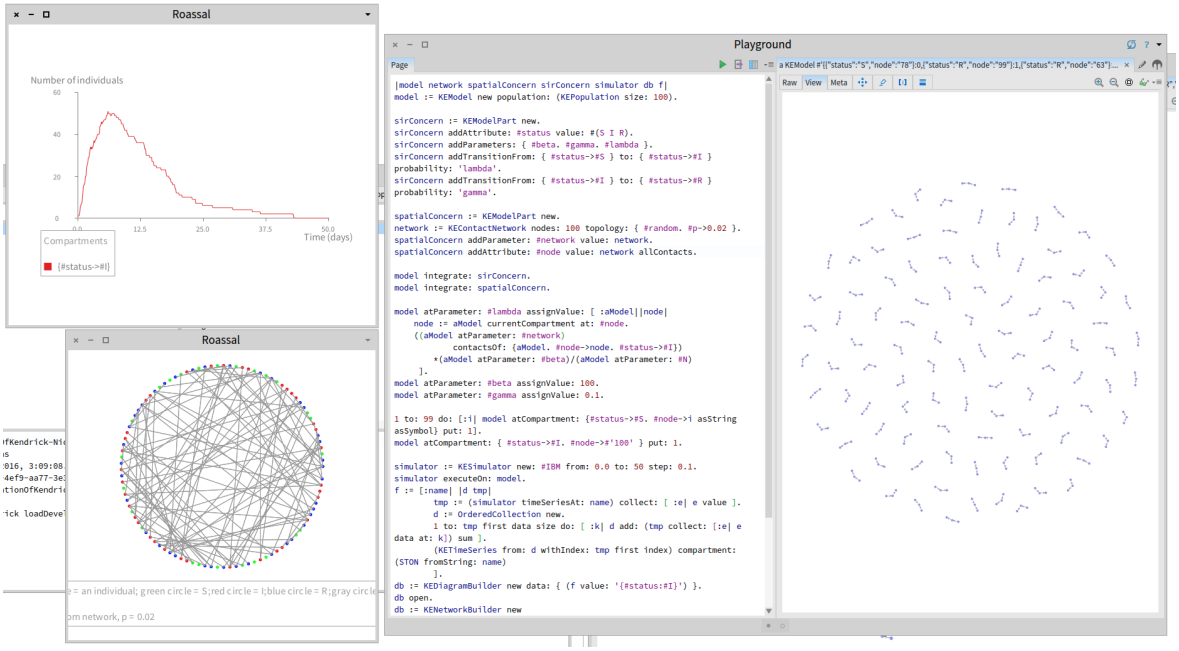


Figure D.3 – La capture d'écran de la spécification et visualisation du modèle de réseau en Kendrick

D.2.2 Modèle multi-espèces d'une maladie vectorielle

Une autre façon pour définir le modèle multi-espèces qui a été présenté dans le chapitre 6. Dans cet exemple, nous définissons un modèle VectorBorneDisease, que nous intégrons avec la préoccupation ThreeSpecies.

```
KendrickModel VectorBorneDisease
  population: 13000;
  attribute: #(status -> S I R);
  parameters: #(mu gamma beta lambda);
  mu: 12.17;
  gamma: 52;
  beta: 1;
  lambda: #(beta*I);
  transitions: #(
    S -- lambda --> I.
    I -- gamma --> R.
    status -- mu --> Empty.
    Empty -- mu --> S.
  ).

Concern ThreeSpecies
  attribute: #(species -> mosquito reservoir1 reservoir2).

Composition VBD3Species
  model: 'VectorBorneDisease';
  concern: 'ThreeSpecies'.

Scenario VBDScr1
  on: 'VBD3Species'.
  S_species: #(9999 1000 2000);
  I_species: #(1 0 0);
  N: #(species);
```

```
28    mu_species: #(12.17 0.05 0.05);
29    beta_species: #(
30      #(0 0.02 0.02)
31      #(0.02 0 0)
32      #(0.02 0 0)
33    );
34    lambda: #(beta*I sum).
35
36  Simulation VBDStc gillespie
37    scenario: 'VBDScr1';
38    from: 0.0;
39    to: 0.5;
40    step: 1/365.
41
42  Visualization VBDViz diagram
43    for: 'VBDStc';
44    data: #(I sqrt);
45    xLabel: 'Time (years)'.
```

D.2.3 Modèle spatial pour Ebola

Ce modèle est le même modèle présenté dans la section D.1.1 en utilisant les API de Kendrick.

```
1   KendrickModel Ebola.
2
3   Concern SIR
4     attribute: #(status -> S I R);
5     parameters: #(lambda beta gamma);
6     lambda: #(beta*I);
7     transitions: #(
8       S -- lambda --> I.
9       I -- gamma --> R.).
10
11  Concern Spatial
12    attribute: #(country -> Liberia Guinea SierraLeone Nigeria Senegal Niger);
13    parameters: #(rho);
14    transitions: #(
15      Liberia -- rho --> Guinea.
16      Liberia -- rho --> SierraLeone.
17      Liberia -- rho --> Senegal.
18      Guinea -- rho --> Nigeria.
19      SierraLeone -- rho --> Senegal.
20      Senegal -- rho --> Niger.
21      Niger -- rho --> Liberia.
22    ).
23
24  Composition SpatialEbola
25    model: 'Ebola';
26    concern: 'SIR';
27    concern: 'Spatial'.
28
29  Scenario EbolaScr1
30    on: 'SpatialEbola';
31    populationSize: 6000;
32    lambda: #(beta*I_country);
33    beta: 0.0002;
34    gamma: 0.1;
35    rho: 0.05;
```

```
36    S_country: #(700 1000 1000 1000 1000 1000);
37    I_country: #(300 0 0 0 0 0).
38
39  Simulation RungeKuttaEbola rungeKutta
40    scenario: 'EbolaScr1';
41    from: 0;
42    to: 100;
43    step: 0.1.
44
45  (Visualization EbolaViz1 africa
46      for: 'RungeKuttaEbola';
47      data: #(country I_country peakOfEpidemic)) open.
```

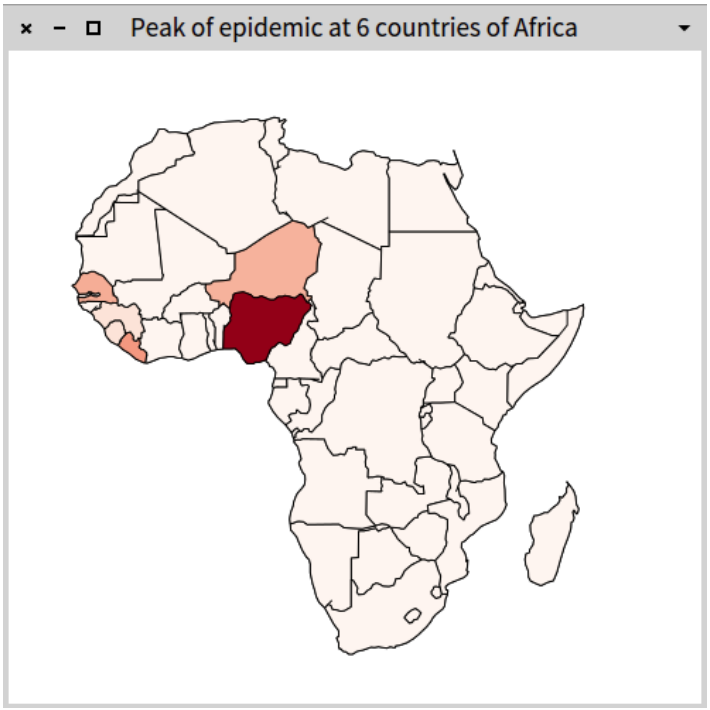


Figure D.4 – Visualisations du modèle Ebola

D.2.4 Modèle multi-espèces spatial de la grippe aviaire

```
1   Concern SEIRS
2       attribute: #(status -> S E I R);
3       parameters: #(beta lambda gamma mu sigma nu);
4       lambda: #(beta*I/N);
5       transitions: #(
6           S -- lambda --> E.
7           E -- sigma --> I.
8           I -- gamma --> R.
9           R -- nu --> S.
10          status -- mu --> Empty.
11          Empty -- mu --> S).
12
13  Concern TwoSpecies
14    attribute: #(species -> humans birds).
15
16  Map IndoChina
17      for: #(country -> Thailand Laos Vietnam Cambodia MyanmarBurma);
```

```
    borders: #(
         #(0 1 0 1 1)
         #(1 0 1 1 1)
         #(0 1 0 1 0)
         #(1 1 1 0 0)
         #(1 1 0 0 0)
).
Concern Spatial
    maps: 'IndoChina';
    withTransitionRate: #(rho).

KendrickModel Influenza.

Composition AvianInfluenza
    model: 'Influenza';
    concern: 'SEIRS';
    concern: 'MultiSpecies';
    concern: 'Spatial'.

Scenario AIScr1
    on: 'AvianInfluenza';
    populationSize: 27500;
    mu_species: #(0.000365 0.00137);
    beta_species: #(
         #(0 0.21)
         #(0 0.42));
    gamma_species: #(0.25 0.233);
    sigma_species: #(0.5 0.67);
    nu: 0.00274;
    rho_species: #(0.03 0.1);
    lambda: #(beta*I_country/N sum);
    N: #(species_country);
    S_species_country: #(
         #(500 500 500 500 500)
         #(4990 5000 5000 5000 5000));
    I_species_country: #(
         #(0 0 0 0 0)
         #(10 0 0 0 0)).

Simulation AIRK rungeKutta
  scenario: 'AIScr1';
  from: 0;
  to: 500;
  step: 1.

Visualization AIViz1 diagram
  for: 'AIRK';
  data: #(I_species);
  legendTitle: 'Total of Infectious';
  legends: #('humans' 'birds');
  xLabel: 'Time (days)'.

Visualization AIViz2 map
    for: 'AIRK';
    data: #(country I_bird_country peakOfEpidemic).
```

D.2.5 Modèle spatial multi-espèces et multi-souches de la grippe aviaire

```
Concern SEIRS
```

```
    attribute: #(status -> S E I R);
    parameters: #(beta lambda gamma mu sigma nu);
    lambda: #(beta*I/N);
    transitions: #(
        S -- lambda --> E.
        E -- sigma --> I.
        I -- gamma --> R.
        R -- nu --> S.
        status -- mu --> Empty.
        Empty -- mu --> S).

Concern TwoSpecies
  attribute: #(species -> humans birds).

Map IndoChina
    for: #(country -> Thailand Laos Vietnam Cambodia MyanmarBurma);
    borders: #(
         #(0 1 0 1 1)
         #(1 0 1 1 1)
         #(0 1 0 1 0)
         #(1 1 1 0 0)
         #(1 1 0 0 0)
).
Concern Spatial
    maps: 'IndoChina';
    withTransitionRate: #(rho).

Concern SEIRSTwoStrains
  extends: 'SEIRS';
  parameters: #(betaA betaB);
  splitStatus: #(I index #(a b)).

KendrickModel Influenza.

Composition 'AvianInfluenza'
  model: 'Influenza';
  concern: 'SEIRSTwoStrains';
  concern: 'TwoSpecies';
  concern: 'Spatial'.

Scenario AIScr2
  on: 'AvianInfluenza';
  populationSize: 27500;
  mu_species: #(0.000365 0.00137);
  betaA_species: #(
    #(0 0.21)
    #(0 0.42));
  betaB_species: #(
    #(0 0.021)
    #(0 0.042));
  gamma_species_Ia: #(0.25 0.233);
  gamma_species_Ib: #(0.0025 0.0023);
  sigma_species_Ia: #(0.5 0.67);
  sigma_species_Ib: #(0.005 0.0067);
  nu: 0.00274;
  rho_species: #(0.03 0.1);
  lambda: #((betaA*(Ia_country/N))+(betaB*(Ib_country/N)) sum);
  N: #(species_country);
  S_species_country: #(
    #(500 500 500 500 500)
    #(4990 5000 5000 5000 5000));
  Ia_species_country: #(
    #(0 0 0 0 0)
```

```
    #(9 0 0 0 0));
  Ib_species_country: #(
    #(0 0 0 0 0)
    #(1 0 0 0 0)).

Simulation AIRK rungeKutta
  scenario: 'AIScr2';
  from: 0.0;
  to: 500;
  step: 1.

Visualization AIViz3 diagram
  for: 'AIRK';
  data: #(Ia_species Ib_species);
  legendTitle: 'Total of Infectious';
  legends: #('birds strains 1' 'humans strains 1' 'birds strains 2' 'humans strains 2');
  xLabel: 'Time (days)'.
```

D.2.6 Modèle spatial multi-espèces spatial de la grippe aviaire incorporant la quarantaine

```
Concern SEIRS
    attribute: #(status -> S E I R);
    parameters: #(beta lambda gamma mu sigma nu);
    lambda: #(beta*I/N);
    transitions: #(
        S -- lambda --> E.
        E -- sigma --> I.
        I -- gamma --> R.
        R -- nu --> S.
        status -- mu --> Empty.
        Empty -- mu --> S).

Concern TwoSpecies
  attribute: #(species -> humans birds).

Map IndoChina
    for: #(country -> Thailand Laos Vietnam Cambodia MyanmarBurma);
    borders: #(
         #(0 1 0 1 1)
         #(1 0 1 1 1)
         #(0 1 0 1 0)
         #(1 1 1 0 0)
         #(1 1 0 0 0)
).
Concern Spatial
    maps: 'IndoChina';
    withTransitionRate: #(rho).

Concern SEIRSQ
  extends: 'SEIRS';
  parameters: #(delta epsilon);
  addStatus: #(Q);
  addTransition: #(I -- delta --> Q);
  addTransition: #(Q -- epsilon --> R);
  addTransition: #(Q -- mu --> Empty);
  lambda: #(beta*I/(N-Q)).

KendrickModel Influenza.
```

```
39
40 Composition AvianInfluenza
41   model: 'Influenza';
42   concern: 'SEIRSQ';
43   concern: 'TwoSpecies';
44   concern: 'Spatial'.
45
46 Scenario AIScr1
47     on: 'AvianInfluenza';
48     populationSize: 27500;
49     mu_species: #(0.000365 0.00137);
50     beta_species: #(
51          #(0 0.21)
52          #(0 0.42));
53     gamma_species: #(0.25 0.233);
54     sigma_species: #(0.5 0.67);
55     nu: 0.00274;
56     rho_species: #(0.03 0.1);
57     lambda: #(beta*I_country/N sum);
58     N: #(species_country);
59     S_species_country: #(
60          #(500 500 500 500 500)
61          #(4990 5000 5000 5000 5000));
62     I_species_country: #(
63          #(0 0 0 0 0)
64          #(10 0 0 0 0)).
65
66 Scenario AIScr3
67   on: 'AvianInfluenza';
68   delta_species: #(0.068 0.055);
69   epsilon_species: #(0.096 0.082);
70   rho_species_Q: #(0.03 0.1 0);
71   lambda: #(beta*(I_country/(N-Q_country)) sum).
72
73 Simulation AIRK rungeKutta
74   scenarios: #(AIScr1 AIScr3);
75   from: 0;
76   to: 500;
77   step: 1.
78
79 Visualization AIViz1 diagram
80   for: 'AIRK';
81   data: #(I_species);
82   legendTitle: 'Total of Infectious';
83   legends: #('humans' 'birds');
84   xLabel: 'Time (days)'.
```

D.3 Modèle de la rougeole généré en code C/C++ à partir d'un modèle Kendrick

Le code C/C++ généré par un visiteur sur un modèle stochastique de la rougeole (Chapitre 3) est le suivant :

```
1 //stochastic model
2 #include <iostream>
3 #include <string>
4 #include <stdio.h>
```

```
5  #include <stdlib.h>
6  #include <math.h>
7  #include <time.h>
8  #include <fstream>
9  #include <sstream>
10
11 using namespace std;
12
13 const double TMAX = 100.0;
14 const double TMIN = 0.0;
15 const double STEP = 0.001;
16 const int NB_EVENTS = 8;
17
18 const int nbOfCompartments = 4;
19
20 double t = TMIN;
21 double initialValue[nbOfCompartments];
22 string compartmentName[nbOfCompartments];
23 double x[nbOfCompartments];
24 double rates[NB_EVENTS];
25
26 const double beta = 2.14e-5;
27 const double sigma = 0.125;
28 const double gamma = 0.143;
29 const double mu = 3.51e-5;
30 const double N = 100000.0;
31 void initializeCompartments()
32 {
33     compartmentName[0] = "I";
34     initialValue[0] = 1;
35     compartmentName[1] = "S";
36     initialValue[1] = 99999;
37     compartmentName[2] = "E";
38     initialValue[2] = 0;
39     compartmentName[3] = "R";
40     initialValue[3] = 0;
41 }
42 double sum(double a[], int n) {
43     double s=0.0;
44     for (int i=0; i < n; i++)
45         s += a[i];
46     return s;
47 }
48 int selectEvent(double sumOfRates, double r) {
49     int event = -1;
50     double sp = 0.0;
51     double p = 0.0;
52     p = r * sumOfRates;
53     for (int i = 0; i < NB_EVENTS; i++) {
54         sp += rates[i];
55         if (p <= sp) {
56             event = i;
57             break;
58         }
59     }
60     return event;
61 }
62 void calculateRate()
63 {
64     rates[0] = (-(gamma*x[0]));
65     rates[1] = (sigma*x[2]);
66     rates[2] = (-((beta*x[1])*x[0]));
67     rates[3] = (-(mu*x[1]));
```

```
68      rates[4] = (-(mu*x[3]));
69      rates[5] = (-(mu*x[2]));
70      rates[6] = (-(mu*x[0]));
71      rates[7] = (mu*N);
72  }
73  void Transition(int pEvent)
74  {
75      switch(pEvent) {
76          case 0:
77              x[0]=x[0]-1;
78              x[3]=x[3]+1;
79              break;
80          case 1:
81              x[0]=x[0]+1;
82              x[2]=x[2]-1;
83              break;
84          case 2:
85              x[2]=x[2]+1;
86              x[1]=x[1]-1;
87              break;
88          case 3:
89              x[1]=x[1]-1;
90              break;
91          case 4:
92              x[3]=x[3]-1;
93              break;
94          case 5:
95              x[2]=x[2]-1;
96              break;
97          case 6:
98              x[0]=x[0]-1;
99              break;
100         case 7:
101             x[1]=x[1]+1;
102             break;
103         default: cout<<"Error"<<endl;
104             break;
105     }
106 }
107 int main(int argc, char *argv[]){
108     double sumOfRates = 0.0;
109     double previousTime = TMIN;
110     double r1 = 0.0;
111     double r2 = 0.0;
112     int event = -1;
113     double tOffset = 0.0;
114     //Initialize all values of x
115     initializeCompartments();
116     for (int i = 0; i < nbOfCompartments; i++)
117             x[i] = initialValue[i];
118     //prepare file for saving data
119     stringstream fname;
120     fname << "data_";
121     for (int i = 0; i < nbOfCompartments; i++)
122         fname << compartmentName[i];
123     fname << "_stc.txt";
124     ofstream f(fname.str().data());
125     cout << "Starting..." << endl;
126     srand(time(0));
127     //Begin of algorithm Gillespie
128     if (f.is_open())
129     {
130         while (t < TMAX) {
```

```
131             //calculation of rates
132             calculateRate();
133             for (int i = 0; i < NB_EVENTS; i++)
134                 if (rates[i] < 0)
135                     rates[i] = rates[i]*(-1);
136             //Sum of rates
137             sumOfRates = sum(rates, NB_EVENTS);
138             //Generation of time for the next calculation
139             r1 = (double)rand()/RAND_MAX;
140             r2 = (double)rand()/RAND_MAX;
141             tOffset = (-1/sumOfRates)*log(r1);
142             //select event
143             event = selectEvent(sumOfRates, r2);
144             Transition(event);
145             if (t > (previousTime + STEP)){
146                 f << t << '\t';
147                 for (int i = 0; i < nbOfCompartments; i++)
148                         f << (x[i]) << '\t';
149                 f << '\n';
150                 previousTime = t;
151             }
152             t += tOffset;
153         }
154         f.close();
155     }
156     cout << "Finished..." << endl;
157 }
```

Annexe E

Grammaire de KENDRICK en EBNF

⟨*KENDRICK-SCRIPT*⟩ : := ⟨*KENDRICK-ENTITIES*⟩+

⟨*KENDRICK-ENTITIES*⟩ : := ⟨*MODEL-DECLARATION*⟩
| ⟨*CONCERN-DECLARATION*⟩
| ⟨*COMPOSITION-DECLARATION*⟩
| ⟨*SIMULATION-DECLARATION*⟩
| ⟨*VISUALIZATION-DECLARATION*⟩

⟨*IDENTIFIER*⟩ : := ⟨*LETTER*⟩ (⟨*LETTER*⟩ | [0-9]+)*

⟨*COMPOSITE-IDENTIFIER*⟩ : := ⟨*IDENTIFIER*⟩ (_ ⟨*IDENTIFIER*⟩)+

⟨*LETTER*⟩ : := [a-zA-Z]

⟨*NUMBER*⟩ : := [0-9]+ (.[0-9]+) ?

⟨*STRING*⟩ : := ' [^] '

⟨*KEYWORD*⟩ : := ⟨*IDENTIFIER*⟩ :

⟨*VALUE*⟩ : := ⟨*NUMBER*⟩ | ⟨*STRING*⟩ | ⟨*ARRAY*⟩

⟨*SHORT-EQ-WITH-OP*⟩ : := ⟨*EQUATION-EXPRESSION*⟩ ⟨*BASIC-OP*⟩*

⟨*BASIC-OP*⟩ : := `sum`|`sqrt`|`size`|`min`|`max`|`avg`|`median`|`mean`

⟨*ARRAY*⟩ : := `#(` ((⟨*VALUE*⟩* | ⟨*SHORT-EQ-WITH-OP*⟩) `)`

⟨*COMMON-EXPRESSION*⟩ : := ⟨*ATTRIBUTE-DEFINITION*⟩
| ⟨*PARAMETERS-DEFINITION*⟩
| ⟨*ASSIGNMENT-CLAUSE*⟩
| ⟨*EQUATIONS-DEFINITION*⟩
| ⟨*TRANSITIONS-DEFINITION*⟩

⟨*ATTRIBUTE-DEFINITIONS*⟩ : := `attribute:` ⟨*ATTRIBUTE-ARRAY*⟩

⟨*ATTRIBUTE-ARRAY*⟩ : := `#(` ⟨*IDENTIFIER*⟩ `->` (⟨*IDENTIFIER*⟩ | ⟨*NUMBER*⟩)+ `)`

⟨*ASSIGNMENT-CLAUSE*⟩ : := ⟨*KEYWORD*⟩ ⟨*VALUE*⟩

⟨*PARAMETER-DEFINITIONS*⟩ : := `parameters:` ⟨*IDENTIFIER-ARRAY*⟩

⟨*IDENTIFIER-ARRAY*⟩ : := `#(` ⟨*IDENTIFIER*⟩+ `)`

⟨*EQUATIONS-CLAUSE*⟩ : := `equations:` ⟨*EQUATIONS-ARRAY*⟩

⟨*EQUATIONS-ARRAY*⟩ : := `#(` (⟨*EQUATION*⟩ .)+ `)`

⟨*EQUATION*⟩ : := ⟨*EQUATION-SIGNATURE*⟩ `=` ⟨*EQUATION-EXPRESSION*⟩

⟨*EQUATION-SIGNATURE*⟩ : := ⟨*IDENTIFIER*⟩ `:` ⟨*IDENTIFIER*⟩

⟨*EQUATION-EXPRESSION*⟩ : := ⟨*TERM*⟩ ((`+`|`-`) ⟨*EQUATION-EXPRESSION*⟩)*

⟨*TERM*⟩ : := ⟨*NUMBER*⟩|⟨*IDENTIFIER*⟩|⟨*COMPOSITE-IDENTIFIER*⟩|(EQUATION-EXPRESSION)|⟨*PRODUCT*⟩

⟨*PRODUCT*⟩ : := ⟨*TERM*⟩ `*` ⟨*TERM*⟩

⟨*TRANSITIONS-DEFINITION*⟩ : := `transitions:` ⟨*TRANSITIONS-ARRAY*⟩

⟨*TRANSITIONS-ARRAY*⟩ : := `#(` ⟨*TRANSITION*⟩+ `)`

⟨*TRANSITION*⟩ : := ⟨*IDENTIFIER*⟩ `-` ⟨*IDENTIFIER*⟩ `->` ⟨*IDENTIFIER*⟩

⟨*MODEL-DECLARATION*⟩ : := `KendrickModel` ⟨*IDENTIFIER*⟩ ⟨*MODEL-BODY*⟩

⟨*MODEL-BODY*⟩ : := (⟨*MODEL-EXPRESSION*⟩ `;`) * ⟨*MODEL-EXPRESSION*⟩ .

⟨*MODEL-EXPRESSION*⟩ : := ⟨*POPULATION-CLAUSE*⟩ | ⟨*COMMON-EXPRESSION*⟩

⟨*POPULATION-CLAUSE*⟩ : := `population:` ⟨*NUMBER*⟩

⟨*CONCERN-DECLARATION*⟩ : := `Concern` ⟨*IDENTIFIER*⟩ ⟨*CONCERN-BODY*⟩

⟨*CONCERN-BODY*⟩ : := (⟨*CONCERN-EXPRESSION*⟩ `;`) * ⟨*CONCERN-EXPRESSION*⟩ .

⟨*CONCERN-EXPRESSION*⟩ : := ⟨*EXTENTION-CLAUSE*⟩ | ⟨*OPERATOR-CLAUSE*⟩ | ⟨*COMMON-EXPRESSION*⟩

⟨*EXTENTION-CLAUSE*⟩ : := `extends:` ⟨*IDENTIFIER*⟩

⟨*OPERATOR-CLAUSE*⟩ : := ⟨*DELAY-CLAUSE*⟩
| ⟨*DIVIDE-CLAUSE*⟩
| ⟨*ADD-CLAUSE*⟩
| ⟨*TRANSITION-CLAUSE*⟩

⟨*DELAY-CLAUSE*⟩ : := `delay: #(` ⟨*IDENTIFIER*⟩ `,` ⟨*TRANSITION*⟩ `,` ⟨*IDENTIFIER*⟩ `)`

⟨*DIVIDE-CLAUSE*⟩ : := `divide: #(` ⟨*IDENTIFIER*⟩, `,` ⟨*IDENTIFIER*⟩ `,` ⟨*IDENTIFIER*⟩ `)`

⟨*ADD-CLAUSE*⟩ : := `addStatus: #(` ⟨*IDENTIFIER*⟩ `+` ⟨*IDENTIFIER*⟩ `)`

⟨*LINK-CLAUSE*⟩ : := `transition: #(` ⟨*TRANSITION*⟩ `)`

⟨*COMPOSITION-DECLARATION*⟩ : := `Composition` ⟨*IDENTIFIER*⟩ ⟨*COMPOSITION-BODY*⟩

⟨*COMPOSITION-BODY*⟩ : := (⟨*COMPOSITION-EXPRESSION*⟩ `;`) * ⟨*COMPOSITION-EXPRESSION*⟩ .

⟨*COMPOSITION-EXPRESSION*⟩ : := ⟨*MODEL-CLAUSE*⟩
| ⟨*CONCERN-CLAUSE*⟩
| ⟨*COMPOSITE-ASSIGNMENT*⟩
| ⟨*COMMON-EXPRESSION*⟩

⟨*MODEL-CLAUSE*⟩ : := `model:` ⟨*STRING*⟩

⟨*CONCERN-CLAUSE*⟩ : := `concern:` ⟨*STRING*⟩

⟨*COMPOSITE-ASSIGNMENT-CLAUSE*⟩ : := ⟨*COMPOSITE-IDENTIFIER*⟩ `:` ⟨*VALUE*⟩

⟨*SIMULATION-DECLARATION*⟩ : := `Simulation` ⟨*IDENTIFIER*⟩ ⟨*SIMULATION-MODIFIER*⟩ ⟨*SIMULATION-BODY*⟩

⟨*SIMULATION-MODIFIER*⟩ : := `RungeKutta` | `AB2` | `AM3` | `BDF4` | `Euler` | `Heun` | `ImplicitMidPoint` | `AB4` | `BeckwardEuler`
| `BDF3` | `Midpoint` | `Trapezoid` | `AB3` | `BDF2` | `AM4` | `Gillespie` | `GPUGillespie` | `TauLeap` | `IBM`

⟨*SIMULATION-BODY*⟩ : := (⟨*SIMULATION-EXPRESSION*⟩ `;`) * ⟨*SIMULATION-EXPRESSION*⟩ .

⟨*SIMULATION-EXPRESSION*⟩ : := ((`from:` | `to:` | `step:`) ⟨*NUMBER*⟩) | `for:` ⟨*IDENTIFIER*⟩

⟨*VISUALIZATION-DECLARATION*⟩ : := `Visualization` ⟨*IDENTIFIER*⟩ ⟨*VISUALIAZATION-MODIFIER*⟩ ⟨*VISUALIZATION-BOI*

⟨*VISUALIZATION-MODIFIER*⟩ : := `diagram` | `pieChart` | `barPlot` | `map`

⟨*VISUALIZATION-BODY*⟩ : := (⟨*VISUALIZATION-EXPRESSION*⟩ `;`) * ⟨*VISUALIZATION-EXPRESSION*⟩ .

⟨*VISUALIZATION-EXPRESSION*⟩ : := `for:` ⟨*IDENTIFIER*⟩
| (`xlabel:` | `ylabel:` | `legendTitle:`) ⟨*STRING*⟩
| `data:` ⟨*SHORT-EQ-WITH-OP*⟩
| `legends:` ⟨*ARRAY*⟩