وزارة التعليم العالي و البحث العلمي

BADJI MOKHTAR-ANNABA UNIVERSITY
UNIVERSITE BADJI MOKHTAR-ANNABA

جامعة باجي مختار– عنابـــة

Faculté des Sciences de l'Ingéniorat **Année 2014-2015**

Département d'Informatique

THESE

Présentée en vue de l'obtention
du diplôme de Doctorat en Sciences

CLASSIFICATION DES SYSTEMES BIO-INSPIRES : VERS UNE APPROCHE BASEE MODELES

Option
Génie Logiciel

Par

Mr Mili Seif Eddine

Directeur de Thèse
Mr Djamel MESLATI Professeur Université Badji Mokhtar-Annaba

DEVANT LE JURY

Président :	Salim GHANEMI	Prof.	Université Badji Mokhtar-Annaba
Examinateurs :	Mahmoud BOUFAIDA	Prof	Université de Constantine 2
	Farid MOKHATI	Prof	Université Larbi Ben M'hidi Oum El Bouaghi
	Hakim BENDJENNA	MCA	Université Larbi Tébessi - Tébessa

REMERCIEMENTS

La réalisation de la thèse est constituée d'une succession d'étapes aux niveaux desquelles différentes personnes interviennent. Ce mémoire, sa rédaction et son évaluation, marquent la fin d'une étape importante qui doit permettre de formaliser de manière rigoureuse et détaillée les réalisations de la thèse et de situer précisément ces réalisations dans un domaine déjà bien fourni.

Je remercie avant tout mon Directeur de thèse, Monsieur Djamel MESLATI, pour m'avoir accompagné tout au long du parcours de la thèse avec une immense disponibilité, pour la confiance qu'il m'a accordé, pour son ouverture et son enthousiasme, pour son sérieux et pour sa convivialité.

Mes remerciements vont également au Professeur Vincent RODIN qui m'a orienté durant les stages que j'ai passé avec les membres de son équipe et qui a contribué beaucoup à la finalisation de cette thèse.

Je remercie également les membres du jury, Pr Salim GHANEMI, Pr Mahmoud BOUFAIDA, Pr Farid MOKHATI et Dr Hakim BENDJENNA d'avoir accepté d'évaluer mon travail de thèse.

Je remercie également tous les membres et responsables du laboratoire LISCO particulièrement ceux qui ont permis l'instauration d'un cadre de travail convenable.

Je remercie les nombreux étudiants qui ont contribué directement ou indirectement à l'achèvement de cette thèse.

Enfin, je tiens par cette simple phrase, chargée de sens pour moi, à remercier tous mes amis, ma grande famille, mes parents, ma femme, mes sœurs, mes tentes et mes amis pour leur présence et leur soutien qui dépasse largement la période de thèse qui vient de s'écouler.

DEDICACES

Je voudrais dédier ce travail d'abord et avant tout à la mémoire de ma grand mère BOUBA et de ma tente KHADOUDJA, que j'aurais tant aimé voir dans l'assistance et qui auraient tant voulu y être.

Je voudrais également dédier ce travail à ma tente NADJIA qui a travaillé et fait tant d'efforts pour que je puisse mener mes études à bien.

A toute ma famille.

ملخص

إن الاستعارة البيولوجية هي القياس بين العالم البيولوجي والعالم الاصطناعي الذي يمكننا من الاستفادة من المقاربات الاصطناعية من خلال تقليد بعض الجوانب البيولوجية وتجاهل الأخرى. الاستعارات البيولوجية، و المسماة أيضا المقربات المستوحاة من البيولوجيا، لا تعتمد فقط على الحقل البيولوجي الذي نتطرق له ولكن أيضا على فهمنا لهذا الحقل والنماذج والوسائل التي نستخدمها لاستخراج عناصر عملية ومفيدة لنمذجة بعض جوانب هذا المجال. هناك اليوم عدد هائل من الاستعارات التي تختلف بطبيعتها ومن المتوقع أن يزيد عددها وفقا لقدراتنا الإلهامية. و أمام هذا العدد المتزايد من الاستعارات يصبح من الضروري تحديد الملامح الرئيسية لكل واحدة من أجل تقييم أثرها العملي، مقارنتها ببعضها، تسهيل تعلمها و استخدامها، اذماجها، وما إلى ذلك. نتعامل في هذه الأطروحة مع مشاكل تصنيف النظم المستوحاة من البيولوجيا باستعمال نموذج POE كأساس للتصنيف مع الاعتماد على الهندسة المقادة بالنماذج لإيجاد مقاربة تسمح لنا بوصف النظم المستوحاة من البيولوجيا. خيارنا بالاعتماد على الهندسة المقادة بالنماذج له ما يبرره، فهي قادرة على وصف مختلف العمليات المعقدة و المنتجات الحرفية الناجمة عن تطوير النظم البرمجية وذلك بشكل موحد. و كنتيجة أولية، أثبت مقاربتنا الوصفية هذه فعاليتها في تميز مجموعة واسعة من الأنظمة المستوحاة من البيولوجيا.

كلمات مُرشدة : تصنيف النظم المستوحاة من البيولوجية، الهندسة المقادة بالنماذج ، تنشؤ الفرد، علم تطور السلالات، علم التخلق.

ABSTRACT

The biological metaphor is an analogy, between the biological world and the artificial world that enables us to benefit from artificial approaches by imitating some biological aspects while ignoring others. The biological metaphors, also called bio-inspired approaches, depend not only, on the biological field considered, but also on our understanding of that field and the paradigms and means we use to extract practical and useful elements to model some aspects of that field. Today, there is a huge number of metaphors which are different by their very nature and this number is expected to increase according to our inspiration capabilities. In front of this increasing number of metaphors it becomes necessary to define the main features of each one in order to evaluate their practical impact, to compare them, to ease their learning and use, to combine them, etc. In this thesis, we deal with taxonomy problems of bio-inspired systems by using the POE model as a basis for taxonomy and concepts of model-driven engineering to find an approach to describe bio-inspired systems. Our choice of the model-driven engineering paradigm is justified by its ability to describe uniformly various intricate processes and artefacts involved in the development of software systems. As a preliminary result, our description approach proved to be effective in characterizing a wide range of bio-inspired systems.

Keywords: Bio-inspired system taxonomy, Model-Driven Engineering, Ontogeny, Phylogeny, Epigeny.

RESUME

On entend par métaphore biologique une analogie qu'on cherche à déterminer entre le monde biologique et le monde artificiel, de façon à pouvoir proposer des approches qui imitent certains aspects du premier, tout en en ignorant d'autres. Fondamentalement, les métaphores ne cherchent pas à reproduire ce qui est biologique, mais plutôt à l'interpréter en fonction de ce qu'il est possible et raisonnable de réaliser. De ce fait, on peut déduire que les métaphores biologiques sont évolutives et dépendent de notre compréhension de la réalité et notre aptitude à en extraire des éléments pratiques et bénéfiques. Ce qui précède laisse sous-entendre, qu'à la diversité des organismes biologiques, viendra s'ajouter une diversité de métaphores pouvant engendrer une multitude de possibilités d'inspiration. Il devient alors indispensable de pouvoir caractériser les différentes approches bio-inspirées à des fins d'évaluation de leur apport concernant un aspect ou un autre. Dans cette thèse, nous traitons des problèmes de classification des systèmes bio-inspirés, nous partons du modèle POE comme base de taxonomie et nous puisons dans les concepts de l'ingénierie dirigée par les modèles pour trouver une approche de description des systèmes bio-inspirés qui utilise à la fois la qualité bio-inspiré du modèle POE et le savoir-faire du paradigme de l'ingénierie dirigée par les modèles (MDE ou Model Driven Engineering). Notre choix du paradigme MDE est justifié par sa capacité à décrire uniformément divers processus complexes et les artefacts impliqués dans le développement des systèmes logiciels. Comme résultat préliminaire, notre approche de description s'est avérée efficace pour caractériser un large éventail de systèmes bio-inspirés.

Mots clés: Taxonomie des systèmes bio-inspirés, Ingénierie Dirigée par les Modèles, Ontogenèse, Phylogénèse, Epigenèse.

TABLE DES MATIERES

LISTE DES FIGURES

LISTE DES PROGRAMMES

CHAPITRE 1 INTRODUCTION GENERALE

1. Contexte de recherche

L'homme a toujours vu la nature comme une source d'inspiration pour la construction de nouvelles machines. En effet, les avions sont une imitation des oiseaux, les sous-marins sont une imitation des poissons et les ordinateurs cherchent à imiter le cerveau de l'humain. L'inspiration de la nature a donnée ces fruits dans plusieurs domaines et plus spécialement en ingénierie informatique où Von Neumann a été l'un des pionniers en la matière.

De nos jours, les systèmes aussi bien matériels que logiciels faisant intervenir des processus inspirés du vivant sont de plus en plus nombreux [Tho 05]. On pourrait citer les travaux s'inspirant des réseaux de neurones et ceux de l'évolution naturelle de Darwin. Ces deux exemples sont le fruit de nombreuses années de recherche, mais ils ne représentent qu'une infime partie de l'effort de recherche consenti au domaine des systèmes bio-inspirés.

Mais pourquoi vouloir s'inspirer du vivant ? La réponse est simple, de nombreux constats montrent que les êtres vivants présentent des caractéristiques telles que l'autoréparation, l'adaptation, l'apprentissage, l'évolution et le développement. Ces concepts, bien utilisés et bien maîtrisés dans le domaine de l'ingénierie aideraient le développement et à la réalisation de machines résistantes aux pannes capable de s'adapter à leurs environnements, etc.

Motivée par les capacités des organismes biologiques, on constate une prolifération des systèmes bio-inspirés. La motivation comprend les trois facteurs suivants :

- La croyance en l'existence de métaphores biologiques qui peuvent apporter des solutions appropriées à divers problèmes.
- Le succès de certaines approches telles que les réseaux de neurones et les algorithmes génétiques.
- L'amélioration sur le plan technologique ainsi que celle des outils et méthodes de développement.

Ces facteurs ont conduit au développement de diverses approches qui s'inspirent des mécanismes naturels ayant fait leurs preuves. Ainsi divers domaines allant de l'intelligence artificielle à la vie artificielle en passant par l'ingénierie des logiciels et celle des architectures matérielles ont longuement puisé des idées variées (métaphores) de la nature.

Le travail présenté ici, s'inscrit dans le cadre de la recherche de caractéristiques communes entre les différentes approches bio-inspirées dans le but d'en élaborer une classification.

2. Problématique

On entend par métaphore biologique une analogie qu'on cherche à déterminer entre le monde biologique et le monde artificiel, de façon à pouvoir proposer des approches qui imitent certains aspects du premier, tout en en ignorant d'autres [Mes 06]. Fondamentalement, les métaphores ne cherchent pas à reproduire ce qui est biologique, mais plutôt à l'interpréter en fonction de ce qu'il est possible et raisonnable de réaliser. De ce fait, on peut déduire que les métaphores biologiques sont évolutives et dépendent de notre compréhension de la réalité et notre aptitude à en extraire des éléments pratiques et bénéfiques.

Ce qui précède laisse sous entendre, qu'à la diversité des organismes biologiques, viendra s'ajouter une diversité de métaphores pouvant engendrer une multitude de possibilités d'inspiration. Il devient alors indispensable de pouvoir caractériser les différentes approches bio-inspirées à des fins d'évaluation de leur apport concernant un aspect ou un autre (ex. adaptabilité dynamique, résistance aux pannes, ...).

Traditionnellement, les approches bio-inspirées sont classées selon les disciplines et les sous disciplines, telles que l'intelligence artificielle, l'intelligence artificielle distribuée, la vie artificielle, le calcul évolutionnaire, la cybernétique, etc. A l'intérieur de chaque discipline/sous discipline, les approches sont classées encore, en utilisant des critères qui reflètent les objectifs du système, son mode opératoire, sa constitution, etc. Par exemple, les systèmes multi-agents sont classés dans l'intelligence artificielle distribuée, en premier lieu, puis comme systèmes réactifs ou cognitifs en second lieu [Fer 95].

Ces classifications reflètent l'évolution de nos inspirations des organismes biologiques, plutôt que les propriétés intrinsèques des systèmes bio-inspirés. Nous citons les inconvénients de ces classifications dans ce qui suit:

- Manque de précision
- Caractère non naturel
- Frontières non justifiées

Ces insuffisances donnent naissance, directement ou indirectement, aux problèmes suivants :

- Les classifications ne prennent pas en considération la nature des approches qui sont inspirées de phénomènes naturels.
- Les classifications ne mettent pas l'accent sur l'environnement où les systèmes opèrent, alors que celui-ci joue un rôle très important.
- Une multitude d'hybridations d'approches qui sont peut être équivalentes.
- L'étude des systèmes bio-inspires devient une tâche ardue.
- La terminologie des systèmes bio-inspirés dans leur ensemble est souvent ambiguë.

3. Motivations

Dans le cadre de l'élaboration de cette thèse, nous nous sommes intéressé particulièrement au modèle POE [Sip 97], qui vise à décrire les systèmes matériels bio-inspirés selon trois axes (dits aussi processus POEtic): Phylogénétique, relatif à l'évolution, Ontogénétique, pour ce qui est de la croissance et de l'autoréparation, et Épigénétique, en ce qui concerne l'apprentissage. Différentes applications peuvent mettre en œuvre un, deux ou trois de ces axes, en fonction du problème à résoudre.

Malheureusement le modèle POE souffre des insuffisances suivantes:

- Certaines définitions utilisées peuvent être sujettes à discussion. C'est le cas, par exemple, lorsque les auteurs considèrent que l'environnement n'a aucun effet sur l'ontogenèse alors qu'en réalité il en a.
- Les processus POEtic peuvent être combinés, mais la classification POEtic ne peut discriminer les diverses formes de combinaison. Par exemple, la phylogenèse et l'épigenèse peuvent être combinées de diverses façons.
- La classification ne laisse pas apparaître la dichotomie individu/espèce comme un critère important alors que celle-ci est importante et permet de comprendre plus facilement certaines approches.

Face à ces insuffisances et le nombre grandissant des systèmes bio-inspirés, il devient important de trouver des critères pouvant caractériser chacune des approches et, éventuellement, conduire à une classification. Nous citons les objectifs suivants comme des motivations pour la recherche d'une classification.

- Caractériser et comparer les approches
- Rechercher les concepts communs et unificateurs
- Faciliter l'étude des systèmes bio-inspirés
- Trouver des directions d'inspiration prometteuses
- Unifier la terminologie des systèmes bio-inspirés
- Aider à la détermination des caractéristiques d'un système en cours de développement

4. Objectifs

Cette thèse regroupe plusieurs domaines : les nouvelles visions du génie logiciel tel que l'ingénierie dirigée par les modèles et les systèmes bio-inspirés. Nous avons cherché à concrétiser notre vision de la taxonomie on se basant sur trois objectifs qui ont guidé nos investigations tout au long de cette thèse :

- Cerner le domaine de la biologie en ayant une vision bio-inspirée.
- Etudier les systèmes bio-inspirés les plus en vue.
- Partant du modèle POE comme base de taxonomie et en puisant dans les concepts de l'ingénierie dirigée par les modèles, trouver une approche de description des systèmes bio-inspirés qui utilise à la fois la qualité bio-inspirée du modèle POE et l'aspect unificateur du paradigme MDA (Model Driven Architecture).

5. Description du contenu

Cette thèse est divisée en sept parties. Outre cette introduction et la conclusion, nous y présentons les systèmes bio-inspirés d'un point de vue biologique et artificiel, puis les grandes familles des systèmes bio-inspirés tel que les réseaux de neurones et les systèmes immunitaires artificiels.

Le formalisme de représentation par les modèles est ensuite introduit, avant d'entrer plus en détail dans l'objectif de cette thèse. Finalement, une nouvelle approche de caractérisation est présentée, de même que différentes représentations de systèmes. La description succincte des chapitres est donnée dans ce qui suit.

Le chapitre 2 met l'accent sur les différents aspects du vivant ayant inspiré les ingénieurs. Puis, il classifie ces aspects selon trois axes, la capacité à évoluer au sien de sa communauté, la capacité à évoluer en tant qu'un individu à travers le temps et la capacité à s'adapter à un environnement changeant.

Dans le chapitre 3, nous nous intéressons à quartes grandes familles de systèmes bio-inspirés. Nous commençons par décrire le système cellulaire, sa définition, ses caractéristiques, puis nous présenterons des systèmes évolutifs et leurs différentes représentations. Vient après le connexionnisme, sa définition et ses différents modèles enfin nous clôturons ce chapitre par les systèmes immunitaires artificiels.

Le chapitre 4 donne, dans un premier temps, le tour de l'approche « Model Driven Architecture » (MDA) et le concept de modèle, et nous décrit les standards sur lesquels l'architecture de MDA est basée ainsi que le processus de développement basé modèles. Dans un second temps, il met l'accent sur la transformation de modèles, son principe et les langages et outils développés pour sa réalisation.

Dans le chapitre 5, nous présentons notre approche et vision des systèmes bio-inspirés et nous proposons des critères qui peuvent caractériser ces dernières. Nous détaillons les concepts d'unité architecturale, ses principes, ses mécanismes et ses propriétés. Nous y montrons comment les unités architecturales peuvent décrire les processus d'un système bio-inspiré.

Le chapitre 6 résume le travail établit dans cette thèse nous y décrivons et modélisons plusieurs systèmes bio-inspirés. Dans un premiers temps, il propose une démonstration de notre démarche dans laquelle on modélise des systèmes bio-inspirés vue sous plusieurs dimensions. Dans un second temps, il montre l'intérêt de cette démarche pour la classification.

6. Références

[Fer 95] J.Ferber, Les Systèmes Multi-Agents, Vers Une Intelligence Collective, Intereditions, Paris, 1995.

[Mes 06] D.Meslati, Mage : Une Approche Ontogénétique De L'évolution Dans Les Systèmes Logiciels Critiques Et Embarqués. Thèse De Doctorat D'état, Université De Annaba, Février 2006.

[Sip 97] M. Sipper & Al, A Phylogenetic, Ontogenetic, And Epigenetic View Of Bio-Inspired Hardware Systems, Ieee Transactions On Evolutionary Computation, Vol. 1, No. 1, April 1997, Pp 83-97.

[Tho 05] Y.Thoma, Tissu Numérique Cellulaire A Routage Et Configuration Dynamiques, Thèse Epfl, No 3226, 2005

CHAPITRE 2 LES SYSTEMES BIO-INSPIRES : DU BIOLOGIQUE A L'ARTIFICIEL

1. Introduction

La nature n'est-elle pas parfaite ? On pourrait le croire en observant l'incroyable diversité des espèces vivantes et leurs comportements face au changement de l'environnement.

Chaque animal a des capacités spéciales qui correspondent, entre autres, à son environnement et à son type de nourriture. Les pates du bouc lui permettent de se déplacer facilement dans les montagnes, l'aigle a une vue perçante, une sorte de loupe étant intégrée au centre de son œil, et les fourmis sont passées maîtres dans la construction d'édifices.

Pourtant tout ceci ne s'est pas fait en quelques jours. L'évolution a pris le temps de générer l'éventail des espèces vivantes, en faisant disparaître les moins adaptées, tout au long d'un long processus d'évolution [Seg 98]. De plus, la faculté d'adaptation permettant au vivant de s'adapter à un environnement changeant est une des caractéristiques essentielles à la survie, aussi bien d'un individu que d'une espèce.

Une modification du comportement est souvent nécessaire afin de faire face aux aléas de la vie, une non adaptation étant alors synonyme de mort certaine. Au niveau d'une espèce, l'adaptation se fait au fil des générations, au prix de nombreux essais.

Par sélection naturelle, les plus aptes à se reproduire ont la chance de transmettre leurs gènes à leur descendance, et ainsi l'espèce reste adaptée à son environnement.

Ce qui précède laisse sous entendre, que nous ne pouvant pas affirmer que la nature est parfaite, si on considère le nombre de morts nécessaires à l'obtention d'une solution adéquate. Mais on peut répondre positivement si on considère la vie ou l'espèce dans son ensemble. En effet, depuis son apparition, la vie n'a jamais disparu. Malgré de multiples extinctions, des espèces ont toujours trouvé le moyen de s'adapter pour survivre.

Par analogie, les machines créées par l'homme semblent parfaites car créées de manière exacte pour un problème particulier. Toutefois, l'adaptation n'y est pas, et une machine n'est

pas capable d'effectuer plus au moins une autre tâche que celle pour laquelle elle a été programmée. N'est-ce donc pas l'adaptation qui fait le succès du vivant ? Et n'est-ce donc pas cela qui manque aux systèmes logiciels ? De plus, outre ces deux niveaux d'adaptation, à l'échelle de l'individu pour l'apprentissage, et de l'espèce pour l'évolution, l'autoréparation est un autre grand atout du vivant. Alors qu'en général, une petite erreur met à mal l'entièreté d'un système artificiel, les êtres vivants sont capables de les gérer, en cicatrisant ou en adaptant l'utilisation d'autres parties du corps [Tho 05].

Au cours de ce chapitre, nous allons présenter différents aspects du vivant ayant inspiré les ingénieurs. Nous verrons qu'il s'agit toujours de systèmes adaptatifs, autant dans le sens d'évolution et d'apprentissage que dans le sens d'autoréparation.

2. Un être vivant c'est quoi ?

Un être vivant est un organisme animé qui présente une forme bien définie, qui respire, s'alimente, se développe et se reproduit. Nous entendons par système biologique, un organisme isolé aussi bien qu'un ensemble composé de plusieurs organismes vivants, semblables ou différents.

L'étude des systèmes biologiques est une tâche difficile, car un être vivant, pris isolément, présente déjà une grande complexité. En effet, dans sa forme la plus élaborée, l'organisme vivant est composé d'organes liés et ordonnés de façon à assurer sa survie et sa reproduction. Plusieurs êtres vivants peuvent s'assembler dans un système où règnent un ordre et des interactions complexes. Dans la nature, tout système est sujet à l'évolution.

Il existe plusieurs définitions de l'évolution biologique. Bien qu'elle désigne un nombre important de concepts, les définitions existantes vont dans le même sens : c'est-à-dire le changement à long terme des espèces.

Dans l'encyclopédie Universalis, l'évolution est définie par :
Le processus par lequel, au cours des âges, se succèdent et s'engendrent, tout en variant, les espèces végétales et animales.

Dans [Rid 96], on trouve la définition suivante :
« Evolution signifie changement dans la structure et le comportement des organismes, au fil des générations. Les aspects divers des organismes actuels, à tous les niveaux depuis la séquence de leur ADN jusqu'à leurs structures macroscopiques ou jusqu'à leur comportements sociaux, sont issus de la modification de ceux de leurs ancêtres ».

Ces définitions excluent le changement qui touche un seul organisme appelé développement. Dans [Rid 96], Ridley enchaîne à sa définition :
« Il ne faudrait cependant pas inclure dans la notion d'évolution tout changement, quel qu'il soit, d'un être vivant. Au sens strict du mot, on ne considère pas comme évolutives les modifications que subit un organisme au cours de son développement ».

L'évolution se distingue du développement par le fait que ce dernier concerne un organisme alors que l'évolution concerne les générations des espèces sur une échelle temporelle plus importante.

Dès sa naissance, et même avant, un organisme se caractérise par un ensemble de propriétés que l'être humain peut percevoir par ses organes sensoriels ou par ses capacités d'analyse et de déduction. Cet ensemble de propriétés, appelé *phénotype*, renferme aussi bien des propriétés physiques que comportementales.

Par exemple, une plante peut être caractérisée par sa taille, sa morphologie, sa saison de floraison, son mécanisme de reproduction. De même, un animal peut inclure dans son phénotype des propriétés telles que la couleur, le poids, la possession des ailes, la stratégie de défense contre les prédateurs, etc.
Durant sa vie, un organisme change continuellement son phénotype et passe d'une phase à une autre. Le développement désigne ce processus de changement continuel et couvre deux fonctions consistant à engendrer la diversité et à assurer la continuité de la vie d'une génération à la suivante.
Ci-après, nous introduisons la notion de génotype, nous montrons son lien avec le phénotype puis nous donnons de plus amples détails sur le processus de développement et d'évolution.

3. Constitution d'un organisme : génotype et phénotype

Les propriétés qui forment un phénotype d'une entité peuvent être réparties en deux sous ensembles.

Le premier, appelé péristase, est constitué des propriétés que l'entité acquiert après sa naissance par interaction avec le milieu où elle vit. Ce sont principalement des propriétés comportementales. Le deuxième sous-ensemble, appelé génotype est formé des propriétés directement transmissibles des ascendants aux descendants. Le génotype d'une entité comprend toutes les propriétés développées indépendamment du milieu d'existence de celle-ci. Ces propriétés sont parfois qualifiées de propriétés innées.

Dans la réalité, il est difficile de dissocier la péristase et le génotype, car l'entité développe toujours son génotype en interaction avec un milieu donné. De même, la péristase se développe dans les limites fixées par le génotype [Mes 06].

Le génotype est développé par l'entité à partir d'un ensemble de gènes, appelé génome, et est supporté par le génome et les propriétés physiques des organes qu'il engendre. La péristase, étant composée de propriétés comportementales acquises en interaction avec le milieu, suite à des multitudes d'expériences, est supportée par des organes ayant une capacité de mémorisation d'apprentissage et/ou de raisonnement logique. Par exemple, pour un être humain, la manière de produire une protéine (codée dans le génome) ou l'intervalle angulaire maximal de rotation du poignet (propriété physique d'organe) font partie du génotype alors que les manières de parler, de marcher ou de réagir à une offense sont supportées principalement par le système nerveux et font partie de la péristase [Mes 06].

3.1. Le processus de développement

Les organismes multicellulaires ne se créent pas complètement formés. En effet, ils sont le fruit d'un processus qui opère des changements progressifs sur un intervalle de temps relativement long. Ce processus est qualifié de processus de développement [Gil 03].

A l'origine de son existence, un organisme se compose d'une cellule unique sur laquelle s'opèrent des divisions répétées, appelées *mitoses*, qui dupliquent la cellule originale en plusieurs. Les mitoses successives conduisent à la création de toutes les cellules d'un organisme multicellulaire. Durant la phase allant de la première cellule à la naissance, l'organisme est appelé *embryon*.

La phase embryonnaire représente la phase la plus importante et la plus spectaculaire d'un organisme car elle fait passer celui-ci d'une phase de quasi-inexistence à une phase où l'organisme exhibe un phénotype complexe.

Dans la plupart des organismes, il n'y a pas d'arrêt de développement après la naissance ni même à l'âge adulte [Gil 03]. Le développement qui suit la naissance est, dans la plupart des cas, relativement moins spectaculaire et se base sur les mêmes principes que le développement durant la phase embryonnaire.

Cependant, il convient de souligner que chez certains organismes vivants tels que l'homme, en plus du développement morphologique, structurel, physiologique etc., un développement comportemental et psychologique important prend le relais.

Ce qui précède explique pourquoi la biologie du développement s'intéresse principalement à l'embryologie [Gil 03].

Le développement du génotype vise deux objectifs principaux :

- Engendrer une diversité de cellule et assurer un certain ordre de celle-ci pour chaque espèce et/ou génération.
- Garantir la continuité de la vie par reproduction permanente de nouveaux organismes au sein de chaque espèce.

Dans le règne animal, la phase embryonnaire est très variable. Cependant, il est possible de ressortir quatre processus importants constituant l'embryogenèse : segmentation, différenciation, morphogenèse et organogenèse.

- Après fécondation, une suite de mitoses extrêmement rapides, appelée segmentation, divise la cellule initiale en une multitude de cellules.
- La segmentation ralentit et une différentiation cellulaire commence à s'opérer. La différentiation à pour effet de spécialiser les cellules en fibres musculaires, cellules de peau, neurones, lymphocytes, … .
- Au fur et à mesure que les cellules se spécialisent, elles effectuent des déplacements importants qui modifient leurs positions respectives. On aboutit ainsi à la formation de trois régions cellulaires appelées feuillets germinatifs :

Le premier feuillet réunit les cellules qui donneront naissance à l'épiderme et au système nerveux. Le deuxième feuillet donnera naissance à la paroi du tube digestif, le pancréas, le foie ; etc. Le troisième feuillet regroupe les cellules qui donneront naissance au cœur, aux reins, aux gonades, au sang et aux tissus conjonctifs formant les os, les muscles et les tendons.

- La formation des différents organes est dite organogenèse. Chaque organe se compose de cellules qui proviennent des trois feuillets. Les cellules des trois feuillets, interagissent les unes sur les autres et opèrent de longues migrations vers leurs lieus définitifs. Ce réarrangement conduit à l'apparition de formes et de structures, c'est la morphogenèse [Mes 06].

3.2. Le processus de l'évolution génétique

La théorie de l'évolution des espèces ou simplement évolutionnisme, remonte principalement aux travaux de Charles R. Darwin [Rid 96]. L'évolutionnisme considère que la création de nouvelles espèces animales ou végétales se fait par filiation directe et continue.

En comparant la faune des îles Galápagos à celle de l'Amérique du sud, Darwin déduisit que les espèces ont la faculté de devenir autres. Sa théorie se base sur la variation et la sélection naturelles.

Comme tout ce qui est porté par le génome, l'évolution du génotype est transmissible par héritage. Cependant, il convient de distinguer entre changement du génotype (i.e. développement) et changement du génome (appelée *mutation*). La première est une mise en œuvre des mécanismes et processus d'évolution portés par certains gènes. La mutation, quant à elle, implique le changement du génome lui-même. Elle intervient le plus souvent lors de la transmission héréditaire. Dans ce cas, elle n'a pas d'effet sur le génotype de l'ascendant mais a pour conséquence le fait que le génome de son descendant sera différent. Le phénotype du descendant sera, par conséquent, différent. Notons aussi, qu'il existe certaines mutations qui affectent les gènes actifs ou en attente de l'être et, par conséquent, altèrent le processus du développement de l'entité elle-même [Lin 91, Lew 99].

Dans la réalité, le mécanisme le plus évolué de reproduction d'une entité, consiste en la création de la première cellule à partir de deux cellules dites de reproduction (appelées aussi *gamètes*) provenant, chacune, d'un ascendant distinct. Après fécondation, les chromosomes semblables ou homologues (i.e. brins de l'ADN pris chacun dans un gamète) s'associent en paires et forment le génome de la première cellule. Dans chaque paire, les gènes se correspondent.

Par exemple le gène responsable de la couleur des yeux dans un chromosome correspondra au même gène dans l'autre chromosome. Le gène qui imposera sa propriété à l'entité sera dit gène dominant et l'autre gène récessif. Le gamète est dit pur car il n'existe pas deux gènes semblables parmi tous ses chromosomes (voir figure1).

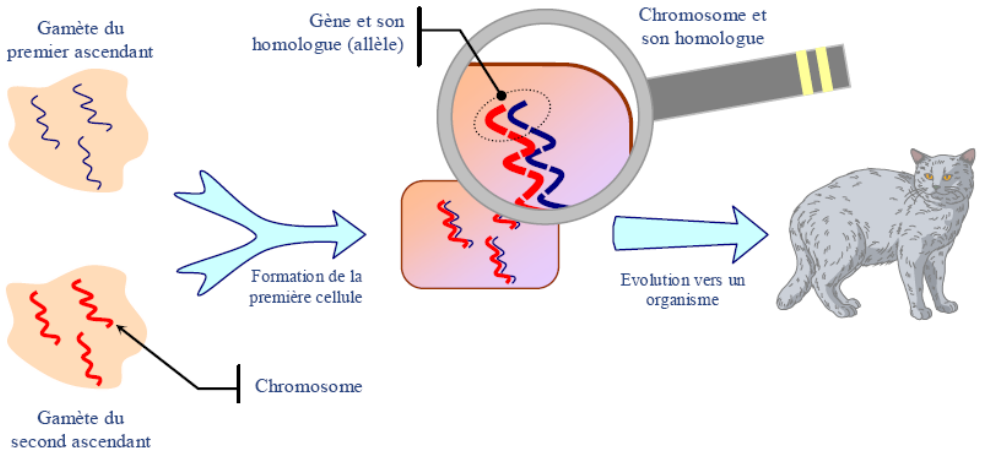


Figure 2.1. Association des chromosomes et des gènes [Mes 06]

Les espèces renferment souvent des variétés qui ont des phénotypes différents. La variété dans le génotype se base principalement sur le concept d'association en paire des chromosomes semblables. En effet, les gènes relatifs à une seule propriété existent en plusieurs variantes (dites *allèles*) qui ont des degrés de dominance différents. Au hasard des fécondations, la première cellule aura un génome où chacun des gènes dominants appartient soit au gamète du premier ascendant, soit à celui du second. Pourvu que les deux gamètes ne soient pas identiques, cela donnera une entité différente de ses ascendants.

De ce qui précède, il apparaît que l'unité de transmission héréditaire du patrimoine génétique est le chromosome. Mais, en réalité, la transmission héréditaire est plus complexe car un gène peut n'avoir d'effet que s'il est associé à un ou plusieurs autres gènes dans son chromosome (i.e. *linkages* ou *relations de dépendance* entre gènes). Cette association peut être bouleversée durant la phase de formation des gamètes par des phénomènes de mutations. Ces dernières constituent une autre source de variété et se répartissent essentiellement en deux catégories : La *mutation génique* qui change la constitution physico-chimique des gènes et la *mutation chromosomique*, où la structure du chromosome est changée par ajout, suppression ou déplacement de gènes.

L'entrecroisement des chromosomes est une forme courante de mutation qui intervient durant la formation des gamètes. Elle consiste en un échange d'un ensemble de gènes entre les chromosomes d'une paire pendant leur séparation.
Sans pression de l'environnement, pratiquement tous les organismes d'une même espèce pourraient, à la fois, muter sans contraintes (avec apparition des caractères phénotypiques correspondants) et survivre. Mais, le climat, la végétation, la nourriture ou même la position géographique avantagent un certain nombre d'organismes dont le phénotype se révèle mieux adapté. Par exemple, les animaux de montagne survivent mieux s'ils ont des poils longs, les fleurs si elles attirent les insectes (pour favoriser la pollinisation), etc. Ainsi, la sélection naturelle agit sur les organismes par le biais de leur phénotype.

3.3. Le processus de l'évolution adaptative

L'adaptation d'une espèce correspond à son aptitude à développer de nouvelles propriétés pour survivre dans un environnement inhabituel. Chaque espèce possède un certain nombre de caractères dits adaptatifs, qui maintiennent son adéquation avec son milieu, autorisant sa survie et sa reproduction. Les caractères adaptatifs correspondent à l'utilisation optimale des conditions et des ressources de l'environnement, la défense adéquate contre les prédateurs et la protection contre toute autre condition défavorable à la survie de l'espèce.

Le résultat direct de cette constatation est que les organismes les moins viables se reproduiront beaucoup moins, et que, de ce fait, leur génotype ne sera pas transmis à la descendance. Ainsi, il se produit une sélection du génotype par l'environnement. Les gènes sont ainsi représentés au cours des générations successives en proportion de la valeur sélective de leurs effets phénotypiques (c'est-à-dire des avantages ou des désavantages qu'ils procurent). Il arrive également que des gènes n'aient aucune influence sur la sélection des organismes d'une population donnée, mais peuvent néanmoins favoriser son adaptation si, soudain, son milieu de vie change (on parle alors de *préadaptation*). Une mutation qui ne confère à un organisme ni avantage ni désavantage est dite *mutation neutre*. Ainsi, d'une génération à l'autre, l'espèce évolue [Mes 06].

Notons pour terminer cette section, qu'un organisme ne se crée jamais du néant. De génération en génération, un patrimoine génétique et comportemental (voire culturel) est transmis avec relativement peu de changements. L'organisme détient une grande partie des propriétés de ses ascendants.

4. La génétique dans le développement et l'évolution

La génétique est une science qui traite des mécanismes de reproduction à tous les niveaux où elle se présente. Qu'elle soit au niveau de la molécule, de la cellule, de tout l'organisme ou de toute une population, l'étude de la reproduction biologique passe par l'étude de la variation de caractères entre organismes.

La génétique distingue les variations acquises des variations génétiques, et, pour ces dernières, il y a une distinction entre les mutations et les recombinaisons. Les variations acquises sont très perceptibles chez les organismes complexes et ne sont pas transmises à la descendance, par exemple, la langue parlée, le comportement social, etc.

Il est actuellement bien admis que toute entité biologique évolue physiquement de sa naissance à sa mort selon un plan minutieusement établi. Ce plan réside dans le noyau de chaque cellule de l'entité sous forme de chromosomes. Chaque chromosome porte un nombre très important de gènes, dont chacun est responsable de l'apparition d'une ou de plusieurs propriétés structurelles et fonctionnelles (par exemple : nombre de doigts, couleur des yeux, plan de fabrication d'une protéine ...). Chimiquement, le chromosome est une très longue molécule appelée ADN (Acide Désoxyribonucléique) et le gène est une portion de cette molécule. L'ensemble des gènes de tous les chromosomes d'une cellule est appelé génome. Pour une espèce donnée, les chromosomes s'associent en paire et leur nombre est immuable. Chaque paire de chromosomes (dits chromosomes homologues ou semblables) est

responsable de l'apparition d'un sous-ensemble disjoint de propriétés du génotype. Par exemple, pour l'être humain, il existe une paire de chromosomes responsable de toutes les propriétés liées à son sexe.

Les biologistes qualifient le génome selon les espèces de programme *génétique fermé* et programme *génétique ouvert*. Un programme fermé n'accepte pas de sensibles changements durant sa traduction en phénotype. De ce fait, une expérience acquise ne peut s'y inscrire (par exemple certains comportements innés chez des animaux simples sont strictement régis par leurs gènes). Chez les organismes complexes, un programme génétique moins contraignant leur donne plus de liberté dans le comportement (par exemple un enfant est génétiquement programmé pour apprendre à parler ; cependant, si la langue qu'il apprendra dépendra du milieu dans lequel il vit, les sons qu'il est capable de produire sont prédéterminés et limités) [Mes 06].

Un comportement complexe n'est jamais directement contrôlé par le génotype, mais c'est un programme comportemental contrôlé par le système nerveux. Ce programme est supporté par les *synapses* (connexions entre les neurones). Les synapses ne sont pas régies par le génome (seul le nombre et la disposition des neurones le sont). De ce fait, un nouveau comportement acquis s'inscrira dans le système nerveux, par formation de nouvelles synapses, et non dans le génome. Notons que les propriétés supportées par les synapses du système nerveux sont plus facilement altérables et évolutives que celles du génotype.

Le développement d'une entité suppose l'existence de plusieurs processus de changements. Les uns, régis par les gènes, échappent au contrôle de l'entité et les autres, tels que l'apprentissage, permettent à l'entité d'acquérir des comportements complexes. Le développement régi par les gènes, mène l'entité d'une simple cellule à un organisme doté d'organes complexes [Mes 06]. Tout au long de sa vie, les interactions avec le milieu influent sur le développement de l'entité (voir figure 2.2).

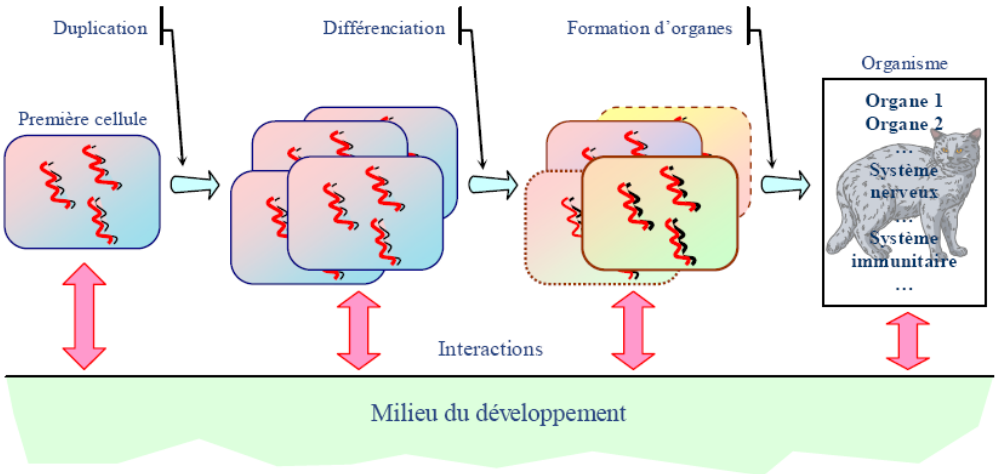


Figure 2.2. Développement d'un organisme et interaction avec le milieu [Mes06]

Tout changement dans le génotype est effectué selon des mécanismes et des processus qui sont codés dans les gènes. Cela sous-entend que ce changement est prévu à l'avance et que

l'instant (relativement à la naissance de l'entité) du déclenchement des mécanismes et processus est aussi fixé dans le génome [Gil 03].

Par exemple, l'être humain passe par différentes phases avant et après sa naissance. Dans chaque phase, il y a développement d'organes et enrichissement du génotype par de nouvelles propriétés. Au début de chaque changement, les mécanismes et processus correspondants sont déclenchés et les gènes qui les supportent passent de l'état passif à l'état actif. Une fois le développement achevé, les gènes concernés redeviennent passifs.

Selon la nature de l'évolution, les gènes sont soit toujours actifs, soit passent d'un état à un autre. Certains gènes ne sont activés que durant un seul intervalle de temps pour toute la vie d'un organisme donné [Lew 99].

5. Systèmes apprenants biologiques

Les biologistes identifient chez le règne animal trois systèmes qui contribuent aux développements et à l'évolution et qui ne sont pas directement contrôlés par les gènes : il s'agit du système nerveux, du système immunitaire et du système endocrinien.

5.1. Le système nerveux

C'est l'ensemble des structures anatomiques caractéristiques du règne animal impliquées dans la réception et dans la transmission des informations provenant de l'environnement, dans la commande des muscles et autres organes effecteurs, et dans la coordination des diverses fonctions vitales [Mes 06].

Sur le plan structurel, le système nerveux se compose de deux parties : le système nerveux central et le système nerveux périphérique. Alors que la première partie regroupe les centres nerveux, la deuxième, comprend les différents nerfs qui partent du système nerveux central pour atteindre chaque recoin de l'organisme.

Sur le plan fonctionnel, le système nerveux se partage en deux sous-systèmes : le système nerveux *somatique* et le système nerveux *autonome* (dit aussi végétatif). Le premier, qui inclut le système volontaire, reçoit et traite les informations en provenance des organes sensoriels et contrôle les postures et mouvements de l'organisme. Le second fonctionne en toute indépendance de la volonté de l'organisme et régule la respiration, les contractions cardiaques, la digestion, etc.
Les éléments de base du système nerveux sont les neurones. Leur rôle est de véhiculer des informations d'une zone de l'organisme à une autre sous forme d'impulsions électriques. Les neurones qui véhiculent les impulsions des organes vers le cerveau sont appelés les neurones *sensitifs*, ceux véhiculant les impulsions dans le sens inverse sont dits neurones *effecteurs*.

Dans le système nerveux périphérique, les neurones sont organisés en fibres, les nerfs ou terminaisons nerveuses. Ces dernières sont sensibles, selon leur type, à des stimuli particuliers tels que la douleur, la chaleur, la sensation visuelle, le toucher, etc. Une fois excitées, elles transmettent l'information au système nerveux central qui déclenche une réponse au niveau des organes, en leur envoyant des impulsions à travers les neurones effecteurs.

Le système nerveux forme un réseau intégré de neurones fortement connectés. Ces connexions ont lieu par le biais d'éléments spécifiques au neurone dits *axone* et *dendrites*. La transmission, proprement dite d'un signal d'un neurone à l'autre se fait par des *synapses*, qui peuvent être soit électriques, soit chimiques. Dans ce dernier cas, le message est transmis par l'intermédiaire d'un composé chimique, dit *neuromédiateur*, qui peut être soit excitateur, soit inhibiteur (i.e. favorise/défavorise l'apparition d'une action).

Le comportement du neurone résulte de l'ensemble des signaux reçus, inhibiteurs et activateurs. Si les signaux activateurs sont majoritaires, le neurone produira un signal nerveux. Si, au contraire, les signaux inhibiteurs sont majoritaires, il bloquera la transmission. C'est ce mode de fonctionnement qui est à la base du traitement de l'information par l'organisme et qui dote ce dernier de propriétés phénotypiques diverses.

5.2. Le système immunitaire

Le système immunitaire se matérialise par l'ensemble des cellules, des tissus et des organes chargés de défendre l'organisme contre certains agents pathogènes, des substances étrangères ou anormales, notamment celles qui font partie des micro-organismes (bactéries, virus, etc.) et celles qui se trouvent à la surface des cellules cancéreuses. Ces substances susceptibles d'induire une réponse immunitaire sont appelées *antigènes* [Mes 06].

Le système immunitaire joue un rôle fondamental et extrêmement complexe, il permet de réagir de façon appropriée à l'infinité d'antigènes différents qui pénètrent ou envahissent l'organisme. De nos jours, les mécanismes physiologiques complexes mis en œuvre dans le système immunitaire sont mieux compris mais pas encore complètement élucidés.

Sur le plan structurel, le système immunitaire comprend des cellules, des substances et des organes. Il existe trois grandes catégories de cellules immunitaires (dites globules blancs) : les *neutrophiles*, les *lymphocytes* et les *macrophages*.

- Neutrophiles : Ce sont des cellules qui détruisent, par des enzymes, les antigènes étrangers et, en particulier, les bactéries. Elles présentent un mécanisme d'action particulier. En effet, sur le lieu d'une infection, les neutrophiles relâchent dans le milieu extracellulaire des fibres protéiques, mêlées à de l'ADN, qui forment un filet fonctionnant comme un piège à bactéries. Prises dans ce piège, les bactéries et leurs toxines sont détruites par certaines protéines du filet, à activité antibactérienne.
- Lymphocytes : Ce sont les cellules les plus importantes du système immunitaire. Elles ont la capacité de se souvenir, chimiquement, d'une exposition antérieure à un antigène. En effet, au cours d'une agression antigénique, certains des lymphocytes se transforment en cellules mémoire, de longue durée, ce qui leur permet, lors d'une nouvelle exposition au même antigène, d'avoir une réponse plus rapide et plus efficace.
- Macrophages : Elles ont pour rôle de pénétrer dans le tissu, de détruire les particules étrangères qui y sont présentes, de conserver certains des composants chimiques des antigènes et de les faire parvenir à la surface de leur membrane. Cette signature antigénique est, par la suite, mémorisée par les lymphocytes.

> Les substances chimiques libres faisant partie du système immunitaire sont de nature peptidique ou protéique, et se trouvent dans le sang ou dans les liquides extracellulaires des tissus. Ces substances exercent soit des fonctions d'anticorps (chacune se combine spécifiquement avec un antigène afin de permettre son élimination de l'organisme), soit des fonctions de régulation de la réponse immunitaire qui confèrent au système immunitaire une activité d'intensité appropriée quand cela est nécessaire.

Les organes et les tissus dont le rôle est spécifiquement immunitaire sont qualifiés de *lymphoïdes*. Ils comprennent la moelle osseuse, le thymus, la rate, les ganglions lymphatiques et différentes structures annexées aux muqueuses. La moelle osseuse et le thymus sont des organes lymphoïdes centraux, car ils sont responsables de la formation initiale des lymphocytes, pendant la vie embryonnaire. Les autres organes lymphoïdes sont dits périphériques.

Sur le plan fonctionnel, une fois qu'un lymphocyte a terminé sa maturation dans la moelle ou le thymus, il se rend par voie sanguine dans un lymphoïde périphérique, par exemple dans un ganglion du système lymphatique où il reste en attente. Dès que le contact avec un antigène est établi, éventuellement par l'intermédiaire d'une cellule présentatrice telle que le macrophage, le lymphocyte est activé, se multiplie et entre en fonction.

Les différentes composantes du système immunitaire interagissent pour produire une réponse immunitaire efficace. En effet, lorsqu'un antigène, provenant de l'extérieur, traverse la première ligne de défense de l'organisme (la peau ou la muqueuse), il déclenche d'abord les phénomènes de l'immunité non spécifique, qui s'exercent de la même façon indépendamment de l'antigène et qui met en action les neutrophiles et les macrophages. Si cette première réponse immunitaire n'est pas suffisante, les macrophages interagissent avec les lymphocytes afin d'activer une immunité spécifique plus efficace. Dans certains cas, le macrophage peut également emmener la signature de l'antigène détruit vers le ganglion lymphatique le plus proche où des anticorps spécifiques sont alors produits et des lymphocytes spécifiques activés. Une fois que le système immunitaire prend le dessus sur l'envahisseur, les mécanismes suppresseurs d'autorégulation entrent en action pour arrêter la réponse immunitaire.

Une des caractéristiques importantes du système immunitaire réside dans sa capacité de mémorisation qui permet à l'organisme de développer, au fil des ans, une résistance aux agents pathogènes.

5.3. Le système endocrinien

Le système endocrinien est l'ensemble de glandes sécrétrices d'hormones et de tissus qui régulent les différentes fonctions de l'organisme. Il joue un rôle critique dans la coordination, la stimulation et l'inhibition de diverses activités physiologiques de l'organisme de sorte qu'elles aient lieu au bon moment et selon une intensité convenable [Mes 06].

Le système nerveux joue aussi un rôle important dans les processus de l'organisme, cependant, les actions du système nerveux s'opèrent plus rapidement et ne durent pas longtemps comme c'est le cas des contrôles que le système endocrinien opère. L'interaction des deux systèmes permet de coordonner leurs activités. Le temps des communications qui ont lieu dans le système endocrinien varie entre secondes et jours et l'effet de certaines actions endocriniennes peut durer de quelques minutes à quelques mois [Nel 05].

Comme objectifs des actions du système endocrinien, on cite :

- Coordonner plusieurs aspects liés à la croissance, le développement et la reproduction
- Aider le corps à réagir aux urgences : infection, traumatisme, stress émotionnel, déshydratation, températures extrêmes, …
- Réguler le métabolisme (ensemble des réactions biochimiques)
- Réguler les contractions cardiaques
- Réguler quelques activités du système immunitaire
- Réguler la composition chimique et le volume du liquide extracellulaire.

Sur le plan structurel, le système endocrinien se compose de glandes endocrines et de tissus qui synthétisent et libèrent, dans l'organisme, des hormones. Ces dernières sont des messagers chimiques, véhiculés par le sang, qui, une fois arrivés aux cellules des organes cibles, y provoquent une réaction. Parmi les glandes endocrines, nous citons : la glande pinéale, la glande pituitaire et l'hypothalamus au niveau du cerveau, la thyroïde et la parathyroïde au niveau du larynx, le thymus au niveau des poumons, les glandes surrénales, etc.

Sur le plan fonctionnel, le système endocrinien est très complexe. A titre indicatif, notons que :

- Une même glande endocrine peut sécréter plusieurs hormones.
- Une hormone donnée peut avoir des effets différents sur différentes cellules cibles.
- Un processus physiologique peut être contrôlé par plusieurs hormones.
- Les glandes endocrines agissent les unes sur les autres afin d'ajuster mutuellement leurs fonctions.

Le système endocrinien fait intervenir plusieurs dizaines d'hormones différentes dont la sécrétion peut être déclenchée par des stimulations produites par des influx nerveux, des variations homéostatiques (concentration en ions, en nutriments, ...), des variations environnementales (stress) ou même d'autres hormones.

Malgré que les hormones soient véhiculées par voie sanguine, i.e. peuvent atteindre tous les organes irrigués par le sang, leurs actions sont sélectives. En effet, une hormone n'agit pas globalement mais spécifiquement dans le corps. Seules les cellules cibles de l'hormone y sont sensibles car elles seules possèdent des récepteurs spécifiques de l'hormone. En d'autres termes, c'est la présence du récepteur hormonal qui confère à la cellule cible sa sensibilité vis à vis de l'hormone.

Les hormones peuvent avoir deux modes d'actions :

- Action via les récepteurs transmembranaires. Au niveau des membranes des cellules, il existe des récepteurs sous forme de protéines qui participent au passage de l'information véhiculée par l'hormone vers le cytoplasme, sans que celle-ci ne pénètre à l'intérieur de la cellule. L'hormone se fixe sur la face externe du récepteur puis une reconnaissance s'opère. Cette reconnaissance est un contact physique basé sur le principe de complémentarité de la forme moléculaire. Suite à l'activation du récepteur, une réponse de la cellule aura lieu.
- Action via des récepteurs intracellulaires. Certaines hormones (dites hormones stéroïdes et hormones thyroïdiennes) n'utilisent pas les récepteurs transmembranaires mais sont capables de traverser la membrane plasmique des cellules cibles. Une fois entrées dans la cellule, les hormones poursuivent leur chemin puis franchissent la membrane du noyau cellulaire et enfin interagissent avec le génome via des récepteurs spécifiques.

Les hormones peuvent avoir un effet important sur la physiologie de la cellule. Cette dernière est capable, à son tour, de moduler positivement ou négativement sa sensibilité aux hormones. En effet, l'augmentation du nombre de récepteurs, par la cellule, accroît la capacité de réponse de la cellule cible alors qu'une réduction du nombre de récepteurs la diminue. Une exposition très faible des cellules aux hormones accroît le nombre de récepteurs et vice versa. Certaines hormones ont même pour rôle de contrôler la synthèse des récepteurs d'autres hormones par la cellule.

Dans la section suivante nous allons faire une synthèse de ce qui a été présentés pour en conclure avec un système de classifications des processus biologiques.

6. La vie selon les trois axes

Comme présenté plus haut, la constitution d'un être vivant est définie par trois aspects, sa capacité à évoluer au sien de sa communauté en s'accouplant pour générer sa descendance, sa capacité à évoluer en temps qu'individu à travers le temps et à développer ses organes jusqu'à maturité et le dernier aspect qui est beaucoup plus psychologique, puisqu'il concerne son adaptation à l'environnement changeant, correspond à sa capacité à apprendre des concepts et des notions qui sont facilement façonnables.

6.1. Ontogenèse

L'ontogenèse traite du développement de l'être à partir d'un œuf fécondé, le zygote. Lors d'une reproduction sexuée, cette cellule contient la moitié des gènes du père et la moitié de ceux de la mère, qui ensemble décrivent la manière dont l'organisme doit se construire. La création d'un organisme entier à partir de l'œuf est une œuvre extraordinairement complexe et sa compréhension complète nécessitera encore de nombreuses années de travail, si toutefois on parviendra à en percer tous les secrets un jour.

L'information nécessaire au développement est stockée dans les chromosomes, qui sont composés d'ADN (Acide Désoxyribonucléique, dont la structure à été découverte en 1953 par Crick et Watson [Wat 53]), et forment le patrimoine génétique d'un être vivant.

Le codage de base correspond à la manière dont sont arrangés les quatre acides nucléiques constituants de l'ADN : la Guanine, l'Adénine, la Cytosine, et la Thymine.

Il est intéressant de constater que le génome humain contient environ 35'000 gènes alors que le corps humain adulte est composé de cent mille milliards de cellules (10^{14}). Il est donc bien clair qu'un gène ne décrit pas une cellule, mais un mécanisme de développement est nécessaire à la traduction de l'information des gènes en un corps entier. Il est également important de constater que le patrimoine génétique est présent dans toutes les cellules de l'organisme (sauf dans les cellules germinales qui n'en contiennent que la moitié). Ceci implique une duplication de l'ADN lors d'une mitose (duplication de la cellule), et permet au corps de disposer de l'autoréparation. En effet, lorsque des cellules meurent ou que le corps est blessé, le corps est capable de se réparer en cicatrisant. Les organismes vivants sont donc les meilleurs exemples de systèmes auto-réparateurs, laissant les machines loin derrière. Nous présentons succinctement les principes directeurs connus actuellement pour le développement de l'embryon [Tho 05].

6.1.1. Position

La fonctionnalité d'une cellule dépend entre autre de sa position dans l'organisme, qui est définie par les interactions qu'elle entretient avec son environnement. Une partie de son génome peut alors être exprimée en fonction de ces interactions. Deux systèmes influençant le comportement de la cellule semblent probables, à savoir la diffusion d'un gradient chimique et la communication intercellulaire [Tho 05].

6.1.2. Gradient chimique

Un composé chimique émis depuis un emplacement précis, la zone polarisante, pourrait se diffuser sur au plus un demi millimètre, ce qui correspond à environ 30 à 50 cellules. Ces gradients chimiques seraient donc utilisés lors des premières phases du développement, le nombre de cellules y étant encore petit. Un processus serait alors déclenché en fonction de la quantité de produit chimique, laissant le reste du développement s'effectuer par un autre mécanisme. A titre d'exemple, les "doigts" numéros 2 à 4 d'un membre d'embryon de poulet sont initiés par une zone polarisante. En ajoutant artificiellement une deuxième zone, il est possible de créer un poulet avec deux fois plus de doigts disposés de façons symétrique avec une organisation de type 4, 3, 2, 2, 3, 4 au lieu de 2, 3, 4 [Tho 05].

6.1.3. Communication intercellulaire

La communication intercellulaire est également importante. En fonction de son état interne, une cellule envoie des signaux chimiques à son voisinage. Ces interactions modifient alors le comportement interne des autres cellules, et de ce fait, de simples communications locales peuvent aider à la création d'un organisme [Tho 05].

6.1.4. Temps

Le temps semble jouer un rôle important dans le développement d'un organisme. Comme l'a observé l'équipe bâloise de Gehring [Geh 02], la formation de l'axe de la colonne vertébrale est dirigée par un ensemble de gènes organisateurs placés dans le génome dans un ordre bien précis. Ces gènes sont exprimés dans l'ordre de leur positionnement, et durant un temps donné. De cette manière, les gènes actifs à un instant précis définissent le type de vertèbre à générer. Toujours basé sur le temps, la création du bras se fait à partir d'un bourgeon comprenant une zone prolifération. Ce bourgeon part du tronc pour créer le bras entier, en générant de nouvelles cellules. Une cellule posséderait une horloge qui s'arrêterait lorsqu'elle quitte la zone prolifération [Tho 05]. Le temps passé dans cette zone définirait alors la position de la cellule dans le bras.

6.1.5. Réseau de régulation de gènes

Finalement, citons les réseaux de régulation de gènes. Il existe en effet plusieurs types de gènes participant au développement, et entre autres les gènes de régulation et les gènes de différenciation. Les premiers génèrent des protéines, qui elles-mêmes activent ou inhibent d'autres gènes, et ceci dans un réseau de dépendance pouvant être inextricable. Cette dynamique de la cellule permet son fonctionnement et dans notre cas précis, le développement de l'organisme [Tho 05].

6.1.6. Cellules souches

Les gènes de différenciation sont exprimés à des stades particuliers, lorsqu'une cellule doit se différencier. En effet, il existe, dans le corps humain, environ 350 différents types de cellules (musculaires, neuronales, ...), qui servent toutes en des endroits bien particuliers de l'organisme. Au début du développement, l'embryon est composé de cellules souches, totipotentes, qui peuvent se différencier en n'importe quel type. Ces cellules ont donc le potentiel de pouvoir effectuer n'importe quelle tâche, jusqu'à leur spécialisation. Ce processus, que l'on croyait irréversible [Nyd 01], est en fait réversible, et des cellules différenciées peuvent redevenir des cellules pluripotentes.

6.2. Phylogenèse

Lors d'une fécondation, chaque parent transmet à sa progéniture la moitié de ses gènes. De cette manière, l'enfant a tendance à ressembler à sa mère et à son père, puisqu'il se développe sur ce substrat génétique qu'il partage en partie avec eux. Ce génome définit en effet une sorte de programme de formation de l'être à partir de la première cellule de base [Tho 05]. Les espèces évoluent donc, de par le mélange incessant de gènes au sein d'une population d'individus génétiquement compatibles. De plus, un individu, pour pouvoir se reproduire, doit être suffisamment adapté à son environnement pour ne pas mourir avant d'avoir atteint sa maturité sexuelle. En conséquence, les parents sont des êtres aptes à la survie, qui auront tendance à donner naissance à une progéniture également capable de s'intégrer dans son environnement.

Ce processus, appelé sélection naturelle, force une population à ne toujours être composée que d'individus bien adaptés.

Nous désirons ici nous étendre quelque peu sur les différentes théories de l'évolution, qui sont trop souvent ignorées au profit de la théorie de Darwin. Nous présenterons ci dessous quelques considérations biologiques autres que celle de Darwin.

6.2.1. Créationnisme

Dieu créa la terre et tous ses habitants en six jours ! Tous les individus d'une espèce vivante seraient donc issus d'un seul couple originel, tel Adam et Eve pour les humains. Dans cette théorie qui n'en est pas vraiment une car non réfutable, l'évolution n'existe pas, seule la disparition d'une espèce étant possible. Bien que peu probable en tant qu'explication du vivant, nombre d'entre-nous considèrent le créationnisme comme étant la seule et unique vérité [Tho 05]. Nous n'y accorderons toutefois que peu d'attention, le fixisme des espèces ne laissant aucune place à une quelconque adaptation, et sa non réfutabilité n'étant pas acceptée des scientifiques.

6.2.2. Fixisme

Vers la fin des années 1700, Georges Cuvier [Buf 02], un Français, élabore le fixisme. Entièrement en accord avec le créationnisme, il le nuance toutefois en introduisant le fait qu'il n'y aurait pas eu qu'une seule création, mais plusieurs. Suite à des catastrophes anéantissant toute vie sur Terre, le Créateur aurait créé de nouvelles espèces, dont celles issues de la dernière création auraient subsisté jusqu'à nos jours. Là encore, toute trace d'évolution est écartée, permettant à Cuvier de figurer dans les bons papiers de l'Eglise et des "bien-pensants". Cette approche n'est toutefois pas une théorie, de par les mêmes arguments que pour le créationnisme [Tho 05].

6.2.3. Transformisme

Ce n'est qu'au début du 19ème siècle que Jean-Baptiste Pierre Antoine de Monet, chevalier de Lamarck, propose une nouvelle théorie. En 1800, il prononcera les mots suivants lors de d'un discours inaugural au Muséum National d'Histoire Naturelle : "Je pourrais prouver que ce n'est point la forme du corps, soit de ses parties, qui donne lieu aux habitudes, à la manière de vivre des animaux ; mais que ce sont au contraire les habitudes, la manière de vivre et toutes les circonstances influentes qui ont avec le temps constitué la forme des animaux." [Tho 05].

Ce discours présente ce qui deviendra le transformisme, à savoir la première théorie évolutionniste. Dans sa *Philosophie Zoologique* [Lam 09] de 1809, Lamarck introduit le concept d'évolution avec hérédité des caractères acquis. Dans cette théorie, un individu ayant appris certains comportements ou dont le corps se serait modifié pourrait transmettre ces changements à sa progéniture. L'exemple le plus fréquemment cité est le cas de la girafe, dont le cou se serait allongé au fil des générations. Le cou d'un individu s'allongerait durant sa vie, afin d'attraper le plus de feuilles possibles, et cette modification physiologique serait transmise à sa descendance.

Malheureusement pour Lamarck, aucune de ses expériences ne put prouver sa théorie. Il avait introduit le concept d'évolution, mais il fallut attendre un demi-siècle pour voir le darwinisme arriver.

Notons que bien que le lamarckisme ait été abandonné depuis de nombreuses années, de récents travaux tels ceux de l'équipe de Lars Olov Bygren [Kaa 02] tendent à montrer qu'il existerait une hérédité liée à des expériences vécues. De même, Yves Coppens, un éminent spécialiste de l'évolution, cité dans Science et Vie, imagine "dans les caryotypes mêmes [les chromosomes, Ndlr] un mécanisme subtil qui serait capable de recevoir de l'information du milieu qui change et de s'en servir, en toute connaissance de cause, pour provoquer, dans la bonne direction, les dites mutations." [Tho 05].

Cette nouvelle orientation des recherches n'en est qu'à ses débuts, et d'autres expériences sont nécessaires avant d'annoncer à grand fracas le retour d'une certaine forme de Lamarckisme. Sur le plan des systèmes artificiels, le concept de Lamarckisme à été étudié par plusieurs équipes de recherche, dont certains des travaux sont résumés dans [Hay 99, Sas 00].

6.2.4. Darwinisme

La théorie de la sélection naturelle, ou darwinisme, est attribuée à Charles Darwin, qui la présente de manière fort complète dans son *Origine des Espèces par voie de sélection naturelle* de 1859.
Et c'est ainsi que fut créée la théorie de la sélection naturelle, selon laquelle les individus les plus féconds engendrent plus de descendants, et donc imposent leurs caractères au fil des générations. L'évolution des espèces serait donc poussée dans la direction des plus aptes à la survie et à la procréation, les caractères biologiques la favorisant étant transmis à la progéniture, et ce grâce à la fécondité différentielle [Tho 05].

Notons que Darwin, à l'instar de Lamarck, croit à une transmission de caractères acquis, la pangenèse. Des gemmules serviraient au corps pour l'envoi d'informations aux parties génitales, et donc les informations transmises à la progéniture pourraient être affectées par les expériences vécues par l'individu [Tho 05].

6.2.5. Néo-darwinisme

Le néo-darwinisme voit sa source dans les travaux de Ernst Haeckel, en fin de 19ème siècle, avant la redécouverte des lois de Mendel. Le terme néo-darwinisme est introduit en 1896 par George Romanes, pour décrire la pensée d'August Weismann, qui est un fervent partisan de cette nouvelle théorie selon laquelle la sélection naturelle est toujours d'actualité, mais qui réfute la transmission des caractères acquis qu'avaient suggéré Lamarck et Darwin. Dans cette lignée, Gustav Fischer, J.B.S. Haldane et Sewall Wright proposent ensuite des modèles mathématiques de la sélection naturelle [Tho 05].

6.2.6. Théorie synthétique

En 1866, Gregor Mendel publie ses résultats, présentant les premiers travaux en génétique expliquant la transmission des caractères héréditaires. Ce n'est toutefois qu'à partir de 1900 que naît la génétique, après que ses travaux aient été reconnus [Tho 05].

Basés sur cette nouvelle science, Théodosiens Dobzhansky [Dob 66], Ernst Mayr [May 82], Georges Gaylord Simpson et Julian Huxley vont, dans les années 1940, lancer les bases de la théorie synthétique, en alliant la génétique et la paléontologie à la théorie néo-darwinienne. Le concept de sélection naturelle reste équivalent, mais la transmission du patrimoine génétique explique ici la transmission des caractères. Les chromosomes des deux parents sont mélangés par un processus appelé crossing-over. De plus, des mutations peuvent intervenir, en modifiant le génome d'un individu de manière aléatoire. Cette théorie explique donc la microévolution (au niveau des individus). Bien qu'adoptée par une grande majorité, elle pose quelques problèmes sur le plan de la macroévolution (au niveau des espèces). En effet, la théorie synthétique est une théorie gradualiste, où l'évolution d'une espèce vers une autre se ferait lentement, par micro changements successifs, et les paléontologues peinent à trouver des fossiles entre certaines espèces. Il existe trop de chaînons manquants que cette théorie ne peut expliquer [Tho 05].

6.2.7. Monstres prometteurs

Deux théories saltationnistes, celle des monstres prometteurs et celle des équilibres ponctués, vont à l'encontre du gradualisme. La première, due à Richard Goldschmidt, fut publiée en 1940 [Tho 05]. Goldschmidt y distingue les petites mutations de la microévolution gardant une continuité dans l'espèce, des grandes mutations de la macroévolution.

Ces dernières généreraient des monstres prometteurs, relativement distants de leurs congénères sur le plan génétique. Ces nouveaux individus seraient alors à la base d'une nouvelle espèce, sans qu'un changement graduel ne puisse être observé [Sor 95].

Cette théorie eut beaucoup de mal à se défendre face aux acteurs de l'approche néo-darwinienne. Ce n'est qu'à partir de 1970 que les progrès en génétique permirent de montrer que la modification d'un seul gène pouvait avoir d'énormes conséquences sur la morphologie d'un individu. Paolo Sordino, Frank van der Hoeven et Denis Duboule, ont entre autre montré que la modification de seulement deux gènes organisateurs a pu transformer une nageoire en une patte lors du passage des poissons à la vie terrestre [Tho 05]. Les monstres prometteurs ont donc pu jouer un rôle non négligeable dans l'évolution des espèces [Sor 95].

6.2.8. Équilibres ponctués

Dans la même optique non gradualiste, Niles Eldredge et Stephen Jay Gould publient en 1972 leurs idées sur la question [Tho 05]. Leur théorie des équilibres ponctués, habituellement attribuée à Gould, consiste en une évolution active sur de très courtes périodes, le reste du temps ne voyant que peu de variations au sein d'une même espèce.

Ces courtes périodes peuvent être dues à un élément tel qu'une catastrophe naturelle, ou un brusque changement de l'environnement. Durant cette transition, soit un grand nombre d'individus ne survivraient pas, et une nouvelle population serait créée à partir d'un sous-ensemble de la population initiale, soit la population serait séparée en sous-populations par un facteur quelconque.
Cette théorie, la plus communément admise par la communauté paléontologique, explique entre autre le manque de fossiles entre deux espèce parentes. La période d'évolution étant très courte, il est effectivement très peu probable de tomber sur un fossile correspondant [Tho 05].

6.2.9. Neutralisme

Quelques temps avant la théorie de Gould, Mooto Kimura publie la *Théorie neutraliste de l'évolution* [Tho 05], jetant un pavé dans la mare de la théorie synthétique. Suite à de nombreuses expériences reprenant certaines des idées de Malécot, Wahlund et Wright, il conclut que la sélection naturelle n'a que peu d'influence quant à l'évolution moléculaire. Cette évolution ne pousserait pas dans la direction d'une meilleure adaptation, mais la plupart des mutations seraient neutres au regard de la sélection naturelle, et l'évolution correspondrait à une dérive génétique aléatoire [Tho 05].

Cette approche est intéressante dans le sens où ce ne sont que le hasard et les probabilités qui guident l'évolution, en créant un grand nombre de possibilités, la sélection n'intervenant que dans l'élimination des inaptes.

6.2.10. Néo-mutationnisme

Un des défauts de la sélection naturelle est le manque de considération des contraintes structurelles, physiques, ou liées au fonctionnement de la cellule. Ces contraintes peuvent dans certains cas forcer l'évolution dans une certaine direction, sans apporter d'avantage sélectif. Un des exemples de cette théorie est la cavité cylindrique centrale présente au milieu des coquilles d'escargots, qui, selon Stephen Jay Gould [Gou 91], serait due à des modalités de développement des mollusques plus qu'à une sélection. Le néo-mutationnisme prend donc en compte les contraintes structurelles. De plus, elle réactualise la théorie des monstres prometteurs en mettant l'accent sur les mutations plus que sur la sélection, remettant le hasard au goût du jour.

6.2.11. Transitionisme

Finalement, une dernière théorie se développe à partir des travaux en génétique moléculaire du développement. Selon cette théorie, développée entre autres par l'équipe de Duboule [Dub 98], les mécanismes d'évolution ont eux-mêmes évolué au cours du temps, passant du gradualisme au ponctualisme. La démonstration se base sur le fonctionnement des gènes en observant que ceux-ci n'ont que peu varié tout au long de l'histoire du vivant. Nous retrouvons en effet les mêmes groupements dans les organismes ancestraux que chez l'être humain. La création d'organismes complexes n'a donc pu se faire que par l'introduction de la multifonctionnalité des gènes, ces derniers ne servant plus à définir un seul paramètre du développement, mais étant actifs dans plusieurs processus. Dès lors, les réseaux de régulation des gènes sont devenus de plus en plus complexes, et la modification d'un gène n'influant

plus sur une seule partie du développement risque fort de compromettre d'autres parties, rendant ainsi l'embryon inapte à la survie. Ceci implique donc une grande résistance des gènes à la variation, peu de celles-ci étant permises [Tho 05]. De ce fait, l'évolution d'une espèce vers une autre, qui dans les temps anciens pouvait se faire de manière gradualiste, ne dispose que de peu de possibilités, et ces possibilités représentent de grands sauts évolutifs de par le fait qu'une modification influe sur beaucoup de mécanismes.

6.3. Epigenèse

Troisième axe de la vie, l'épigenèse constitue ce que nous appelons communément l'apprentissage. Notre savoir, qui définit notre comportement, est composé d'éléments innés et acquis. Les premiers nous sont "imposés" par nos gènes, qui semblent coder certaines réactions du type réflexes. Les seconds correspondent à l'ensemble des expériences accumulées durant la vie d'un individu, qui l'enrichissent et influent sur ses décisions courantes. Cet apprentissage nécessite un système nerveux capable d'adaptation, ce que quasiment tous les animaux pluricellulaires possèdent. L'être humain dispose quant à lui d'un cerveau volumineux composé de quelques 10 à 30 milliards de neurones [Tho 05].

Le neurone est principalement une cellule particulière ayant quatre fonctions, à savoir, transmettre de l'information extérieure ou intérieure, analyser cette information, et potentiellement la mémoriser. Ils servent à récupérer les données sensorielles tels que la lumière pénétrant les yeux ou le toucher sur la peau, à traiter et stocker cette information, et à agir sur les muscles de manière à faire se mouvoir le corps. Ils sont composés d'un corps cellulaire, d'un axone capable de transmettre de l'information sur une longue distance, et de dendrites récupérant de l'information d'autres neurones.

Il existe dans le cerveau une foule de types de neurones différents. Leur fonctionnement de base est toutefois toujours semblable, et utilise des potentiels électriques.

Un neurone est capable d'émettre un signal électrique transmis à ceux qui lui sont connectés.

Cette émission dépend de l'état interne de la cellule ainsi que des stimuli reçus, et est typiquement activée lorsqu'un seuil de potentiel est atteint.

Il est intéressant de noter que l'épigenèse est fortement liée à l'ontogenèse, le cerveau n'étant que partiellement formé à la naissance (la capacité crânienne augmente ensuite de 4,3 fois) [Mic 03]. Durant le développement, les neurones se différencient et se disposent en couches dans le cerveau, grâce à leur capacité migratoire. Ils sont formés sur la surface cérébrale, puis migrent vers le centre en s'accrochant à des cellules gliales, qui les guident dans la bonne direction. Ils s'arrêtent à un emplacement correspondant à leur fonction, créant ainsi les différentes couches cérébrales.

Les neurones sont reliés entre eux par des synapses qui sont créées à raison de 40000 par secondes à la naissance. Un fait important est qu'une synapse ne perdure que si elle reçoit un feed-back environnemental, c'est-à-dire participe à un traitement d'information. Et alors que

nous pensions que lorsque la construction du cerveau était achevée plus aucun neurone ne pouvait être généré, de récents travaux montrent qu'il existe une forme de neurogenèse, des neurones, des axones, et des dendrites se créant durant toute la vie de l'individu [Tho 05].

7. Conclusion

Dans ce chapitre, nous avons présenté les processus de développement et d'évolution des systèmes biologiques tels qu'ils sont vus par les biologistes. Cependant, nous nous sommes restreints à quelques notions clés, vu la complexité de ces systèmes.

Ainsi, nous pouvons résumer ce chapitre par le fait que l'évolution biologique se partage en développement et en évolution des espèces. Le développement d'un organisme représente le processus qui fait passer ce dernier de la phase d'une cellule unique à la phase d'un organisme composé d'organes ordonnés, capable de survivre et de se reproduire. L'évolution génétique permet à une espèce de se perpétuer tout en s'adaptant à un environnement changeant.

A l'issue de ce tour biologique, il ressort les aspects importants suivants [Mes 06]:

- La continuité : Le processus de développement est un processus continu qui façonne un organisme conformément à un plan préalablement établi.
- L'autonomie : Les organismes présentent une certaine autonomie dans l'évolution, dans le sens où le milieu intervient mais dans les limites de l'ouverture qu'offre un programme génétique.
- La spécialisation. Malgré le fait que la génétique joue un rôle prépondérant dans le développement et l'évolution, on constate qu'au fur et mesure des évolutions (en comparant plusieurs espèces [Rid 96]) le système nerveux, le système immunitaire et le système endocrinien ont émergé comme des systèmes spécialisés qui interagissent avec l'environnement et où chacun joue un rôle spécifique.
- Le hasard : L'évolution génétique des espèces se base sur des mutations et des recombinaisons aléatoires. Ainsi, au hasard des évolutions du génome, l'espèce évolue librement vers de nouveaux horizons. Le milieu intervient par la sélection des organismes les plus aptes.
- Le temps : L'aboutissement à un organisme complexe et parfaitement adapté à son environnement est le résultat d'un long processus d'évolution.

Les systèmes biologiques renferment des mécanismes et des processus longuement éprouvés qui leur ont permis de survivre face à un environnement changeant et contraignant. Ce fait a conduit l'être humain à les considérer comme source d'inspiration dans la conception des systèmes artificiels.

8. Références

[Buf 02] E.Buffetaut , cuvier le découvreur du monde disparus, Belin pour la science, Paris, Octobre 2002.

[Dob 66] T.Dobzhansky, l'homme en évolution, livre édition Flammarion, Paris 1966.

[Dub 98] D.Duboule, A.S.Wilkins, The evolution of bricolage, Trends in Genetics Journal, Vol14, Fevrier 1998, pp54-59.

[Geh 02] W.Gehring, M.Affolter et T.Burglin, Homeodomain Protiens, Annual Reviews of Earth and Planetary Sciences, 2002, pp487-523.

[Gil 03] S.F.Gilbert, Developmental biology, Livre 7eme edition, Sinauer associates Inc. Publishers, 2003.

[Gou 91] S.J.Gould, La vie est belle les surprises de l'évolution, livre édition science, Paris, 1991.

[Hay 99] B.Hayes, experimental Lamarckism, American scientist Journal, Vol. 87, N°6, November-December, 1999, pp494-498.

[Kaa 02] G.Kaati , L.O.Bygren et S.Edvinsson, cardiovascular and diabetes mortality determined by nutrition during parents and grandparents slow growth period, european journal of human genetics, London, 2002, pp682-688.

[Lam 09] J.Lamarck philosophie zoologique, livre, paris, 1809.

[Lew 99] B.Lewin, Genes VII, Livre Oxford University Press, Oxford, 1999

[Lin 91] F.Lints, Génétique, 3eme édition, Lavoisier Technique et Documentation, 1991.

[May 82] E.Mayr, speciation and macroevolution, Journal of evolution, 1982, pp1119-1132.

[Mes 06] D.Meslati, Mage : une approche ontogénétique de l'évolution dans les systèmes logiciels critiques et embarques thèse de doctorat université de Annaba Algérie 2006.

[Mic 03] E.Michel, Les neurones et créativité, revue littéraire Lieux d'être, n° 36, automne, Paris, 2003.

[Nyd 01] U.Nydegger, Hematologie la cellule souche d'ici à là et retour, Forum medical suisse N° 51/52, décembre 2001

[Rid 96] M. Ridley, Evolution, Livre 2eme edition, Blackwell scientific publications Ltd, Oxford, 1996.

[Sas 00] T.Sasaki et M.Tokoro, comparison between lamarckian and darwinian evolution on a model using neural networks and genetic algorithms, Journal of knowledge and information systems, London, juin 2000, pp201-222.

[Seg 98] R.Segev, E.Ben Jacob from neurons to brain: adaptive self writing of neurons, journal of complex systems, 1998, pp67-78.

[Sor 95] P. Sordino, F.Van Der Hoeven et D. Duboule, Hox Gene Expression in Teleost Fins and the Origin of Vertebrate Digits . Journal of Nature, juin 1995, pp678–681.

[Tho 05] Y.Thoma, Tissu numérique cellulaire à routage et configuration dynamiques, Thèse EPFL, no 3226, Lausanne Suisse, 2005.

[Wat 53] J.Watson, F.Crick, A structure for deoxyribose nucleic acid , Journal of Nature, Avril 1953, pp737-738.

Chapitre 3 Les Grandes Familles Des Systemes Bio-Inspires

1. Introduction

Systèmes multi agents, système de neurones artificiels, algorithmes génétiques et systèmes immunitaires artificiels ont tous un point commun. Chacun de ces systèmes présente un paradigme inspiré de la biologie. Cette inspiration des systèmes biologiques a donnée naissance à une multitude de systèmes dits bio-inspirés.

Ce chapitre propose une synthèse de quelques systèmes bio-inspirés et définit les concepts et termes clés de ces dernières. Les sections seront organisées comme suit :

- Nous allons exposer des automates cellulaires et des concepts qui les constituent.
- Nous aborderons le sujet des systèmes évolutionnaires et les termes clés qui les définissent.
- Puis nous présenterons des systèmes connexionnistes
- Et nous terminerons le chapitre avec les systèmes immunitaires artificiels

2. Les systèmes cellulaires

La plus simple partie d'un système qui peut être considéré comme vivante est la cellule. Bien que les cellules soient déjà complexes, la forme la plus complexe est l'organisme multicellulaire qui est formé de plusieurs cellules. Dans les organismes multicellulaires presque toutes les cellules possèdent le même génome mais la morphologie et la fonction de deux cellules peut être différente. Cette différence est expliquée par le fait que l'état de chaque cellule ne dépend pas seulement de sa composition génétique mais de l'état de la cellule quand elle a été crée. Les systèmes multicellulaires sont des exemples de systèmes qui sont composés de plusieurs copies d'une unité fondamentale « la cellule » et dont les interactions produisent un comportement global complexe [Dar 08].

2.1. Le Jeu de la vie

À l'origine, le « Jeu de la vie » fut présenté comme un jeu mathématique. Sa description va nous permettre de matérialiser et mieux comprendre ce que sont les automates cellulaires [Ren 00].

A l'instar des espaces cellulaires d'Ulam, le « Jeu de la vie » se présente sous la forme d'une grille constituée de cellules:(Voir l'exemple de la figure 3.1).

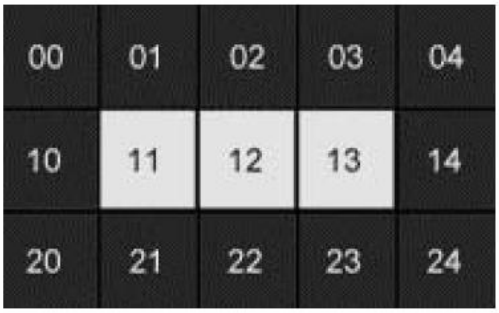


Figure 3.1. Exemple de configuration de départ [Ren 00]

L'univers est limité ici à un rectangle de 5 par 3. Pour faciliter l'explication, nous avons numéroté les cellules de 0 à 4 en horizontal et de 0 à 2 en vertical. Les cellules claires sont actives.

Dans le Jeu de la vie, une cellule est considérée comme voisine si elle est contiguë, y compris en diagonale.

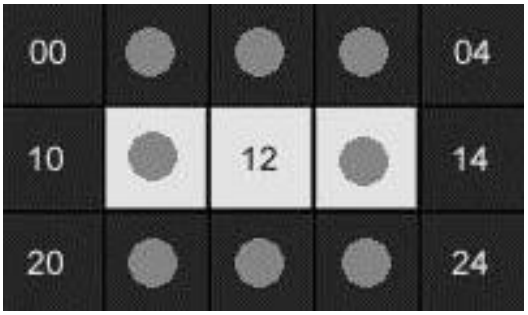


Figure 3.2. Détermination du voisinage [Ren 00].

La figure 3.2 montre le voisinage de la cellule 12. En l'occurrence, sur les huit voisines, deux sont actives.

Les règles du Jeu de la vie sont simples :

- Une cellule inactive entourée de 3 cellules actives devient active (« naît ») ;
- Une cellule active entourée de 2 ou 3 cellules actives reste active
- Dans tous les autres cas, la cellule « meurt » ou reste inactive.

On peut interpréter ces règles en considérant qu'une naissance nécessite un certain rassemblement de population (3 en l'occurrence), que les cellules ne peuvent survivre à un trop grand isolement (moins de 2 voisines) et qu'une trop forte concentration (plus de 3 voisines) les étouffe.

Les automates cellulaires fonctionnent de manière *discrète*. C'est-à-dire que le temps s'écoule par à-coups. Ceci signifie dans notre cas, à la génération *t*, chaque cellule examine son

environnement et détermine son état futur. Quand l'ensemble des cellules a été traité, et seulement à ce moment là, toutes les cellules passent à l'état calculé. On simule ainsi un traitement parallèle [Ada 98].

Illustrons ce mécanisme à partir de la configuration précédente :

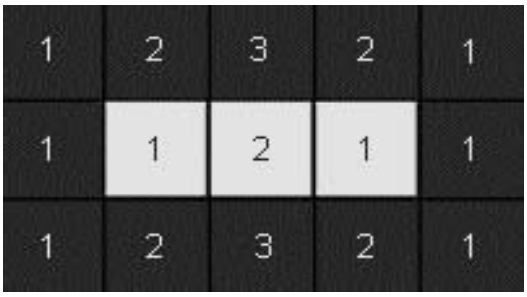


Figure 3.3. Valeurs de voisinage [Ren 00].

Dans le schéma précédent, le nombre de voisines actives est noté pour chaque cellule :

- Les cellules inactives 00, 04, 10, 14, 20 et 24 ont une voisine active et restent inactives.
- Les cellules inactives 01, 03, 21 et 23 ont deux voisines, elles ne changent donc pas.
- Les deux cellules inactives restantes (02 et 22) ont trois voisines actives, la règle 1 s'applique : elles naissent.
- Les cellules actives 11 et 13 n'ont qu'une voisine active : elles meurent.
- Enfin la cellule active 12 ayant deux voisines actives elle reste en vie.

À la génération suivante, seules les cellules 02, 12 et 22 seront donc actives.

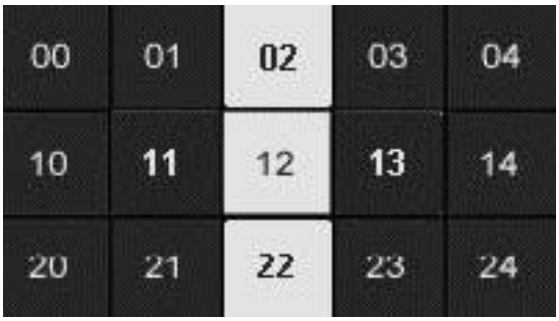


Figure 3.4. Seconde génération [Ren 00].

On met ici en évidence les trois propriétés fondamentales des automates cellulaires :

- *Le parallélisme* : Un système est dit parallèle si ses constituants évoluent simultanément et de manières indépendantes.
- *La proximité* (locality) : Le nouvel état d'une cellule ne dépend que de son état actuel et de l'état du voisinage immédiat.
- *L'homogénéité* : Les lois sont universelles, c'est-à-dire communes à l'ensemble de l'espace de l'automate [Wol 02].

2.2. Les composants de base

Nous utiliserons l'inspiration fournie par les tissus cellulaires biologiques pour définir les éléments d'un système cellulaire. On abstrait le tissu cellulaire comme une collection de cellules qui va devenir notre espace d'état ou de recherche. Les états internes de la cellule biologique sont représentés par des nombres ou symboles qui prennent leurs valeurs d'un ensemble simple. Les règles et les interactions complexes entre les cellules sont représentées par des fonctions mathématiques et des règles logiques [Dar 08].

Dans des termes plus formels un système cellulaire est composé des éléments suivants :

2.2.1. L'espace cellulaire

C'est le nom qu'on donne à un ensemble de cellules dans le système. En général, c'est un tissage de n dimensions. Quand on fait des traitements sur les systèmes cellulaires dans un certain niveau d'abstraction le treillis est considéré comme infini, par contre toute implémentation doit être dans un espace fini de cellules. La figure 5 montre quelques types d'espaces cellulaires. Les espaces cellulaires de plus de 3 dimensions sont rarement implémentables parce que le nombre de treillis croit exponentiellement avec le nombre de dimensions [Har 01].

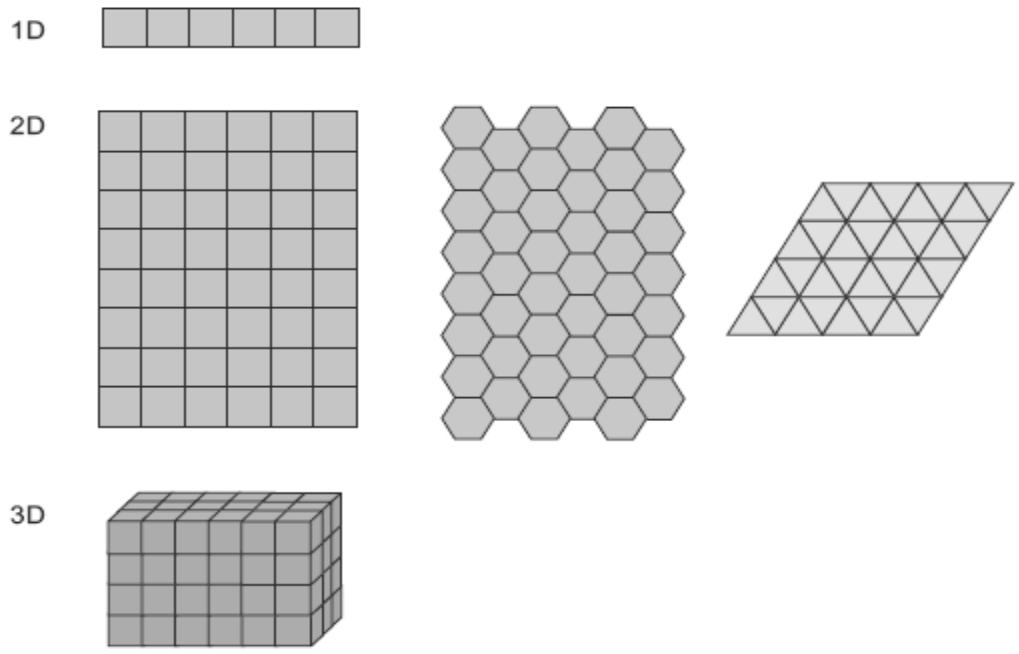


Figure 3.5. Espace cellulaire à une, deux et trois dimensions

2.2.2. Les variables du temps

L'effet dynamique des systèmes cellulaires dans l'axe du temps peut être modélisé par des variables discrètes ou continues.

2.2.3. Etats et ensemble des états

L'état de la cellule représente les informations spécifiant les conditions actuelles de la cellule. C'est une mémoire qui stocke le passé de la cellule. Donc, c'est l'historique de la cellule qui

peut influencer le futur du système cellulaire. L'ensemble des états S est l'ensemble des valeurs acceptables. Souvent un état particulier est spécifié comme état stable s0 qui représente l'inactivité de la cellule. Dans la plupart des systèmes cellulaires l'état de la cellule est représenté par des valeurs numériques. Parfois un n-uplet de nombres numériques au lieu d'une seule variable. L'avantage de l'utilisation des n-uplets est que chaque variable peut représenter un aspect de la cellule [Har 01].

2.2.4. Le voisinage

Le voisinage d'une cellule c'est un ensemble de cellules incluant la cellule elle-même où son état est directement influençable par ces derniers. La forme et la taille du voisinage peuvent être arbitraires. Donc, le voisinage d'une cellule est composé d'un nombre limité de cellules adjacentes parce que les systèmes cellulaires échangent les informations localement. La méthode la plus simple est de spécifier la distance du voisinage dans l'espace cellulaire. La figure 3.6 montre le principe du voisinage et illustre quelques types de voisinage [Har 01].

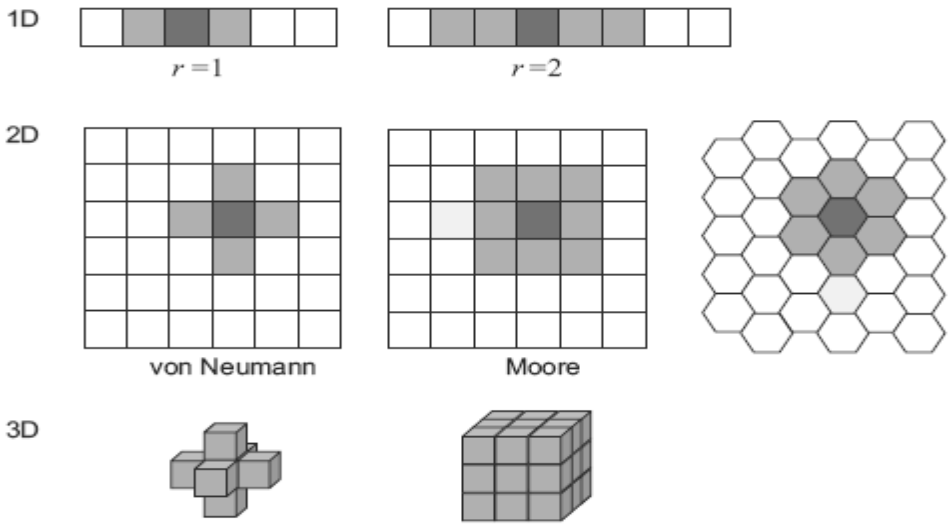


Figure 3.6. Quelques types de voisinage pour des représentations à une, deux et trois dimensions.

2.2.5. La fonction de transition

La fonction de transition est une fonction qui spécifie comment l'état de la cellule se développe dans le temps. Elle dépend surtout de l'état des cellules qui sont dans son voisinage et sur la position de la cellule et du temps. Si la fonction de transition est la même pour toutes les cellules ou si elle ne dépend pas du temps le système cellulaire est dit homogène.

2.2.6. Les conditions de frontière

Si l'espace cellulaire a une limite, les cellules en frontière peuvent manquer de quelques unes des cellules exigées pour former le voisinage. Plusieurs types de solutions existent pour traiter ces problèmes :

2.2.6.1. Périodique

La solution la plus simple est d'éliminer les frontières pour cela on va coller les frontières ensemble comme le montre la figure 3.7.

2.2.6.2. L'assignement

Une autre stratégie est de définir un voisin virtuel. La cellule virtuelle ajoutée pour compléter le voisinage doit être assigné dans un état qui dépend de l'état actuel du système cellulaire.

2.2.6.3. Copiage

On peut créer une cellule virtuel en copiant l'état de la cellule avoisinante, cette méthode s'appelle condition frontière adiabatique.

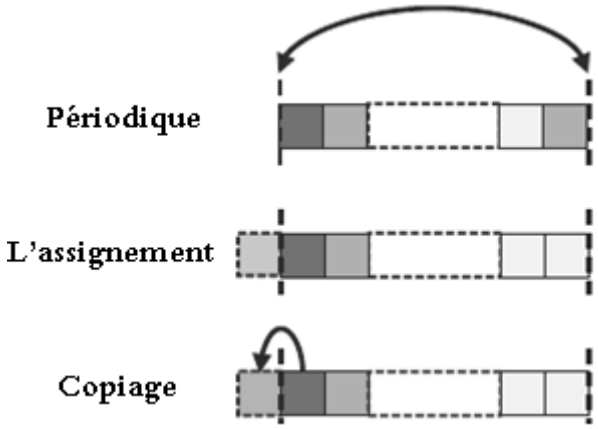


Figure 3.7. Les principales méthodes d'élimination de frontières sur un espace d'une dimension

2.2.7. Les conditions initiales

Il est nécessaire de spécifier l'état initial des cellules avant de commencer l'exécution « maître à jour les cellules ».

2.2.8. Les conditions d'arrêt

Elles spécifient quand la mis à jour des cellules du système doit s'arrêter, généralement quand l'état des cellules devient cyclique [Har 01].

2.3. Les automates cellulaires

Le type le plus simple et le plus populaire des systèmes cellulaires sont les automates cellulaires qui possèdent toutes les spécificités citées plus haut. Un automate cellulaire a des variables de temps de type discret, un voisinage finis, un ensemble d'états finis, et une mise à jour synchrone de l'ensemble des cellules. La séquence d'entiers S={0,…,k-1} est souvent utilisée comme ensemble d'état avec s_0=0 qui représente l'état stable. La fonction de transition F « souvent dite règle de transition » détermine l'état $s_i(t+1)$ de la i ^{éme} cellule au temps t+1 en fonction de l'état des cellules voisins Ni au temps t, autrement dit :

$$si(t+1) = \varphi(sj(t): j \in Ni \qquad (1)$$

Les règles de transition (équation 1) peuvent être représentées par une table de transition qui spécifie toutes les configurations possibles du prochain état de la cellule. Si le nombre des états est k et le nombre de voisines est n alors le nombre de configurations possibles de

voisinage est k^n . Donc la représentation des règles de transition sous forme de table devient impossible si le nombre d'état et des voisines augmentent. De plus pour chaque configuration de voisinage nous avons k façon de le spécifier, donc nous avons $k^{k(n)}$ différente règles de transition pour un automate à k états.

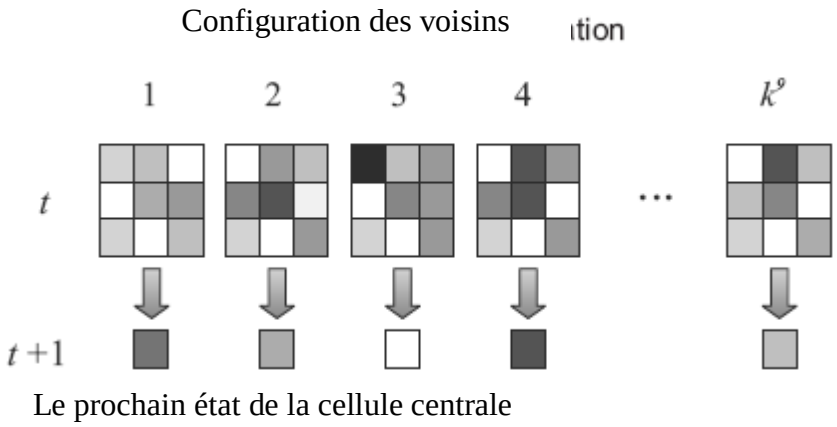


Figure 3.8. Tables de transition d'un automate cellulaire à deux dimensions

2.3.1. Quelques types de règles pour les automates cellulaires

L'univers des règles de transition est très large, on peut lister les règles de transition les plus communes.

2.3.1.1. Totale

On suppose que les états sont représentés par des nombres, une règle est dite totale si elle dépend de la somme des valeurs des états des cellules voisines. Une règle totale peut s'écrire (équation 2):

$$si(t+1) = \varphi(\sum_{j \in ni} sj(t)) \qquad (2)$$

2.3.1.2. Totale externe

Une règle est dite totale externe si elle dépend de la valeur de l'état de la cellule qui est mise à jour « la cellule du centre » et aussi de la somme des cellules voisines (équation 3).

$$si(t+1) = \varphi(si(t), \sum_{\substack{j \in ni \\ j \neq i}} sj(t)) \qquad (3)$$

2.3.1.3. symétrique

Une règle est symétrique si le changement de position des voisines ne l'affecte pas. Par exemple les règles de type total et totale externe sont symétriques car la cellule voisine peut être à gauche, à droite, en haut, … . Son effet sera le même sur la cellule qu'on souhaite mettre-à-jour.

2.3.1.4. L'état nul

Une règle est dite état nul si les voisines stables engendrent un état stable pour la cellule considérée.

2.3.2. Modélisation d'un automate cellulaire

Pour définir un modèle d'automate cellulaire nous procédons de la façon suivante :

- Attribuer un espace cellulaire
- Attribuer des variables temporelles
- Définir le voisinage
- Définir l'ensemble des états
- Définir les règles de transitions
- Définir les conditions d'élimination de frontières
- Définir les conditions initiales et finales
- Mise à jour des états des cellules jusqu'à atteindre la condition d'arrêt

2.3.3. Autres types d'Automates Cellulaires

Le Jeu de la vie vu plus haut n'est qu'un type d'automate cellulaire parmi une infinité. Il est en effet possible de jouer sur l'ensemble des règles qui régissent l'univers de l'automate cellulaire [Gut 94].

Le paramètre le plus évident est le nombre de dimensions. Rien n'oblige en effet à considérer des environnements à deux dimensions. L'analyse théorique des automates cellulaires s'est essentiellement effectuée à partir d'automates à une dimension. En réduisant le nombre de dimensions, on limite l'explosion combinatoire, donc le nombre d'automates possibles. Si l'on considère le cas simple d'un voisinage de trois cellules, soit la cellule concernée et ses voisines de droite et de gauche, dans un automate à une dimension et deux états, il n'existe que 2 puissance $2^3 = 256$ règles possibles. La représentation des automates à une dimension (soit une ligne), utilise la seconde dimension pour représenter le temps. À chaque génération, une nouvelle ligne est ajoutée au-dessous de la précédente, on peut visualiser ainsi la dynamique de ce type d'automate.

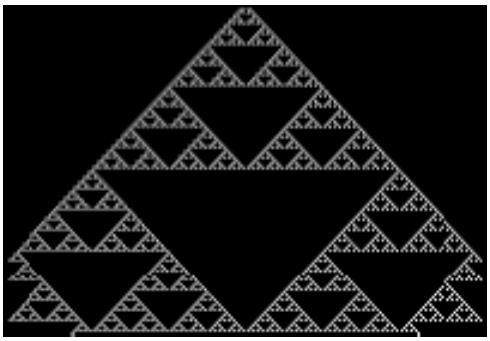


Figure 3.9. Exemple d'automate à une dimension (Triangle de Pascal) [Ren 00].

Il est naturellement possible de créer des automates à trois dimensions voire plus.

Il est également possible de jouer sur la détermination du voisinage. Si l'on considère les automates à deux dimensions, les voisinages les plus courants sont:

- *Von Neumann* : on considère les seuls voisins Nord/Sud/Est/Ouest.
- *Moore* : on ajoute les diagonales. C'est le cas du Jeu de la vie.
- *Moore étendu* : on étend la distance de voisinage au-delà de un.
- *Margolus* : on considère des ensembles de 2x2 éventuellement alternés.

C'est ce type de voisinage qui est utilisé dans la simulation du comportement des gaz.

Par exemple, l'automate de Fredkin qui utilise un voisinage de Moore est basé sur la parité du voisinage. C'est un automate de type *sommatif*, c'est-à-dire que l'état des cellules dépend du nombre de voisins actifs, indépendamment de leurs positions. En l'occurrence, il n'y a reproduction que si la valeur de voisinage est impaire. Cet automate a la propriété remarquable de reproduire toute configuration de base en neuf exemplaires. La règle de Fredkin est généralisable à plus de deux dimensions [Heu 98].

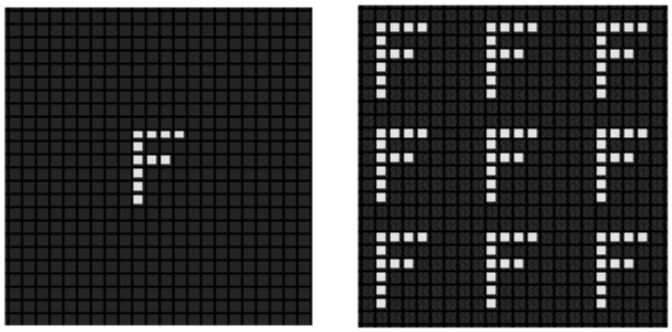


Figure 3.10. Fredkin génération 0 et Fredkin génération 8 [Ren 00].

Il est également possible de jouer sur le nombre d'états. Rien n'oblige en effet à se cantonner aux deux états vie/mort. *Brian's Brain* par exemple, présenté par Brian Silverman en 1984 utilise trois états (vie/fantôme/mort) pour engendrer une grande diversité de planeurs complexes au sein de configurations graphiques étonnantes.

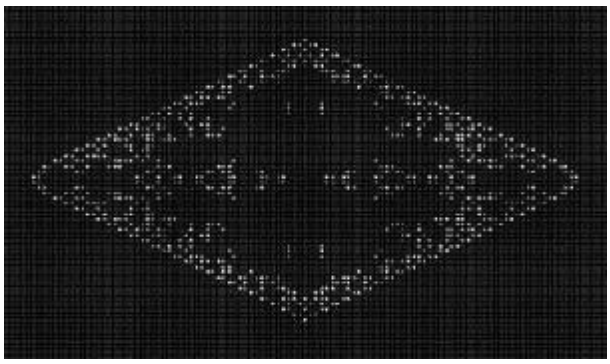


Figure 3.11. Planeurs Brian's Brain [Ren 00].

Des règles plus complexes sont imaginables. On peut par exemple construire des automates stochastiques dont les règles de transition intègrent une fonction de probabilité.

D'une manière générale, on peut construire tout type d'automate en jouant sur les règles *structurelles* et *fonctionnelles*. Les premières définissent la structure spatiale du réseau d'automates, soit son nombre de dimensions, le mode d'arrangement des cellules (carré, hexagonal, ... dans un automate à deux dimensions) et le mode de détermination du voisinage. Les secondes déterminent le nombre d'états et les règles de transition. Le choix de ces deux types de règles permet de construire un univers adapté à l'objectif recherché [Heu 98].

3. Les systèmes évolutifs

L'évolution artificielle inclut un large ensemble d'algorithmes qui prennent tous inspiration des principes de l'évolution naturelle et de la génétique dans le but de trouvé des solutions aux problèmes d'optimisation, découvrir des nouveaux programmes de calcul, modéliser des circuits électroniques, …

La plupart des systèmes évolutifs sont basés sur quatre principaux piliers de l'évolution naturelle :

- entretien d'une population
- création de diversité
- mécanisme de sélection
- processus d'héritage.

Dans les systèmes évolutifs, le phénotype d'un individu qui a subi le processus de sélection est la solution du problème. Quand au génotype, c'est la représentation génétique de la solution et qui est transmise par les générations et manipulé par les opérateurs génétiques.

Comme nous l'avons mentionné avant, la différence entre l'évolution naturelle et artificielle est que l'évolution naturelle n'a pas d'objectifs précis au contraire de l'évolution artificielle. La résolution du problème dans les systèmes évolutifs se base en particulier sur le processus de sélection qui se divise en deux étapes : une évaluation du phénotype qui fourni une valeur dite fitness et l'opérateur de reproduction qui fait des copies du génotype correspondant au phénotype avec la plus grande fitness.

La structure d'un algorithme évolutionnaire consiste en une simple procédure d'itération d'une population de plusieurs individus.

Les phénotypes sont évalué selon une fonction de fitness prédéfinie, le génotype du meilleur individu est dupliqué plusieurs fois et modifier par les opérateurs génétiques (mutation, croisement), et les nouveaux génotypes sont insérés dans la population à la place des anciens. Cette procédure se répète jusqu'à ce qu'une solution satisfaisante soit trouvée.

Les algorithmes évolutionnaires sont applicables à plusieurs domaines, si une représentation génétique cohérente peut être formulée. Ces algorithmes peuvent être couplés avec d'autres méthodes pour élever la qualité de la solution [Dar 08].

Il y a plusieurs types d'algorithmes évolutionnaires, qui sont souvent classifiés pour différentes raisons, par le type d'opération de mutation. Pour ne pas rentrer dans les détails nous allons donner les étapes pour formuler un algorithme évolutionnaire :
Les algorithmes évolutionnaires se décomposent généralement en 7 étapes (voir Figure 3.12) :

- **Etape 1** : Création d'une population de base de μ individus (les individus étant des solutions au problème à résoudre). Cette population peut-être créée aléatoirement ou grâce à une autre heuristique ou métaheuristique.
- **Etape 2** : Evaluation des individus composant la population par la fonction objective (qui doit juger de la qualité des solutions).
- **Etape 3** : Sélection de λ individus dans la population pour le croisement. Un individu peut-être sélectionné plusieurs fois. Le rapport entre λ et μ dépend généralement des politiques de remplacement que nous verrons ultérieurement.
- **Etape 4** : Croisement des λ individus sélectionnés.
- **Etape 5** : Mutation d'une partie des λ enfants (individus issus du croisement). Le *taux de mutation* est le pourcentage d'enfants qui vont subir une mutation, c'est un paramètre de l'algorithme.
- **Etape 6** : Evaluation par la fonction objective des λ enfants résultants des croisements et des mutations.
- **Etape 7** : Sélection de μ individus parmi la population de $\mu + \lambda$ individus (parents et enfants). Cette sélection se fait selon une politique de remplacement dont il existe plusieurs types et que nous verrons ultérieurement. A la fin de cette étape, si un critère d'arrêt est atteint, l'algorithme s'arrête et le ou les meilleurs individus sont sélectionnés. Dans le cas contraire, on recommence le processus à l'étape de sélection (étape 3).

3.1. Caractéristiques des systèmes évolutifs

3.1.1. La représentation génétique

La représentation génétique, connue aussi sous le nom de codage génétique, décrit les éléments du génotype et comment cet élément est mis en relation avec le phénotype.

Une bonne représentation génétique devrait avoir des opérateurs de mutation et de croisement qui ont une grande probabilité de générer de meilleurs individus, et tous les génotypes conçu on une grande probabilité de couvrir tous l'espace de travail pour générer une solution optimale. Par exemple si on veut faire évoluer un circuit électronique digital, la représentation génétique doit utiliser un alphabet discret qui a une correspondance avec les composant du circuit et par conséquent facilite la relation entre génotype et phénotype.

Nous donnons dans ce qui suit quelque représentation génétique qui font la correspondance entre génotype et phénotype.

3.1.1.1. Représentation discrète

L'individu est représenté par une séquence de i valeurs discrètes d'un alphabet avec une cardinalité k. Par exemple, l'algorithme génétique [Hol 75, Gol 89], qui est un cas particulier des algorithmes évolutionnaires, utilise souvent un alphabet binaire avec une cardinalité k=2.

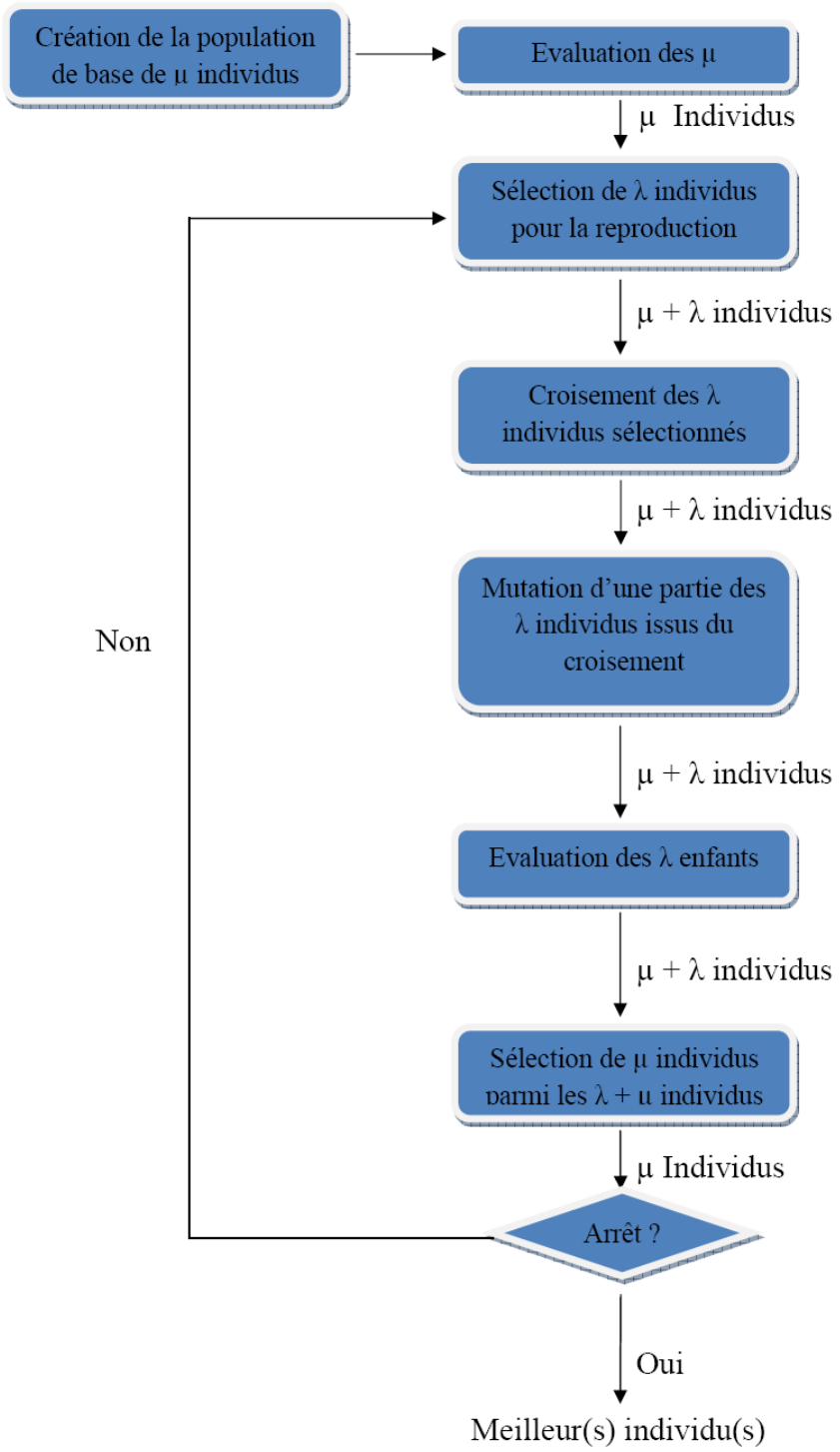


Figure 3.12. Schéma type d'un algorithme évolutionnaire [Lam 09].

Dans quelque cas, cette représentation binaire peut être interprétée directement comme phénotype, tel que la description d'une chaîne de caractère d'un FPGA « un circuits électronique programmable ». Cependant, il est souvent nécessaire de transformer la représentation binaire à différents types de phénotypes.
La représentation binaire du génotype peut être mappé sur plusieurs phénotypes, tel que nombre entier, nombre réel, etc.

On considère par exemple le génotype 01010100. Cette représentation peut être mapper vers un nombre entier (formule 4):

$$0 * 27 + 1 * 26 + 0 * 25 + 1 * 24 + 0 * 23 + 1 * 22 + 0 * 21 + 0 * 20 = 84 \qquad (4)$$

Il peut être aussi mapper vers un nombre réel R dans un rang [min, max] on le décode en premier lieu comme un nombre entier « i »puis on applique la formule R=i/max-min). Le même génotype peut être utilisé pour décrire des jobs schedullés, chaque position du génotype correspond à un job différent. 0 pour un job exécuté le matin et 1 pour un job exécuté l'après midi.

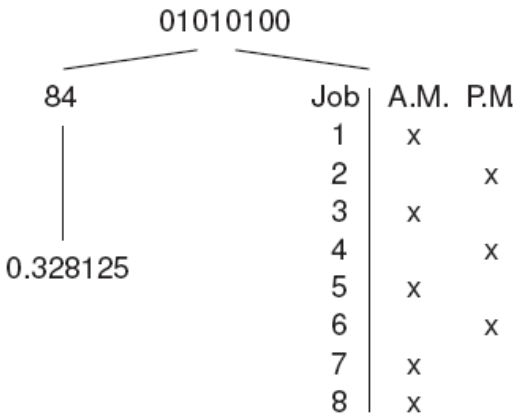


Figure 3.13. Représentation binaire d'un génotype et ses différentes interprétations

3.1.1.2. Représentation des valeurs réelles

Le génotype est vu comme un ensemble de nombres N de valeurs réelles, ces nombres sont généralement représentés en virgule flottante.
Cette représentation est utilisée pour des solutions qui requièrent des degrés de précision de haut niveau, comme les paramètres pour décrire une courbe d'aile d'un avion par exemple.

3.1.1.3. Représentation en arbre

Cette représentation est utilisée pour décrire des structures hiérarchiques avec des points de branchement et des conditions. Elle est notamment utilisée pour décrire des circuits électroniques.
Il existe plusieurs autres représentations mais nous avons donné ici les plus utilisés.

3.1.1.4. L'évolutivité

La représentation génétique affecte l'évolution du système c'est-à-dire la probabilité de générer une amélioration en appliquant les opérateurs génétiques. La représentation génétique

est définie pour s'adapter à la nature du problème. Les génotypes de longue taille couvre un large espace de recherche mais la probabilité pour produire une amélioration est faible par rapport a une représentation plus courte [Wag 96].

Un autre problème apparait lorsque la dimension de l'espace de recherche et le nombre de solutions sont prédéfinis ou constants, cela limite la progression de l'évolution à travers les générations.
Le résultat de l'évolution et de la transformation de génotype vers phénotype est plus intéressant si le phénotype se compose de plusieurs composants comme les robots autonomes.

3.1.2. La représentation de la population

La population initiale doit être suffisamment vaste et variée pour assurer que les individus aient différentes valeurs de fitness. Parce que si tous les individus ont la même valeur de fitness on ne peut pas faire de sélection « tous les individus se valent ». La taille d'une population dépend des propriétés de l'espace de recherche et du coût de calcul pour l'évolution de touts les individus sur plusieurs générations. Un problème se pose si la plupart des génotypes ont la même valeur de fitness ou si la mutation ne produit pas une amélioration nécessaire pour la taille de la population. Dans la littérature on trouve souvent des populations du rang de cent à quelque milliers d'individus. Si l'évaluation des individus requière des expérimentations du monde réel comme le cas de l'évolution des robots, la population est souvent de taille inférieure à cent individus.

Pour le cas de la représentation binaire, chaque génotype est crée par une génération aléatoire de séquences de un et de zéro. Même processus pour la représentation réelle mais avec un intervalle prédéfini. Cependant, pour la représentation réelle ce sont des valeurs réelles [Sch 92].

La représentation du génotype en arbre est construite à travers un processus récursif qui développe aléatoirement tous les nœuds de l'arbre. On commence avec le nœud racine en sélectionnant une fonction parmi l'ensemble des fonctions. Pour chaque argument sélectionné il est soit sélectionné dans l'ensemble des fonctions soit dans l'ensemble des terminaux (il est considéré comme feuille de l'arbre).

3.1.3. La fonction d'adaptation

La fonction d'adaptation associe un score à chaque phénotype dans la population. Celui-ci est le résultat d'un processus de croissance ou de modification d'un processus non génétique comme la variation de toute une vie ou l'apprentissage. La fonction d'adaptation évalue indirectement la qualité de développement et le processus d'apprentissage.

Il existe deux aspects très importants qui rentrent dans la conception d'une fonction d'adaptation: le choix et la combinaison des composant d'adaptation et la façon avec la quelle la fonction est évalué.

L'évaluation des individus est souvent la plus consommatrice de temps d'exécution dans les algorithmes évolutionnaire. La qualité de la solution dépend de la bonne évolution des individus.

3.1.4. La sélection et la reproduction

Le rôle de la sélection est de choisir un grand nombre de descendance pour la population. Les contraintes de sélection indiquent le pourcentage des individus qui créent la descendance de la prochaine population. Une sélection hautement contrainte implique qu'un petit pourcentage d'individus va être sélectionné pour la reproduction. Bien que cette stratégie donne des résultats rapides, elle provoque aussi le risque que la population perde rapidement de diversité et converge vers un minimum local. C'est pour cela qu'il faut maintenir une balance entre contraintes de sélection et les facteurs créateurs de diversité comme la mutation [Dar 08].

3.1.4.1. La sélection proportionnelle

Dans ce type de sélection, la probabilité p(i) qu'un individu i soit sélectionné est donnée par le ratio entre sa valeur d'adaptation f(i) et la somme de la valeur de d'adaptation de tous les individus (équation 5):

$$p(i) = \frac{f(i)}{\sum_i^N f(i)} \qquad (5)$$

Par conséquent le nombre de génération a atteindre pour un individu i est N*p(i) où N est la taille de la population. La figure ci-dessous est une représentation de la sélection proportionnelle.

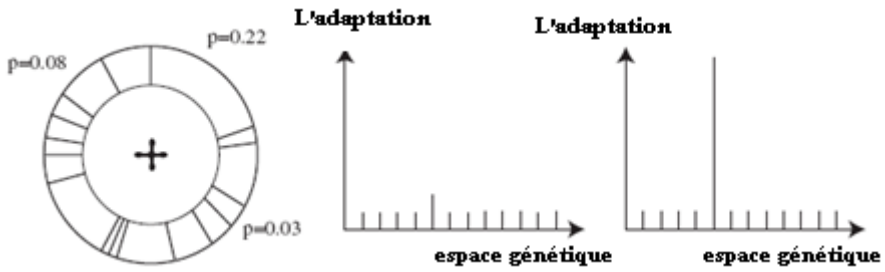


Figure3.14. Représentation de la sélection propositionnelle

Dans la figure 3.14 à gauche, la représentation de la roulette de sélection, au centre la représentation de la fitness de chaque individu et les individus ont la même fitness, à droite un individu domine la population La figure de la roulette est divisée en plages qui représente les probabilités de chaque individu dans la population, pour faire la sélection on tourne la roulette et la plage ou elle s'arrête nous prenons cet individu, l'opération est répétée N fois où N représente la taille de la population. Cette méthode a un inconvénient majeur, si la fitness des individus est pratiquement la même et si un des individus a une valeur d'adaptation nettement supérieure aux autres « il aurait la plus grande plage dans la roulette et la diversité est perdu ».

3.1.4.2. La Sélection basée sur les classes

Ce type de sélection ne soufre pas des inconvénients de la méthode précédente parce qu'il est basé sur la classe de l'individu dans la population au lieu de sa valeur d'adaptation. Elle consiste à classer les individus du meilleur au mauvais et attribuer une probabilité de reproduction à chaque individu proportionnel à sa classe. Par conséquent peu importe la différence fine entre les individus, le meilleur aura toujours une probabilité plus grande de se reproduire.

3.1.4.3. La sélection basée sur les classes tronquées

Ce type est une variation du type précédent qui consiste à prendre seulement les top n individus dans la classe et faire le même nombre de reproduction pour chaque individu sélectionné. Par exemple on pourrait prendre les 20 meilleurs individus d'une population de 100 et les reproduire 5 fois dans le but d'obtenir une nouvelle population. A condition que n ne soit pas trop petit (risque d'une convergence prématuré), cette méthode assure pour les individus qui ont obtenu une fitness relativement basse mais encore meilleure que la plus mauvaise, qu'ils auront un même nombre de reproduction que les meilleurs. Cette méthode est assez utile quand les individus ne peuvent pas être évalués exhaustivement et par conséquent la valeur de fitness de quelque individu ne reflète pas forcement leur valeur d'adaptation réelle.

3.1.4.4. La sélection basée sur le tournoi

Elle consiste à organiser un tournoi entre un groupe d'individu de sous ensemble de la population. Premièrement la sélection débute par le choix aléatoire de k individu où k est la taille du tournoi. L'individu avec la meilleure fitness parmi les k individus est sélectionné pour la reproduction. Tous les autres individus peuvent être sélectionnés une autre fois pour être dans un nouveau tournoi. Les tournois continus jusqu'à ce qu'une nouvelle population soit créé.

3.1.4.5. La sélection par remplacement général

Cette méthode est de loin la plus utilisée, la nouvelle génération produite remplace complètement l'ancienne. Cependant si l'espace de recherche est complexe où l'évolution de la fitness est très variante implique que les bons individus peuvent être perdus dans la génération future. Dans ce cas un remplacement de population stratégique est employé, nommé remplacement élitiste qui consiste à maintenir un certain nombre d'individus « les meilleurs » pour la prochaine population.

3.1.5. Les opérateurs génétiques

Les opérateurs génétiques simulent l'effet de la mutation biologique dans le génotype. Ces opérateurs sont conçus pour modifier le génotype mais pas sa taille. Autre types d'opérateurs génétiques peuvent modifier la taille du génotype comme la suppression, l'insertion et la duplication qui sont utilisés pour une représentation plus avancé pour faire évoluer un problème bien spécifique.
Les opérateurs génétiques introduisent la diversité dans la population et permettent l'exploration de nouvelles solutions.

3.1.5.1. Le croisement

Le croisement assure que les descendances héritent des caractéristiques des parents parce qu'on crée une paire de demi-génome et on fait une recombinaison.

Cette opération est connue sous le nom de recombinaison. La nouvelle descendance créée est le résultat d'une recombinaison d'un ensemble de paires de génotype divisés aléatoirement est échangés par l'opération de croisement selon une probabilité p. l'opération de croisement possède différents formes.

3.1.5.1.1. Le croisement par une seule fragmentation

Il peut être appliqué à la représentation réelle. Il consiste à faire une coupure aléatoire sur les deux génomes et faire le changement des deux fragment entre les génomes (figure 3.15.a). Le croisement par fragmentation multiple consiste à faire plusieurs coupures pour les deux génomes et puis faire le changement des fragments [Lam 07].

3.1.5.1.2. Le croisement uniforme

Ce type de croisement a le même principe que le croisement par fragmentation multiple mais la seule différence c'est que les fragments doivent avoir la même longueur (figure 3.15.b)

3.1.5.1.3. Le croisement arithmétique

Dans ce type de croisement, au lieu d'avoir deux génomes on obtient un seul. Le principe de ce type de croisement est de faire la moyenne entre chaque élément des deux génomes pour obtenir un troisième. Cette technique est utilisée généralement avec la représentation réelle (figure 3.15.c).

3.1.5.1.4. Le croisement par séquence

Ce type de croisement peut être utilisé dans les cas où le génome représente une séquence. Par exemple le problème du voyageur de commerce, où chaque génome doit contenir tous les codes des villes et sans répétition. Dans ce cas une contrainte est à respecter lors de l'opération du croisement. Le principe ici est que l'opérateur de croisement crée un seul génotype en prenant une partie du premier génome fragmenté et on lui ajoutant des fragments du deuxième génome (figure 3.15.d).

3.1.5.1.5. Croisement pour la représentation en arbre

Ce type de croisement est utilisé avec la représentation en arbre. Il consiste à faire une coupure aléatoire au niveau des nœuds de chaque génome et de faire un changement des parties fractionnées entre ces deux génomes pour obtenir deux nouveaux (figure 3.15.e).

3.1.5.2. La mutation

Comme le croisement la mutation est une autre forme pour créer la diversité. Elle opère au niveau de l'individu. La mutation est une petite modification du génotype qui permet l'évolution et l'exploration des solutions existantes. L'opération de mutation permet de parcourir tout l'espace de recherche pour atteindre une solution. La mutation est utile pour

éviter le problème du minimum local et atteindre davantage de progrès dans des situations où on a une grande couverture de la population. Ces performances ne peuvent être atteintes par l'opération de croisement. Cependant, le nombre et la taille de mutation doivent être relativement longs pour empêcher la perte des solutions découvertes auparavant. Plusieurs formes de mutation existent, nous les présentons ci-après.

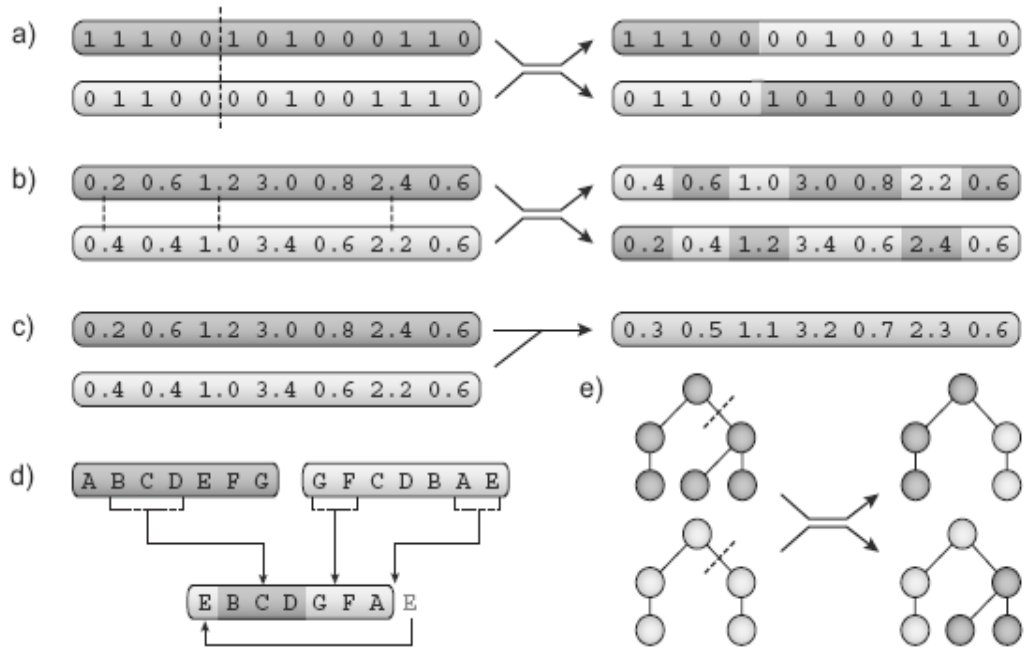


Figure 3.15. Représentation des opérations de croisement a)croisement par un seul point de fraction, b) croisement uniforme, c) croisement arithmétique, d) croisement par séquence, e) croisement pour la représentation en arbre.

3.1.5.2.1. La mutation des valeurs binaires

La mutation des valeurs binaires consiste à changer la valeur des bits sélectionnés (de zéro vers un et vis versa) comme dans la figure (3.16.a).

3.1.5.2.2. La mutation des valeurs réelles

Pour la représentation des valeurs réelles la mutation consiste à sélectionner les positions à modifier dans le génome et ajouter une valeur aléatoire extraite de la distribution de Gauss $N(0, \sigma)$, où 0 est la moyenne et σ est la variance (figure 3.16.b).

3.1.5.2.3. La mutation pour les séquences

Dans le cas de la mutation pour les séquences comme le problème du voyageur de commerce par exemple, elle consiste à faire un swap de deux positions aléatoirement choisi d'un génome (figure 3.16.c).

3.1.5.2.4. La mutation pour la représentation en arbre

Dans ce dernier cas la mutation consiste à faire un changement d'un nœud sélectionné avec un autre élément du même ensemble (figure 3.16.d). Si le nœud sélectionné est un terminal, il va être remplacé aléatoirement par un autre élément choisi parmi l'ensemble des terminaux.

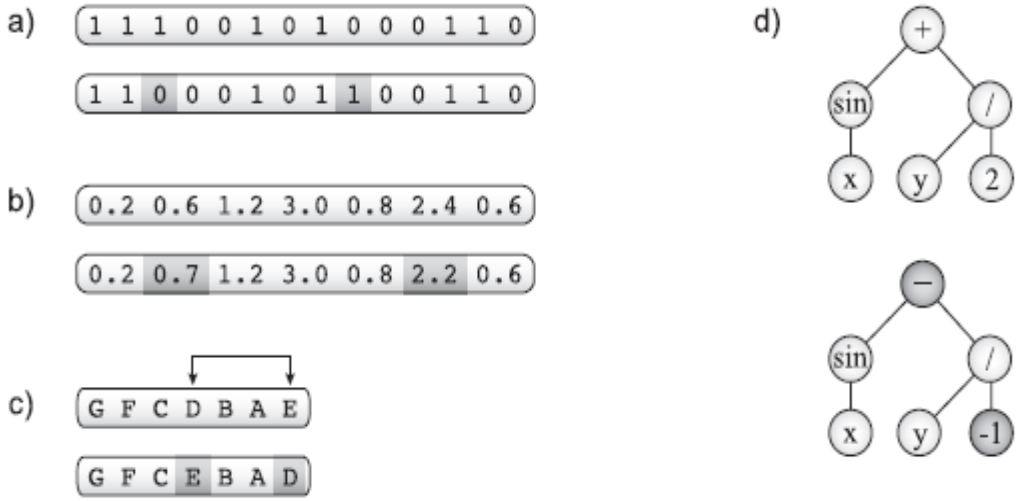


Figure 3.16. Représentation des opérations de mutation a) mutation des valeurs binaires, b) mutation des valeurs réelles, c) mutation pour les séquences, d) mutation pour la représentation en arbre.

3.1.6. Sélection pour le remplacement

Après avoir obtenu une population de λ enfants à partir de μ parents. Et avoir effectué les mutations sur les enfants, il faut une stratégie pour sélectionner les individus qui feront partie de la population de la génération suivante (c'est-à-dire à la prochaine itération). Plusieurs approches existent, nous allons en présenter quelques unes [Lam 09].

3.1.7. Remplacement générationnel

Pour ce cas de figure, il faut que le nombre d'enfants soit égal au nombre de parents ($\lambda = \mu$). La nouvelle génération est exclusivement composée des enfants et tous les parents sont remplacés. C'est la première approche proposée pour les algorithmes génétiques (et ce pour son analogie approximative avec l'évolution des êtres vivants). On peut aisément constater la possibilité que le meilleur individu disparaisse au profit de descendants moins performants, il est donc nécessaire de garder en mémoire le meilleur individu tout au long de la recherche, même si celui-ci ne participe pas au processus évolutif [Dre 05].

3.1.8. Remplacement déterministe

Cette méthode est celle proposée à l'origine pour les stratégies d'évolution. Dans ce cadre, λ est usuellement bien plus grand que μ. On choisit donc les μ meilleurs enfants (parmi λ) qui vont constituer la nouvelle population. Dans le cas où $\lambda = \mu$, cette méthode est équivalente au remplacement générationnel [Lam 09].

3.1.8.1. Remplacement stationnaire

Pour cette méthode, le nombre d'enfants est très petit (un ou deux), ils remplacent à chaque étapes le où les pires individus parmi les parents. Cela implique une convergence très lente. Cette méthode est utilisée dans l'*approche parisienne* des algorithmes évolutionnaires car elle permet une évolution graduelle et pas trop brusquée de la solution [Lam 09].

3.1.9. Remplacement élitiste

Le remplacement élitiste a pour principe de garder les parents qui sont plus performants que les enfants. L'approche la plus simple et de ne pas faire de distinction entre parents et enfants et de garder les μ meilleurs individus parmi les $\lambda + \mu$ individus de la population. On peut aussi garder le parent le plus performant pour remplacer le moins performant des enfants [Bel 08].

3.2. Les types d'algorithmes évolutifs

Comme nous l'avons dit plus haut, il existe plusieurs types d'algorithmes évolutifs qui diffèrent généralement par le choix de la représentation génétique et des opérateurs. Le choix optimal d'un algorithme, dépend des propriétés du problème à résoudre. En d'autres termes, il n'y a pas un algorithme meilleur que d'autres [Eib 03, Mic 96].

3.2.1. Les algorithmes génétiques

Ce type d'algorithmes utilise généralement la représentation binaire pour décrire les individus et met le point sur l'opération de croisement [Hol 75]. La programmation génétique fait partie aussi de la classe des algorithmes génétiques et elle utilise souvent la représentation en arbre pour représenté les individus surtout dans le domaine de la construction de circuits électroniques [Koz 92]. Une autre catégorie des algorithmes génétiques est la programmation évolutive qui utilise directement les paramètres qui définissent un phénotype et par une légère perturbation « mutation » selon la distribution de Gauss fait évoluer la population d'individus. La programmation évolutive utilise souvent la sélection basée sur le tournoi et n'utilise pas de croisement [Fog 66].

3.2.2. Le modèle Island

Ce type d'algorithmes maintien un ensemble de populations qui évolue en parallèle et échange des individus à chaque génération. L'idée ici est de laisser l'évolution se poursuivre dans différents parcours et dans différentes populations en maintenant la diversité dans les populations et en exploitant les synergies potentielles par un changement d'individus entre les populations [Whi 98]. Le modèle Island est approprié pour une implémentation parallèle pour un problème linéairement séparable ou les meilleures solutions découvertes dans les différentes populations peuvent être recombinés. Les paramètres essentiels dans ce modèle sont le nombre de population et le nombre et la fréquence de migration des individus à travers les populations.

3.2.3. L'évolution d'états stables

Dans ce type d'algorithmes les individus qui obtiennent la mauvaise fitness sont remplacés par les descendances des individus qui ont obtenu les meilleures valeurs de fitness [Sys 89]. L'idée ici est de maintenir à l'intérieur de la population les meilleures solutions trouvées et d'ajouter toujours les meilleurs individus. Il a été prouvé par les tests que ce type d'algorithmes est approprié lors de l'utilisation d'une petite population.

3.2.4. Le recuit simulé

C'est une fonction d'optimisation qui se base sur une perturbation aléatoire des candidats et une décision sur une certaine probabilité de retenir les solutions qui ont été engendrées « après l'opération de mutation ». Le recuit simulé prend son inspiration du processus de refroidissement progressif du métal. Dans la simulation le métal est le candidat « une des solutions » et ses paramètres sont aléatoirement initialisés [Kir 83]. Ce candidat va subir une mutation et si son énergie « équivalente à l'inverse de la valeur de fitness » est basse par rapport à l'étape précédente, le nouveau candidat remplace l'ancien. L'opération s'arrête quand la température du recuit est proche de zéro. La différence entre les algorithmes génétiques et le recuit simulé est que le premier opère sur une population d'individus et l'autre sur un seul individu.

3.2.5. Les populations basées sur l'apprentissage incrémental

Ce type opère sur une seule chaîne de caractère génétique qui représente la totalité des individus de la population. L'algorithme suppose que les génotypes de touts les individus ont la même taille et sont utilisés en codage binaire. Pi représente la probabilité de trouvée un 1 à la position i du génotype. Généralement à l'initialisation de la population tous les Pi sont égaux à 0,5, on sélectionne les individus qui ont la meilleure valeur de fitness pour la reproduction et en recalculant les nouveaux Pi, une nouvelle population est crée par échantillonnage. On continue ainsi jusqu'à ce qu'on atteigne un critère d'arrêt prédéterminé [Bal 95].

4. Le connexionnisme

L'être humain possède la capacité de mémoriser, d'apprendre, de raisonner et de calculer. Ces capacités sont supportées par son cerveau. Les enfants, par exemple, sont capables de distinguer les couleurs, les contours, les nuances, etc. Ces tâches sont triviales pour les enfants, cependant, la réalisation de telles tâches par une machine demeure très complexe, voir impossible actuellement. Le cerveau qui donne aux enfants cette puissance de raisonnement et de calcul est composé d'un ensemble d'unités (neurones) reliées entre elles par des connexions. Sachant que la puissance de calcul d'une unité biologique est beaucoup plus faible que celle d'un micro-processeur de dernière génération, la question qui se pose, comment est-ce possible qu'un réseau d'unités disposant d'une capacité de calcul faible puisse effectuer des tâches aussi complexes ?

Les études ont montré que le système nerveux est constitué de neurones dont chacun peut avoir plusieurs milliers de connexions avec d'autres neurones, on réalise alors la complexité des réseaux nerveux, supports matériels de l'ensemble de nos pensées, de nos comportements et de nos raisonnements. Les scientifiques ont déduit que c'est grâce aux connexions que le système nerveux peut réaliser des tâches complexes. C'est en se basant sur ces résultats que les approches connexionnistes ont été fondées.

Le principe du connexionnisme repose sur le fait qu'une activité collective d'unités simples qui s'influencent mutuellement peut engendrer un comportement varié et complexe. Le but du

connexionnisme est de rendre compte de la cognition humaine par les réseaux de neurones artificiels.

Les réseaux neuronaux sont des modèles conçus pour simuler le fonctionnement du cerveau. Ils consistent en un réseau de petits nœuds qui fonctionnent ensemble afin de former des schémas d'information. Ils ont un potentiel énorme et semblent actuellement rencontrer un grand succès dans plusieurs domaines de l'intelligence artificiel.

4.1. Du Neurone Biologique au Neurone Formel

Un neurone biologique est une cellule nerveuse dont la fonction est de transmettre un signal électrique dans certaines conditions (Figure 3.17). Le corps d'un neurone (soma) est relié d'une part à un ensemble de dendrites (entrées du neurone) et d'autre part à un axone, partie étirée de la cellule, qui représente sa sortie. Le neurone est généralement connecté aux neurones qui l'environnent par l'intermédiaire de synapses (jonction axone/dendrite entre deux neurones). Il reçoit au niveau de ses dendrites les signaux électriques des neurones en amont (ou en provenance de l'environnement extérieur), propagés par les axones de ces derniers. Les charges électriques s'accumulent dans le neurone jusqu'à dépasser un certain seuil, et à ce moment la transmission du signal électrique se déclenche via son axone vers d'autres neurones en aval. Si, au contraire, les signaux électriques ne dépassent pas le seuil, la transmission se bloque.

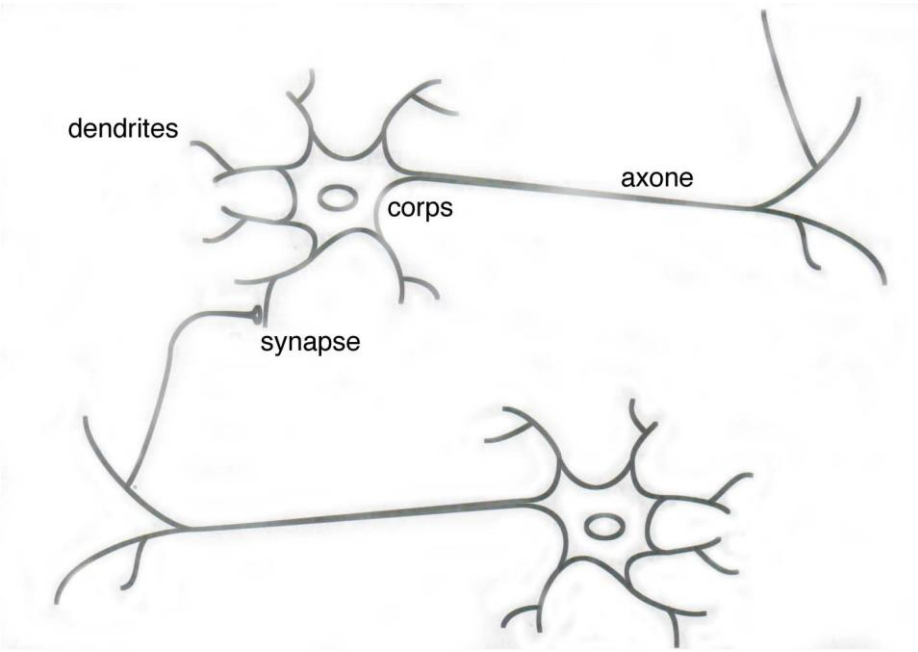


Figure 3.17. Neurone biologique.

Il est à remarquer que les connexions synaptiques entre deux neurones ne sont pas toutes de la même efficacité et de la même qualité. Ainsi l'entrée associée à une certaine dendrite du neurone pourra avoir plus d'influence qu'une autre sur la valeur de sortie, en d'autres termes, les connexions synaptiques ont un effet important sur le fonctionnement et la valeur de sortie

des neurones (notion de connexionnisme [Cou 07]). On peut représenter donc l'efficacité d'une connexion synaptique par un poids *W*, sorte de coefficient s'appliquant au signal d'entrée. Un poids positif aura tendance à accentuer une entrée (synapse excitatrice), tandis qu'un poids négatif viendra l'inhiber (synapse inhibitrice).

Pour imiter le neurone biologique, le neuropsychiatre McCulloch et le logicien Pitts ont fait appel à un modèle mathématique de base (i.e. un neurone formel) illustré à la figure 3.18 [Mcc 43].

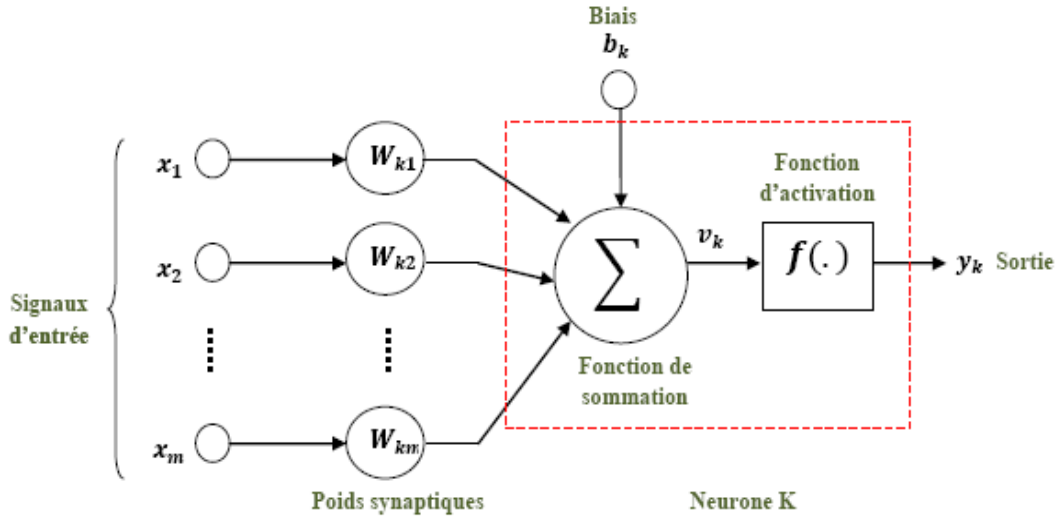


Figure 3.18. Neurone formel

Un neurone formel est une unité élémentaire de traitement qui forme la base des réseaux de neurones artificiels. On peut identifier trois éléments de base qui caractérisent un modèle de neurones [Hay 99]:

- Un ensemble de synapses (ou connexions), dont chacun est caractérisé par un poids $w_{k,j}$ où k représente le neurone en question, et j représente le neurone source du signal x_j (i.e. neurone d'entrée). Les poids synaptiques d'un neurone artificiel peuvent avoir des valeurs positives ou négatives.
- Un intégrateur (ou fonction de sommation) qui effectue la somme pondérée des signaux d'entrée. la pondération consiste à multiplier les signaux d'entrée par les poids synaptiques correspondants.
- Une fonction d'activation (ou fonction de transfert) pour limiter l'amplitude de la sortie d'un neurone. Typiquement, la sortie d'un neurone prend sa valeur dans l'intervalle [0,1] ou alternativement [-1,1].

Le neurone k illustré à la figure 18 effectue une somme des entrées x_1, x_2 , … , x_m qui lui parviennent, pondérées par les poids $w_{k,1}$, $w_{k,2}$, … , $w_{k,m}$ correspondants. Formellement, on peut décrire un neurone k en utilisant les deux équations suivantes :

$$u_k = \sum_{j=1}^{m} w_{k,j}\ x_j \qquad (6)$$

$$= w_{k,1}\ x_1 + w_{k,2}\ x_2 + \cdots + w_{k,m}\ x_m$$

Et

$$y_k = f(v_k + b_k) \qquad (7)$$

$\boldsymbol{u_k}$ est la sortie de l'unité de sommation. On ajoute éventuellement à cette sortie une valeur de référence, qui représente le *seuil d'activation* de neurone, notée par $\boldsymbol{\theta_k}$. Le résultat $\boldsymbol{v_k}$ décrit par l'équation (8), va ensuite être transformé par une fonction d'activation f qui produit la sortie $\boldsymbol{y_k}$ du neurone. $\boldsymbol{y_k}$ forme ce qu'on appele le niveau d'activation de neurone (ou le potentiel du neurone). Les entrées x_j et les poids synaptiques $w_{k,j}$ sont des valeurs réelles.

$$v_k = n_k + b_k \qquad (8)$$

Dans un neurone biologique, le seuil θ a une valeur négative. Cependant, dans un neurone formel, θ peut avoir une valeur positive, si c'est le cas, le terme *biais*, noté par b_k , est usuellement utilisé. Le seuil (ou biais) a l'effet d'augmenter ou diminuer la valeur de sommation des entrées $\boldsymbol{n_k}$. Il est considéré comme un paramètre extérieur du neurone artificiel. Pour des raisons de simplification, le seuil peut être inclus dans la somme des entrées en ajoutant une entrée supplémentaire x_0 (généralement fixée à 1), pondérée par un poids $w_{k,0}$ tel que $\boldsymbol{b_k} = w_{k,0}\ x_0$ (figure 3.19). La formule d'activation et la combinaison de l'équation (6) et (8) deviennent alors :

$$v_k = \sum_{j=0}^{m} w_{k,j}\ x_j \qquad (9)$$

Et

$$y_k = f(v_k) \qquad (10)$$

Les deux modèles illustrés à la figure 3.18 et 3.19 ont une apparence différente mais ils sont mathématiquement équivalents.

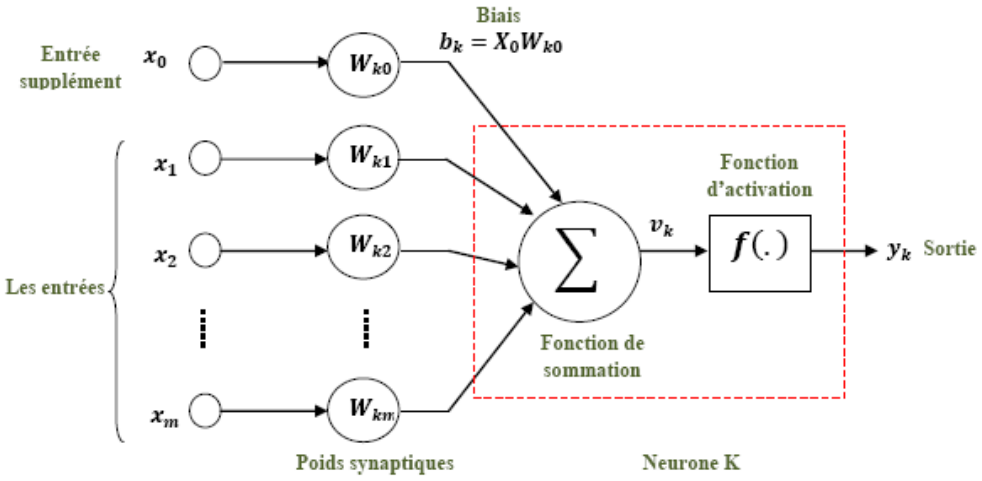


Figure 3.19. Neurone formel avec entrée supplémentaire

On peut aussi récrire l'équation (6) sous forme vectorielle :

$$u_k = W_k{}^T \mathrm{X}. \qquad (11)$$

$W_k = [w_{k,1}\ w_{k,2}\ \ldots w_{k,m}\ ]^T$ représente le vecteur des poids du neurone, tandis que $\mathrm{X} = [x_1\ x_2\ \ldots x_m\ ]^T$ représente le vecteur des *m* entrées du neurone. On multiplie donc chaque valeur d'entrée x_j par la composante des poids correspondante $w_{k,j}$ (produit scalaire) et on ajoute le biais.

On combinant la formule (11) et (7) on obtient:

$$y_k = f\left(W_k{}^T \mathrm{X} + b_k\right). \qquad (12)$$

Il est possible aussi d'adopter une forme générale en remplaçant W^T par une matrice $\mathrm{W_k} = W_k{}^T$ d'une seule ligne :

$$y_k = f(\mathrm{W_k X} + b_k). \qquad (13)$$

Malgré sa structure simple, le neurone de McCulloch et Pitts représente un outil puissant de calcul. Ils ont démontré qu'un assemblage synchrone de ces neurones permet, en principe, de réaliser n'importe quel calcul effectué ordinairement par un ordinateur [Mcc 43].

Initialement, le neurone formel proposé par McCulloch et Pitts était implémenté avec une fonction d'activation de type seuil. Mais il a été rapidement généralisé de différentes manières. Différents types de fonctions peuvent donc être utilisés comme des fonctions d'activation (tableau 3.1). Les fonctions seuil, linéaire et sigmoïde sont les plus souvent utilisées.

La fonction seuil permet de fournir une sortie comprise entre 0 et 1 en appliquant un seuil sur la somme pondérée des entrées. Si Cette somme dépasse le seuil alors la fonction retourne la valeur 1 (équivalant à l'activation du neurone), sinon elle renvoie 0 (le neurone est au repos). Cette fonction permet alors de prendre des décisions binaires.

La fonction de transfert linéaire est plus simple, elle affecte directement son entrée à sa sortie (y= v). Dans le contexte d'un neurone, l'application de cette fonction signifie que la sortie du neurone corresponde à son niveau d'activation.

La fonction sigmoïde est une fonction croissante, continue et bornée. À l'opposé de la fonction seuil (ou n'importe quelle fonction à échelon) dont la sortie délivrée est une valeur binaire, cette fonction permet de fournir une valeur réelle. Dans les problèmes où la sortie désirée doit être une valeur réelle, la fonction sigmoïde est convenable et permet de répondre aux contraintes exprimées [Bla 96].

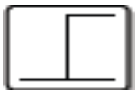
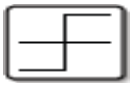
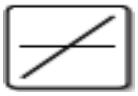
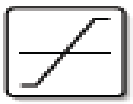
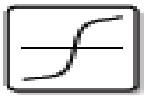
Nom de la fonction	Relation d'entrée/sortie	Représentation
seuil	a = 0 si n < 0 a = 1 si n 0	
Seuil symétrique	a = −1 si n < 0 a = 0 si n = 0 a = 1 si n 0	
Linéaire	a = n	
linéaire saturée	a = 0 si n < 0 a = n si 0 n 1 a = 1 si n > 1	
linéaire saturée symétrique	a = −1 si n < −1 a = n si − 1 n 1 a = 1 si n > 1	
linéaire positif	a = 0 si n < 0 a = n si n 0	
Sigmoïde	a = $\frac{1}{1+exp^{-n}}$	
Tangente Hyperbolique	a = $\frac{e^n - e^{-n}}{e^n + e^{-n}}$	
Compétitive	a = 1 si n maximum a = 0 autrement	

Table 3.1. Fonctions de transfert y =f(v).

4.2. Réseau de neurones artificiels

Un réseau de neurones artificiel (RNA) est constitué de plusieurs neurones fortement interconnectés et fonctionnant en parallèle, qui réalise chacun un traitement simple indépendamment des autres, mais dont l'ensemble fait émerger des propriétés globales dignes d'intérêt, cela signifie que les capacités du réseau excèdent celles de ses éléments [Bla 96].

Les neurones constituant un RNA sont organisés souvent en couches. Un RNA peut contenir un ou plusieurs couches et chaque couche peut contenir S neurones (où Sϵℕ). On peut

distinguer la couche d'entrée qui recevra les données en entrée, les couches cachées et la couche de sortie qui donne le résultat obtenu après le traitement de données. Les couches cachées sont les couches qui se situent entre la couche d'entrée et la couche de sortie. Un réseau de neurones peut donc être représenté par un graphe dirigé et pondéré dont les nœuds sont des unités de calcul (neurones formels), et dont les arcs pondérés sont des poids synaptiques (figure 3.20).

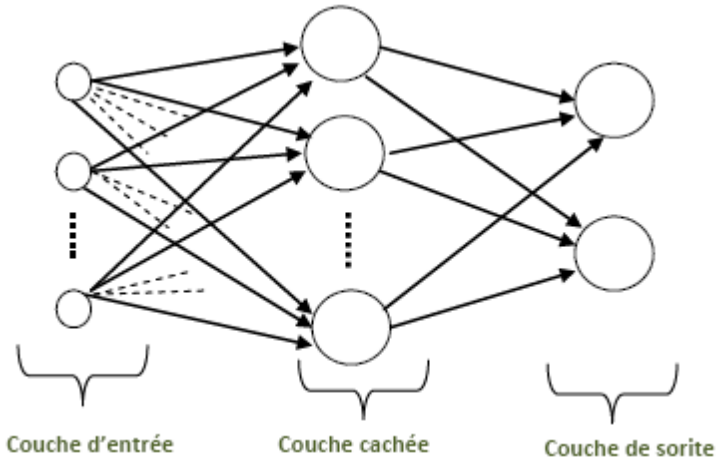


Figure 3.20. Réseau de neurones artificiels.

Les réseaux de neurones artificiels sont dérivés de leurs correspondants en biologie. Ces derniers se caractérisent par des propriétés leur permettant d'accomplir des tâches spécifiques et complexes que les approches actuelles ont de réelles difficultés à accomplir (e.g. reconnaissance des formes, compréhension d'une langue) [Ben 07]. La topologie d'interconnexion des neurones, l'efficacité des connexions synaptiques, les stimuli provenant de l'environnement, etc. sont des facteurs qui jouent un rôle fondamental dans le fonctionnement de notre système nerveux. En effet, la complexité d'un cerveau humain ne peut pas être reproduite, les RNAs ne représentent qu'une abstraction de haut niveau de cette complexité.

Le fait que les RNAs soient des approches bio-inspirées basées sur le fonctionnement du cerveau humain rend ces réseaux un outil puissant de calcul qui se caractérise par des propriétés très intéressantes, dont nous décrivons brièvement les plus importantes [Aza 00].

- **Apprentissage et adaptabilité.** L'apprentissage est le processus d'adaptation des paramètres d'un système pour donner une réponse désirée à une entrée ou stimulation quelconque [Tor 97]. Dans les RNA, l'ajustement des paramètres est généralement réalisé de façon itérative, en minimisant une mesure de l'erreur qui représente la différence entre la réponse obtenue et la réponse correcte, jusqu'à ce que le réseau élève puisse imiter aussi bien que possible le professeur. Un réseau de neurones entrainé à un environnement spécifique, peut facilement être réentrainé pour s'adapter aux modifications de son environnement [Hay 99]. Ainsi, un réseau de neurones qui

opère dans un environnement instable (i.e. les conditions environnementales changent continuellement à travers le temps), peut être conçu pour changer ses poids synaptiques en temps réel.

- **Parallélisme.** Il est bien entendu que les neurones constituant un RNA peuvent effectuer un calcul simultanément [Ben 07]. C'est une propriété importante du point de vue implémentation. Puisqu'il est difficile d'accélérer le fonctionnement d'une unité de calcul (i.e. un neurone formel), l'idée est de distribuer les tâches complexes à un grands nombre de telles unités travaillant en parallèle. Il faut noter aussi qu'il est possible que la réalisation de certaines tâches par des réseaux de neurones soit de manière séquentielle où à chaque top horloge un seul neurone effectue le calcul [Zem 03].
- **Mémoire distribuée.** La mémoire permet l'acquisition, le stockage et la restitution d'informations. Dans les RNAs, un élément de mémoire n'est pas physiquement localisable comme dans la mémoire d'un ordinateur ordinaire, mais il est distribué sur tout le réseau [Bel 92]. En effet, les informations acquises par les RNAs sont entreposées et encodées dans la force des connections (i.e. poids synaptiques) [Ben 07].
- **Tolérance aux fautes.** L'une des propriétés intéressantes des RNA est leur capacité de tolérance aux fautes. Cette capacité est due à la nature répartie de l'information sur toutes les connexions ainsi que le traitement parallèle effectué par les RNA. En conséquence, si un neurone ou ses connexions sont endommagés, le réseau exhibe une dégradation de performance (Gracefull degradation) plutôt qu'une défaillance catastrophique [Hay 99]. Ainsi, si une petite partie des données en entrée manque ou est déformée, la performance se dégrade seulement légèrement. Cette dégradation est due à l'inexactitude ou l'incomplétude des données [Bur 91].
- **Généralisation.** La généralisation peut être définie par la capacité d'élargir les connaissances acquises après apprentissage à des données nouvellement rencontrées par le réseau de neurones [Zem 03]. Cette capacité est fondée essentiellement sur la similitude de nouvelles situations à celles que le système a rencontré précédemment. Les systèmes connexionnistes permettent de reconnaître des formes approximatives et de les comparer à des patterns appris. Si un stimulus n'est pas trop éloigné de la forme apprise durant la phase d'apprentissage, alors il sera reconnu comme étant cette forme. Un RNA qui possède une bonne généralisation réagit correctement lorsque des stimuli, qui n'ont pas été vus lors de l'apprentissage, sont présentés. Les réseaux de neurones artificiels sont convenables pour les problèmes où la compréhension est limitée ou incomplète et où existe une abondance de données [Mes 06].
- **Non-linéarité.** Un neurone artificiel peut être linéaire ou non-linéaire selon la fonction d'activation adoptée. Cependant, un réseau de neurones constitué d'une interconnexion de neurones linéaires est non-linéaire [Hay 99]. Cette propriété est très importante car la majorité des problèmes rencontrés ont souvent une nature non-linéaire.

Les chercheurs ont essayé de construire des modèles qui exhibent ces propriétés et qui possèdent ces capacités. Cette tendance a contribué à la réalisation d'une grande quantité de modèles connexionnistes qui possèdent tout ou au moins une partie des propriétés désirées.

Les applications des réseaux de neurones sont très nombreuses et touchent plusieurs domaines. Parmi ces applications, nous citons :

- **Reconnaissance.** Un réseau peut servir à reconnaître des caractères. Cela est déjà utilisé à la Poste pour lire les codes postaux, ou même dans certaines banques pour lire les chèques. Il est aussi possible de retrouver la prononciation des mots à partir d'un texte.
- **Prévision.** On utilise de plus en plus les réseaux de neurones artificiels pour faire des prévisions en marketing (prédiction de comportement, de possibilité de vente d'un produit, etc.) ou pour le trafic routier, etc. Mais les prévisions en météo ou en bourse sont trop compliquées à réaliser.
- **Contrôle.** Par exemple, on peut contrôler la qualité des produits dans une industrie.
- **Approximation de fonction.** Les fonctions trop compliquées peuvent être approximées, grâce au réseau, par une somme de fonctions plus simples comme des polynômes ou des sigmoïdes.
- **Robotique.** Certains robots sont dotés de réseaux de neurones. Des entreprises japonaises les utilisent déjà pour les produits électroménagers et informatiques.
- **Optimisation de trajectoires.** En intégrant les paramètres tels que le vent, les conditions extérieures, ... On peut, par exemple, déterminer quelle est la meilleure trajectoire pour un avion, une fusée, etc.
- **La commande de processus.** Commander un processus, c'est imposer à celui-ci un comportement défini à l'avance en fonction des signaux de commandes; l'ensemble, commande plus processus peut donc être considéré comme un système qui réalise une fonction (non-linéaire) qu'un réseau de neurones peut approcher.
- **La classification.** Les réseaux de neurones artificiels sont des bons candidats pour réaliser des tâches de classification.

Nous n'avons représenté qu'un sous ensemble représentatif des applications possibles des réseaux de neurones. Ils ont encore beaucoup d'autres usages.

4.3. Classification des approches connexionnistes

Devant la diversité des approches connexionnistes [San 98], il devient nécessaire de faire une classification de ces approches selon leurs propriétés afin de parvenir à une bonne compréhension de la différence entre eux et à un bon choix d'une approche orientée vers un objectif spécifique.

La classification des approches connexionnistes existantes que nous adoptons dans notre travail, est restreinte et se base essentiellement sur quelques attributs et critères spécifiques tels que le type d'apprentissage et l'architecture d'interconnexion des neurones. Cette classification va permettre de comparer les différentes approches pour arriver à définir les

avantages et les inconvénients de leurs utilisations pour résoudre tel ou tel problème, ainsi que les solutions possibles pour remplir leurs limitations (e.g. modèles hybrides [San 98]).

4.3.1. Type d'apprentissage

La capacité d'apprentissage est une particularité désirable et puissante du réseau de neurones. Elle représente un processus dynamique et itératif permettant de modifier les paramètres d'un réseau (les poids de ses liaisons) en réaction aux stimuli qu'il reçoit de son environnement (les valeurs d'entrée). L'apprentissage permet d'améliorer les performances du réseau (i.e. en fonction d'une certaine entrée, il offre la bonne sortie) car à chaque erreur qu'il fait, il subit une correction qui le fait réagir différemment s'il est confronté à la même situation. Les règles permettant de réaliser un tel processus d'adaptation constitue ce qu'on appelle l'algorithme d'apprentissage du réseau. La première théorie fondamentale pour l'apprentissage a été proposée en 1949 par Donald Hebb [Hebb 49], qui décrit dans son ouvrage « The Organization of Behaviour » une règle simple qui consiste à renforcer ou à affaiblir les connexions en fonction de l'activité des neurones qu'elles relient.

Aujourd'hui, il existe au moins 23 types de règles d'apprentissage [Has 95]. Les approches d'apprentissage connexionnistes peuvent être classées selon le degré de contrôle donné à l'opérateur en trois catégories:

4.3.2. Apprentissage Supervisé

Un professeur fournit un grand nombre de couples de référence précis de type {entrée, sortie désirée pour cette entrée} dans le but d'habituer le réseau à se comporter correctement à l'apparition de chaque stimulus. Le réseau sera donc capable de mesurer la différence entre son comportement actuel et le comportement de référence, de produire un signal d'erreur, et de corriger ses paramètres via une procédure itérative de façon à réduire cette erreur. Les RNAs usuels qui adoptent ce type d'apprentissage sont le perceptron multicouches (PMC) et le réseau à fonction de base radiale (RBR).

4.3.2.1. Apprentissage par renforcement

L'apprentissage par renforcement ou parfois aussi dite semi-supervisé, ressemble à l'apprentissage supervisé sauf que dans le premier, l'utilisateur ne possède que des indications imprécises (e.g. échec/succès du réseau) sur le comportement final désiré. Ce type d'apprentissage ne dispose pas d'un signale d'erreur, ce qui rend la prise de décision et le changement des paramètres du réseau très difficile. Dans ce cas, un processus d'essais et d'erreurs doit être implanté et la tâche d'apprentissage doit passer par une étape d'exploration où l'on essaie des directions aléatoires de changement, et une étape d'exploitation où l'on prend une décision. Cependant, ce processus en deux étapes peut retarder considérablement l'apprentissage. De plus, l'usage de ce type d'apprentissage est difficile à mettre en œuvre.

4.3.2.2. Apprentissage non supervisé

Il se caractérise par l'absence du professeur, donc l'absence d'un signal d'erreur. Le réseau doit apprendre par lui-même à créer des classes de stimuli similaires pour les entrées qui proviennent de l'environnement en recherchant des critères de ressemblance (i.e. élaborer une

représentation interne de l'espace des données d'entrée et catégoriser les données en détectant les similarités et les différences à partir de la représentation interne) [Vir 01]. Les réseaux à apprentissage non supervisé les plus utilisés et étudiés sont les cartes de Kohonen [Gat 07] L'avantage de ce type de réseaux réside dans sa capacité d'adaptation reconnue comme une auto-organisation (self-organizing) [Koh 87].

L'apprentissage non supervisé est surtout utilisé pour la reconnaissance des formes, le traitement du signal et l'analyse factorielle.

Généralement, l'apprentissage connexionniste nécessite une grande quantité de données (exemples d'apprentissage). Une fois la phase d'apprentissage terminée, une étape de validation peut être effectuée en fournissant un ensemble de données non encore présentées pour obtenir une généralisation [Gat 07]. Cette étape permet à la fois d'évaluer les performances du système connexionniste et de détecter le type de données qui posent des problèmes. Si les performances ne sont pas satisfaisantes, il faudra soit modifier la base d'apprentissage, soit modifier l'architecture du réseau.

4.3.3. Topologie du réseau

Les réseaux de neurones peuvent être classés selon la manière dont les neurones sont interconnectés. Plusieurs architectures différentes de RNAs ont été étudiées ces dernières années [Pre 96]. Chaque architecture possède des particularités propres et peut servir à des applications spécifiques. Malgré la diversité de leurs architectures, les réseaux de neurones peuvent être regroupés en deux formes principales de topologie:

4.3.3.1. Réseaux non bouclé

Feedforward neural network, les neurones dans ces réseaux sont structurés en couches. Les neurones qui reçoivent les stimuli de l'environnement sont les neurones d'entrées qui appartiennent à la couche d'entrée, tandis que les neurones qui effectuent les dernières opérations sont appelés neurones de sortie et appartiennent à la couches de sortie. Les autres neurones, placés entre les deux couches précédentes, sont en nombre variable et appartiennent à des couches cachées dont le nombre est déterminé en fonction de la puissance de calcul nécessaire et la fonction désiré. Généralement, le nombre de couches cachées ne dépasse pas une, voir deux. La structure générale d'un réseau feedforward consiste donc en une couche d'entrée, une ou plusieurs couches cachées et une couche de sortie. Le nombre de neurones dans chaque couche ainsi que le nombre de couches cachées seront choisis selon plusieurs critères. Parmi ces critères on cite :

- La tâche à effectuer qui joue un rôle principal dans le choix du nombre de couches cachées, le nombre de neurones dans la couche de sortie et dans la couche d'entrée. Par exemple, pour un réseau utilisé pour classifier un ensemble d'objet en groupes, on va avoir autant de neurones de sortie que les groupes (chaque neurone représente un groupe). Si le réseau est destiné à reconnaître des caractères, le nombre de neurones d'entrée sera égale au nombre de pixels utilisé pour représenter n'importe quel caractère présenté au réseau. Ainsi, le nombre de neurones de sortie sera fixé en fonction de nombre de caractères appris par le réseau. Le nombre de couches cachées

ne dépasse pas une, voir deux, car les réseaux ayant deux couches cachées peuvent représenter toute forme de fonctions.

- La complexité et la quantité de données présentées en entrée qui peuvent intervenir dans le choix du nombre de neurones cachés. Un nombre réduit de neurones peut engendrer un mauvais traitement des données (underfitting) tandis qu'un nombre assez grand de neurones cachés peut conduire à un mauvais apprentissage parce que les informations contenues dans l'ensemble d'apprentissage sont insuffisantes pour faire apprendre la tâche désignée à tous les neurones cachés (overfitting).
- Le temps d'apprentissage qui est également l'un des points pris en considération. Un nombre grand de neurones peut augmenter le temps d'apprentissage lorsque les données d'apprentissages sont suffisantes. Cette incrémentation engendre un mauvais apprentissage pour le réseau.

La caractéristique principale des réseaux non bouclés est l'absence de rétroaction, c'est-à-dire que chaque neurone d'une couche reçois ses entrées provenant de la couche précédente et envoie ses sorties vers la couche suivante sans aucune rétroaction (i.e. sans retour en arrière) [Dre 04]. Un cas particulier de cette classe de réseaux est le RNA *à* une couche (Single layered network). C'est un réseau constitué d'une couche d'entrée composée de m neurones, une couche de sortie constituée de S neurones, et les connexions pondérées entre-elles (voir figure 3.21).

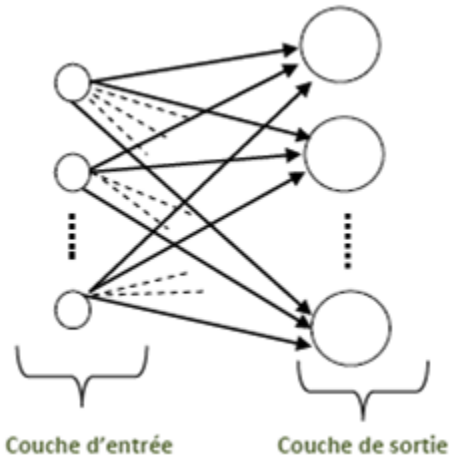


Figure 3.21. Réseau à une couche.

On désigne par une couche la couche de sortie qui contient les neurones opérationnels. On ne compte pas la couche d'entrée parce que aucune opération n'est effectuée au niveau de cette couche (i.e. elle est considérée comme l'interface du réseau avec l'environnement extérieur) [Hay 99]. Les S neurones de la couche de sortie sont connectées directement aux m entrées (i.e. un réseau totalement connecté). Le Perceptron et l'Adaline (ADAptive LInear NEuron) sont deux exemples très connus de ce type de réseaux. On dit qu'un RNA est totalement connecté dans le cas où chaque neurone dans chaque couche du réseau est connecté avec tous les neurones de la couche adjacente suivante. Dans le cas où quelques liens de connexion manquent, on dit que le réseau est partiellement connecté.

4.3.3.2. Réseaux bouclés

Connus aussi sous le nom de réseaux récurrents (figure 3.22). Ils peuvent avoir une ou plusieurs couches comme les réseaux non bouclés, mais la différence entre les deux se situe dans la présence d'au moins une connexion qui relie la sortie d'un neurone avec l'entrée d'un autre neurone d'une même couche ou d'une couche inférieur (on parle d'un feedback). Il est possible aussi d'avoir une connexion qui relie la sortie d'un neurone avec son entrée. Les réseaux bouclés sont appropriés pour la modélisation des comportements dynamiques et des aspects temporels, et nécessite des algorithmes particuliers pour l'apprentissage. On peut citer comme exemples, les réseaux TDNN (pour Time Delay Neural Networks), réseaux de Jordan, les réseaux SRN (Simple Recurrent Network), modèle de Hopfield, etc.

Les réseaux de neurones peuvent être aussi classés selon l'évolution de leurs architectures durant l'apprentissage. En fonction de ce critère, on distingue les deux classes suivantes :
Réseaux statiques. Avant la tâche d'apprentissage, la structure de ces réseaux est définie préalablement, c'est-à-dire que le nombre de neurones et la structure d'interconnexions sont statiques et ne changent pas lors de l'apprentissage. Ce type de réseaux a quelques limitations : la spécification de la topologie de réseau et le nombre de neurones appropriés pour résoudre un problème donné dès le début, est presque impossible. Ainsi, il n'existe pas de méthode optimale pour fixer le nombre exacte d'unités à employer.

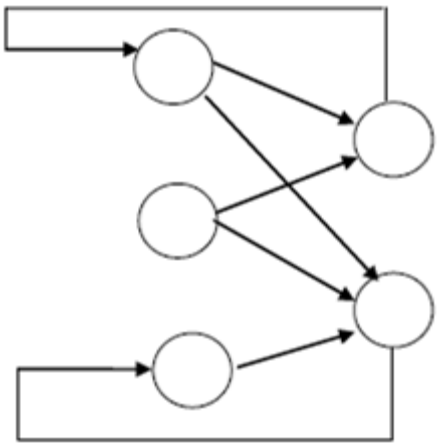


Figure 3.22 Réseau bouclé (récurent).

4.3.3.3. Réseaux dynamiques

Le nombre de neurones et la structure d'interconnexions peut varier dans les réseaux possédant une topologie dynamique (réseaux ontogéniques) [Bur 91]. Les changements effectués dans la structure des réseaux ontogéniques peuvent être un ajout ou une suppression de neurones et d'interconnexions pour arriver à résoudre un problème donné.

IL reste à noter que les neurones formels peuvent aussi être classés selon la fonction de transfert interne utilisée pour calculer l'activation. Le type de fonction d'activation employé (e.g. linéaire, seuil, sigmoïde, tangente Hyperbolique, etc.) joue un rôle principal dans la

détermination de la fonctionnalité de chaque neurone et, par conséquent, la détermination du type de réseau.

4.4. Réseaux de neurones usuels

Dans cette section, on décrit quelques réseaux de neurones parmi les plus utilisés. Nous commençons par le perceptron multicouches qui est largement utilisé dans des problèmes complexes (e.g. reconnaissance de chiffres). Ensuite, nous présentons brièvement le réseau à fonction de base radiale qui est un réseau non bouclé à apprentissage supervisé, le réseau de Khonen qui est un réseau bouclé et le réseau de Hoperfield.

4.4.1. L'approche du perceptron multicouche (PMC)

Le perceptron multicouche est un réseau de neurone parmi les plus utilisés surtout pour des problèmes d'approximation, de classification et de prédiction [Wei 91]. Il est généralement constitué de deux ou trois couches de neurones totalement connectés. Un perceptron multicouches a trois caractéristiques principales [Hay 99]:

- Le modèle de chaque neurone dans le réseau inclut une fonction d'activation non linéaire (en excluant les neurones d'entrée). La fonction usuellement utilisée pour satisfaire la non-linéarité est la fonction sigmoïde. C'est une caractéristique importante qui permet au perceptron multicouche de résoudre des problèmes complexes.
- Le réseau contient une ou plusieurs couches cachées lui permettant d'apprendre des tâches complexes par l'extraction des caractéristiques signifiantes de l'ensemble des entrées.
- Le réseau exhibe un degré de connectivité élevé, déterminé par les synapses du réseau. Des changements dans la connectivité du réseau impliquent des changements, soit des connexions synaptiques, soit de leurs poids.

Par la combinaison de ces caractéristiques, le perceptron multicouches devient un outil puissant de calcul. Cependant, ces mêmes caractéristiques rendent l'analyse de ce type de réseau une tâche difficile et leur apprentissage un défis pour les chercheurs.

4.4.1.1. Le perceptron

Un cas particulier plus simple du PMC est le perceptron simple à une seule couche. C'est la forme la plus simple de RNA, et permet de classifier correctement des objets appartenant à deux classes linéairement séparables (i.e les deux classes doivent être suffisamment séparées de telles sortes que la frontière de décision entre les deux consiste en un hyperplan). L'origine de perceptron remonte aux travaux de Rosenblatt, présentés dans son livre [Ros 59]. Il est considéré comme le premier réseau à apprentissage. Rosenblatt s'est inspiré de la règle de Hebb pour la modification des poids.

Le perceptron dans sa version originale est constitué d'un seul neurone qui possède un seuil avec un ensemble de poids synaptiques ajustables (il repose sur le modèle de McCulloch et Pitts). Minsky et Papert ont démontré qu'un perceptron entrainé par une procédure d'apprentissage dite *règle du perceptron* (Perceptron Learning Rule), qui est destinée à ajuster les poids synaptiques dans la bonne direction lorsqu'une mauvaise classification

survient (ce type d'apprentissage est supervisé et les entrées fournies lors de l'apprentissage doivent appartenir à deux classes linéairement séparables), la règle converge, en un nombre fini d'itérations, vers un hyperplan séparateur correcte définie par l'équation (14) (voir figure 3.23).

$$\sum_{j=1}^{m} w_j \; x_j \; + b=0 \qquad (14)$$

Le perceptron composé d'un seul neurone ne permet que d'effectuer la classification binaire. Par l'élargissement de la couche de sortie du perceptron en incluant plus d'un seul neurone, il sera possible de discriminer plusieurs classes (les classes doivent être linéairement séparables) [Bla 96].

L'intérêt principal d'un réseau de neurone est sa capacité d'apprentissage. Il est bien entendu que la majorité des réseaux de type perceptron sont à apprentissage supervisé. Dans le cas d'un perceptron simple ayant une fonction d'activation bien déterminée et dont les poids et biais sont initialement inadaptés et attribués aléatoirement, la réaction du perceptron à une entrée donnée ne fournira pas la sortie désirée car il est ignorant. Une phase d'apprentissage supervisé est indispensable pour ajuster les paramètres du réseau (poids et biais) de telle sorte que à chaque entrée du réseau la sortie obtenue soit plus proche de la sortie désirée et l'opération d'ajustement va continuer jusqu'à l'obtention d'un comportement correcte du perceptron.

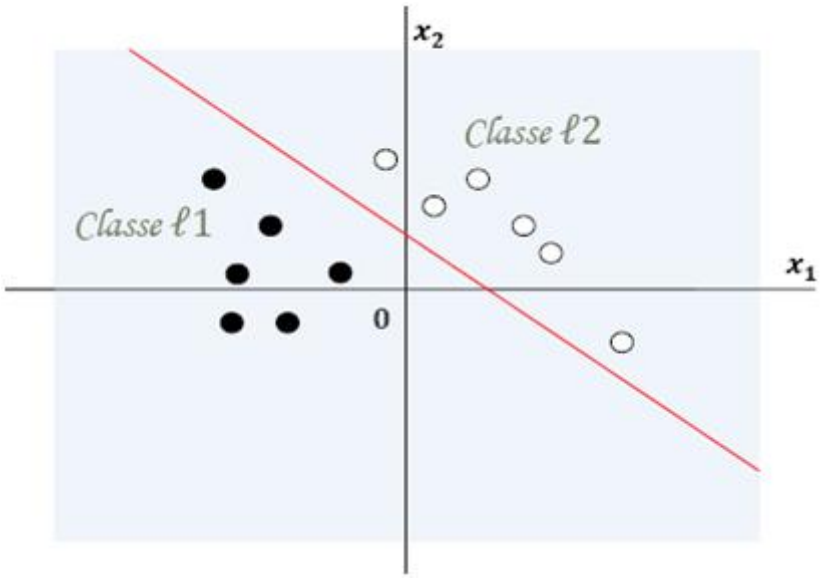


Figure 3.23. Un hyperplan séparateur pour une classification à deux dimensions

La règle du perceptron converge vers une solution, si celle-ci existe, mais vers n'importe laquelle parmi les différentes solutions possible. Le réseau résultant est parfois sensible au bruit (figure 3.24) puisque la frontière de décision se retrouve souvent trop proche des objets d'apprentissage (l'algorithme s'arrête dès que tous les objets sont biens classés). Des règles d'apprentissage plus puissantes sont alors utilisées. Parmi ces règles, on trouve la règle du

moindre carré (Least Mean Square). Elle minimise une erreur (dite l'erreur quadratique moyenne) de sorte que la frontière de décision a tendance à se retrouver aussi loin que possible des prototypes (i.e. les classes).

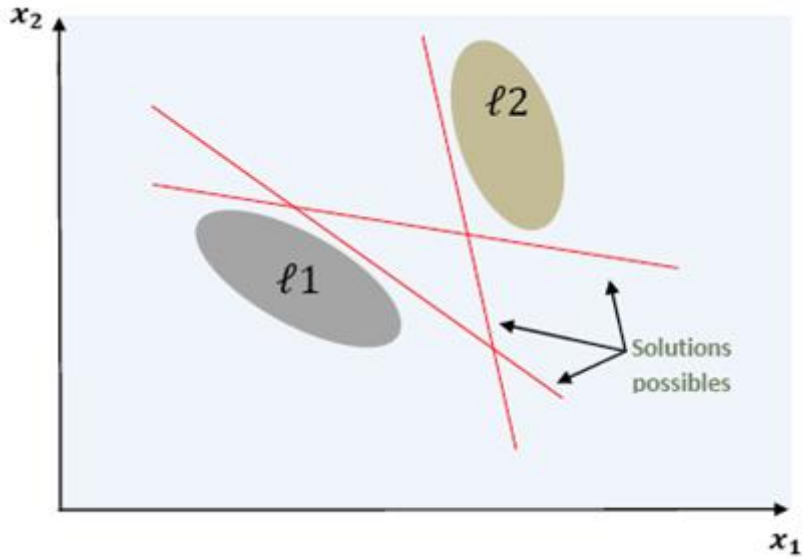


Figure 3.24. Exemple de solutions possibles de la règle du perceptron

4.4.1.2. Limitations et solutions

L'incapacité du Perceptron simple à une couche à résoudre des problèmes intéressants (des problèmes fortement non linéaires en générale) est démontrée par Minsky et Papert [Min 75]. Ainsi, plusieurs problèmes de classification nécessitent une partition non linéaire de l'espace d'entrée que ces perceptrons ne peuvent pas trouver. Un exemple très utilisé pour illustrer cette limitation est le problème du XOR illustré à la figure 3.25. Il consiste à classer en deux classes quatre points, dont chacun est une représentation graphique de la sortie de la fonction logique « `ou exclusif` », qui prend en entrée deux informations (deux nombres qui peuvent valoir 0 ou 1 chacun) et retourne 1 si l'une de ses deux entrées vaut 1, 0 sinon. Comme le montre la figure 3.25, les deux classes (la classe des points en noire et celle des points en blancs) ne sont pas linéairement séparables et ne peuvent pas être apprises par un perceptron simple à une seule couche de neurones.

Pour résoudre ce problème, il suffit de rajouter une couche cachée contenant deux neurones. Chaque neurone (sauf les neurones d'entrée) adopte une fonction d'activation de type seuil [Hay 99].

Les perceptrons multicouches, qui consistent à mettre en cascade deux couches ou plus de perceptrons, sont capables de résoudre des problèmes complexes non linéairement séparables.

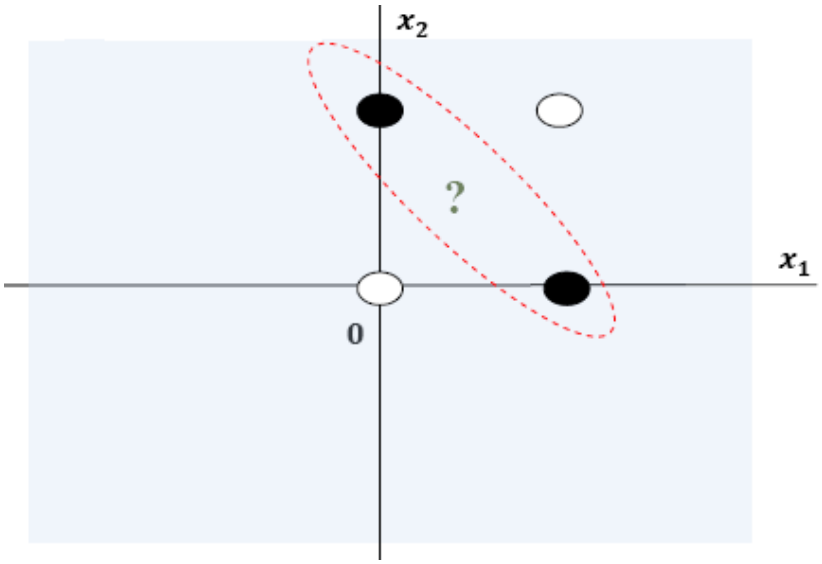


Figure 3.25. Problème du XOR

4.4.2. Le Réseaux à fonction de base radiale (RBF)

Un réseau à fonction de base radiale (RBF pour Radial Basis Function) est un modèle connexionniste simple à mettre en œuvre et est très utilisé pour la régression et la discrimination [Bla 96]. Il représente l'un des deux principaux réseaux de neurones supervisés (l'autre étant le Perceptron Multicouches).

Un RBF est constitué uniquement de 3 couches, une couche d'entrée qui n'effectue aucun traitement, une couche de sortie qui contient une fonction linéaire et une couche cachée qui contient les neurones RBF. Chaque couche est totalement connectée avec la couche suivante, il s'agit donc d'une spécialisation d'un PMC (figure 3.26). Chaque neurone RBF contient une gaussienne (figure 3.27) qui est centrée sur un point de l'espace d'entrée. Pour une entrée donnée, la sortie du neurone RBF est la hauteur de la gaussienne en ce point. La fonction gaussienne permet aux neurones de ne répondre qu'à une petite région de l'espace d'entrée, région sur laquelle la gaussienne est centrée.

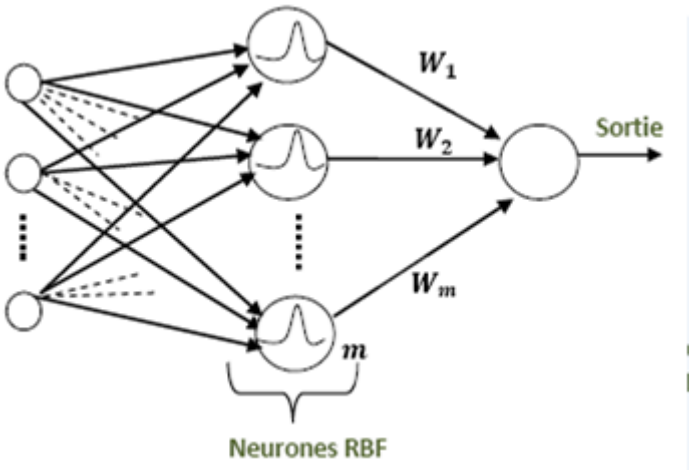


Figure 3.26. Réseau RBF

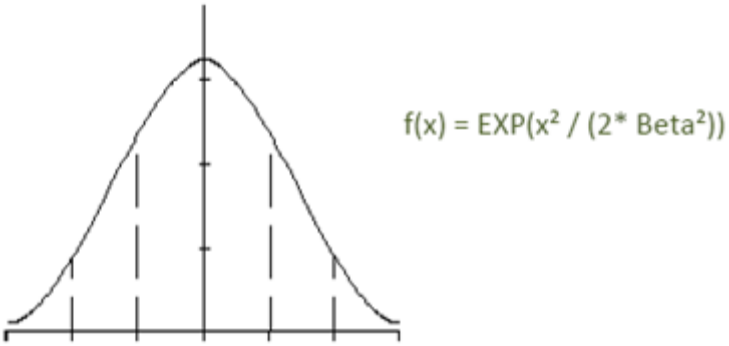


Figure 3.27. Fonction gaussienne

La sortie du réseau est simplement une combinaison linéaire des sorties des neurones RBF multipliés par le poids de leur connexion respective (équation 15):

$$\sum_{j=1}^{m} w_j \; s_j \qquad (15)$$

Il ya quatre paramètres principaux à régler dans un réseau RBF :

- Le nombre de neurones RBF (nombre de neurones dans l'unique couche cachée).
- La position des centres des gaussiennes de chacun des neurones.
- La largeur de ces gaussiennes.
- Le poids des connexions entre les neurones RBF et le(s) neurone(s) de sortie.

Toute modification de l'un de ces paramètres entraîne un changement du comportement du réseau.

4.4.3. Réseau de Hopfiled

Les réseaux de Hopfield sont des réseaux récurrents et entièrement connectés (figure 3.28). Dans ce type de réseau, chaque neurone est connecté à tous les autres neurones par une connexion symétrique et il n'y a aucune différenciation entre les neurones d'entrée et de sortie. Ils fonctionnent comme une mémoire associative non-linéaire et sont capables de trouver un objet stocké en fonction de représentations partielles ou bruitées [Gat 07].

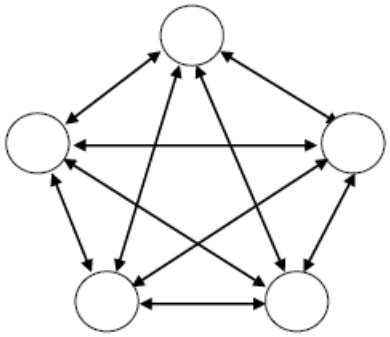


Figure 3.28. Réseau de Hopfiled

Dans une mémoire informatique classique, une information est retrouvée à partir d'une clé arbitraire. Par opposition, une donnée entreposée dans une mémoire associative est accessible à partir d'informations qui lui sont associées. La fonction d'une mémoire associative est de restituer une information en tenant compte de sa perturbation ou de son bruit. L'information doit alors se rapprocher d'une information apprise ou connue.

L'application principale des réseaux de Hopfield est l'entrepôt de connaissances mais aussi la résolution de problèmes d'optimisation. Le mode d'apprentissage utilisé par ces réseaux est le mode non-supervisé.

4.4.4. Carte auto-organisatrices de Kohonen

Les réseaux de neurones à auto-organisation, et en particulier les cartes auto-organisatrices de Kohonen sont des réseaux à apprentissage non supervisé qui constituent une classe importante des réseaux de neurones, Ils ont reçu une importance particulière depuis les travaux de Von der Malsburg et Kohonen [Koh 87].

La carte de Kohonen est un réseau constitué de deux couches. Chaque neurone de la couche d'entrée est relié à chaque neurone de la carte de Kohonen (figure 3.29).

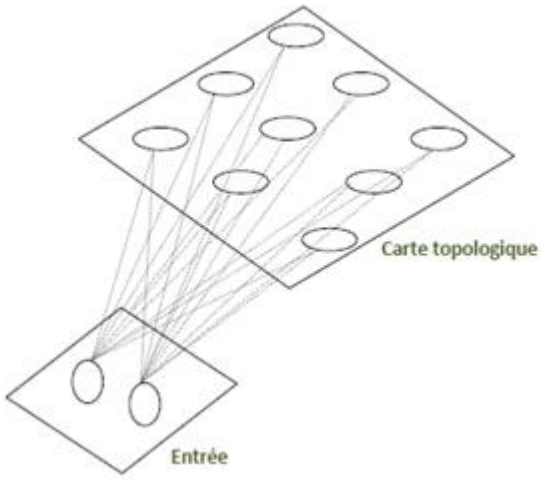


Figure 3.29. Réseau de Kohonen

Dans la couche d'entrée, les neurones correspondent aux variables décrivant les observations. La couche de sortie, elle, est le plus souvent organisée sous forme de grille (de carte) de neurones à 2 dimensions. Chaque neurone représente un groupe d'observations similaires.

Le réseau de Kohonen est donc une technique de classification automatique. L'objectif est de produire un regroupement de manière à ce que les individus situés dans la même case soient semblables, les individus situés dans des cases différentes soient différents. En effet, les neurones de la couche de sortie sont organisés de manière à ce que deux cellules adjacentes dans la grille correspondent à des groupes d'observations proches dans l'espace de

représentation initial. On parle de cartes auto organisatrices (SOM pour Self Organisation Map).

5. Les systèmes immunitaires

La biologie est la source d'inspiration de plusieurs méta heuristiques. Ainsi, les théories de l'évolution et le comportement collectif des insectes sociaux ont permis le développement de la plupart des algorithmes évolutionnaires comme les algorithmes génétiques, les colonies de fourmis, et l'optimisation par essaims d'abeilles. Les Systèmes Immunitaires Artificiels (SIA) en constituent aussi une autre classe particulière qui découle directement de la biologie et plus précisément du fonctionnement du système immunitaire des vertébrés. Souvent, les Systèmes Immunitaires Artificiels sont classés comme étant une nouvelle branche de l'"Intelligence Informatique"(Computational Intelligence) [Das 06]. Comme les autres algorithmes inspirés de la biologie, les systèmes immunitaires artificiels utilisent les idées inspirées de la biologie pour le développement des outils informatiques dans le but de résoudre différents problèmes dans l'ingénierie.

Dans cette section, nous allons introduire cette nouvelle méthode, sa métaphore d'inspiration, ses principales caractéristiques, et nous présenterons ses principaux domaines d'application.

5.1. Le Système Immunitaire Biologique

Le système immunitaire biologique est un système robuste, complexe et adaptatif qui protège l'organisme contre les agressions d'organismes extérieurs. Ce système très complexe est constitué de plusieurs cellules, molécules et organes travaillant ensemble pour construire un mécanisme d'identification capable de catégoriser toutes les cellules (ou molécules) dans le corps comme étant de cellules appartenant au *soi* ou du *non-soi*.

Le non-soi est donc l'ensemble de toutes les molécules différentes du soi et susceptible d'entraîner des réactions immunitaires ; l'élément du non-soi reconnu est un antigène.

Les *antigènes* sont des agents infectieux qui ont pour effet de perturber le fonctionnement normal du système immunitaire naturel. Toute cellule étrangère est considérée comme un antigène. Plus précisément un antigène est une substance douée de la propriété de provoquer une réponse immunitaire. Ainsi, le système immunitaire peut considérer, comme faisant partie du non-soi, les bactéries, les virus, les parasites, les cellules dégénérées, les tissus provenant d'une autre personne (greffes), mais également les aliments, les médicaments, le matériel implantable, etc [Med 07].

Ainsi, les groupes sanguins, et surtout le Complexe Majeur d'Histocompatibilité constituent pour chaque individu des marqueurs de son identité [Car 06].

5.1.1. Immunité innée et Immunité adaptative

En termes biologiques, l'immunité se réfère à des conditions dans lesquelles l'organisme est capable de résister face aux différentes maladies infectieuses. Les cellules et molécules

responsables de l'immunité constituent le *système immunitaire* biologique. La collection et la coordination des différentes réponses de chaque cellule et molécule en présence de pathogènes sont appelées la *réponse immunitaire* [Das 06].

Le système immunitaire biologique peut être envisagé comme étant un système de protection multi niveau, dont chaque niveau fournit des mécanismes de défense différents pour la détection, la reconnaissance et les réponses [Dec 99].
Trois principaux niveaux existent, incluant la barrière anatomique, l'immunité innée (non spécifique) et l'immunité adaptative (spécifique). Une fois que l'immunité adaptative reconnaît la présence d'un ennemi, elle déclenche deux types de réponses : humorale ou à médiation cellulaire (cellulaire).

5.1.1.1. Immunité innée (non spécifique)

L'immunité innée constitue un mécanisme invariable présent dès la naissance de l'être humain. Des conditions physiologiques comme le pH, la température, les cellules et les barrières anatomiques, comme la peau et le système de respiration fournissent des conditions inappropriées pour la vie des organismes étrangers [Car 06].

Dans la plupart des cas, l'immunité innée est capable de détruire les différents pathogènes à la première rencontre.

5.1.1.2. Immunité adaptative (spécifique)

C'est l'immunité résultant de la présence dans l'organisme de récepteurs d'antigène spécifiques à l'égard d'un antigène déterminé. Elle réagit donc sur les cellules étrangères déjà identifiées dans une réponse précédente comme étant des cellules du non-soi et selon le type de l'infection l'immunité adaptative se décompose en deux types : humorale ou cellulaire. Deux des cellules les plus importantes dans ce processus sont les cellules B et les cellules T. Les deux sont issues de la moelle de l'os [Med 07].

- L'immunité humorale est assurée par le transfert d'humeurs d'un homme à l'autre par exemple. C'est le principe du vaccin ou du *sérum (liquide acellulaire qui persiste après la coagulation du sang)* qui est capable de neutraliser et de précipiter les toxines et d'agglutiner les bactéries.
- L'immunité cellulaire, comme son nom l'indique, ne peut être assurée par un simple transfert de sérum. Par contre, elle nécessite le transfert de cellules.

Remarque : En raison de l'efficacité de l'immunité innée, l'immunité adaptative n'est lancée qu'en cas d'urgence. En effet, l'immunité adaptative peut prendre du temps pour lancer une réaction efficace. C'est donc la tâche de l'immunité innée de réagir plus rapidement et de mettre l'attaquant sous contrôle jusqu'à ce que l'immunité adaptative soit capable de réagir plus efficacement.
L'immunité innée et l'immunité adaptative sont inter-liées et ont une influence l'une sur l'autre. Si l'immunité innée échoue dans l'élimination d'un organe étranger, le corps peut lancer une réponse adaptative ou spécifique pour un certain type d'antigènes [Das 06].

5.1.2. Cellules Immunologiques

Les réponses immunitaires sont assurées par les *globules blancs* ou l*eucocytes*, cellules toutes issues de la moelle osseuse. Ces cellules se distinguent selon leur origine et leur rôle fonctionnel. Dans ce qui suit, nous présentons les principales cellules impliquées dans une réponse immunitaire [Med 07].

5.1.2.1. Les lymphocytes

Les lymphocytes (un type de globule blanc) sont des cellules du système immunitaire qui possèdent des récepteurs capables de se lier spécifiquement à un antigène unique (chaque lymphocyte peut reconnaître un certain type de protéine), permettant ainsi de reconnaître une cellule étrangère à l'organisme. Ces cellules sont de grande importance dans une réponse immunitaire. Les principaux lymphocytes sont les cellules T et les cellules B .Toutes les deux sont des variantes des globules blancs et résident dans la moelle des os de vertébrés. Elles sont assez similaires, mais elles diffèrent par la relation qui leur permet de reconnaître les antigènes et par leur rôle fonctionnel.

- Les cellules B sont responsables de la production et la sécrétion des anticorps. Chaque cellule B peut produire seulement un seul type d'anticorps particulier. Les cellules B sont capables de reconnaître les antigènes libres dans la solution. Les cellules T quant à elles requièrent des antigènes pour être présentées aux autres cellules accessoires [Dec 99].
- Les cellules T sont de trois types : cellules T aideuses qui sont essentielles pour l'activation des cellules B ; les cellules T tueuses qui s'attâchent aux organismes étrangers et leur injectent des produits toxiques causants leur mort ; et les cellules T suppresseuses qui empêchent les actions des autres cellules immunitaires et donc évitent les réactions allergiques et les maladies auto-immunitaire [Dec 99].

5.1.2.2. Les anticorps (immunoglobulines)

Les anticorps sont des variantes de cellules B. Leur rôle est de s'attâcher aux antigènes, ce qui conduit à leur éventuelle élimination par les autres cellules. Chimiquement, un anticorps est composé par deux chaînes (H et L) comprenant chacune une grande partie constante et une partie variable qui forme la spécificité de l'anticorps [Med 07]. La partie variable est le site de reconnaissance et de fixation sur un antigène.

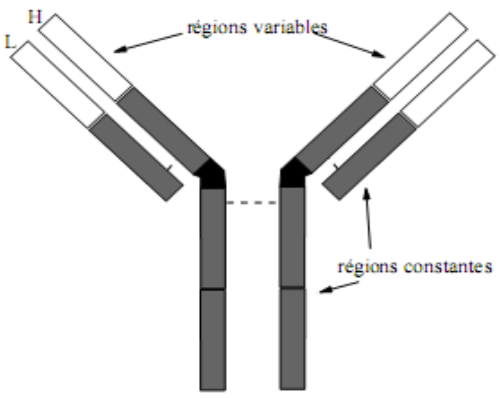


Figure 3.30. Représentation d'un anticorps [Car 06]

Comme l'illustre la figure 30, un anticorps est non seulement capable de s'attâcher à deux antigènes en même temps grâce aux deux portions de la région variable mais aussi peut jouer deux fonctions différentes. En effet, la région variable s'attâche aux différentes formes antigéniques. La région constante peut s'attâcher à un récepteur spécifique dans la surface d'autres cellules immunitaires [Dec 99].

5.1.3. Caractéristiques d'un système immunitaire

D'un point de vue informatique, le système immunitaire peut être considéré comme étant une source d'inspiration très riche du fait qu'il présente des notions d'apprentissage, de mémoire, d'adaptabilité, de reconnaissance de forme et d'auto-organisation. Ses principales caractéristiques peuvent être résumées comme suit [Tim 05] :

5.1.3.1. La discrimination entre le soi et le non-soi (reconnaissance de formes)

Le système immunitaire est capable de reconnaître, d'identifier et de répondre à une grande variété de formes. Il est capable donc de distinguer entre les cellules de soi et les cellules de non-soi (voire figure 3.31).

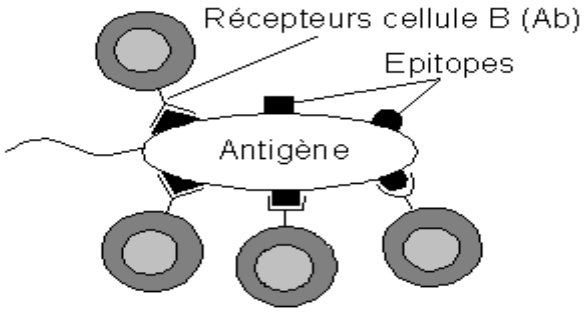


Figure 3.31. Reconnaissance de formes dans le système immunitaire référence [Dec 99].

En effet, comme illustré sur la figure précédente, un antigène peut contenir plusieurs épitopes (formes). Quant aux cellules B, elles ne présentent qu'un seul type de récepteurs. Cependant, le système immunitaire est capable de développer plusieurs types de récepteurs. Ce principe sera détaillé dans les sections qui suivent.

5.1.3.2. Extraction des caractéristiques

Le système immunitaire est capable d'extraire les caractéristiques des antigènes par filtrage des molécules bruits qui provoquent chez lui une réponse immunitaire, avant d'être présenté à d'autres cellules immunitaires y compris les lymphocytes.

5.1.3.3. L'apprentissage et la mémorisation

Le mécanisme d'*hyper mutation somatique* suivi d'un processus de sélection dur permet au système immunitaire d'améliorer et de raffiner sa réponse à un antigène envahisseur. Ce processus est appelé processus de *maturation de l'affinité*. La maturation de l'affinité permet au système immunitaire de devenir de plus en plus efficace pour la reconnaissance de formes. Quant à la mémoire immunitaire, lors d'une réponse immunitaire à un antigène, quelques

ensembles de cellules et molécules sont souscrits avec une durée de vie grandissante afin qu'elles donnent une réponse puissante et rapide à une infection ultérieure causée par le même antigène. Ce processus connu comme étant la *maturation de la réponse immunitaire*, permet d'augmenter la durée de vie des cellules qui ont fait leur preuve lors de la réponse immunitaire. Pour cela le système immunitaire génère des cellules mémoires et des molécules pour combattre ces antigènes lors d'une rencontre ultérieure [Med 07].

5.1.3.4. La distributivité

La distributivité est une caractéristique intrinsèque dans le système immunitaire. Il n'existe aucun point de contrôle central. Chaque cellule est spécifiquement stimulée et répond à tout antigène non encore reconnu qui peut attaquer l'organisme à tout endroit.

5.1.3.5. L'auto organisation

L'auto-organisation caractérise un processus au cours duquel une structure émerge, au niveau global uniquement, d'un grand nombre d'interactions entre les composants de niveau local du système. De plus, les règles spécifiant les interactions entre les composants du système sont suivies en utilisant uniquement des informations locales, sans référence au modèle global [Cam 00].
Suite à une stimulation antigénique, une réponse immunitaire se déclenche, crée donc des nouvelles cellules et molécules pour neutraliser l'effet de cette stimulation. Le système immunitaire est capable de revenir à son état normal et robuste, jusqu'à ce qu'il ait besoin d'une autre réponse à un autre antigène.

5.1.3.6. Le méta dynamisme

Le système immunitaire crée régulièrement des nouvelles cellules et molécules, et élimine celles qui sont devenues trop vieilles ou qui ne sont pas très utilisées.

5.1.3.7. La communication

Dans le système immunitaire, la diffusion de signaux et le dialogue sont deux types de communication possibles. La diffusion de messages est passée par une composante immunologique à une autre sans feedback. Le dialogue quant à lui se présente quand le système immunitaire échange continuellement des signaux moléculaires avec ses semblables [Das 06].

5.1.4. Principaux modèles et théories immunologiques

Le système immunitaire biologique est un système très riche par les différentes théories et modèles visant à expliquer et représenter leur fonctionnement. Dans ce qui suit, nous présentons les principaux modèles proposés dans la littérature.

5.1.4.1. Théorie de la sélection clonale

La sélection clonale fut développée par F. M. Burnet en 1959, et utilisée par le système immunitaire naturel pour définir les grands traits d'une réponse immunitaire à une stimulation antigénique [Bur 78]. Elle établit l'idée que seules les cellules qui reconnaissent les antigènes sont sélectionnées pour proliférer. Les cellules sélectionnées sont sujettes à un processus de

maturation d'affinité, pour améliorer leur affinité vis-à-vis des antigènes sélectionnés. En effet, quand l'organisme est exposé à un antigène, quelques cellules B répondent en produisant des anticorps.

Chaque cellule B sécrète un seul genre d'anticorps qui est relativement spécifique pour l'antigène. En se liant à ces anticorps et avec un second signal de cellules accessoires, comme les cellules T aideurs, l'antigène provoque la division de la cellule B sélectionnée (reproduction) et mature en des cellules sécrétant des anticorps.

5.1.4.1.1. Principe général de la sélection clonale

Lors d'une réponse immunitaire réussie, un lymphocyte ayant reconnu une cellule du non-soi va être stimulée à proliférer (en produisant des clones d'elle-même). La prolifération est rendue possible grâce au processus de division cellulaires (les cellules se divisent elles-mêmes).

Durant la reproduction, les clones (des copies conformes) de cellules B doivent subir un processus d'hyper mutation. Les antigènes stimulent les cellules B pour proliférer et mûrir en un anticorps final en secrétant des cellules nommées cellules plasma. En plus de la prolifération et la différentiation en cellules plasma, les cellules B actives avec une grande affinité antigénique sont sélectionnées pour devenir des cellules mémoires avec une durée de vie très longue. Ces cellules de mémoires circulent à travers le sang, lymphe, et tissu. Quand elles sont exposées à un second antigène stimulateur, elles commencent à se différencier en cellules plasma capables de produire des anticorps avec une grande affinité, présélectionné pour un antigène spécifique qui a stimulé la première réponse. La figure suivante illustre ce processus de prolifération et de différenciation [Med 07].

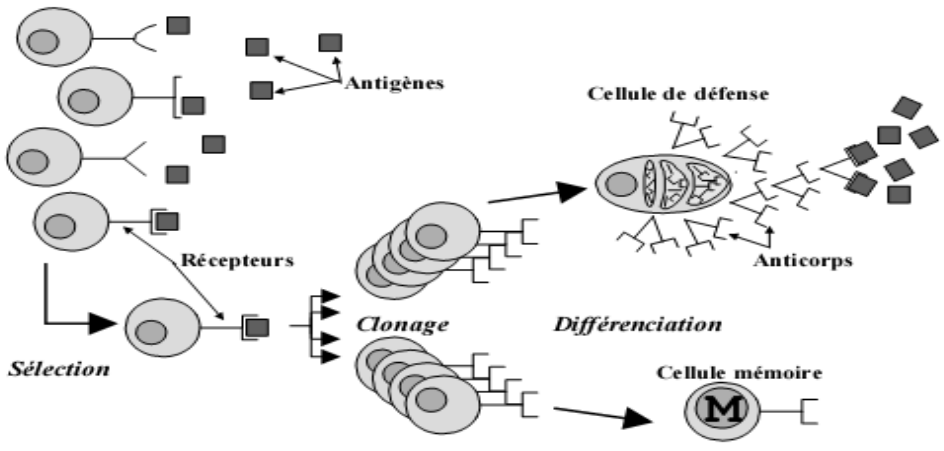


Figure 3.32. Principe de la sélection clonale [Dre 03]

En résumé, les caractéristiques principales de la théorie de la sélection clonale (Burnet 1978) sont les suivantes [Tim 05, Dec 03] :

- Les nouvelles cellules sont des copies de leurs parents (clones) sujettes au mécanisme d'hyper mutation somatique.

- Elimination de nouvelles cellules différenciées présentant des récepteurs réagissant contre le soi.
- Prolifération et différentiation en contact avec les cellules matures et les antigènes.

5.1.4.1.2. Renforcement de l'apprentissage et de la mémoire

L'apprentissage dans le système immunitaire implique l'augmentation de la taille de la population et l'affinité des lymphocytes qui ont fait leur preuve en reconnaissant tout antigène [Dec 00].

Durant la vie de l'homme, l'organisme souvent rencontre le même antigène plusieurs fois. La première infection à un antigène qui provoque une réponse immunitaire adaptative est guidée par des petits clones de cellules B, chacune produisant un anticorps d'affinité différent. L'effectivité de la réponse immunitaire à une seconde rencontre du même antigène est considérablement améliorée en mémorisant quelques meilleures affinités [Dec 02a].

Une caractéristique importante de la mémoire immunitaire est qu'elle est associative : des cellules B adaptées à un certain type d'antigène Ag1 présentent une seconde réponse rapide et plus efficace non seulement pour Ag1, mais aussi pour tout antigène Ag2 qui a une structure proche de Ag1.

5.1.4.1.3. Hyper mutation somatique

Le développement de la mémoire immunitaire est rendu possible grâce au processus de maturation de cellules T. Ce processus requiert que le site d'interaction antigénique avec les anticorps soient structurellement différent par rapport à celui présent dans une réponse primaire.

Des changements aléatoires sont introduits aux gènes de Ag-Ab responsables de l'interaction. Un tel changement peut introduire une nouvelle caractéristique permettant d'augmenter l'affinité de l'anticorps. C'est cette nouvelle caractéristique qui va être sélectionnée pour entrer dans la population de cellules mémoires.

5.1.4.2. Théorie de la sélection Négative/Positive

Appelé aussi mécanisme de discrimination entre le soi et le non-soi, elle permet au système immunitaire de différencier entre les cellules internes des cellules étrangères. En effet, les cellules T ont des récepteurs sur leur surface qui sont capables de détecter les cellules étrangères (antigènes). Durant la génération de cellules T, les récepteurs sont produits par un processus de réarrangement génétique pseudo aléatoire. Donc elles subissent un processus d'interdiction nommé sélection négative dans le thymus, quand les cellules T réagissent contre une protéine de soi et sont détruites. Donc seules celles qui ne s'attâchent pas aux protéines de soi sont autorisées à quitter le thymus. Ces cellules T matures circulent à travers le corps pour effectuer des fonctions immunologiques et protéger le corps contre les antigènes étrangers. Ce processus d'interdiction est très important dans la discrimination entre le soi et le non-soi [Das 99].

5.1.4.3. La théorie du Réseau Immunologique (idéotypique)

La théorie des réseaux immunologiques a été développée par Jerne en 1974 [Jer 74]. L'hypothèse est que le système immunitaire maintient un réseau idéotypique de cellules B interconnectées pour la reconnaissance d'éventuels antigènes. Ces cellules se stimulent et se suppriment en même temps l'une l'autre afin de stabiliser le réseau. Deux cellules B sont connectées si l'affinité qu'elles partagent est supérieure à un certain seuil, et la solidité de la connexion est directement proportionnelle à l'affinité qu'elles partagent [Das 06].

5.2. Le Système Immunitaire Artificiel

Les Systèmes Immunitaires Artificiels (SIA) ont émergé en 1990 comme étant une nouvelle branche de l'Intelligence Informatique (Computation Intelligence).
Plusieurs définitions ont été données pour définir le domaine des Systèmes Immunitaires Artificiels :

- Selon de Castro & Timmis [Dec 02b], « les Systèmes Immunitaires Artificiels sont des systèmes adaptatifs, inspirés par les théories immunologiques et à partir des fonctions, des principes et des modèles immunitaires observés, appliqués à la résolution de problèmes ».
  Cette définition souligne une relation stricte entre la théorie immunologique et les systèmes immunitaires artificiels : l'applicabilité [Dec 03].
- Selon Dasgupta, «Les systèmes immunitaires artificiels sont des méthodologies intelligentes inspirées de systèmes immunitaires pour la résolution de problèmes» [Das 99].

De Castro et Timmis affirment que tout algorithme immunitaire doit vérifier les trois critères suivants :

- Il doit inclure au moins un modèle de composant immunologique comme les lymphocytes.
- Il doit être conçu à partir des idées théoriques ou expérimentales immunologiques.
- Il doit être dirigé vers la résolution de problème.

5.2.1. Le Framework de De Castro et Timmis pour les SIA

Dans le monde de l'ingénierie, il est devenu commode de définir des principes et des processus de base ainsi que de développer des processus génériques aidant à la résolution de problèmes dans un domaine donné.

De Castro et Timmis [Dec 03], propose un framework pour simplifier le processus de correspondance entre les systèmes immunitaires biologiques et les systèmes immunitaires artificiels. Le corps de ce Framework est composé de trois éléments :

- Représentation des composants du système.
- Un ensemble de mécanismes pour l'évaluation de l'interaction de composants avec l'environnement et entre eux (une fonction de l'affinité).
- Procédure d'adaptation qui gouverne la dynamique du système ; i.e. comment son comportement varie avec le temps.

Les différentes étapes sont illustrées dans la Figure 3.33.

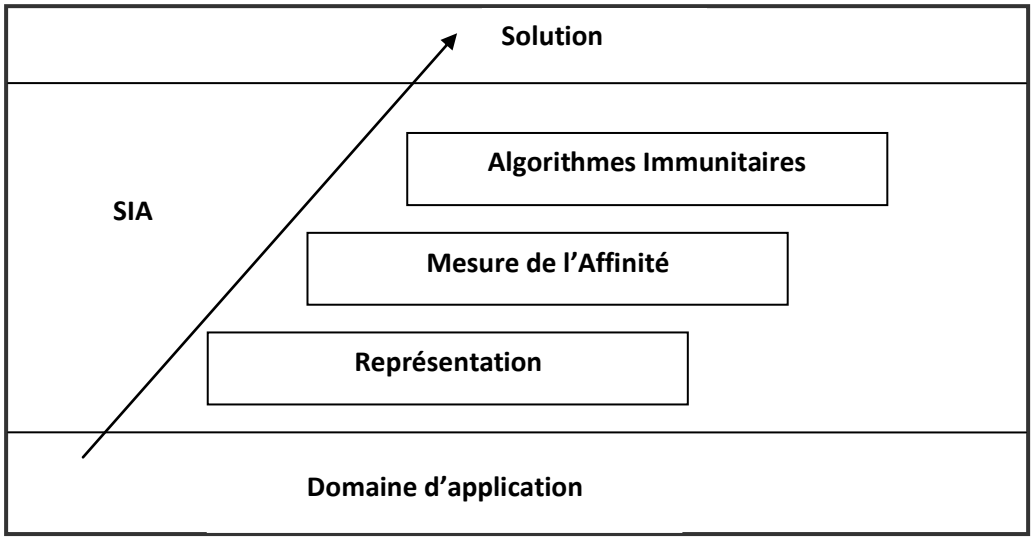


Figure 3.33. Un framework de SIA pour l'ingénierie et sa structure multi niveau [Dec 03]

Ce framework utilise une approche multi-niveaux, dont le domaine d'application influe sur la représentation. Cette dernière influe sur la mesure de l'affinité. Cette approche multi-niveaux a été ensuite généralisée par Stepney et al. Un framework conceptuel générique a été proposé [Ste 04].

5.2.2. Espace de Formes

Le concept d'espace de formes a été introduit pour la première fois par Perelson et Oster en 1979 pour une description quantitative de l'interaction entre les cellules et les molécules de systèmes immunitaires et les antigènes. L'idée était qu'il est possible de décrire la forme d'un récepteur d'anticorps m avec un ensemble de L paramètres physiques (largeur, longueur, charge électrique dans le site d'interaction). Donc un point dans un espace de L dimensions décrit la forme du récepteur. Mathématiquement $m=<m_1, m_2, \ldots m_L>$ est un point d'un espace de formes à L dimensions, m appartient à S^L [Med 07].

Etant donné un système immunitaire avec N anticorps différents, l'espace de formes de ce système contient N points répartis sur un volume V de taille finie avec seulement quelques longueurs, largeurs et charges électriques (figure 3.34).

La notion d'espace de formes a été ensuite généralisée et adoptée dans le domaine des systèmes immunitaires artificiels. Ainsi dans un espace de L dimensions :

- Chaque forme moléculaire m à L dimension correspond à un point de l'espace
- Les molécules ayant des formes semblables sont voisines
- Les molécules complémentaires sont proches mais appartiennent à des plans différents.

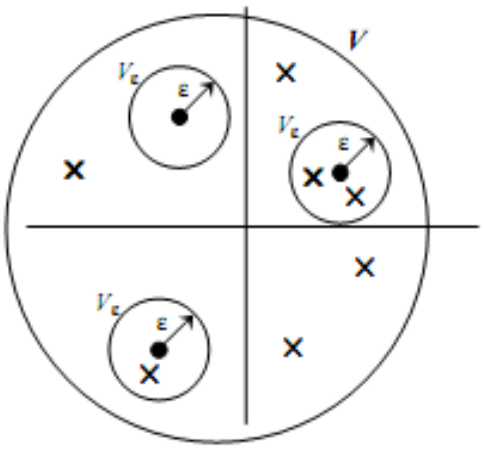


Figure 3.34. Espace de formes Perelson & Oster 1979 [Dec 99]

Un point m=<m_1, m_2, … m_L> peut représenter donc une solution informatique à un problème donné. Ainsi selon ce codage, un anticorps est représenté par un point Ab=<Ab_1, Ab_2, … Ab_L> et un antigène par un point Ag=< Ag_1, Ag_2,…, Ag_L>.

Cette représentation peut se voir comme étant un vecteur de caractéristiques ou une chaîne d'attributs. Ce vecteur peut contenir donc un ensemble de :

- Nombres (réels, entiers, binaires,…)
- Symboles, crées à partir d'un alphabet donné.

5.2.3. Mesure du degré d'interaction entre les anticorps et les antigènes

La mesure de l'affinité dans un espace de formes tient son origine du fait que les récepteurs de cellules lymphocytes s'attâchent aux différentes formes antigéniques pour permettre de donner un moyen par lequel est estimé le degré de correspondance entre une solution candidate et les données [Med 07].

Un point dans l'espace de formes (une cellule artificielle), est entouré par un volume, dans lequel un anticorps peut interagir avec un antigène. Cet entourage par un volume est caractérisé par epsilon, appelé *seuil de contre réaction*, et la surface dans le volume est appelée la *région de reconnaissance*. Une fonction mathématique est nécessaire pour déterminer le degré de correspondance entre deux points. Généralement cette mesure est relative à la distance [Aic 03].

Cette distance peut être une distance Euclidienne, de Hammig ou de Manhattan par exemple. Ainsi, si un anticorps est représenté par Ab=<Ab_1, Ab_2, … Ab_L> et l'antigène Ag=< Ag_1, Ag_2,…, Ag_L> la distance est mesurée par :

- La distance Euclidienne (équation 16):

$$D = \sqrt{\sum_{i=1}^{L}(Abi - Agi)2} \qquad (16)$$

- La distance de Hamming (équation 17):

$$D = \sum_{i=1}^{L} \delta\, i \; Where \; \delta = 1 \; if \; Abi \neq Agi \; et \; 0 \; sinon \qquad (17)$$

- La distance de Manhattan (équation 18) :

$$D = \sum_{i=1}^{L} |Abi - Agi| \qquad (18)$$

5.2.4. Algorithmes Immunitaires

Comme illustré dans les sections précédentes, le système immunitaire naturel est riche en théorie et modèles qui peuvent être des sources d'inspiration d'une grande variété d'algorithmes. Ainsi une classification des algorithmes immunitaires qui ont été développé a été donnée par De Castro et al [Dec 03] (figure 3.35).

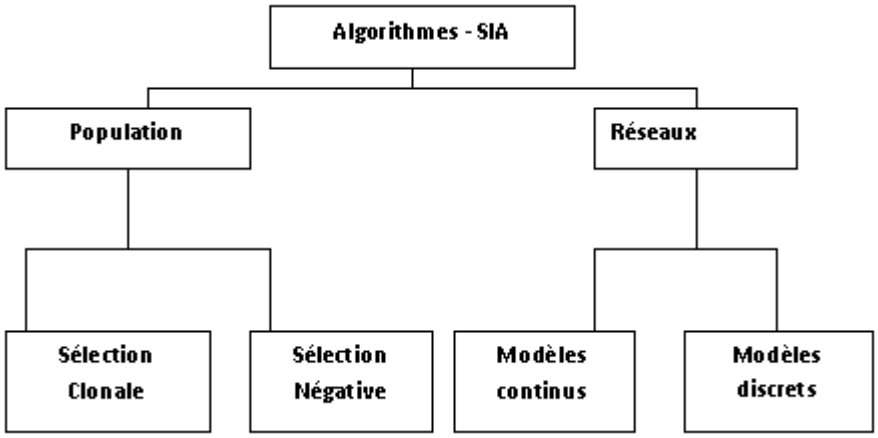


Figure 3.35. Une taxonomie des algorithmes de SIA

Selon cette classification les théories de sélection clonale et de sélection négative / positive sont classées dans les algorithmes à base de populations. Quant aux modèles de réseaux immunologiques, ils sont divisés en deux groupes : discret et continu.

5.2.4.1. Algorithme de la sélection clonale

Le principe ou la théorie de la sélection clonale, est l'algorithme utilisé par le système immunitaire pour décrire les principales caractéristiques d'une réponse immunitaire à une stimulation antigénique. Elle établit l'idée que seules les cellules qui ont pu reconnaître l'antigène prolifèrent, et donc seront sélectionnées par rapport aux autres [Dec 99].

Plusieurs algorithmes basés sur la théorie de sélection clonale ont été proposés. Parmi ces algorithmes nous citons : CLONALG [Dec 00/02a/b/03], et CLONCLAS [Whi 03]. D'une part, l'algorithme CLONALG est le premier algorithme basé sur la théorie de sélection clonale et d'autre part la théorie de sélection clonale a été largement popularisé par les travaux de De Castro et al.
CLONALG existe en deux versions, une pour la reconnaissance des formes et une autre pour le problème d'optimisation multi modale.

5.2.4.2. Démarche de la Sélection Négative/Positive

Un des propos des systèmes immunitaires est de reconnaître toutes les cellules (ou molécules) et de pouvoir donc catégoriser les cellules comme appartenant au soi ou au non-soi. Les cellules du non-soi à leur tour seront catégorisées afin d'introduire le mécanisme de défense approprié.

L'idée principale de la sélection négative est de fournir une tolérance aux cellules de soi. Elle représente la capacité du système immunitaire à détecter les antigènes inconnus sans réagir sur les cellules du soi [Das 06].

Partant des idées de la sélection négative de cellules T, Forrest, et al. (1994) sont à l'origine du développement d'un algorithme de détection d'anomalie basé sur la sélection négative de cellules T dans le thymus. Cet algorithme est appelé algorithme de sélection négative et son application originale a été dans le domaine de la sécurité informatique. L'idée principale de cet algorithme est de générer un ensemble de détecteurs afin de pouvoir détecter une violation d'une norme au sein d'un ensemble d'éléments prédéfinis.

5.2.4.3. Réseau Immunologique

Le système immunitaire maintient un réseau de cellules B interconnectées pour la reconnaissance des antigènes. Ces cellules en même temps se stimulent et s'éliminent les unes les autres pour maintenir la stabilité du réseau. Deux cellules B sont interconnectées si les affinités qu'elles partagent sont supérieures à un certain seuil, et la solidité de la connexion est directement proportionnelle à l'affinité qu'elles partagent.

Deux modélisations théoriques ont été données pour décrire le principe des réseaux immunologiques grâce à la modélisation mathématique de l'interaction de cellules au sein de la population. Cette modélisation a été dirigée vers deux voix : discrète et continu. Le modèle continu de Jerne, Farmer et al [Far 86] et Varela et al [Var 88] utilise les équations différentielles pour la modélisation de la manière dont la concentration de lymphocytes peut varier sur la base de la stimulation, la suppression, la génération et la destruction de cellules. Le modèle de Farmer est le plus significatif pour les SIA, il décrit la modélisation mathématique de la population dynamique de chaîne de bits dans l'espace de Hamming [Dec 03, Far 86] (équation 19).

$$si = \sum_{j=1}^{M}(1 - Di,j) + \sum_{k=1}^{n}(1 - Di,k) - \sum_{k=1}^{n} Di,k \qquad (19)$$

Avec:
M : le nombre d'antigènes
n : le nombre de cellules B interconnectées
Di, j : la distance entre un antigène j et une cellule B i.

En se basant sur cette théorie, Timmis & Neal proposent en 2000 l'algorithme d'apprentissage de réseaux immunologique RAIN, qui utilise l'équation précédente pour la détermination des niveaux de stimulation pour chaque cellule (i) dans la population durant chaque itération de l'algorithme.

Les réseaux immunologiques ont déjà été appliqués avec succès dans des domaines variés dont, à titre d'exemple, : le data mining [Hun 96] et le diagnostic [Ish 96].

5.2.4.4. La Théorie du Danger

La théorie du danger a été proposée la première fois par Polly Matzinger, en 1995, comme une nouvelle technique émergente de la théorie classique de soi et non-soi initialement appliquée à la détection des intrusions [Mat 02].

L'idée principale derrière la théorie du danger est que :

- Le système immunitaire réagit toujours contre le non-soi, sauf si le non soi est non dangereux ;
- Le système immunitaire tolère toujours le soi sauf si le soi est dangereux.
  Plusieurs tentatives de définition de cette nouvelle vision de la théorie classique de soi et non-soi ont été données [Med 07] :
  Selon Cayzer : « La théorie du danger suggère que le système immunitaire réagit aux menaces en se basant sur la corrélation des différents signaux d'alarme, fournissant une méthode d'"agrandissement" de la réponse immunitaire, i.e. lier directement à l'attaquant » [Cay 05].
  Selon Aickin :« La théorie des dangers suppose que le système immunitaire réagit aux différentes menaces en se basant sur la corrélation des différents signaux (dangers), fournissant une méthode de justification de la réponse immunitaire pour se lier directement à l'attaquant » [Aic 03].

Selon Dasgupta : « Dans la théorie de danger, la réponse immunitaire est déterminée par la présence ou l'absence des signaux d'alarmes, quelques signaux d'alarmes comme les dommages dans les tissus déclenchent plusieurs réactions et réponses immunitaires. L'idée principale est de reconnaître la présence d'un danger au lieu de reconnaître une cellule de non-soi. » [Das 06]

5.2.4.5. Bibliothèques de gènes

Les bibliothèques de gènes sont de mécanismes biologiques pour la génération et la maintenance de la diversité dans les systèmes immunitaires [Med 07].

Les gènes qui encodent les récepteurs de cellules sont hérités par les cellules comme étant des fragments de gènes. Ces fragments sont hérités aléatoirement puis concaténés pour la génération de la diversité. Cependant, les anticorps sont constitués par deux régions différentes : chaîne « hard » et chaîne « lourd », ce qui requière des gènes différents pour les encoder pour préserver les structures. Dorénavant, les séquences d'ADN héritées ne peuvent être concaténées d'une manière entièrement aléatoire, sauf si ces fragments de gènes sont pris à partir d'une bibliothèque de gènes spécifique [Cay 05, Cay 06].
L'utilisation des bibliothèques des gènes présente plusieurs avantages parmi lesquels [Cay 05] :

- Améliorer la couverture de l'espace de non-soi par un meilleur placement des détecteurs (anticorps) au lieu d'une génération aléatoire.
- Réduire le coût de génération des détecteurs en évitant plus efficacement les cellules de soi.
- Donner une description plus efficace de la population des antigènes.
- Aider la distribution avec les antigènes développés.

6. Conclusion

Dans ce chapitre nous avons présenté quarte types de systèmes dites bio-inspirés, nous avons donné pour chacun les caractéristiques qui le définissent et pour en conclure nous pouvons dire que :

- Les automates cellulaires sont des structures abstraites qui permettent d'étudier des univers virtuels dont on maîtrise l'ensemble des lois. Ils contribuent à la connaissance de notre propre Univers : « Ainsi, les systèmes physiques et biologiques complexes peuvent-ils reposer sur les mêmes classes universelles que les modèles mathématiques idéaux fournis par les automates cellulaires. La connaissance du comportement des automates cellulaires peut amener à des résultats plus généraux concernant le comportement des systèmes naturels complexes ».
- Les algorithmes évolutionnaires sont un outil puissant pour l'optimisation. En plus du fait de s'inspirer de l'évolution des espèces et d'avoir une « légitimité » biologique, ils disposent d'une base théorique solide qui permet leur mise en application dans divers domaines. Nous avons vu par ailleurs qu'il y a de très nombreuses variantes d'algorithmes évolutionnaires, et cela à cause de la disponibilité de plusieurs options à chaque étape (plusieurs modes de codage, plusieurs possibilités de mutation, plusieurs méthodes de croisement, etc.) et il serait intéressant de les réunir dans une même application où le choix de chaque étape serait donné à l'utilisateur. Cela permettrait de comparer leur performance et de juger de l'efficacité et de la primauté d'une version par rapport à une autre ainsi que de l'importance des divers paramètres.
- Les réseaux connexionnistes possèdent une capacité d'apprentissage. Ils peuvent apprendre à partir des exemples et sont capables d'effectuer des tâches complexes, et pourtant, ils sont constitués de neurones formels à faible capacité de calcul. Les connexions jouent donc un rôle fondamental dans les approches connexionnistes. C'est grâce à elles que les réseaux de neurones artificiels arrivent à résoudre différents problèmes.
- Les Systèmes Immunitaires Artificiels constituent une nouvelle approche d'optimisation robuste et flexible qui peut être appliquée à une grande variété des problèmes. La sélection négative, la sélection clonale et le réseau immunologique restent les modèles les plus utilisés. Des nouveaux modèles comme la théorie de dangers et les bibliothèques des gènes restent en cours de développement et sont moins utilisés. Il n'existe pas d'algorithmes standards bien définis pour chaque modèle. Souvent les implémentations varient selon le problème traité.

7. Références

[Ada 98] C.Adami, Introduction to Artificial Life, Ai and Society Journal Springer-Verlag V13, New-York Etas-Unis, 1998, pp450-451.

[Aic 03] P.Aickelin, P.Bentley, S.Cayzer, J.Kim, J.McLeod, Danger Theory: The Link between AIS and IDS?, 2nd International Conference on Artificial Immune Systems, Napier University, Edinburgh, UK, 1-3 Septembre 2003, pp 147-155.

[Aza 00] F.Azam, Biologically Inspired Modular Neural Networks. PhD thesis, Faculty of the Virginia Polytechnic Institute and State University USA, May 2000.

[Bal 95] S.Baluja, R.Caruana, Removing the genetics from the standard genetic algorithm, Proceedings of the Twelfth International Conference on Machine Learning, San Mateo, CA Etas-Unis, 22 Mai 1995,pp38-46.

[Bel 08] A.Belaïd, Implémentation d'une méthode de classification non supervisée de données à l'aide d'algorithmes génétiques évolutionnaires, Mémoire d'ingénieur en informatique INI Algerie 2008.

[Bel 92] E. Belhaire, Contribution à la réalisation électronique de Réseaux de Neurones Formels : Intégration Analogique d'une machine de Boltzmann, Thèse de doctorat, Université de Paris XI Orsay Paris France, Février 1992.

[Ben 07] I.Benkermi, Modèle et algorithme d'ordonnancement pour architectures reconfigurable dynamiquement. Thèse de doctorat, Université de Rennes 1 Rennes France, Janvier 2007.

[Bla 96] F.Blayo et M.Verleysen, Les réseaux de neurons artificiels, Presses Universitaires de France, 1ère édition Que Sais-je No 3042, France, 1996.

[Bur 91] G. Burel, réseaux de neurones en traitement d'images: Des Modèles Théoriques aux Applications Industrielles. Thèse de doctorat, Université de Bretagne Occidentale, Décembre 1991.

[Cam 00] G.Camazine, S.Deneubourg, J.Franks, N.Sneyd, J.Theraulaz, and E.Bonabeau, Self-Organization in Biological Systems, Livre édition Princeton University Press, New Jersey Etas-Unis, 2000.

[Car 06] B.Caroline, Modélisation mathématique de la réponse lymphocytaire T spécifique à une infection virale, Thèse de Doctorat, l Ecole Nationale Supérieure des Mines de Saint-Etienne et de l'Université Jean Monnet de Saint-Etienne France, le14 avril 2006.

[Cay 05] S.Cayzer, J.Smith, A.R.Marshall, T.Kovacs, What Have Gene Libraries Done for AIS? , 4th International Conference on Artificial Immune Systems, Banff, Canada, 14-17 Aout 2005.

[Cay 06] S.Cayzer & J.Smith, Gene Libraries: Coverage, efficiency and diversity, 6th Conference of Adaptation in Artificial and Biological Systems University of Bristol, Bristol, Angleterre, 03-06 Avril 2006.

[Cou 07] P.Coulibaly, F.Anctil, B.Bobée, Prévision hydrologique par réseaux de neurones artificiels : état de l'art, Revue canadienne de génie civil. Vol. 26, 1999, pp. 293-304.

[Dar 08] F.Dario, C.Mattiussi, Bio-Inspired Artificial Intelligence Theories, Methods and Technologies, Livre edition The MIT Press Cambridge, Massachusetts, London 2008.

[Das 06] D.Dasgupta, Advances in Artificial Immune Systems, IEEE Computational Intelligence Magazine, Novembre 2006, pp40-49.

[Das 99] D.Dasgupta, S. Forrest, Artificial Immune Systems in Industrial Applications, the Second International Conference on Intelligent Processing and Manufacturing of Materials (IPMM), Honolulu, Etas-Unis, 10-15 Juillet 1999.

[Dec 00] L.N.De Castro, F.J.Von Zuben, Artificial Immune Systems: Part II A Survey of Applications, Technical Report RT DCA 02/00, FEEC/UNICAMP, Brazil, 2000.

[Dec 02a] L.N.De Castro, F.J.Von Zuben, Learning and Optimization Using the Clonal Selection Principle, the Special Issue on Artificial Immune Systems of the journal IEEE Transactions on Evolutionary Computation, Vol. 6, N°3, Juin 2002.

[Dec 02b] L.N.De Castro, J.Timmis, Artificial Immune Systems: A Novel Approach to Pattern Recognition, In L Alonso J Corchado and C Fyfe, livre edition Artificial Neural Networks in Pattern Recognition, University of Paisley, Ecosse, Janvier 2002, pp67-84.

[Dec 03] L.N.De Castro, J.Timmis, Artificial Immune Systems as a Novel Soft Computing Paradigm. In the Soft Computing Journal, V7, Issue 7, Juillet 2003.

[Dec 99] L.N.de Castro, F.J.Von Zuben, Artificial Immune Systems: Part I Basic Theory and Applications. Technical Report – RT DCA 01/99, FEEC/UNICAMP, Brazil,1999.

[Der 04] G. Dreyfus, J.M. Martinez, M. Samuelides, M. B. Gordon, F. Badran, S. Thiria, L. Hérault, Réseaux de neurones, méthodologie et applications, sous la direction de Gérard Dreyfus, Livre 2ème édition, Eyrolles, avril 2004.

[Dre 03] J.Dreo, A.Perowski, P.Siarry, E.Taillard, Métaheuristiques pour l'optimisation difficile, Livre Edition Eyrolles 2003.

[Dre 05] J.Dréo, A.Petrowski, P.Siarry , E.Taillard, Métaheuristiques pour l'optimisation difficile, livre Editions Eyrolles, 2005.

[Eib 03] A.E.Eiben, J.E.Smith, Introduction to Evolutionary Computing. Livre edition Natural Computing Series Springer-Verlag, Berlin Allemagne, 2003.

[Fog 66] L.J.Fogel, A.J.Owens, M.J.Walsh, Artificial Intelligence through Simulated Evolution, Livre édition Wiley, New York Etas-Unis, 1966.

[Gat 07] M.L. Gatet, Intégration de réseaux de neurones pour la Télémétrie Laser. Thèse de doctorat, Université de Toulouse France, Septembre 2007.

[Gol 89] D.E.Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Livre edition Addison-Wesley, Redwood City, CA Etas-Unis,1989.

[Gut 94] H.A.Gutowitz, C.G.Langton, Methods for designing Cellular Automata with Interesting Behavior, 1994.

[Har 01] F.Harold, The Way of the Cell: Molecules, Organisms, and the Order of Life, Oxford University Press, Oxford, 2001.

[Has 95] M.H. Hassoum, Fundamentals of artificial neural networks. MIT Press, Cambridge Angleterre, 1995.

[Hay 99] S. Haykin, Neural networks A comprehensive foundation, livre 2eme edition Pearson, 1999.

[Heu 98] J.C.Heudin, L'évolution au bord du chaos, Livre édition Hermès, Paris, 1998.

[Hol 75] J.H.Holland, Adaptation in Natural and Artificial Systems, Livre édition University of Michigan Press, Ann Arbor, Etas-Unis, 1975.

[Kir 83] S.Kirkpatrick, C.D.Gelatt, M.P.Vecchi, Optimization by simulated annealing. Journal of Science, 1983, pp671-680.

[Koh 87] T. Kohonen, Self-organization and associative memory, Livre 2eme Édition Springer-verlag, Berlin, 1987.

[Koz 92] J.R.Koza, Genetic Programming: On the Programming of Computers by Means of Natural Selection, Livre edition MIT Press, Cambridge, MA Etas-Unis, 1992.

[Lam 09] M.L.Lamali, Approches évolutives pour la classification non supervisée de données, Mémoire de fin d'études pour l'obtention du diplôme d'ingénieur d'état en informatique, Ecole national Supérieure d'Informatique (ESI) Oued-Smar, Alger 2009.

[Mat 02] S.Matzinger, The Danger Model: A renewed sense of self Science, Journal of Science, 2002, pp301-304.

[Mcc 43] W.S.Mcculloh, W.Pitt, A logical calculus of the ideas immanent in nervous activity, Bulletin of Mathematical Biophysics, 1943.

[Med07] M.Medjoudj, Approche de Partitionnement et Ordonnancement à base de Systèmes Immunitaires Artificiels Mémoire de fin d'études pour l'obtention du diplôme d'ingénieur d'état en informatique Institut National de formation en Informatique (I.N.I) Oued-Smar, Alger 2007.

[Mes 06] D.Meslati, Mage : une approche ontogénétique de l'évolution dans les systèmes logiciels critiques et embarques thèse de doctorat université de Annaba Algérie 2006.

[Mic 96] Z.Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, Livre 3rd edition, Springer-Verlag, Berlin 1996.

[Min 75] M.Minsky, A framework for Representing Knowledge. In: The Psychology of Computer Vision, Livre edition Wiston , MacGraw-Hill, New York Etat-Unis, 1975.

[Pre 96] L.Prechelt, A quantitative study of experimental evaluations of neural networklearning algorithms: current research practice, The Official Journal of the International Neural Network Society, European Neural Network Society & Japanese Neural Network Society V9, Avril 1996, pp457-462.

[Ren 00] J.P.Rennard support de cours les automates cellulaires http://www.rennard.org/alife 2000.

[Ros 59] F. Rosenblatt, Principles of Neurodynamics, Livre édition Spartan Books, New York Etas-Unis, 1959.

[San 98] F.Santos Osório, un système hybride neuro-symbolique pour l'apprentissage automatique constructif, Thèse de doctorat, Institut national polytechnique de Grenoble France, février 1998.

[Sch 92] N.Schraudolph, R.K.Belew, Dynamic parameter encoding for genetic algorithms, éditeur Machine Learning, 1992.

[Ste 04] S.Stepney, R.E.Smith, J.Timmis and A.M.Tyrrell., Towards a Conceptual Framework for Artificial Immune Systems, In the proceedings of the Third International Conference on Artificial Immune Systems, Catania, Italy,13-16 Sptembre, 2004.

[Sys 89] G.Syswerda, Uniform crossover in genetic algorithms. In Proceedings of the Third International Conference on Genetic Algorithms, Morgan Kaufmann, San Mateo, CA Etas-Unis, 04-07 Juin 1989, pp2-9.

[Tim 05] J.Timmis, M.Neal, Once more unto the breach: Towards artificial homeostasis, Recent Developments in Biologically Inspired Computing, chapter in Book Recent Developments in Biologically Inspired Computing edition L N De Castro and F J Von Zuben, 2005, pp340-365.

[Tor 97] J.M.Torres-Moreno, Apprentissage et généralisation par des réseaux de neurones : étude de nouveaux algorithmes constructifs. Thèse de doctorat, Institut national polytechnique de Grenoble, France, septembre 1997.

[Vir 01] B.Virole, Résaux de neurones et Psychométrie. Editions du Centre de Psychologie Appliquée, Juin 2001.

[Wag 96] G.P.Wagner, L.Altenberg, Complex adaptations and the evolution of evolvability. Journal of Evolution, 1996, pp967-976.

[Wei 91] S.M.Weiss, C.A.Kulikowski, Computer systems that learn, Morgan Kaufmann Publishers Inc., San Mateo, Californie Etas-Unis, 1991.

[Whi 03] A.White, M.Simon, Improved Pattern Recognition with Artificial Clonal Selection? , 2ed International Conference on Artificial Immune Systems (ICARIS), Napier University, Edinburgh, UK, 1-3 Septembre 2003.

[Whi 98] D.Whitley, S.Rana, R.B.Heckendorn, The island model genetic algorithm: On separability, population size and convergence, Journal of Computing and Information Technology, 1998, pp33-47.

[Wol 02] S.Wolfram, A New Kind of Science, Livre édition Wolfram Media, 2002.

[Zem 03] M.R.Zemouri, Contribution à la surveillance des systèmes de production à l'aide des réseaux de neurones dynamiques : Application à la e-maintenance, Thèse de doctorat, Université de Franche-Comté, Novembre 2003.

CHAPITRE 4 LE FORMALISME DE REPRESENTATION PAR LES MODELES

1. Introduction

Les systèmes informatiques n'ont jamais été autant au centre de la stratégie des entreprises qu'aujourd'hui. Les fonctionnalités qu'ils offrent, leur facilité d'utilisation, leur fiabilité, leur performance et leur robustesse sont les qualités reines qui permettent aux entreprises d'être compétitives. L'ingénierie du logiciel fournit sans cesse de nouvelles technologies facilitant la mise en œuvre de ces systèmes tout en accroissant leurs qualités. A travers ce chapitre, nous présentons l'approche « Model Driven Architecture » (MDA) et le concept de modèle, et nous décrivons les standards sur lesquels l'architecture de MDA est basée ainsi que le processus de développement basé modèles. Par la suite, nous nous focalisons sur la transformation de modèles, leur principe et les langages et outils développés pour la réaliser.

2. Les modèles

Avant de préciser en quoi les modèles sont intéressants pour nous, il est intéressant de préciser la définition du terme *modèle*.

Le dictionnaire de la langue française en ligne TLFI *(Trésor de la langue française informatisé)* donne les définitions suivantes du mot « modèle »*:*

- Arts et métiers : représentation à petite échelle d'un objet destiné à être reproduit dans des dimensions normales.
- Épistémologie : système physique, mathématique ou logique représentant les structures essentielles d'une réalité et capable à son niveau d'en expliquer ou d'en reproduire dynamiquement le fonctionnement.
- Cybernétique : système artificiel dont certaines propriétés présentent des analogies avec des propriétés, observées ou inférées, d'un système étudié et dont le comportement est appelé, soit à révéler des comportements de l'original susceptibles de faire l'objet de nouvelles investigations, soit à tester dans quelle mesure les propriétés attribuées à l'original peuvent rendre compte de son comportement manifeste.

Dans un contexte informatique, la définition suivante est proposée : Les modèles d'applications informatiques sont des représentations, à différents niveaux d'abstraction et selon plusieurs vues, de l'information nécessaire à la production et à l'évolution des applications [Mou 05].

Les modèles offrent de nombreux avantages dont le plus important est qu'ils permettent est de spécifier différents niveaux d'abstraction, facilitant la gestion de la complexité inhérente aux applications.

Les modèles très abstraits sont utilisés pour représenter l'architecture générale d'une application ou sa place dans une organisation, tandis que les modèles très concrets permettent de spécifier précisément des protocoles de communication réseau ou des algorithmes de synchronisation. Même si les modèles se situent à des niveaux d'abstraction différents, il est possible d'exprimer des relations de raffinement entre eux. Véritables liens de traçabilité, ces relations sont garantes de la cohérence d'un ensemble de modèles représentant une même application.
La diversité des possibilités de modélisation ainsi que la possibilité d'exprimer des liens de traçabilité sont des atouts décisifs pour gérer la complexité.

Un autre avantage incontestable des modèles est qu'ils peuvent être présentés sous format graphique, facilitant d'autant la communication entre les acteurs des projets informatiques [Bla 05].

La représentation graphique de ces modèles offre un gain significatif de productivité.

3. Architecture basée modèles

Au fur et à mesure de l'apparition de nouvelles machines beaucoup plus puissantes, les applications logicielles deviennent de plus en plus complexes et de grande taille. Cette évolution engendre une difficulté lors de la tâche de développement et de maintenance et augmente significativement la complexité des logiciels (plus de dépendances entre les éléments d'un système logiciel) ainsi que la complexité de leurs développements. De par sa nature, la maintenance ou l'évolution des produits logiciels est difficile à effectuer. Cette difficulté est encore pire lorsque le produit à modifier, soit pour le maintenir ou l'évoluer, atteint un degré de complexité élevé. Ces problèmes nécessitent une approche interdisciplinaire qui permet de développer les systèmes logiciels de telle sorte qu'ils soient plus facilement et faiblement changés dans les phases qui suivent le développement [Mes 06].

L'ingénierie dirigée par les modèles (MDE pour Model-Driven Engineering) est l'une des nouvelles approches proposées pour résoudre les problèmes de développement, de maintenance et de l'évolution. Elle fait partie des approches dites basées processus du fait qu'elles visent directement le processus de développement et de maintenance. L'ingénierie des modèles engendre une modification profonde dans le processus de production et de maintenance [Mes 06].

Cette approche fait suite à la proposition nommée MDA (pour Model Driven Architecture) faite par l'OMG (Object Management Group) en novembre 2000 [Omg 04]. L'objectif principal de cette approche est de séparer la logique métier (partie fonctionnelle) de la partie implémentation. De par le fait que la logique métier est stable, indépendante des plateformes d'exécution, et ne subit que peut de modifications à travers le temps, il est donc intéressant de séparer les deux parties parce que cette séparation va conduire à une meilleure portabilité, interopérabilité, réutilisation et productivité [Bla 05]. Ainsi, tout changement effectué sur une des préoccupations (i.e. fonctionnalités métiers, règles de gestions et plateforme d'exécution) n'aura aucun effet sur les autres.

Le point clés de l'approche MDA est l'utilisation massive des modèles. Un modèle peut être défini comme une description conceptuelle d'un système et de son environnement [Mes 06]. MDA consiste à décrire les systèmes en utilisant des modèles. Ces derniers peuvent être de différents types : Les modèles décrivant la logique métier qui visent à représenter la partie fonctionnelle des systèmes sans entrer dans les détails des plateformes d'exécution (e.g. J2EE, .Net, PHP, etc.) et les modèles incluant les détails techniques des plateformes.
L'utilisation des modèles présente plusieurs avantages [Bla 05]. Le modèle fournit plusieurs nivaux d'abstraction allant de modèle très abstrait, tel que la présentation de l'architecture générale d'un système logiciel, au modèle très concret, tel que la spécification d'un protocole de communication réseau. La possibilité de présenter les modèles sous format graphique est un avantage indéniable du fait qu'elle facilite la communication entre les acteurs des projets informatiques et augmente la compréhension et la maitrise de la complexité.

Ainsi, l'approche basée modèles emploie des outils et des langages standardisés pour créer les modèles (e.g. UML). Par conséquent, la communication et l'échange par les modèles seront grandement simplifiés, ce qui laisse un effet important sur les processus de développement et de maintenance [Mes 06].

Bien entendu, les modèles peuvent appartenir à différents niveaux d'abstraction. L'approche MDA fournit un point clés puissant qui permet de transformer un modèle d'un niveau d'abstraction à un autre, on parle de la transformation des modèles. Par exemple, un modèle concret (e.g. modèle de codage) peut être obtenu par la transformation d'un modèle abstrait (e.g. modèle de conception) en raffinant ce dernier par l'ajout de plus de détails (e.g. détails concernant une plateforme technologique). Cette possibilité apporte des bénéfices remarquables car le fait qu'on soit capable d'automatiser le processus de développent et de maintenance par la transformation automatique des modèles, engendre un gain important en temps et en coût.

Ces avantages apportés par l'approche MDA ont dégagé un courant de généralisation qui considère qu'un système logiciel n'est qu'un ensemble de modèles donnant chacun une facette de ce dernier, et que les activités liées à l'ingénierie des logiciels et leur maintenance ne sont en fait que des transformations de modèles [Béz 04].

4. Architecture et processus de MDA

L'approche MDA se base sur plusieurs standards universellement utilisés dans le domaine de génie logiciel, et vise un large éventail d'applications. La figure 1 présente une vision générale de MDA. Elle illustre que l'architecture du MDA se découpe en quatre couches. Au centre, on cite le standard UML (Unified Modeling Language), MOF (Meta Object Facility) et CWM (Common Warehouse Metamodel). La deuxième couche contient le standard XML-XMI (XML Metadata Interchange) et un ensemble de middlewares (Java, .Net, CORBA et web services). La troisième couche propose des services permettant de gérer les transactions, la sécurité, les évènements, les référentiels et autres services. La dernière couche contient des frameworks spécifiques au domaine d'application (médecine, Transport, Finance, commerce électronique, Télécommunication, Espace, manufacture,…) [Bla 05] (voir Figure 4.1).

Figure 4.1. Architecture de l'approche MDA.

4.1. Les standards de l'OMG

L'approche MDA se base sur un ensemble de standards adoptés par l'OMG, incluant UML, MOF, XMI/XML et CWM, qui permettent la création et la gestion des modèles (voir [Bla 05] pour plus de détail).

4.1.1. Unified Modeling Language (UML)

UML est un langage graphique et visuel pour la spécification, la construction et la documentation de tous les éléments (artefacts) d'un système logiciel. UML peut aussi être considéré comme un langage de modélisation général qui peut être utilisé avec les méthodes basées objets et basées composants, et appliqué dans tous les domaines (e.g. santé, finance, télécommunication, etc.) et accepté par toutes les plateformes d'implémentation (e.g. J2EE, .NET) [Bar 05].

UML fournie un ensemble de diagrammes (i.e. une représentation graphique) qui représente chacun un aspect particulier du système à modéliser [Bar 05]. La combinaison de différents diagrammes permet d'obtenir une vue complète de tous les aspects caractérisant un système informatique (aspect statique, aspect dynamique, aspect fonctionnel). Dans ses versions précédentes, UML n'a adopté que neuf diagrammes. Plus tard, UML dans sa version 2.0 est enrichi par quatre nouveaux diagrammes (voir figure 4.2).

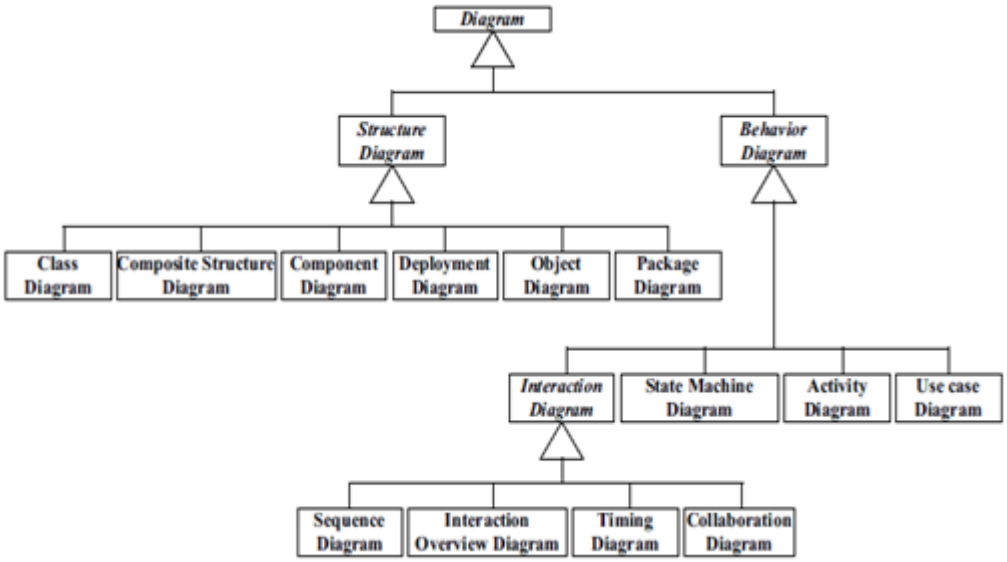


Figure 4.2. Les différents diagrammes UML [Bla 05].

Le langage UML occupe une place importante dans MDA. L'approche MDA préconise UML comme étant le langage utilisé pour créer les modèles indépendants des plateformes par le fait que UML est indépendant de toute plateforme d'implémentation, qu'elle soit J2EE, .Net, PHP, etc. En plus des modèles UML, MDA propose la définition d'une adaptation du langage UML à un domaine spécifique (e.g. domaine des EJB), on parle alors de la notion de profil *UML*. Un profil UML cible une plateforme d'exécution permettant, donc, d'adapter UML à une plateforme d'exécution. Grace aux profils UML, il est possible d'utiliser UML pour élaborer des modèles dépendants des plateformes d'implémentation.

UML est un langage de modélisation dans le sens où il définit un ensemble de concepts (i.e. éléments de modélisation) permettant la réalisation des modèles UML. En effet, le concept sur lequel UML est fondé est le métamodèle. Un métamodèle peut être définit comme une description formelle de tous les concepts d'un langage. Il permet de définir non seulement les éléments de modélisation, mais aussi la sémantique de ces éléments (i.e. leur définition et le sens de leur utilisation), ce qui limite grandement l'ambigüité et incite la construction d'outils. Un métamodèle UML définit la structure que doit avoir tout modèle UML. En d'autre terme, un modèle UML est tout modèle qui est conforme au métamodèle UML. Il faut noter qu'un métamodèle UML est lui-même décrit par un méta-métamodèle.

4.1.2. Meta Object Facility (MOF)

MOF est un formalisme de modélisation. Un formalisme définit les éléments de modélisation nécessaire à l'élaboration des modèles ainsi que les relations entre ces éléments [Mof 10]. On peut donc faire une analogie entre le formalisme de modélisation et le métamodèle. MOF est considéré ainsi comme un métaformalisme dans le sens où il offre la possibilité de modéliser les formalismes de modélisation eux-mêmes (i.e. un formalisme de modèles de formalisme). MOF est appelé aussi un méta-métamodèle. On distingue donc trois niveaux de modélisation : le niveau modèle, le niveau métamodèle et le niveau méta-métamodèle. MDA ne monte pas dans les niveaux méta car il suffit d'utiliser un métamétamodèle MOF pour l'exprimer lui-même. En d'autres termes, un niveau méta plus haut n'est plus nécessaire.

La figure 4.3 illustre les différents nivaux de MDA qui forment une architecture appelée « architecture MDA à quatre niveaux ».

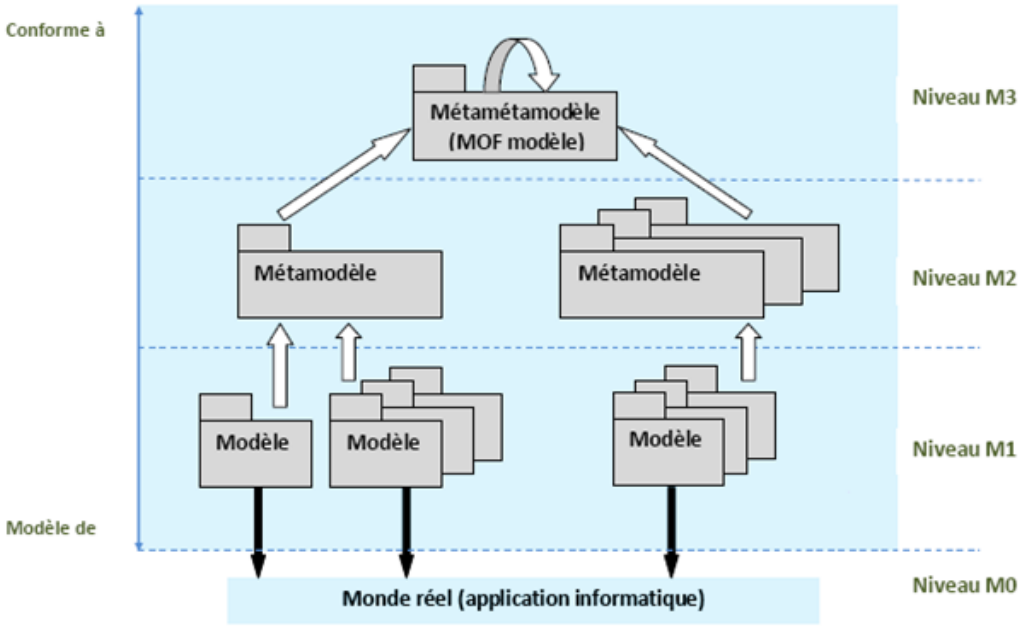


Figure 4.3. L'architecture à quatre niveaux de MDA

Le niveau M0 contient les entités du monde réel à modéliser. Le niveau M1 comprend les différents modèles décrivant les entités du monde réels. Dans le niveau M2 se trouve les métamodèles exprimant les modèles du niveau M1. Le niveau M3 contient le méta-métamodèle qui permet non seulement de définir les métamodèles du niveau M2 mais aussi de décrire le niveau M3 lui-même. Il reste à noter que MDA considère tous les éléments appartenant aux niveaux M1, M2 et M3 comme des modèles.

MOF représente les métamodèles sous forme de diagrammes de classes. Pour des raisons de différentiation, les classes qui constituent un métamodèle sont appelées métaclasses, et les objets instances des métaclasses sont appelés méta-objets. Une métaclasse possède un nom et contient des attributs et des opérations, aussi appelés respectivement méta-attributs et méta-opérations. Les métaclasses sont reliées par des méta-associations. Les méta-attributs et les paramètres des méta-opérations ayant un type de donnée (dataType) qui peut être un booléen,

un tableau, une chaîne de caractères, etc. La construction d'un type non primitif est possible grâce aux constructions fournies par MOF (énumérations, alias et structures). Les navigations entres les méta-objets est possible grâce au concept de référence. Il permet de naviguer à travers les méta-associations. Le concept de package (métapackage) regroupe les différents éléments d'un métamodèle.

Les concepts cités précédemment possèdent un nombre de propriétés :

- La métaclasse : a un nom, peut hériter d'une ou plusieurs classes, peut être abstraite, peut être considérée comme racine (root) ou feuille (leaf) d'un arbre d'héritage, etc.
- Le méta-attribut : a un nom, a un type (métaclasse ou type de données), possède une portée (scope), peut être ou non modifiable (IsChangeable), peut être considéré comme dérivé (isDerived), possède une multiplicité (multiplicity), peut être ordonné ou non (is_ordered) et l'information is_unique spécifie que l'ensemble des valeurs du méta-attribut ne contient pas de doublons.
- La méta-opération : possède un nom, possède une portée, contient zéro ou plusieurs métaparamètres, peut définir un type de retour (métaclasse ou type de données), peut générer une ou plusieurs exceptions, etc.
- La méta-association : possède un nom, contient forcément deux extrémités, etc.
- Le métapackage : a un nom, peut contenir zéro ou plusieurs métaclasses, contient zéro ou plusieurs autres métapackages, méta-associations qui relient les métaclasses et types de données nécessaires, peut hériter d'un ou plusieurs autres métapackages, peut importer un autre package, etc.
- Référence : a un nom, un type, une multiplicité, etc.

Vu qu'il n'existe pas une notation spécifique pour élaborer un métamodèle, on peut utiliser la notation UML (diagramme de classes UML). La figure 4.4 représente une partie de MOF (version1.4) sous forme de diagramme de classes. Ce diagramme définit la structure que doit avoir tout métamodèle, c'est-à-dire, il représente le métamétamodèle (MOF modèle).

Dans sa version 2.0, le métamodèle MOF est constitué de deux parties : EMOF (Essential MOF), pour l'élaboration des métamodèles sans association, et CMOF (Complete MOF) pour les métamodèles avec associations. Comme il est presque impossible de faire la distinction entre un diagramme de classes représentant un métamodèle MOF et un diagramme de classes représentant un modèle UML, il devient évident d'essayer de rapprocher ces deux standard au niveau des diagrammes de classes ce qui contribue à l'élaboration d'un métamodèle commun entre MOF et UML nommé UML2.0 Infrastructure [Bar 05].

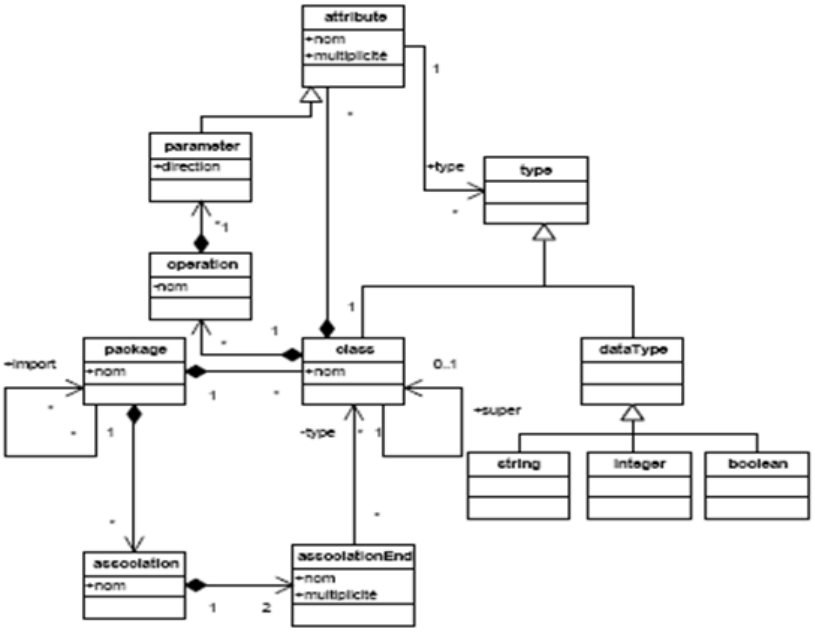


Figure 4.4. MOF (v1.4) sous forme de diagramme de classe [Bla 05].

En résumé, MOF est un standard de méta-modélisation qui définit les éléments nécessaires de modélisation, la structure et la syntaxe des métamodèles utilisés pour réaliser des modèles.

4.1.3. Object Constraint Language (OCL)

OCL est un langage qui vise l'ajout de contraintes pour exprimer la sémantique statique des modèles et métamodèles [Ocl 05]. Un métamodèle (ou modèle) n'a pas toujours l'expressivité suffisante pour décrire toutes les contraintes sur les méta-éléments qu'il contient. L'OMG préconise l'utilisation du langage textuel OCL pour décrire des contraintes non capturées dans le métamodèle, sous forme d'expressions. L'ajout de ces contraintes permet de créer des modèles et métamodèles clairement et précisément définies et non ambigües. Les expressions OCL sont rattachées à un contexte (i .e. un élément de modélisation) et peuvent être évaluées pour retourner une valeur (vraie ou faux).

Elles n'ont aucun effet de bord et ne permettent ni de créer ni de supprimer ou modifier les objets d'un modèle. Les expressions OCL servent dans plusieurs situations :

- Dans le contexte d'une class UML, par exemple, une contrainte OCL peut être utilisée pour exprimer un invariant (mot-clé inv). Cela signifie que la contrainte OCL doit être tout le temps vraie pour toutes les instances de la classe.
- OCL fournit la possibilité de définir des pré-conditions (mot-clé pre) et post-conditions (mot-clé post) sur les opérations pour que le système modélisé reste dans un état cohérent quand on exécute ces opérations.
- OCL permet de contraindre la valeur retournée par une opération ou une méthode, et rend possible l'obtention de la valeur d'un attribue d'une classe (ou d'un méta-attribue d'une méta-classe).
- A travers le concept de collection, le langage OCL facilite la description des ensembles d'éléments et leur manipulation en fournissant un ensemble de fonctions. Les collections OCL ayant plusieurs sortes selon l'ordre et le nombre d'apparition des

éléments constituant l'ensemble décrit par la collection (Bag, Set, OrderedSet et Sequence) [Ocl 05].

OCL fournit donc une bonne solution pour exprimer des contraintes sur les modèles mais pas pour décrire le comportement ou la sémantique des modèles (i.e. l'état de modèle reste inchangé). Il est important de noter que le langage OCL est indépendant de toute plateforme d'exécution et les contraintes OCL sont indépendantes des langages de programmation.

4.1.4. XML Metadata Interchange (XMI)

XMI est un standard qui permet de représenter concrètement des modèles sous forme de documents XML bien formé. Un document XML bien formé est construit moyennant le langage XML (eXtensible Markup Language) qui fournit un ensemble de concepts (e.g. les balises XML) permettant la structuration du document. Le document XML est un document textuel et est dit bien formé dans le sens où il est bien structuré. La traduction d'un modèle en un document XML est appelée sérialisation du modèle au format XML.

XMI fournit la possibilité de décrire les métamodèles en utilisant les DTDs XML (DTD pour Document Type Definitions). Une DTD XML est un document textuel qui définit comment les balises XML sont structurées dans un document XML ainsi que leurs relations d'inclusion. Chaque modèle représenté par un document XML est conforme à une DTD correspondante (métamodèle). XMI définit un ensemble de règles qui permettent de générer automatiquement une structuration de balises XML (e.g. DTD) à partir d'un métamodèle. La figure 5 illustre les opérations de généralisation et sérialisation [Bla 05].

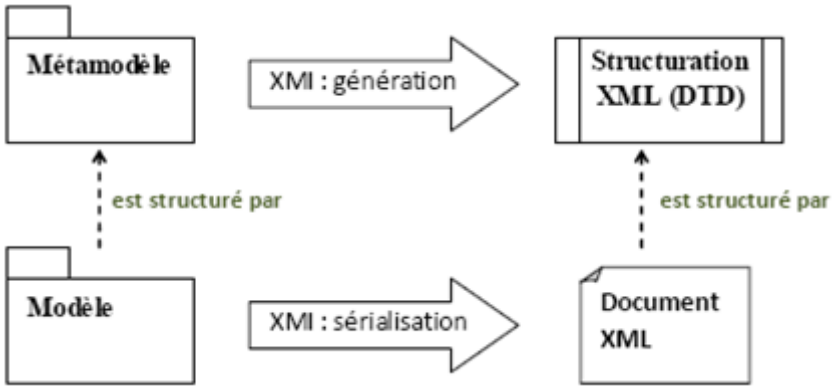


Figure 4.5. XMI et la structuration de balises XML [Bla 05]

4.1.5. Common Warehouse Metamodel (CWM)

CWM est un standard qui permet la définition des métamodèles qui décrivent à la fois des métadonnées métiers et des métadonnées techniques. Il facilite l'échange des métadonnées et les entrepôts de données (Warehouse), où se trouvent les métadonnées, entre les plateformes supports et les outils des entrepôts de données. CWM apporte l'interopérabilité entre divers logiciels par l'utilisation du format CWM de tel sorte qu'un logiciel émetteur traduit ses métadonnées vers le format CWM, et un logiciel récepteur extrait ces métadonnées par la traduction du format CWM reçu.

Ces standards sur lesquels l'approche MDA est basée, s'enrichissent mutuellement et chacun peut apporter des concepts et des éléments dont les autres ont besoin. Le MOF fournit le concept de métamodèle et de métamétamodèle via lesquels nous pouvons définir la structure et la syntaxe de tous les métamodèles comme les métamodèle UML et CWM. Ainsi, MOF utilise le diagramme de classes défini par UML pour représenter les métamodèles. OCL, à son tour, enrichit les diagrammes UML par une puissance d'expression afin de décrire des contraintes qu'UML ne permet pas de d'exprimer. OCL peut aussi être utilisé aux niveaux méta. Vu que les modèles sont des entités abstraites qui ne peuvent être ni stockées ni échangées sur un support informatique, XMI propose des solutions convenables pour résoudre ce problème que les standards MOF et UML ne résolvent pas [Bla 05].

4.2. Processus MDA

Le principe clé de MDA consiste à utiliser les modèles aux différentes étapes du cycle de développement d'un système logiciel. MDA préconise la réalisation de différents types de modèles : les modèles d'exigences (CIM), les modèles d'analyse et de conception (PIM) et les modèles de code (PSM) [Bla 05].

4.2.1. Le modèle d'exigences Computation Independent Model (CIM)

La première chose à faire lors de la construction d'une nouvelle application est bien entendu de spécifier les exigences du client. Bien que très en amont, cette étape doit fortement bénéficier des modèles.

L'objectif est de créer un modèle d'exigences de la future application. Un tel modèle doit représenter l'application dans son environnement afin de définir quels sont les services offerts par l'application et quelles sont les autres entités avec lesquelles elle interagit.

La création d'un modèle d'exigences est d'une importance capitale. Cela permet d'exprimer clairement les liens de traçabilité avec les modèles qui seront construits dans les autres phases du cycle de développement de l'application, comme les modèles d'analyse et de conception. Un lien durable est ainsi créé avec les besoins du client de l'application.

Les modèles d'exigences peuvent même être considérés comme des éléments contractuels, destinés à servir de référence lorsqu'on voudra s'assurer qu'une application est conforme aux demandes du client.

Il est important de noter qu'un modèle d'exigences ne contient pas d'information sur la réalisation de l'application ni sur les traitements. C'est pourquoi, dans le vocabulaire MDA, les modèles d'exigences sont appelés des CIM (Computation Independent Model), littéralement « modèle indépendant de la programmation ».

Avec UML, un modèle d'exigences peut se résumer à un diagramme de cas d'utilisation.

Ces derniers contiennent en effet les fonctionnalités fournies par l'application (cas d'utilisation) ainsi que les différentes entités qui interagissent avec elle (acteurs) sans apporter d'information sur le fonctionnement de l'application.

Dans une optique plus large, un modèle d'exigences est considéré comme une entité complexe, constituée entre autres d'un glossaire, de définitions des processus métier, des exigences et des cas d'utilisation ainsi que d'une vue systémique de l'application.

Si MDA n'émet aucune préconisation quant à l'élaboration des modèles d'exigences, des travaux sont en cours pour ajouter à UML les concepts nécessaires pour couvrir cette phase amont.

Le rôle des modèles d'exigences dans une approche MDA est d'être les premiers modèles pérennes. Une fois les exigences modélisées, elles sont censées fournir une base contractuelle variant peu, car validée par le client de l'application. Grâce aux liens de traçabilité avec les autres modèles, un lien peut être créé depuis les exigences vers le code de l'application [Bla 05].

4.2.2. Le modèle d'analyse et de conception abstraite Platform Independent Model (PIM)

Une fois le modèle d'exigences réalisé, le travail d'analyse et de conception peut commencer. Dans l'approche MDA, cette phase utilise elle aussi un modèle.
L'analyse et la conception sont depuis plus de trente ans les étapes où la modélisation est la plus présente, d'abord avec les méthodes Merise et Coad/Yourdon puis avec les méthodes objet Schlear et Mellor, OMT, OOSE et Booch. Ces méthodes proposent toutes leurs propres modèles. Aujourd'hui, le langage UML s'est imposé comme la référence pour réaliser tous les modèles d'analyse et de conception.

Par conception, il convient d'entendre l'étape qui consiste à structurer l'application en modules et sous-modules. L'application des patrons de conception, ou Design Patterns, du GoF (Gang of Four) fait partie de cette étape de conception. Par contre, l'application de patrons techniques, propres à certaines plates-formes, correspond à une autre étape. Nous ne considérons donc ici que la conception abstraite, c'est-à-dire celle qui est réalisable sans connaissance aucune des techniques d'implémentation [Bla 05].

4.2.3. Le modèle de code ou de conception concrète Platform Specific Model (PSM)

Une fois les modèles d'analyse et de conception réalisés, le travail de génération de code peut commencer. Cette phase, la plus délicate du MDA, doit elle aussi utiliser des modèles. Elle inclut l'application des patrons de conception techniques.

MDA considère que le code d'une application peut être facilement obtenu à partir de modèles de code. La différence principale entre un modèle de code et un modèle d'analyse ou de conception réside dans le fait que le modèle de code est lié à une plate-forme d'exécution.

Dans le vocabulaire MDA, ces modèles de code sont appelés des PSM (Platform Specific Model).

Les modèles de code servent essentiellement à faciliter la génération de code à partir d'un modèle d'analyse et de conception. Ils contiennent toutes les informations nécessaires à l'exploitation d'une plate-forme d'exécution, comme les informations permettant de manipuler les systèmes de fichiers ou les systèmes d'authentification.

Il est parfois difficile de différencier le code des applications des modèles de code. Pour MDA, le code d'une application se résume à une suite de lignes textuelles, comme un fichier Java, alors qu'un modèle de code est plutôt une représentation structurée incluant, par exemple, les concepts de boucle, condition, instruction, composant, événement, etc.
L'écriture de code à partir d'un modèle de code est donc une opération assez triviale.

Pour élaborer des modèles de code, MDA propose, entre autres, l'utilisation de profils UML. Un profil UML est une adaptation du langage UML à un domaine particulier. Par exemple, le profil UML pour EJB est une adaptation d'UML au domaine des EJB. Grâce à ce profil, il est possible d'élaborer des modèles de code pour le développement d'EJB.

Le rôle des modèles de code est principalement de faciliter la génération de code. Ils sont donc essentiellement productifs mais ne sont pas forcément pérennes. L'autre caractéristique importante des modèles de code est qu'ils font le lien avec les plates-formes d'exécution. Cette notion de plate-forme d'exécution est très importante dans MDA car c'est elle qui délimite la fameuse séparation des préoccupations [Bla 05].
Le processus MDA (figure 4.6) consiste en une succession d'étapes dont les modèles et les transformations des modèles jouent un rôle principale : construction du CIM, transformation en PIM, transformation en PSM, puis génération du code [Mes 06].

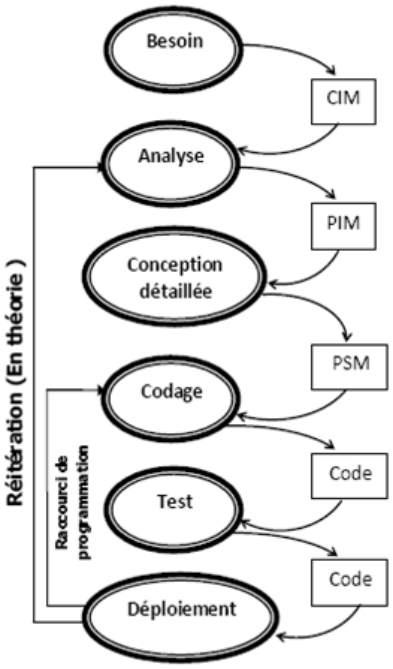


Figure 4.6. Processus de développement MDA [Mes 06]

- Création de CIM. L'élaboration d'une nouvelle application commence toujours par une étape de spécification des exigences. Cette étape conduit à la construction d'un ou plusieurs modèles d'exigence (CIM).
- Création de PIM. Une fois les modèles CIMs créés, on arrive à l'étape d'analyse et de conception abstraite. A ce niveau de développement, les modèles PIMs sont construis partiellement à partir des CIMs. C'est dans cette étape que la logique métier de l'application est exprimée.
- Création de PSM. Dans l'étape de conception détaillée, un mapping de PIM vers une plateforme d'implémentation est effectué. Le résultat de cette étape est la création des PSMs par la transformation des PIM en ajoutant les informations concernant la plateforme.
- Génération du code. Les modèles de code élaborés à l'étape précédente sont suffisamment détaillés pour faciliter la génération du code (i.e. traduction en un formalisme textuel). Grâce à des outils de transformation, la génération du code peut être effectuée automatiquement.

Les avantages du processus MDA par apport au processus de développement classique résident dans le fait que l'utilisation de modèle comme un point de référence tout au long du cycle de développement a grandement consacrée la séparation des préoccupations, ce qui conduit à une pérennisation de la logique métier des entreprises. Ainsi, la possibilité d'automatiser les transformations de modèles a remarquablement augmenté la productivité de cette logique métier. Le passage de la conception à l'implémentation est plus facile grâce à l'intégration des informations concernant les plateformes d'exécution dans les transformations de modèles. Les transformations jouent donc un rôle central dans les processus MDA.

5. Transformation de modèles

Les transformations de modèles sont au cœur de l'approche MDA. OMG définit la transformation de modèles dans le contexte de MDA par le processus qui convertit un modèle à un autre modèle d'un même système [Bie 10]. Une définition plus générale considère la transformation de modèles comme une fonction qui prend en entrée un ou plusieurs modèles sources et qui fournit en sortie un ou plusieurs modèles cibles [Bie 10]. Les éléments qui composent une transformation de modèles sont illustrés à la figure 4.7.

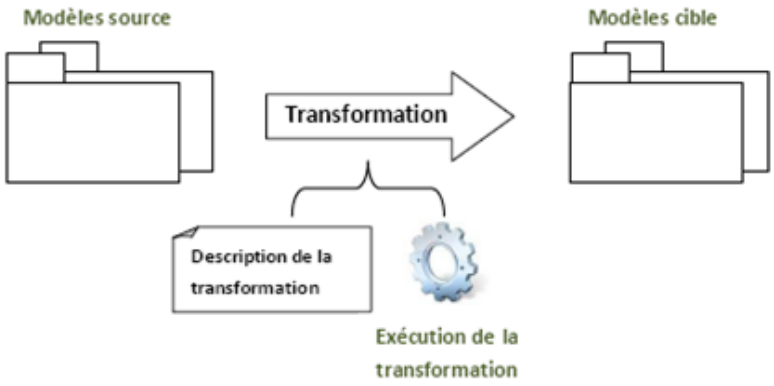


Figure 4.7. Les éléments d'une transformation de modèles

Toute transformation de modèles doit contenir les éléments suivants [Bie 10]:

- Modèle source. Chaque transformation peut avoir un ou plusieurs modèles sources en entrée. Le modèle source est conforme à un métamodèle source.
- Modèle cible. Chaque transformation peut avoir un ou plusieurs modèles cibles. Le modèle cible est conforme à un métamodèle cible. Le terme modèle cible est seulement utilisé dans le contexte d'une transformation de type modèle-à-modèle.
- Description de la transformation. La description de la transformation est écrite dans un langage de transformation de modèles qui fournit un vocabulaire et une grammaire avec une sémantique bien définie pour réaliser des transformations de modèles. Le langage de transformation de modèles est basé sur une approche de transformation qui peut être impérative, opérationnelle, fonctionnelle, déclarative ou relationnelle. La plupart des langages de transformation sont basés sur des règles. En d'autre terme, la transformation de modèle n'est qu'un ensemble de règles de transformation.
- Exécution de la transformation. L'exécution d'une transformation de modèles peut être effectuée par un outil de transformation (e.g. SmartQVT, ATL, etc) qui interprète la description de la transformation. Il consiste à appliquer la description sur le modèle source pour obtenir le modèle cible. Les étapes typiques de l'exécution sont :
    - Identifier les éléments à transformer dans le modèle source.
    - Pour chaque élément identifié, produire l'élément cible associé.
    - Produire l'information de traçabilité qui relie les éléments source et cible affectés et la règle de leur transformation.

La description de la transformation peut être exprimée par un modèle. MDA préconise la modélisation des transformations de modèles elles-mêmes. Tout comme les modèles d'entrée et de sortie, un modèle de transformation est conforme, à son tour, à un métamodèle de transformation qui permet d'exprimer les liens entre les concepts des métamodèles d'entrées et celles des métamodèles de sortie. La figure 8 montre les différents niveaux méta d'une transformation de modèles.

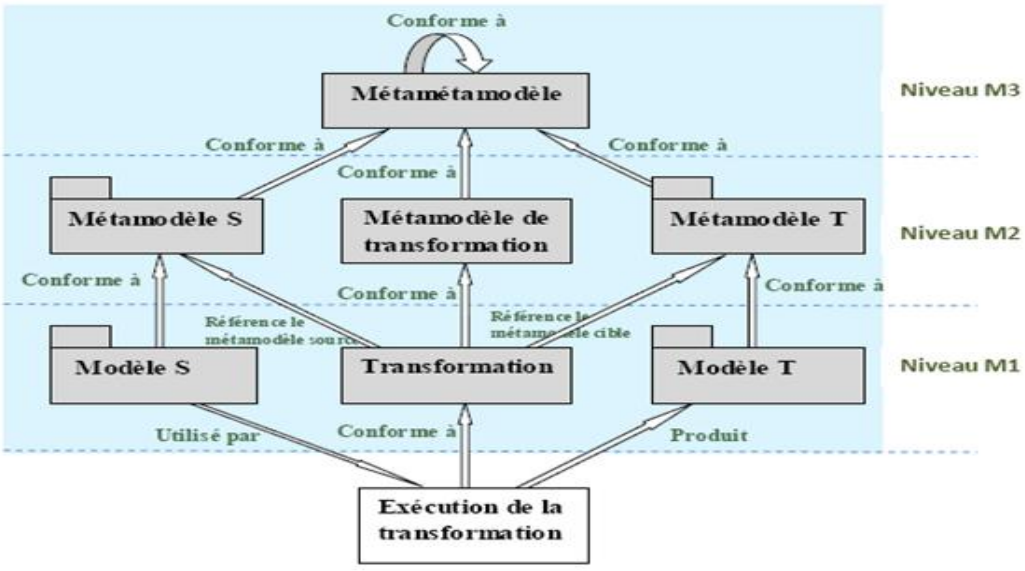


Figure 4.8. Les Niveaux méta de la transformation de modèles

6. Classification des approches de transformation

Les approches de transformation de modèle ont été classées en se basant sur plusieurs points de vue. Chaque point de vue permet une classification particulière. Selon [Jou 05], il existe deux types de transformation de modèle : les transformations de type *«modèle vers modèle»* et les transformations de type *«modèle vers code»*.

6.1. Transformations de type modèle vers modèle

Les transformations de type modèle vers modèle sont aujourd'hui moins maîtrisées. Cependant, elles ont beaucoup évoluées au cours de ces dernières années, et plus particulièrement depuis l'apparition de MDA [Sol 00].
Quand on trouve un grand espace d'abstraction entre un PIM et un PSM, il serait plus facile de générer des modèles intermédiaires au lieu d'aller directement vers le PSM cible. Ces modèles peuvent être utiles pour l'optimisation ou bien pour des buts de débogage. Ces transformations sont utiles pour le calcul des différentes vues du système et leur synchronisation et aussi pour des buts de vérification et de validation.

Une transformation de modèle est principalement caractérisée par la combinaison des éléments suivants: des règles de transformation, une relation entre la source et la cible, un ordonnancement des règles, une organisation des règles, une traçabilité et une direction. Ces éléments sont expliqués dans ce qui suit.

6.1.1. Spécification

Certaines approches de transformation fournissent un mécanisme dédié de spécifications, tel que des pré-conditions et des post-conditions exprimées en OCL. Des spécifications particulières de transformation peuvent représenter une fonction entre les modèles source et cible et peuvent être exécutables. Mais généralement, les spécifications décrivent des relations et elles ne sont pas exécutables [Sol 00].

6.1.2. Les règles de transformation

Une règle de transformation définit la manière dont un ensemble de concepts du méta-modèle source est transformé en un ensemble de concepts du méta-modèle destination. Une règle de transformation est composée de deux parties : un côté gauche (LHS, Left Hand Side) qui accède au modèle source, et un côté droit (RHS, Right Hand Side) qui accède au modèle cible. Une règle comporte également une logique, qui exprime des contraintes ou calculs sur les éléments des modèles source et cible.

Une logique peut avoir la forme déclarative ou impérative. Une logique déclarative consiste à spécifier des relations entre les éléments du modèle source et des éléments du modèle cible. Une logique impérative correspond le plus souvent, mais pas nécessairement, à l'utilisation de langages de programmation pour manipuler directement les éléments des modèles par le biais d'interfaces dédiées. Optionnellement, une règle, peut être bidirectionnelle, peut comporter des paramètres, ou encore peut nécessiter la construction de structures intermédiaires [Sol 00].

6.1.3. Organisation des règles

Les règles peuvent être organisées selon une structure dépendante du modèle source ou du modèle cible [Sol 00].

6.1.4. Ordonnancement des règles

L'ordonnancement des règles peut se baser également sur des conditions, des itérations ou sur une séparation en plusieurs phases et certaines règles ne pouvent être appliquées que dans certaines phases [Sol 00].

6.1.5. Relation entre les modèles source et cible

Pour certains types de transformations, la création d'un nouveau modèle cible est nécessaire. Pour d'autres, la source et la cible sont le même modèle, ce qui revient en fait à une modification de modèle.

6.1.6. Augmentation

Elle a un rapport avec la capacité de mettre à jour les modèles cibles existants qui sont basés sur des changements dans les modèles source.

6.1.7. Direction.

Les transformations peuvent être unidirectionnelles ou bidirectionnelles. Dans le premier cas, le modèle cible est calculé ou mis à jour sur la base du modèle source uniquement. Dans le second cas, une synchronisation entre les modèles source et cible est possible [Sol 00].

6.1.8. Traçabilité

Les transformations peuvent archiver les corrélations entre les éléments des modèles source et cible. Certaines approches fournissent des mécanismes dédiés pour supporter la traçabilité. Dans les autres cas, le développeur doit implémenter la traçabilité de la même manière qu'il crée n'importe quel autre lien dans un modèle [Sol 00].

Pour ce type de transformation, nous distinguons cinq types d'approches, ce sont :

- Approche par manipulations directes.
- Approche relationnelle.
- Approche basée sur la transformation de graphes.
- Approche basée sur la structure.
- Approche hybride.

6.2. Transformations de type modèle vers code

Ce type de transformation peut être considéré comme un cas particulier de la transformation de type «modèle vers modèle» (il suffit de fournir un méta-modèle pour le langage de programmation cible).

Dans ce type de transformation, il existe deux approches de transformations : les approches basées sur le principe du *visiteur (Visitor-based approach)* ou celles basées sur le principe des *patrons (Template-based approach).*

- Les approches, reposant sur le principe du «visiteur», consistent à traverser le modèle en lui ajoutant des éléments (mécanismes visiteurs) qui réduisent la différence de sémantique entre le modèle et le langage de programmation cible. Le code est obtenu en parcourant le modèle enrichi pour créer un flux de texte.
- Les approches, basées sur le principe des «patrons», sont actuellement les plus utilisées. Le code cible contient des morceaux de méta-code utilisés pour accéder aux informations du modèle source. La majorité des outils MDA couramment disponibles supportent ce principe de génération de code à partir de modèles. Parmi les outils basés sur ce principe, nous pouvons citer : *OptimalJ et XDE* (qui fournissent la transformation modèle vers modèle aussi), *JET, ArcStyler* et *AndroMDA* (des générateurs de code qui reposent notamment sur la technologie ouverte *Velocity* pour l'écriture des patrons).

Les travaux réalisés dans le domaine de la transformation de modèle ne sont pas récents et peuvent être chronologiquement classés selon plusieurs générations en fonction de la structure de données utilisée pour représenter le modèle, il existe trois générations qui sont: Transformation de structures séquentielles d'enregistrement, Transformation d'arbres et Transformation de graphes.

6.3. Plan de classification pour les problèmes de transformation de modèles

Les langages de transformation de modèles ont été classifiés par Czarnecki et al [Cza 06], et par Mens et al [Men 06]. Ainsi, différents outils de transformation de modèles ont été évalués par Huber [Hub 08]. Il a conclu qu'il n'existe pas un outil de modélisation absolument meilleur par rapport aux autres outils. Presque pour chaque problème de transformation de modèles, il existe un outil de transformation de modèles correspondant. Pour cette raison, on définit un plan de classification de ces problèmes. On désigne par problème de transformation de modèles le problème qu'on veut résoudre en utilisant une transformation de modèles. La classification peut être utile pour le choix du langage et l'outil de transformation les plus appropriés pour résoudre un problème donné. La classification est basée sur un ensemble de propriétés qui caractérisent les problèmes de transformation [Bie 10].

6.3.1. Changement d'abstraction

La transformation de modèles peut changer le niveau d'abstraction entre le modèle source et le modèle cible. Selon le type de transformation, Le niveau d'abstraction des modèle peut être changé ou pas. On distingue deux types de transformation (figure 4.9) :

- La transformation verticale qui change le niveau d'abstraction. Le changement peut réduire le niveau d'abstraction par l'ajout de détails au modèle source, on parle d'une transformation de raffinage. Le changement peut aussi augmenter le niveau d'abstraction par la suppression des détails, on parle d'une transformation d'abstraction.
- La transformation horizontale ne change pas le niveau d'abstraction du modèle. Cependant, elle peut modifier sa représentation (un métamodèle différent).

6.3.2. Changement de métamodèle

Selon le changement de métamodèle, on définit deux types de transformation (figure 4.9):

- Transformation endogène Les modèles source et cible de la transformation sont conformes au même métamodèle.
- Transformation exogène Les modèles source et cible de la transformation sont conformes à des métamodèles différents.

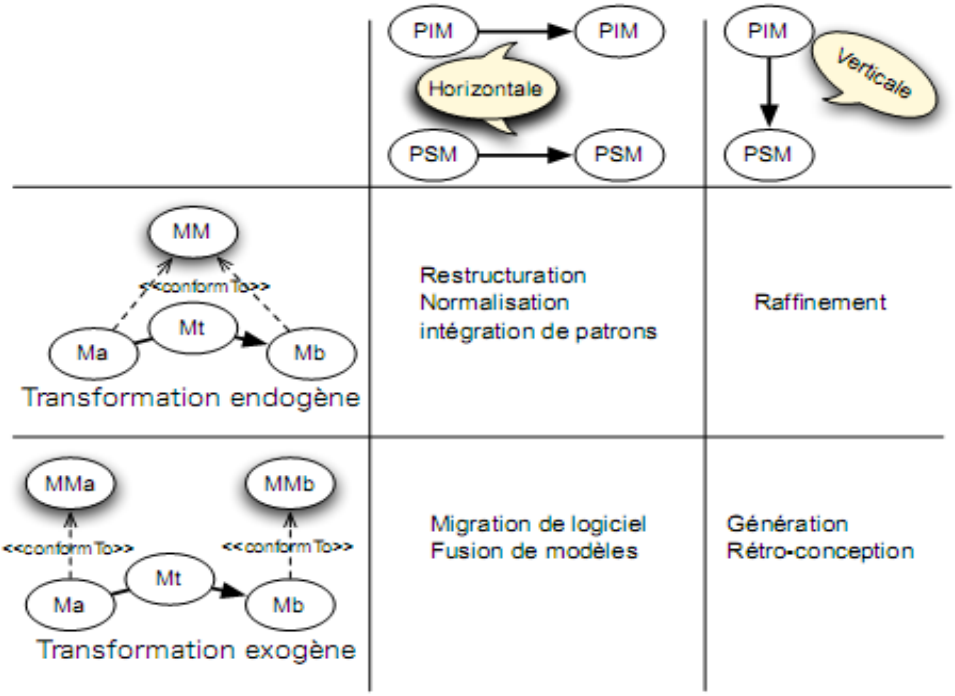


Figure 4.9.Types de transformation et leurs principales utilisations

6.3.3. Le nombre de modèles

Une transformation de modèle peut avoir plusieurs modèles source et plusieurs modèles cibles. Le nombre minimum de modèles inclus dans la transformation est égale à un. La transformation ne change qu'une partie spécifique du modèle source (In-place transformation). La majorité des transformations incluent deux modèles. Cette transformation génère un modèle cible différent du modèle source (Out-place transformation). Le dernier type de transformation peut inclure plusieurs modèles sources et combine les informations trouvées dans ces modèles pour produire le modèle cible (intégration). Plusieurs modèles cibles peuvent aussi être produits par ce genre de transformation [Bie 10].

6.3.4. L'espace technologique

Les modèles sont représentés en utilisant différents espaces technologiques [Bie 10]. Exemples : XML (XSD), MOF (OMG), EMF (Eclipse), DSL (Microsoft), etc.

6.3.5. Le type cible

Le type cible d'une transformation peut être soit un modèle, ou bien, un texte.

6.3.6. Préservation des propriétés

Parfois, les modèles source et cible d'une transformation possèdent une propriété commune qui n'est pas modifiée par la transformation. La propriété peut être de divers types :

- Préservation de la sémantique. Elle consiste à préserver le sens des modèles (Les métamodèles source et cible sont similaires). Par exemple, la transformation qui vise l'amélioration des performances ou le refactoring est une transformation qui préserve la sémantique des modèles.
- Préservation du comportement. La transformation d'un modèle à un code est une transformation qui préserve le comportement mais pas la sémantique.
- Préservation de la syntaxe. La syntaxe définit une notation ou un format selon lequel le modèle est représenté. Généralement, la transformation qui préserve la syntaxe est horizontale et endogène.

7. Spécification des règles de transformation

Ce qui différencie les approches permettant l'élaboration des transformations de modèles est la façon dont sont spécifiées les règles des transformations. Aujourd'hui, trois approches répertoriées permettent la spécification de ces règles de correspondance. Nous les décrivons dans ce qui suit.

7.1. Approche par programmation

Consiste à utiliser les langages de programmation orientée objet. L'idée est de programmer une transformation de modèles de la même manière que l'on programme n'importe quelle application informatique. Ces transformations sont donc des applications informatiques qui ont la particularité de manipuler des modèles. Ces applications informatiques utilisent les interfaces de manipulation de modèles. Cette approche est la plus utilisée car elle est très puissante et fortement outillée [Bla 05].

7.2. Approche par template

Consiste à définir les canevas des modèles cibles souhaités en y déclarant des paramètres. Ces paramètres seront substitués par les informations contenues dans les modèles sources. On appelle ces canevas des « modèles cibles paramétrés» ou des « modèles templates ». L'exécution d'une transformation consiste à prendre un modèle template et à remplacer ses paramètres par les valeurs d'un modèle source. Cette approche nécessite un langage particulier permettant la définition des modèles Template. Sa puissance réside principalement dans le langage de définition des modèles Template. De tels langages sont en cours d'élaboration et n'ont pas encore la maturité des langages de programmation orientés objets utilisés dans la première approche [Bla 05].

7.3. Approche par modélisation

Consiste à appliquer les concepts de l'ingénierie des modèles aux transformations des modèles elles-mêmes. L'objectif est de modéliser les transformations de modèles et de rendre

les modèles de transformation pérennes et productifs et d'exprimer leur indépendance vis-à-vis des plates-formes d'exécution [Bla05].

8. Outils et Langages de transformation de modèles

Plusieurs outils et langages sont proposés pour réaliser les transformations et les mettre en œuvre [Bla 05]. Nous décrivons ci-après quelques uns.

8.1.1. Query View Transformation (QVT)

QVT est un langage standardisé de transformation de type modèle-à-modèle (model-to-model transformation language) adopté par OMG en 2005 [QVT]. Il permet la définition de métamodèle permettant l'élaboration des modèles de transformation. QVT permet de :

- Soumettre des requêtes sur des modèles (Query). Les requêtes permettent la sélection des éléments sur un modèle. Le langage utilisé par QVT est OCL légèrement modifié et étendu avec une syntaxe différente et simplifiée.
- Produire des vues des métamodèles (modèles déduits d'un autre pour en révéler des aspects spécifiques). Elles peuvent être définies en utilisant une requête. Une vue est éventuellement conforme à un métamodèle restreint spécifique à cette vue.
- Opérer des transformations sur les modèles.

La syntaxe abstraite de QVT (métamodèle) est décrite en MOF et sa syntaxe concrète est textuelle ou graphique [Bla 05]. QVT supporte plusieurs scenarii d'exécution (transformations unidirectionnelles, transformations bidirectionnelles, etc.).

Grâce à une architecture hybride, QVT propose les avantages combinés des approches déclaratives et des approches impératives. De part ses capacités d'expression, la partie déclarative permet dans de nombreux cas courant d'écrire facilement des transformations entre modèles, en particulier lorsque la transformation est constituée d'un ensemble de règles relativement simples. Néanmoins certains types de transformations se prêtent mal à un style déclaratif. Pour ce genre de transformation, QVT propose alors un style impératif.

Pour un style déclaratif, QVT utilise deux langages différents. Un langage de haut niveau nommé Relations qui consiste à effectuer une correspondance entre les éléments des modèles source et cible de la transformation, et un langage de bas niveau nommé Core qui est plus simple mais avec le même pouvoir d'expression de transformation que Relations.

Pour un style impératif, QVT utilise le langage impératif Operational Mappings qui permet la mise en œuvre d'une relation et l'ajout de primitives déclaratives inspirées en partie d'OCL. Operational Mappings définit des transformations unidirectionnelles exprimées de manière impératives. Il définie une signature indiquant les modèles impliqués dans la transformation et il définit une opération principale pour son exécution (main). Ce langage étend le langage Relations avec des constructions impératives et des expressions OCL à effets de bords. QVT inclut ainsi une implémentation nommée Black Box (boite noire) qui permet de connecter et d'exécuter du code externe durant l'exécution de la transformation. Ce mécanisme permet d'implémenter des algorithmes complexes et aussi la réutilisation de bibliothèques externes

[Bla 05]. La figure 10 illustre l'architecture de QVT. Dans la littérature, beaucoup d'outils implémentent le standard QVT. Parmi eux, nous citons : SmartQVT, MediniQVT, etc [Dia 09].

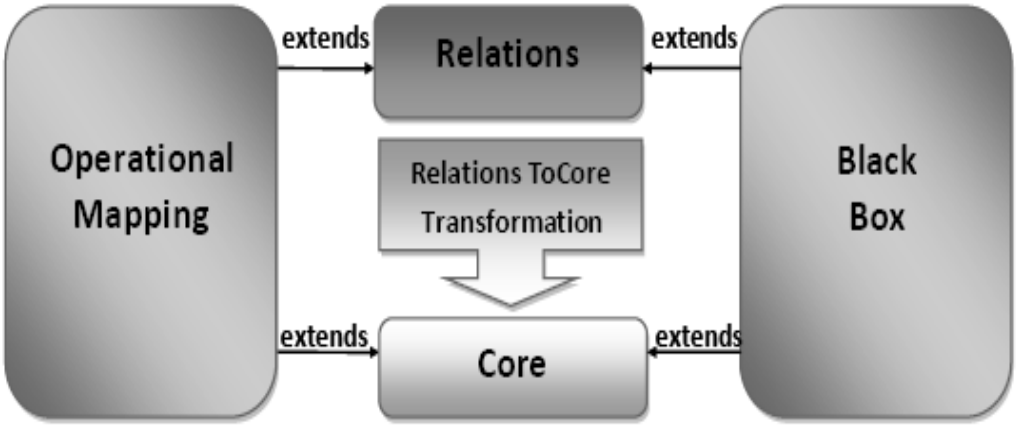


Figure 4.10. Architecture de QVT

8.1.2. Atlas Transformation Language (ATL)

ATL est un langage de transformation de type modèle-à- modèle proposé par le groupe ATLAS [Atl 06]. ATL est considéré comme un langage hybride puisqu'il mélange les deux types de constructions déclarative et impérative. Une transformation ATL est composée d'un ensemble de règles qui décrivent comment initialiser et créer les éléments constituant un modèle cible. Le langage est défini par un modèle MOF pour sa syntaxe abstraite et possède également une syntaxe concrète présentée sous forme textuelle.

ATL utilise les expressions OCL à effet de bord pour manipuler et accéder aux éléments d'un modèle. Ces expressions peuvent retourner une valeur de type simple ou complexe (élément d'un modèle, collection, etc.). Elles permettent la navigation entre les éléments d'un modèle et d'appeler des opérations sur ceux-ci. La navigation ne peut se faire que sur des éléments initialisés, on ne pourra jamais naviguer sur des éléments du modèle cible.

L'exécution d'une transformation est effectuée par un moteur de transformation ATL qui est une machine virtuelle. Comme la Java Virtual Machine, la machine virtuelle ATL dispose de son propre jeu d'instruction. Elle est indépendante du langage de transformation ATL et ne s'intéresse qu'au code de la transformation compilée.

Un modèle de transformation ATL est conforme au métamodèle ATL (Figure 4.11). La définition d'un modèle de transformation se fait dans un fichier portant l'extension " .atl ". L'entête de fichier se déclare de la manière suivante (Listing 4.1) :

```
Module <nom de module>
Create OUT : <métamodèle cible> from IN: <métamodèle source>
```

Listing 4.1. Déclaration de l'entête de la transformation

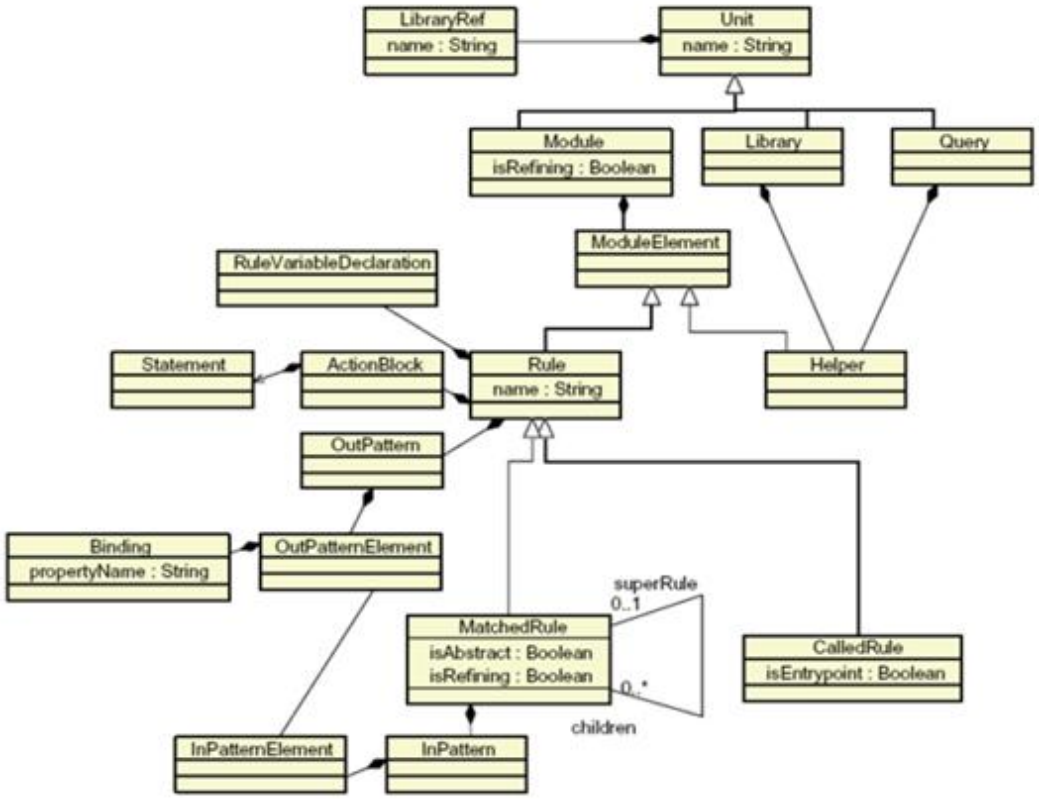


Figure 4.11. Métamodèle ATL simplifié

La section de l'entête définit le nom de module de transformation et le nom des variables qui correspondent aux modèles source et cible. Elle décrit ainsi le mode d'exécution de module. Le mode décrit dans le listing au-dessus est un mode normal. On peut exécuter le module en mode de raffinement (refining mode) dans le cas où la transformation souhaitée nécessite la modification d'une petite partie du modèle. Les éléments qui n'ont subi aucune modification dans le modèle source seront copiés au modèle cible. Le développeur ne spécifie que les changements qui seront effectués sur le modèle source [Atl 06].

Pour décrire la transformation d'un élément du métamodèle source en un élément du métamodèle cible, on utilise une règle qui se définit selon la syntaxe du listing 4.2 :

```
rule <nom de la règle> {
from
<pattern source>
to
<un ou plusieurs patterns cibles>
}
```

Listing 4.2. Déclaration d'une règle de transformation.

Le pattern cible se compose d'un ensemble de déclarations de variables qui correspondent aux éléments cible à générer, suivies d'attâchements qui indiquent pour chaque élément cible, comment instancier ses propriétés (Listing 4.3) :

```
<nom de l'élément cible> :<type> mapsTo <élément source associé>(
<attâchement>
  . . .
)
```

Listing 4.3. Pattern cible

Un attâchement a la forme donnée en Listing 4.4 :

```
<propriété de l'élément cible> <- <élément source>
```

Listing 4.4. La forme d'un attâchement

Implicitement, la propriété de l'élément cible généré prendra comme valeur le résultat de la transformation de l'élément source selon la règle qui correspond à ce dernier.

ATL fournit également la possibilité de créer et de détruire des éléments intermédiaires selon les besoin de la transformation, ainsi que des constructions conditionnelles (if, switch), et itérative (while) avec une syntaxe proche de Java.
Le concept de « helper» définit par ATL permet de définir des méthodes avec des paramètres et un type de retour, et des attributs (sans paramètres). Le corps d'un helper est défini à l'aide d'expressions OCL, il prend la forme suivante (Listing 4.5 et 4.6).

```
helper def :<nom de l'opération> (<paramètres> :<type>) : <type de
retour> =
 <expression OCL>;
```

Listing 4.5. Méthode en ATL

```
helper def :<nom de l'attribut>: <type de retour>=
<expression OCL>;
```

Listing 4.6. Attribut en ATL

ATL est disponible en tant que plugin dans le projet Eclipse M2M [Atl 06]. IL ne supporte que les transformations unidirectionnelles et est adapté pour interpréter des métamodèles décrits à l'aide d'EMF (Eclipse Modeling Framework) ou MDR (Meta Data Repository de NetBeans). Afin d'assurer l'indépendance d'ATL par rapport aux autres outils de modélisation, le plugin ATL met à disposition le langage KM3 (Kernel Meta-Metamodel), qui est une forme textuelle et simplifiée de EMOF, et permet de décrire des métamodèles et des modèles. Les modèles décrits au format KM3 peuvent être transformés en un format XMI Ecore, le format qui correspond dans l'EMF au modèle MOF. EMF, à son tour, fournit un

éditeur qui permet de manipuler les modèles décrits au format XMI sous forme arborescente avec des possibilités de mise à jour.

8.1.3. Kermeta

Kermeta est un langage de métametamodélisation développé à l'Université de Rennes. Il dispose d'un environnement de développement de métamodèles basé sur EMOF dans un environnement Eclipse. Il permet non seulement de décrire la structure des metamodèles, mais aussi leur comportement. Dans Kermeta, un langage est défini pour écrire des transformations. La syntaxe de transformations en Kermeta est proche de JAVA et dispose d'une structure uniquement impérative pour écrire des transformations.

8.1.4. Model-In-Action Transformation (MIA-T)

MIA-T est un outil de transformation de modèle développé par Mia-Software, filiale de Sodifrance. C'est à la fois un moteur de transformation, un dépôt de métamodèles et un environnement de développement. Cependant, il est possible d'utiliser le moteur de transformation en dehors de cet environnement via l'API disponible. Un projet de transformation est composé d'un ensemble de règles. Ces règles sont décomposées en deux partie : une requête qui sélectionne les éléments dans les modèles sources et une action qui créée et initialise les éléments dans les modèles cibles. Ces deux parties peuvent-être écrites dans des langages différents en fonction des besoins. MIA-T possède trois langages pour l'écriture des transformations : deux langages déclaratifs, à savoir MIA-TL et RL-TL, ainsi que Java qui peut-être appelé pour des traitements complexes nécessitant un langage impératif [Mia 10].

8.1.5. XMF-Mosaic

XMF-Mosaic est un outil destiné à développer des langages de modélisation et à déployer les outils nécessaires au travail avec ces langages incluant éditeurs, analyseurs de code et transformateurs. Dans XMF-Mosaic, les métamodèles sont considérés comme des langages spécifiques à un domaine et sont appelés modèles de domaine. XMF-Mosaic est régi par un langage appelé XOCL (eXtensible Object Constraint Language) qui est un langage basé sur XML pour représenter les contraintes OCL dans les modèles UML. Parallèlement, le logiciel fournit un ensemble d'outils graphiques qui permettent de créer des métamodèles, de construire des modèles correspondants et de décrire des transformations. Pour la transformation de modèles, XMF-Mosaic propose le langage Xmap. A l'aide de ce langage, on peut définir un ensemble de règles de transformation ou mappings définis chacun entre un ou plusieurs éléments d'un métamodèle source et un élément d'arrivée du métamodèle cible. Le logiciel dispose d'un outil graphique pour définir le squelette d'un mapping qui permet d'indiquer les métaclasses de départ, les métaclasses d'arrivée et de définir des dépendances entre mappings, lorsque l'un d'entre eux peut être amené à faire appel à une autre règle de transformation.

8.1.6. Eclipse et EMF

Eclipse est un environnement de développement intégré (Integrated Development Environment) dont le but est de fournir une plateforme modulaire pour permettre de réaliser des développements informatiques [Bud 04].
Eclipse utilise énormément le concept de modules nommés "plug−ins" dans son architecture. D'ailleurs, hormis le noyau de la plateforme nommé "Runtime", tout le reste de la plateforme est développé sous la forme de plug−ins. Ce concept permet de fournir un mécanisme pour l'extension de la plateforme et ainsi fournir la possibilité à des tiers de développer des fonctionnalités qui ne sont pas fournies en standard par Eclipse.
Eclipse possède de nombreux points forts qui sont à l'origine de son énorme succès dont les principaux sont :

- Une plate-forme ouverte pour le développement d'applications et extensible grâce à un mécanisme de plug-ins.
- Plusieurs versions d'un même plug-in peuvent cohabiter sur une même plateforme.
- Un support multi langage grâce à des plug-ins dédiés : Cobol, C, PHP, C# , ...
- Support de plusieurs plate-formes d'exécution : Windows, Linux, Mac OS X, ...
- Malgré son écriture en Java, Eclipse est très rapide à l'exécution grâce à l'utilisation de la bibliothèque SWT.
- Les nombreuses fonctionnalités de développement proposées par le JDT (refactorisation très puissante, complétude du code, nombreux assistants, ...).
- Une ergonomie entièrement configurable qui propose selon les activités à réaliser différentes «perspectives».
- Un historique local des dernières modifications.

Eclipse Modeling Framework (EMF) est la partie Model du pattern MVC (Model-Vue-Controler) (à noter que le framework ne propose pas de visuel pour représenter le modèle) :

- EMF est un «framework» qui traite des modèles : cela peut s'entendre ici sous le sens qu'EMF offre à ces utilisateurs un cadre de travail pour la manipulation des modèles.
- EMF permet de stocker les modèles sous forme de fichier pour en assurer la persistance.
- EMF permet de traiter différents types de fichiers : conformes à des standards reconnus (XML, XMI) et aussi sous des formes spécifiques (code Java).
- EMF ne propose pas d'outil graphique (de dessin) pour la modélisation.

L'objectif général d'EMF est de proposer un outillage qui permet de passer du modèle au code Java automatiquement. Pour cela le framework s'articule autour d'un modèle (le Core Model).
EMF va proposer plusieurs services :

- La transformation des modèles d'entrées (à gauche sur la Figure 4.12) présentés sous diverses formes, en Core Model
- La gestion de la persistance du Core Model
- La transformation du Core Model en code Java

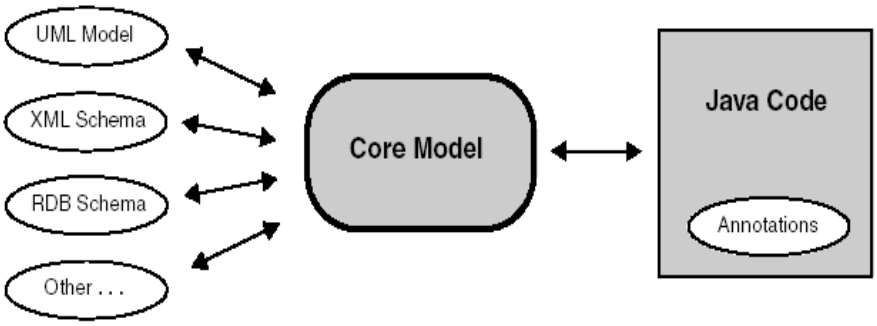


Figure 4.12. Organisation générale d'EMF [Bud 04]

8.1.7. Graphical Modelling Framework (GMF)

GMF permet de créer un modeleur (DSM4). GMF est un pont entre EMF et GEF. Pour créer le modeleur, GMF se base sur un méta-modèle défini en Ecore. Ensuite, plusieurs transformations sont appliquées pour définir :

- Le graphisme (indépendant du méta-modèle) définissant la manière dont sont représentées les classes (fichier .gmfgraph).
- La génération de la palette d'outils (fichier .gmftool).
- Le mapping indiquant quelle classe est modélisée par quel nœud et par quel outil. Le mapping est donc très important puisqu'il fait la relation entre les éléments du méta-modèle et les outils (fichier .gmfmap).

8.1.8. Autres

Plusieurs langages et outils de transformation de modèles existent actuellement [Bie 10], nous citons : Moflon, TCS (Textual concret Syntax), XText, Tefkat, Lxfamily, MOLA, MT, SiTra, ModfLog, GreAT, GenGen, BesBag, UMT, UMLX, ATOM VIATRA, BOTL, XDE Transformations, Codagen Architect Transformations, OptimaJ Transformations, MPS Transformation, Microsoft DSL Tool Transformations, AndroMDA, JET, FUUTje, GMT, Jamda, Fujaba Transformations, TXL, Stratego, etc.

9. Avantages de l'approche MDA

Maintenant que nous avons introduit l'approche MDA nous pouvons donner quelques avantages de cette démarche.

Commençons par rappeler brièvement ce qui s'est passé ces dernières années dans le domaine de la construction des applications réparties.

Les premiers travaux effectués à l'OMG pour faciliter la construction et la maintenance des applications réparties ont porté sur l'élaboration de la spécification CORBA. L'objectif de CORBA était de fournir un environnement standard et ouvert permettant à tout type d'application d'interopérer. À l'époque (1990), il fallait résoudre tous les problèmes d'interopérabilité en standardisant les communications réparties et les interfaces des différentes entités composant l'application répartie [Wes03].

Pour diverses raisons, CORBA n'a pas été un franc succès. D'autres intergiciels ont à leur tour tenté de résoudre le problème de l'interopérabilité en offrant toujours plus de services aux concepteurs d'applications (EJB, DCOM, Web Services).

De cette succession d'intergiciels est né le paradoxe des intergiciels. En effet, les intergiciels qui ont été initialement conçus pour faire face à la complexité des applications réparties ont finalement apporté beaucoup plus de complexité qu'ils n'en ont enlevée. Le problème n'est pas tant leur propre complexité, qui est inévitable, que le fait que les applications qui les utilisent dépendent fortement des services offerts par ces intergiciels.
De ce fait, changer d'intergiciel devient un cauchemar tant l'application est engluée dans l'intergiciel. Certains vont jusqu'à appeler spaghettiware les applications réparties qui utilisent les intergiciels. L'évolution d'une application a un coût prohibitif, car il faut inévitablement plonger en son cœur pour couper tous les liens avec les intergiciels que l'on veut changer.
Pour résoudre ce problème, la solution envisagée a consisté à tout mettre en oeuvre pour assurer de manière industrielle la fameuse séparation des préoccupations entre le métier des applications et la technique des intergiciels. De là est né MDA.
Les trois avantages de l'approche MDA sont les suivants :

- La pérennité du savoir-faire, afin de permettre aux entreprises de capitaliser sur leurs métiers sans avoir à se soucier des techniques.
- Les gains de productivité, afin de permettre aux entreprises de réduire les coûts de mise en œuvre des applications informatiques nécessaires à leurs métiers.
- La prise en compte des plates-formes d'exécution, afin de permettre aux entreprises de bénéficier des avantages des plates-formes sans souffrir d'effets secondaires [Bla 05].

10. Conclusion

Dans ce chapitre, nous avons présenté l'approche MDA et les concepts de modèle, méta-modèle, métamétamodèle et transformation de modèles. Nous avons expliqué comment l'utilisation de modèles tout au long du cycle de développement et de maintenance résout divers problèmes (complexité, compréhension, productivité, etc.).

Nous avons présenté aussi l'architecture générale de MDA et les différents standards de l'OMG sur lesquelles cette architecture est basée. Nous avons clairement montré que la transformation de modèles est au cœur de l'approche MDA. Elle permet d'aller d'un niveau d'abstraction très élevé (CIM) jusqu'à un niveau d'abstraction bas (code). La transformation de modèles peut être de différents types (horizontale, verticale, endogène, exogènes, etc.) et le choix entre eux se fait en fonction du problème posé. Plusieurs outils et langages de transformation de modèles existent actuellement dont chacun est considéré comme approprié pour tel ou tel type de problème. Nous avons donc présenté un ensemble de langages et outils parmi les plus connus de nos jours.

11. Références

[Atl 06] ATLAS, Complex data management in distributed systems, http://www.sciences. univnantes. fr/lina/ATLAS/, 2006.

[Bar 05] F. Barbier, UML 2 ET MDE Ingénierie des modèles avec études de cas, Livre édition Dunod, 2005.

[Bie 10] M.Biehl, Literature Study on Model Transformations, Journal of Royal Institute of Technology, Tech. Rep, July 2010.

[Bla05] X.Blanc, MDA en Action ingénierie logicielle guidée par les modèles, livre éditions Eyrolles, 2005.

[Bud 04] F.Budinsky, S.D.Merks, Eclipse Modeling Framework, livre edition Addison Wesley, The Eclipse series, 2004.

[Cza 06] K.Czarnecki, S.Helsen, Feature-based survey of model transformation approaches, IBM Systems Journal, V45, N 3, 2006, pp621-645.

[Dia 09] S.Diaw, R.Lbath, B.Coulette, Etat de l'art sur le développement logicie basé sur les transformations de modèles, Université de Toulouse France, 2009.

[Hub 08] P. Huber, The model transformation language jungle - an evaluation and extension of existing approaches, Memoire Master, Universite de Technische vienne Autriche, May 2008.

[Jou 05] F.Jouault et I.Kurtev, Transforming Models with ATL, In Satellite Events at the Models 2005 Conférence, Proceedings of the Model Transformations in Practice Workshop, volume 3844 de Lecture Notes in Computer Science Springer, Montego Bay Jamaica, 2005.

[Men 06] T.Mens K.Czarnecki, P.Van Gorp , Applying a model transformation taxonomy to graph transformation technology, in Proceedings of the International Workshop on Graph and Model Transformation , V152, Tallinn Estonia, mars 2006, pp143-159.

[Mes06] D.Meslati, Mage : une approche ontogénétique de l'évolution dans les systèmes logiciels critiques et embarques thèse de doctorat université de Annaba Algérie 2006.

[Mia 10] MIA-Software. MIA-Transformation and MIA-Generation website . http://www.mia-software.com.

[Mof 10] http://www.omg.org/mof/

[Mou 05] I.Mounier , X. Blanc, UML2 Pour Les Développeurs, livre édition Eyrolles, 2005.

[Ocl 05] Object Constraint Language Version 2.0, the official homepage. http://www.omg.org/spec/OCL/2.0/, 2005.

[Sol 00] R.Soley, Model Driven Architecture (MDA), Draft 3.2. Object Management Group, Inc, 27 novembre 2000.

[Wes03] A.Wesley, A.Kleppe, MDA Explained: The Model Driven Architecture: Practice and Promise, Livre edition Wesley, 2003.

Chapitre 5 Vers une Classification des Systemes Bio-Inspires Basee sur le Formalisme MDA

1. Introduction

De nos jours, le développement des systèmes informatiques a atteint une complexité telle que leur synthèse soulève plusieurs problèmes dont la solution excède les capacités intellectuelles humaines [Mes 06]. Les systèmes biologiques sont aussi très complexes, cependant, ils ont développé, à travers des millions d'années, des caractéristiques désirables et des mécanismes éprouvés qui leur permettent de survivre dans un environnement changeant et contraignant. Fascinés par ces systèmes, les chercheurs accordent un intérêt grandissant à toute forme d'inspiration biologique visant à faciliter la construction des systèmes artificiels et à les doter de caractéristiques biologiques.

Face aux nombreuses métaphores biologiques utilisées actuellement et le nombre grandissant des systèmes bio-inspirés, il devient important de trouver des critères pouvant caractériser chacune des approches et, éventuellement, conduire à une classification.

Dans ce chapitre, nous discutons la problématique et les motivations qui nous poussent à établir une taxonomie, nous présentons le modèle POEtic, ces caractéristiques, nous introduisons notre approche, ces caractéristiques et nous terminons par une synthèse.

2. Problématique

Trouver une classification n'est pas une tâche facile, vu qu'il faut atteindre partiellement certains objectifs (trouver quelques similitudes et quelques différences et essayer de les façonner et de les étudier pour extraire des classes proprement dites) avant de parvenir à une classification. Ceci n'est pas un dilemme, mais plutôt un processus incrémental où on classe pour atteindre partiellement certains objectifs puis on raffine la classification. Ce processus conduira à l'amélioration des approches bio-inspirées et, à terme, promouvra une synergie interdisciplinaire qui correspond mieux à leur nature.

Traditionnellement, les approches bio-inspirées sont classées selon les disciplines et les sous disciplines, telles que l'intelligence artificielle, l'intelligence artificielle distribuée, la vie artificielle, le calcul évolutionnaire, la cybernétique, etc. A l'intérieur de chaque discipline/sous discipline, les approches sont classées encore, en utilisant des critères qui reflètent les objectifs du système, son mode opératoire, sa constitution, etc. Par exemple, les systèmes multi-agents sont classés dans l'intelligence artificielle distribuée, en premier lieu, puis comme systèmes réactifs ou cognitifs en second lieu [Fer 95].

Ces classifications reflètent l'évolution de nos inspirations des organismes biologiques, plutôt que les propriétés intrinsèques des systèmes bio-inspirés. Nous citons les inconvénients de ces classifications dans ce qui suit:

- Manque de précision. Elles empêchent la caractérisation des approches qui peuvent être réellement différentes. Par exemple, l'approche d'un système multi-agents où la structure des agents évolue en utilisant un processus phylogénétique est différente d'une approche où la coordination entre agents émerge d'un processus phylogénétique [Ada 98]. Cet inconvénient prendra des dimensions encore plus importantes avec l'utilisation de mécanismes et métaphores multiples et leurs hybridations.
- Caractère non naturel. Alors même que certains mécanismes biologiques deviennent de plus en plus utilisés, il semble difficile de maintenir une correspondance entre les systèmes artificiels et leur contrepartie naturelle. Même si cela n'a pas d'effet sur leur efficacité, l'aspect naturel est une qualité qui facilite la compréhension de ces systèmes. Par exemple, lorsqu'on utilise un processus évolutionnaire dans un agent, il n'est pas facile d'identifier ce qui correspond à un individu et ce qui correspond à une espèce. A première vue, l'agent est ce qui correspond à l'individu. Cependant, pour un individu la phylogenèse n'est significative.
- Frontières non justifiées. L'utilisation des frontières disciplinaires ne reflète pas les tendances courantes et constitue un obstacle à la synergie des approches. Actuellement les approches sont hybrides dans leur grande majorité. En d'autres termes, elles entrecoupent ces frontières.

Ces insuffisances donnent naissance, directement ou indirectement, aux problèmes suivants :

- Les classifications ne prennent pas en considération la nature des approches qui sont inspirés de phénomènes naturelle.
- Les classifications ne mettent pas l'accent sur l'environnement où les systèmes opèrent, alors que celui-ci joue un rôle très important.
- Des difficultés à caractériser les approches qui sont de nature inspirés de phénomènes naturels.
- Une multitude d'hybridations d'approches qui sont peut être équivalentes.
- L'étude des systèmes bio-inspires devient une tâche ardue.
- La terminologie des termes dans les systèmes bio-inspirés est ambiguë.

Au vu de ces insuffisances, il convient de noter que la solution consiste a :

- Trouver une classification qui se base sur des phénomènes naturelle (Vue en chapitre 2).
- Modélisation des phénomènes naturelle (Vue en chapitre 2) en utilisant les aspects de l'ingénierie des modèles (MDA).
- Prendre comme critère important l'environnement où le système opère.

Nos investigations dans le sens de ces besoins nous ont amené à voir et à étudier une approche que nous considérons proche de notre problématique, nous décrivons celle-ci dans la section suivante.

2.1. L'approche POE

L'approche POE, est une méthode qui vise à décrire les systèmes matériels bio-inspirés selon trois axes : Phylogénétique, relatif à l'évolution, Ontogénétique, pour ce qui est de la croissance et de l'autoréparation, et Épigénétique, en ce qui concerne l'apprentissage. Différentes applications peuvent mettre en œuvre un, deux ou trois de ces axes, en fonction du problème à résoudre.

2.1.1. Phylogenèse

Les théories de l'évolution décrites précédemment (voir chapitre 2), et principalement le néo-darwinisme, ont conduit à la création des algorithmes évolutionnistes. Ces algorithmes permettent de résoudre des problèmes d'optimisation pour lesquels une solution déterministe ne peut être trouvée en un temps raisonnable. A titre d'exemple citons celui du voyageur de commerce, qui consiste à trouver le chemin le plus court pour la visite de n villes. Le nombre de solutions possibles y est de l'ordre de n !, qui grandit de façon non polynomiale avec le nombre de villes. L'exécution d'un algorithme testant toutes les configurations possibles d'un tel système n'est dès lors pas réaliste pour de grandes valeurs de n.

Les problèmes de ce type sont appelés NP-difficiles et nécessitent un temps de calcul non-polynômial, qui croît donc très vite en fonction de la taille du problème. Ils sont caractérisés par un espace de recherche très grand, sur lequel on ne peut pas effectuer une simple descente de gradient, à la manière d'une escalade de montagne, où, en suivant la pente il est aisé d'arriver au sommet. En robotique, par exemple, l'optimisation de paramètres ou la tâche de contrôle peuvent également ne pas être forcément résolues de manière déterministe.

Bien qu'utilisés pour l'optimisation, il est important de noter qu'ils ne fournissent pas obligatoirement la solution optimale à un problème. Le résultat est plutôt une solution acceptable, qui approche assez l'optimum pour satisfaire le problème.

Le premier algorithme évolutionniste, appelé algorithme génétique [Hol 75], fut proposé par John Holland. De nombreuses adaptations furent développées, comme la programmation génétique [Koz 92] de John Koza, visant à laisser un programme informatique être généré automatiquement à partir d'une tâche à effectuer, la Particle Swarm Optimization [Ken 95], ou les Stratégies Evolutives [Rec 73]. Un bon résumé des différentes approches peut être trouvé dans l'article de Back [Bac 97]

2.1.2. Ontogenèse

Inspirés par la manière dont les êtres vivants se développent et cicatrisent, l'ontogenèse vise la création de systèmes capables de croître, de s'auto-organiser, et de s'auto-réparer. Les deux buts sous-jacents d'un système ontogénétique dépendent du type d'application à réaliser. Le premier est la scalabilité des algorithmes évolutionnistes, et le deuxième est l'autoréparation, ou la tolérance aux pannes, des systèmes matériels [Tho 05].

Nous pouvons dire que la scalabilité des algorithmes évolutionnistes pose problème lorsque le génome devient trop grand. Dans le cadre de systèmes évolutifs, une méthode d'ontogenèse (ou développement) permet alors d'avoir à disposition un mapping particulier du génotype au phénotype. Le principe est identique aux mécanismes du vivant : les cellules d'un être humain ne peuvent être décrites individuellement dans le génome, et donc un système ontogénétique est obligatoire pour la création des milliards de cellules qui nous constituent. Les systèmes artificiels multicellulaires, c'est -à- dire dont la fonctionnalité peut être décomposée en plusieurs parties relativement semblables, peuvent être conçus de la même manière [Tho 05].

En 1968, Lindenmayer introduisit les systèmes de réécriture, appelés L-systèmes [Lin 68]. Basés sur des chaînes de caractères, et partant d'un axiome de départ, ces systèmes appliquent à chaque pas de temps des règles de production à la chaîne de caractères. Chaque caractère peut représenter une cellule, ou une liaison entre cellules, et l'évolution de la chaîne de caractères correspond donc à la création d'un organisme multicellulaire. Ce type de mécanisme développemental a souvent été utilisé en conjonction des algorithmes évolutionnistes, afin de réduire la taille du génome. A titre d'exemple, citons Kitano [Kit 90], qui a utilisé des grammaires génératives basés sur des L-systèmes pour créer des graphes représentant la structure d'un réseau de neurones. Son approche a montré une meilleure scalabilité par rapport à un encodage direct de la structure du réseau, mais il faut remarquer que cette simplification du codage n'implique pas que toute la structure du réseau peut être générée de cette manière. Gruau [Gru 92] a également utilisé des grammaires génératives pour créer des réseaux de neurones. Cependant, au lieu de réécrire des caractères, ce sont les neurones qui sont directement affectés par les règles de réécriture. Notons finalement que l'implémentation matérielle des systèmes de réécriture n'est pas des plus évidentes, car de simples règles locales ne permettent pas d'implémenter ces mécanismes de façon efficace.

La tolérance aux pannes peut également tirer profit des systèmes ontogénétiques.
Le fait de disposer de systèmes multicellulaires évite la présence d'un contrôle global, qui, s'il flanche, met en péril tout le fonctionnement du système. De plus, la communication intercellulaire, qui est utilisée lors du développement, peut également l'être lors de l'autoréparation [Tho 05].

Les chimies artificielles, domaine de recherche en pleine expansion, en sont un excellent exemple. Miller [Mil 04] a créé une approche améliorée ensuite par Capcarrère et Ozturkeri [Ozt 04], qui permet de générer un drapeau français ou allemand dans une grille régulière de cellules. Partant d'une seule cellule, et en se basant sur des interactions locales, le drapeau se construit automatiquement. Chaque chimie particulière est générée grâce à un algorithme

évolutionniste qui se charge de trouver des règles de comportement cellulaire adaptées. Le fait important est ici que l'altération de l'état d'une ou plusieurs cellules peut être contrecarrée par l'action de la chimie artificielle. Le drapeau retrouve alors sa forme initiale, ou une forme quasiment identique [Tho 05].

Finalement, toujours sur le plan de l'autoréparation, citons le projet Embryonique [Man 00, Man 98, Tem 97], développé au Laboratoire de Systèmes Logiques par l'équipe du professeur Mange. Il a vu la réalisation d'un circuit capable d'exécuter n'importe quelle tâche, ainsi que de s'auto-réparer et de s'auto-répliquer. Ce circuit est une sorte de FPGA dont chaque élément de base, appelé molécule, est un multiplexeur et une bascule [Tho 05].

Un système d'autoréparation au niveau moléculaire est implémenté en matériel, et permet, en plaçant des colonnes de molécules en attente, de faire face à des erreurs du matériel. Ce substrat électronique est ensuite utilisé pour l'implémentation d'un organisme multicellulaire, une cellule étant composée d'un certain nombre de molécules [Tho 05].

Chaque cellule contient le génome complet de l'organisme, de manière à ce qu'au lancement du système, chaque cellule soit totipotente. Un système de différenciation basé sur des coordonnées à deux dimensions permet alors à la cellule de sélectionner une tâche à accomplir, et ce même système est utilisé lors de l'autoréparation cellulaire.

En effet, lorsque les molécules ne peuvent plus supporter de nouvelles fautes, la cellule meurt, sa colonne cellulaire entière est mise hors service, et les colonnes suivantes sont toutes décalées, jusqu'à atteindre une colonne de cellules totipotentes [Tho 05].

Toutes ces colonnes se redifférencient et le circuit peut continuer à fonctionner, sans que sa fonctionnalité n'ait été altérée.

L'approche Embryonique est fortement inspirée du vivant, de par la structure multicellulaire du système, la présence du génome dans chaque cellule, la croissance de l'organisme, et de par l'autoréparation. En ce sens, l'approche POEtic est une sorte de continuité de ce paradigme, auquel nous désirons ajouter de l'évolution et de l'apprentissage.

Concernant les faiblesses de l'approche Embryonique, nous pouvons en citer deux [Tho 05], qui serviront lors de la réalisation du tissu POEtic. Premièrement, la structure multicellulaire et le système de différenciation sont trop figés. Le système de coordonnées utilisé oblige la destruction d'une colonne entière de cellules dans le cas d'une réparation cellulaire, ce qui pourrait être évité dans un système plus souple. Deuxièmement, la structure moléculaire offre une faible fonctionnalité, ce qui implique la réquisition d'un grand nombre de molécules, en comparaison des FPGAs actuels, pour l'implémentation de systèmes peu complexes.

2.1.3. Epigenèse

Troisième et dernier axe, l'épigenèse se propose de s'inspirer de la manière dont les neurones biologiques fonctionnent pour créer leurs correspondants artificiels. L'adaptation s'y effectue durant la vie d'un seul individu, en comparaison des mécanismes phylogénétiques, qui font

intervenir toute une population. De nombreuses approches à l'apprentissage sont possibles, comme les réseaux de neurones, qui sont capables de modéliser nombre de phénomènes, de s'adapter à de nouvelles données, et de généraliser les exemples fournis lors de l'apprentissage [Tho 05].

Deux familles de neurones artificiels ont retenu l'attention des chercheurs. Premièrement, les neurones standards, qui calculent immédiatement une sortie en fonction de leurs entrées, et deuxièmement, les neurones à impulsions, qui travaillent avec une composante temporelle [Tho 05].

Les trois axes de l'approche que nous venons de survoler peuvent évidemment être combinés : PO (Phylogénèse-Ontogénèse), PE (Phylogénèse-Epigénèse), OE (Ontogénèse-Epigénèse), POE (Phylogénèse-Ontogénèse-Epigénèse).

Observons maintenant les implications de telles combinaisons dans le cadre des implémentations matérielles :

- PO. L'analyse des axes phylogénétiques et ontogénétiques des êtres vivants est en pleine évolution, les récentes découvertes des gènes organisateurs influant grandement sur les théories évolutionnistes [Hol 99]. Une nouvelle discipline, appelée evo-devo [Goo 00], est d'ailleurs née de la combinaison de ces deux axes.

  Les systèmes artificiels se doivent de prendre le même chemin, la manière dont un individu est créé à partir de son génome influe grandement sur l'efficacité de l'algorithme génétique relatif. Le codage morphogénétique, introduit par Roggen et Floreano [Rog 03] en est un exemple. Un système cellulaire sert de base à la création d'un individu. Des émetteurs, dont l'emplacement dans le substrat est défini par le génome, sont la base d'un gradient chimique qui décroît avec la distance à la source (Figure 5.1). Plusieurs gradients peuvent être considérés (typiquement quatre), et servent ensuite à la cellule lors de la phase de différenciation, pour accéder au bon gène. Le grand avantage de cette approche est la scalabilité, le génome n'étant pas influencé par la taille du tableau de cellules, et dans plusieurs cas les performances sont meilleures qu'un codage direct [Tho 05].

  La CAM-brain machine de De Garis [Deg 93, Deg 97] est un autre exemple de système PO. Il s'agit d'un réseau de neurones implémentés sur FPGAs, dont la structure est évolutive. Le génome stocke la manière dont le réseau se crée, et un automate cellulaire est en charge de l'ontogenèse. Nous plaçons cette machine dans la combinaison PO, car, bien que basée sur des neurones, ceux-ci ne sont pas capables d'apprendre.

  Haddow, Tufte, et Remortel, ont, quant à eux, utilisé des L-systèmes pour réduire la taille du génome, dans une application en matériel évolutif [Had 01].

  De manière générale, les systèmes PO doivent être capables d'exécuter un algorithme évolutionniste, et présenter une forme de croissance, réduisant ainsi la taille du

génome, ou d'autoréparation, offrant au système la propriété de tolérance aux pannes. Alors que de nombreuses approches liées à la réduction de la taille du génome ont vu le jour, les systèmes auto-réparables de type PO ne sont pas fréquents.

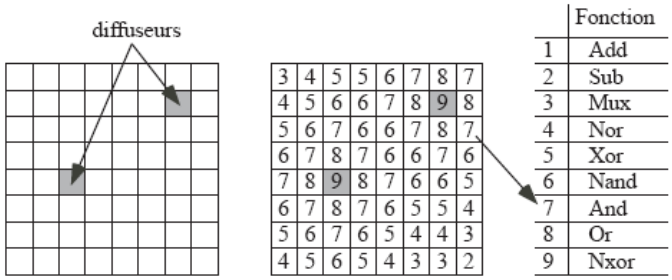


Figure 5.1. Gradient chimique propagé par le codage morphogénétique. La fonctionnalité de la cellule est ensuite définie par une table, qui constitue une partie du génome [Tho 05]

- PE. Certains réseaux de neurones artificiels ne possèdent pas de capacité d'apprentissage, les chercheurs préférant utiliser des algorithmes génétiques pour la modification des poids synaptiques ou de la topologie du réseau (par exemple, les Block-Based Neural Networks de Moon [Moo 01]). Nous ne pouvons toutefois pas arguer qu'il s'agisse d'un système PE, l'apprentissage standard étant inexistant.

  Pour pouvoir créer des systèmes PEs, nous devons faire appel à des réseaux de neurones permettant l'implémentation d'algorithmes d'apprentissage, et qui sont donc des systèmes adaptables. L'évolution peut y apporter un plus en modifiant la structure du réseau, les poids synaptiques initiaux d'un individu, ou les règles d'apprentissage.

  Un excellent résumé des différents travaux allant dans ce sens peut être trouvé dans l'article de Xin [Yao 99], et nous pouvons citer ici deux exemples, [Ast 00] et [Seg 98]. Dans le cas de la robotique évolutive, de tels systèmes ont déjà montré leur avantage sur ceux ne mettant en jeu qu'un des deux mécanismes [Flo 98].

  Parmi les systèmes tirant parti de ces deux axes, nous pouvons citer l'algorithme SOS, développé par Mesot [Mes 03]. Des machines à états finis sont adaptées grâce à un algorithme d'apprentissage par renforcement et un algorithme génétique. L'algorithme génétique fournit des configurations de départ qui servent de base à l'apprentissage, et la combinaison des deux approches offre de meilleurs résultats que l'utilisation de chacune séparée.
- OE. Les réseaux de neurones sont intrinsèquement tolérants aux pannes, de par leur propriété de généralisation. Toutefois, leur structure cellulaire permet l'ajout d'autoréparation au système, de la même manière que pour une application PO.

  Une deuxième manière de relier les deux axes est de considérer la neurogenèse comme faisant partie de l'axe ontogénétique. Peu de réseaux de neurones artificiels font intervenir des topologies variables par croissance ou destruction de neurones [Bor 00, Per 99, Teu 01]. Toutefois, cette propriété, que nous retrouvons dans le règne animal,

devrait apporter de nouvelles solutions aux systèmes d'apprentissage. Un mécanisme de gestion de croissance et destruction doit donc être mis en place au niveau matériel, afin que le réseau puisse se modifier en fonction de ses interactions avec l'environnement.

- POE. Il n'existe pas, de système matériel connu combinant les trois axes (voir figure 5.2). La réalisation d'un tel système devra donc mettre en jeu de l'évolution, de l'apprentissage, de la croissance, et de la tolérance aux pannes, le tout dans un même circuit électronique. Il devrait sans doute s'agir d'un réseau de neurones capables d'apprentissage, et dont la topologie pourrait être modifiable par un algorithme génétique ou par les neurones eux-mêmes. Un mécanisme de croissance et de tolérance aux pannes au niveau cellulaire devrait également y être présent, pour que le réseau puisse se construire de manière autonome, sur la base d'un génome, et qu'un neurone puisse être remplacé par une cellule en attente dans le cas de la présence de matériel défectueux.

  La topologie du réseau doit pouvoir varier durant son fonctionnement, et ce de manière non centralisée. Les neurones doivent être capables de créer de nouvelles cellules, et d'initier de nouvelles connexions. Pour ce faire, l'utilisation d'un circuit programmable capable de modifier sa connectique de manière autonome est nécessaire [Tho 05].

  Outre une fonctionnalité universelle, il doit offrir un système de routage géré par les cellules. De plus, les cellules doivent pouvoir y modifier leur comportement de façon autonome, afin que les cellules totipotentes puissent se différencier, en fonction des tâches à effectuer, et notamment lors de processus d'autoréparation.

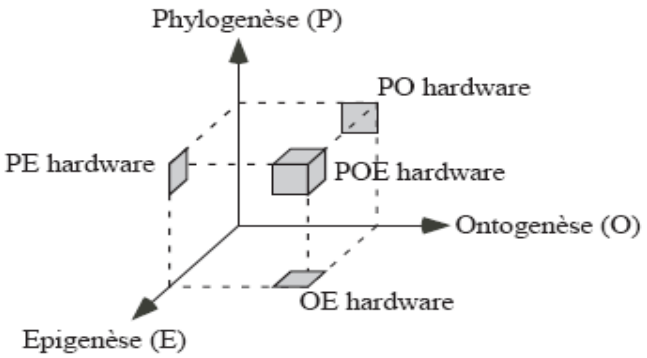


Figure 5.2. La représentation tridimensionnelle du modèle POE

2.2. Critiques et discution

Apres notre investigation dans le domaine des systèmes bio-inspirés et leurs taxonomies nous avons constaté que la majorité des classifications actuelles sont dédiées aux disciplines ou sous disciplines. A notre connaissance seul le travail vu en section 2.1 va dans le sens de ce que nous proposons. Les auteurs utilisent le modèle Poetic pour classer les systèmes bio-inspirés [Sip 97]. La classification Poetic est elle même bio-inspirée et s'accommode d'un large éventail d'approches. Cependant, on peut relever les insuffisances suivantes:

- Certaines définitions utilisées peuvent être sujettes à discussion. C'est le cas, par exemple, lorsque les auteurs considèrent que l'environnement n'a aucun effet sur l'ontogenèse alors qu'en réalité, il en a.
- Les processus Poetic peuvent être combinés, mais la classification Poetic ne peut discriminer les diverses formes de combinaison. Par exemple, la phylogenèse et l'épigenèse peuvent être combinées de diverses façons.
- La classification ne laisse pas apparaître la dichotomie individu/espèce comme un critère important alors que celle-ci est importante et permet de comprendre plus facilement les approches.
- La classification Poetic se limite aux systèmes matériels.

Notre classification se base sur la classification Poetic. Elle peut être considérée comme un raffinement qui utilise les processus Poetic comme un critère discriminant principal, mais ajoute un ensemble de critères pour caractériser un grand nombre d'approches hybrides.

Les biologistes, de leur côté, sont confrontés à un dilemme entre deux types de classifications. La première est la classification phylogénétique, qui repose sur les relations ancestrales entre individus pour les séparer en groupes. La seconde est la classification phénotypique, qui se base sur l'observation des comportements et des caractéristiques pour classer les espèces [Uni 04]. Alors que les deux classifications sont utilisées en biologie, la classification phylogénétique est sans intérêt dans le cas des systèmes bio-inspirés, vu qu'elle n'apporte rien à leur étude ou leur amélioration.

Notons que notre classification peut s'énoncer comme suit : C'est une nouvelle approche de voir et de concevoir des systèmes bio-inspirés en se basant en premier lieu sur l'ingénierie des modèles présentée dans le chapitre 4.

3. Motivation et objectifs De Travail

Face aux nombreuses métaphores biologiques utilisées actuellement et le nombre grandissant des systèmes bio-inspirés, il devient important de trouver des critères pouvant caractériser chacune des approches et, éventuellement, conduire à une classification [Mes 06]. Dans ce qui suit, nous présentons les raisons qui nous ont poussées à rechercher des critères discriminantes universels pouvant conduire à une classification des approches.

3.1. Caractériser et comparer les approches

La caractérisation d'une approche consiste à définir l'objet ou le concept qui distingue cette dernière des autres. Par exemple le faite de définir le concept de la parole comme caractère ou trait, permet de caractériser en grande partie les être humains des animaux.

Caractériser les approches permet aussi de faire une hiérarchisation d'approches. Dans le contexte de notre travail nous souhaitons caractériser les approches bio-inspirées afin d'arriver à faire une comparaison.

3.2. Rechercher les concepts communs et unificateurs

Un concept est une représentation générale et abstraite de la réalité d'un objet, d'une situation ou d'un phénomène; il n'est pas exactement synonyme de notion car plus abstrait. Nous voulons dire par concept dans le cadre de notre travail tout composant qui permet de bien identifier un système bio-inspiré, par exemple : individu, population…

Le but de toute taxonomie et de faire une comparaison, celle-ci ne peut se faire que si les objets à comparer ont au moins un concept commun, une bonne classification doit faire apparaitre les points communs entre les différents systèmes et aussi distinguer les différences entres les eux.

3.3. Faciliter l'étude des systèmes bio-inspirés

La nature nous surprendra toujours, et les biologistes ne cesse de découvrir des aspects surprenant à l'intérieur des organismes, qu'ils soient humains, animaux ou végétaux, face à ces découvertes, il apparait tous les jours des propositions de modélisation de ces dernières afin de les utiliser dans les produits conçus par l'homme.

Les systèmes s'inspirant du réseau de neurones humains, les systèmes s'inspirant de la théorie de l'évolution de Darwin et tant d'autre ont connu un succès dans le domaine de l'informatique et de l'industrie par leurs facteurs d'adaptation et d'évolution. Ce qui précède laisse sous entendre qu'il y aurait une multitude de systèmes inspirés du vivant et il devient nécessaire de les étudier pour en optimiser les fonctionnalités, une bonne étude de ces systèmes nécessite forcement une bonne taxonomie pour pouvoir aborder et étudier la distinction entre les différents concepts de chaque système.

3.4. Trouver des directions d'inspiration prometteuses

Au cours de notre investigation nous avons étudiez l'approche de classification POEtic, qui divise les systèmes bio-inspirés en trois axes, mais les auteurs de cette démarche ont permis par leur découverte de trouver d'autre directions de recherche dans le domaine de la bio inspiration comme par exemple l'hybridation à deux axes du système de classification (ontogénétique et phylogénétique, ontogénétique et épigénétique,…).

Cet approche de taxonomie a permis la découverte d'autres systèmes bio-inspires mais cette fois en se basant sur des systèmes bio-inspirés et non pas sur des systèmes biologiques.

Une bonne taxonomie permet de faire évoluer la recherche dans le domaine ou elle est utilisée et donne de bonnes directions d'investigation.

3.5. Unifier la terminologie des systèmes bio-inspirés

Une terminologie est l'ensemble des termes, rigoureusement définis, qui sont spécifiques à une science, une technique, on un domaine particulier de l'activité humaine. Comme vu dans la section 2, il existe une multitude de systèmes inspirés du vivant et d'autres inspirés eux-mêmes d'autres systèmes bio-inspirés et l'utilisation des termes d'un système à l'autre est

parfois ambiguë. Par exemple le concept « individu » est considéré comme équivalant à « neurone » dans les réseaux de neurones artificiels et comme équivalant à « agent » dans les systèmes multi agents.

Une bonne taxonomie nécessite de repérer, d'analyser, d'unifier et, au besoin, de créer des termes pour répondre aux besoins d'expression.

3.6. Aider à la détermination des caractéristiques d'un système en cours de développement

Lorsque les concepts d'une approche sont bien déterminés, ils deviennent eux-mêmes des éléments qui guident le developpement d'un système donné. En effet, on se posera des questions sur tel ou tel concept de l'approche et quel est son équivalent dans le système en cours de developpement. Par exemple, dans les approches évolutionistes, on distingue des concepts clés : Individu, codage, croisement et mutation, fonction d'adaptation et selection. Lors du developpement d'un système, on ne manquera pas d'implémenter la mutation car on sait préalablement que cette dernière évite de converger rapidement vers un minimum/maximum local.

4. Principe et concepts de notre approche

Trouver un ensemble de critères discriminants n'est pas une tâche aisée. En effet, le nombre des approches bio-inspirées ne cesse de grandir et diverses métaphores sont utilisées.
L'absence d'un consensus sur les définitions et concepts utilisés rend la tâche encore plus difficile.

Avant d'extraire l'ensemble des critères, nous avons entrepris, dans un premier temps, une analyse et une étude comparative des systèmes et approches bio-inspirées (décrites au chapitre 3). Puis nous avons étudié le modèle de classification proposée par Sipper et all [Sip 97] qui se base sur la biologie pour bâtir les trois axes de l'approche POEtic (section 2.1). Le reste de cette section sera consacré à notre méthodologie de classification qui s'inspire fortement des travaux de [Sip 97].

Durant l'élaboration de cette thèse nous avons constaté que les systèmes bio-inspirés, que nous avons considérés, possèdent des comportements différents quant à leurs exécutions. Ce comportement particulier permet de discriminer les systèmes les uns des autres.

Chaque système bio-inspirés à un comportement différent, par exemple les systèmes évolutionnaires commence par établir une population d'individus puis font évoluer cette dernière à travers une série d'opérateurs (mutation et croisement) jusqu' à arriver à une population satisfaisante. Nous pouvons dire ici qu'un système évolutionnaire possède trois comportements. De leur coté les automates cellulaire nous permettent de faire évoluer une plateforme composée de plusieurs cellules identiques avec deux comportements qui peuvent définir un système.

Un réseau de neurone peut être décrit comme un ensemble de neurones connecté entre eux formant ainsi un réseau que celui-ci va adapter à chaque fois pour réaliser une tâche. D'âpres cette définition, nous pouvons dire qu'un réseau de neurones possède deux comportements. Les systèmes à base de colonie quand à eux peuvent être décrits comme un ensemble d'individus à but différents (comportement local), formant une population dans le but de réaliser des tâches globales (comportement globale).

Apres avoir observé les comportements des différents systèmes, l'idée nous est venue de les décomposer en un ensemble de ce que on peut appeler composants de base, en assemblant ces derniers on constituera un système bio-inspirés.

Un composant de base doit permettre d'encapsuler un comportement particulier du système.
Trois comportements de base peuvent être distingués pour les systèmes évolutionnaires :

- La création de la population, qui se compose d'un ensemble d'individus ayant des caractéristiques particulières.
- La sélection des individus selon des critères distinctifs.
- La reproduction de certains individus pour l'amélioration de la population.

Deux comportements peuvent être distingués pour les automates cellulaires :

- La création de la plateforme de cellule.
- L'évolution des cellules à travers des règles d'évolution.

Deux comportements peuvent être distingués pour les réseaux de neurones :

- Création du réseau.
- Adaptation du réseau à travers les entrées définies par l'utilisateur. La tâche de l'adaptation, après mure réflexion, nous a semblé trop abstraite puisque le réseau de neurone procède à un comportement d'ajustement et d'interprétation des résultats. Nous avons alors considéré l'adaptation comme la composition de deux comportements de base qui sont, l'ajustement du réseau et l'interprétation de celui-ci. Prenons l'exemple d'un réseau de neurones utilisé pour la reconnaissance de formes, la première étape va consister à créer le réseau et définir le nombre de neurones et de couches et par conséquent les neurones d'entrée et les neurones de sorties. Après cette étape viens l'adaptation qui est constituée de l'ajustement des poids et l'interprétation du réseau jusqu'à arrivé a un bon résultat.

Les systèmes à base de colonies d'insectes, par exemple les colonies de fourmis possèdent trois comportements :

- Le développement d'une solution possible, prenant exemple sur un système à base de colonie pour la résolution du problème du voyageur de commerce ici une solution possible serait le choix d'un trajet aléatoire.
- L'interprétation des solutions fournie par chaque individu de la colonie, dans le cas du voyageur de commerce déterminé les partis les plus parcourues de chaque trajet.
- L'ajustement des solutions trouvé, par exemple dans le cas du voyageur de commerce on procède à l'élimination des mauvaises solutions.

Ce qui précède laisse sous entendre que pour décrire l'ensemble des systèmes vue précédemment, on a besoin de cinq composants de base qui sont le développement, la sélection, la reproduction, l'interprétation et l'ajustement.

4.1. Description de l'aspect comportemental d'un système bio-inspirés

Trois parties sont impliquées dans les systèmes bio-inspirés: les processus, la structure architecturale et l'environnement où le système opère. Par conséquent, caractériser le système revient à caractériser, par des critères, chacune des trois parties. L'architecture se compose de tous les modèles, toutes les unités architecturales ainsi que leur agencement.
Dans ce qui suit nous donnerons une définition formelle de ce qu'est un composant de base et nous schématiserons les différentes vues qui interviennent dans la description de tout système bio-inspiré.

4.1.1. Composant de base ou unité architecturale

Un composant de base ou unité architecturale est un composant permettant d'encapsuler un certain comportement particulier d'un système bio-inspirés ou d'un certain point de vue, un mécanisme qui permet de faire passer un système d'un état vers un autre état.
Plus formellement, l'unité architecturale se compose d'un ensemble de modèles d'entrée, d'une transformation et d'un ensemble de modèles de sortie produits par la transformation. Les transformations peuvent être guidées par les modèles en entrée, un modèle de paramètres et un modèle de l'environnement représentant des événements qui peuvent, par exemple, déclencher ou arrêter la transformation. Les transformations peuvent être de plusieurs types et avoir un nombre de paramètres quelconque. Nous décrivons dans ce qui suit les composants de base que nous avons identifiés.

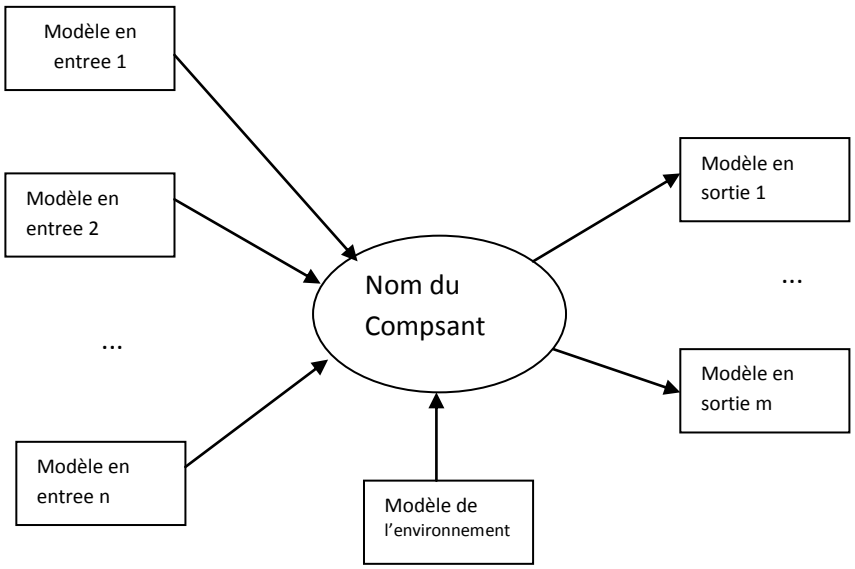


Figure 5.3. L'unité architecturale

4.1.1.1. L'unité architecturale de développement

L'unité architecturale de développement est une unité spécifique qui peut être utilisée dans le processus ontogénétique.
Formellement, l'unité de développement peut être écrite en utilisant une notation fonctionnelle comme suit:

$$Develop(D, M, P, E) \rightarrow M'$$

D est un modèle descriptif qui guide la transformation (i.e. le génome). M est le modèle à transformer (i.e. la partie innée d'un phénotype). Pendant l'ontogenèse, le modèle de sortie d'une itération est le modèle d'entrée de l'itération suivante. (i.e. M et M' sont deux états consécutifs du même modèle). Notons, qu'au départ, la partie innée peut être inexistante. L'opération effectuée par la transformation à chaque itération est déterminée par D, l'état courant de M, le contenu du modèle de paramètres P et celui de l'environnement E. Selon le cas, la transformation peut opérer une division cellulaire, une différentiation, une migration, etc.
Les stimuli de l'environnement utilisés dans l'unité architecturale de développement peuvent être de plusieurs types, allant d'une température qui dépasse un certain seuil à une blessure ou une défaillance dans une partie du système. Dans ce dernier cas, la transformation opère une réparation selon le contenu de la partie du génome (i.e. le modèle D) dédiée à cette tâche. Du point de vue de l'abstraction, l'unité architecturale de développement produit un modèle de sortie M qui a une abstraction supérieure à celle du modèle D.

L'expression $Develop(D, M, P, E) \rightarrow M'$, signifiant que M' est obtenu à partir de M par une transformation basée sur D. P et E sont, respectivement, les paramètres de la transformation et les stimuli de l'environnement.
En notant le niveau d'abstraction de M par Abs(M) on peut caractériser le développement par :

$$(Abs(M) > Abs(D)) \wedge (Abs(M') > Abs(D)) \wedge (Abs(M') \approx Abs(M))$$

Autrement dit l'abstraction du modèle M est supérieure à celle de D, par exemple dans les réseaux de neurones on voit une évolution de topologie ce qui est exprimé par un modèle M plus abstrait que le modèle de développement initial, lors de la phase d'apprentissage le réseau décrit par le modèle M va évoluer vers un réseau décrit par le modèle M', ici la topologie ne change pas il ya que les poids des arc du réseau qui changent on peut dire que le modèle M et M' on une abstraction équivalente.

4.1.1.2. L'unité architecturale de sélection

L'unité de sélection permet la sélection d'un sous ensemble de modèles à partir d'un ensemble de modèles donnés en entrée. Les modèles eux même ne sont pas altérés. Les paramètres de la transformation incluent certains attributs tels que le seuil de sélection.

Formellement, l'unité de sélection peut être écrite en utilisant la notation fonctionnelle comme suit:

$$Select(SM, S, P, E) \rightarrow S'$$

Où SM est modèle contenant la description de la sélection, S est un ensemble d'individus et S' est un sous ensemble de S contenant les individus de S sélectionnés selon le contenu de SM. P et E. P et E sont, respectivement, les paramètres de la transformation et les stimuli de l'environnement.
Du point de vue niveau d'abstraction, les éléments des ensembles S et S' ont la même abstraction.

$$Abs(S) \approx Abs(S')$$

Prenons le cas d'un algorithme génétique la sélection des individus consiste à choisir un sous groupe d'un groupe déjà existant ; on voit très bien ici qu'aucune modification n'a été établis pour les individus sélectionnés, par conséquent les deux modèles S et S' ont la même abstraction.

4.1.1.3. L'unité architecturale de reproduction

La reproduction permet la recombinaison des modèles d'entrée moyennant les opérateurs génétiques (i.e. entrecroisement et la mutation) pour produire les modèles de sorties. Les paramètres de la transformation consistent en un ensemble d'éléments tels que le taux de mutation, le type d'entrecroisement, la méthode d'entrecroisement ...

Formellement, l'unité de production peut être écrite en utilisant la notation fonctionnelle comme suit:

$$Reproduce(RM, S, P, E) \rightarrow S'$$

Où RM est un modèle qui contient une description de la reproduction. S et S' sont des ensembles de modèles. Chaque élément de S' est obtenu de S (selon RM, les paramètres P et l'environnement E) à partir d'un ou de plusieurs éléments de S moyennant des mutations et des entrecroisements. Du point de vue niveau d'abstraction, les éléments des ensembles S et S' ont la même abstraction.
Par exemple dans le cas d'un système évolutionnaire le modèle P peut définir le type de représentation des individus qui est de type binaire pour les algorithmes génétiques et en arbre pour la programmation génétique

4.1.1.4. L'unité architecturale d'ajustement

L'ajustement peut être écrit:

$$Adjust(M, Ph, P, E) \rightarrow Ph'$$

Le modèle Ph' est obtenu de Ph après ajustement décrit par le modèle M. Ph et Ph' ont le même niveau d'abstraction.
D'un point de vue général, l'unité d'interprétation et celle d'ajustement ont la même expression fonctionnelle que l'unité de développement. Cependant, elles sont différentes lorsqu'on considère l'abstraction des modèles d'entrée et de sortie. Alors que le développement conduit à une abstraction plus grande, l'ajustement maintient le même niveau d'abstraction entre les modèles d'entrée et de sortie.

Par exemple lors de la partie apprentissage dans le cas d'un réseau de neurone le changement des poids de liaison se fait au moyen d'un modèle d'ajustement M qui donnera un nouveau modèles PH' mais sans plus au moins changé la topologie du réseau. Le modèle E représente dans ce cas là les exemples introduits au réseau pour l'apprentissage.

4.1.1.5. L'unité architecturale d'interprétation

L'unité d'interprétation accepte des modèles exécutables et des modèles de données en entrée et produit un modèle de données en sortie. L'interprétation peut être écrite formellement comme suit :

$$\text{Interpret}(Ph, I, P, E) \rightarrow O$$

Le modèle O est obtenu en transformant le modèle I selon la description donnée en Ph. Le niveau d'abstraction de I et O sont les mêmes. Cependant, comparé à Ph, ils peuvent avoir un niveau d'abstraction différent.
Par exemple dans le cas d'un réseau de neurones multicouches, le modèle P peut représenter les paramètres qui permettent le calcul du signal de chaque neurone caché.
Le processus d'interprétation et dans le cas d'un réseau de neurone permet d'interpréter les résultats d'apprentissage, si on fait subir au réseau un mauvais apprentissage, le modèle du résultat O va être complètement différent de la description PH et par conséquent le niveau d'abstraction entre ces deux modèles peut être différents
Apres avoir observé les unités architecturales et la description des comportements qu'ils permettent, nous pouvons déduire une représentation graphique plus lisible est semi formelle des composant de base, nous décrivons ci-dessous nos observations et les principaux concepts de cette nouvelle représentations.

- Les modèles constituant l'entrée d'une unité architecturale ne sont pas tous obligatoires.
- Les modèles constituant les sorties d'une unité architecturale ne sont pas tous obligatoires, il suffit d'avoir au minimum un modèle d'entrée et un modèle de sortie.
- Une unité architecturale peut être vue comme une transformation, elle doit posséder un nom et une ou plusieurs règles.

Pour ces raisons nous représentons une unité architecturale par le schéma de la figure 5.4.

La figure 5.4 représente une schématisation semi formelle d'une unité architecturale où des rectangles étiquetés représentent les modèles soit en entrée soit en sortie de la transformation. Une unité architecturale est représentée par un package qui est identifiés par un nom, elle accepte des modèles en entrée par le biais d'un pentagone rond vers l'extérieur, deux pentagones d'entrée peuvent exister dans une transformation : un pentagone sans remplissage qui définit que l'entrée est obligatoire et un pentagone avec un fond noir qui indique que le modèle en entrée est facultatif.
Pour la partie sortie les modèles sont connecté à des chevrons, deux sortes de chevrons existent, ceux sans remplissage qui indiquent que le modèle en sortie est obligatoire et ceux avec un remplissage noir qui indiquent que le modèle en sortie est facultatif.

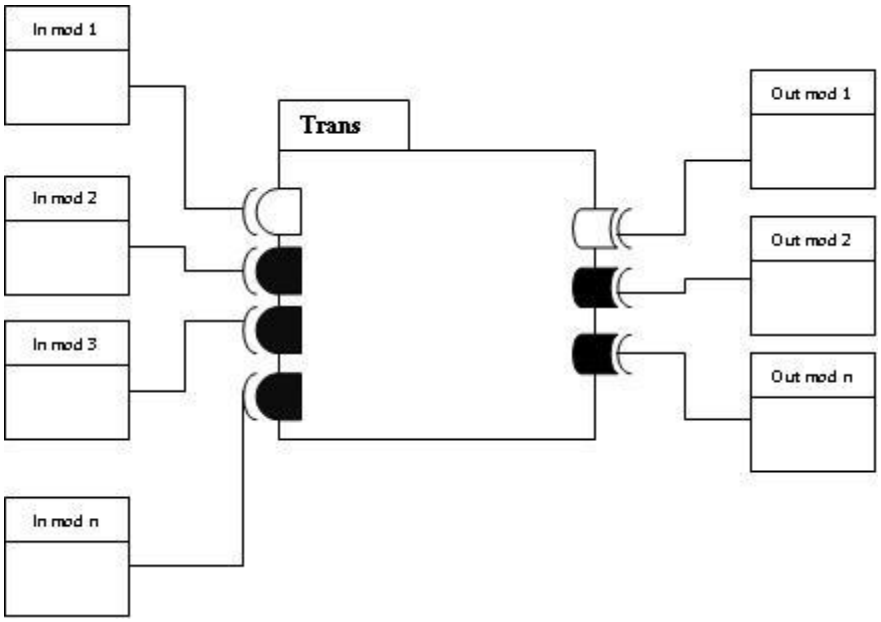


Figure 5.4. Représentation semi formelle de l'unité architecturale

La section suivante va présenter une description des vues architecturelles inspirées de la biologie.

4.1.2. Description des points de vue architecturaux selon les concepts inspirés de la biologie

D'après nos investigations dans le domaine de la bio inspiration, cinq vues architecturelles sont définies pour identifier un système bio-inspiré ci-après nous donnerons une description de chaque vue et sa représentation en composant de base.

4.1.2.1. La vue ontogénétique

La vue ontogénétique décrit l'ontogenèse : Comment la morphologie d'un individu au sein d'un système change au fil du temps.
Cette vue peut être modélisé par le constructeur de développement. Formellement l'unité de développement s'écrit avec la notation fonctionnelle :

$$Develop(D, M,S,E) \rightarrow M'$$

Où M' est obtenue de M par une modification selon la description D.

4.1.2.2. La vue phylogénétique

La vue phylogénétique est une vue architecturale en relation avec les caractéristiques de la phylogénèse du système. Cette dernière ne concerne pas la composition interne de l'individu mais plutôt la composition de cet individu au sein d'une population et son évolution d'une génération à l'autre.
Cette vue est construite en utilisant deux types d'unité architectural l'unité de reproduction et l'unité de sélection.

Select(SM, Reproduce(RM,S,P,E),P',E)→S'

4.1.2.3. La vue épigénétique

La vue épigenetique décrit l'épigenèse d'un système. L'épigenèse d'un système implique tout aspect du système qui est influencé par l'environnement. Cette dernière décrit comment la composition d'un individu change en fonction de l'environnement.
La vue épigenetique peut être modélisée par deux unités architecturales l'interprétation et l'ajustement.

Adjust (AM, Interpret (IM,S,P,E),P',E)→ S'

4.1.2.4. la vue morphogénétique

Est une vue utilisée pour décrire la structure des composants et des connecteurs qui font un individu. Contrairement à la vue ontogénétique, la vue morphogénétique décrit la structure statique des composants d'un individu et les connecteurs et aussi comment la structure peut changer au fil du temps [Van09]. Tous les éléments qui composent un individu se réfèrent au génotype de l'individu. Chaque composant peut être conçu en utilisant quelques vues. La vue morphogénétique utilise deux unités architecturales le développement et la reproduction.

Develop (D, Reproduce(RM,S,P,E),P',E)→S'

4.1.2.5. la vue environnementale

Comprend toutes les entités situées en dehors du système et desquels les composants du système sont en mesure de recevoir des feedbacks. Dans le cas de l'apprentissage par renforcement des changements des poids seront décrits dans la vue environnementale [Wey 07]. Le point de vue environnemental utilise une unité architecturale, celle de l'interprétation.

Interpret (IM,S,P,E) →S'

Après avoir fait une description des vues architecturales, nous entamons la description des processus formant l'approche POEtic décrite en section 2. Nous avons remarqué les insuffisances suivantes:

- Un processus POEtic est un processus itératif mais les composants de base vue plus haut ne permettent pas la représentation de cette itération
- Un processus POEtic est un processus certes itératif mais il va aussi vers le raffinement, par exemple dans un système évolutionnaire la population est raffinée d'une façon itérative.

Ces deux points nous ont conduits à proposer d'autres composants de base qui permettent la construction de processus complexes. La section 4.1.3 présentera les autres composants de base aidant à la modélisation des processus POEtic.

4.1.3. Autres Composants de base

Les unités architecturales précédentes ainsi que les processus Poetic qu'elles composent peuvent être combinés d'une multitude de manières selon l'objectif du système. A cette fin, on a besoin d'autres unités architecturales dites *constructives*. Nous les présentons ci-après:

- Assign (M, M'): assigne la valeur de M' à M
- Iterate (C, AU): exécute de façon répétée AU jusqu'à ce que la condition C soit satisfaite
- (AU1, AU2, ..., AUn): exécute séquentiellement AU1, AU2, ..., AUn où AUi est une unité simple ou composée
- (AU1 || AU2 ||... || AUn): exécute en concurrence AU1, AU2, ..., AUn où AUi est une unité simple ou composée
- GetElem (S): Retire et retourne un élément de l'ensemble S
- AddElem (I, S): Ajoute l'élément I à l'ensemble S

4.1.4. Description des processus POEtic

Moyennant les constructions architecturales précédemment décrites, on peut exprimer les processus Poetic comme suit:

4.1.4.1. Ontogenèse

Iterate (C, Assign(Ph, Develop(G,Ph, P, E)))

Cette expression signifie refaire l'opération de développement sur Ph, en utilisant G, jusqu'à ce que la condition C soit satisfaite. Ph et G sont deux modèles représentant le phénotype et le génome. Initialement Ph est vide.

Les figures 5.5 et 5.6 représentent la schématisation du processus ontogénétique sous forme de composants de base, pour des raison d'espace et de lisibilité nous avant décomposé ce processus sur deux figures la figure 5.5 montre le processus ontogénétique sous forme de : iterat(C,assign(PH,X)).

La figure 5.6 quand à elle, décrit la partie assign(PH, X) ou X et détaillé en develop (G,PH,P,E)

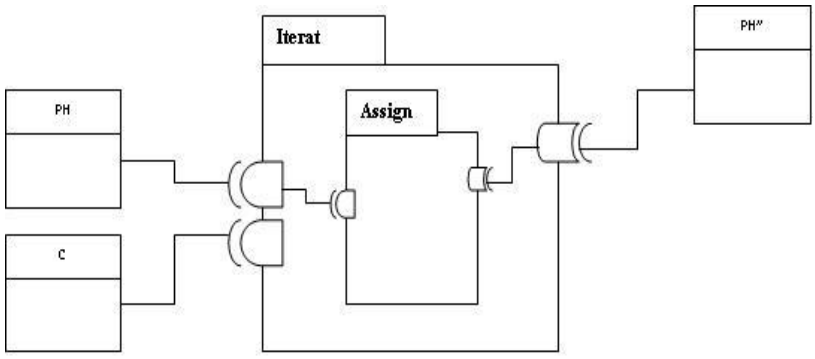


Figure 5.5. Représentation semi formelle des composants iterat et assign

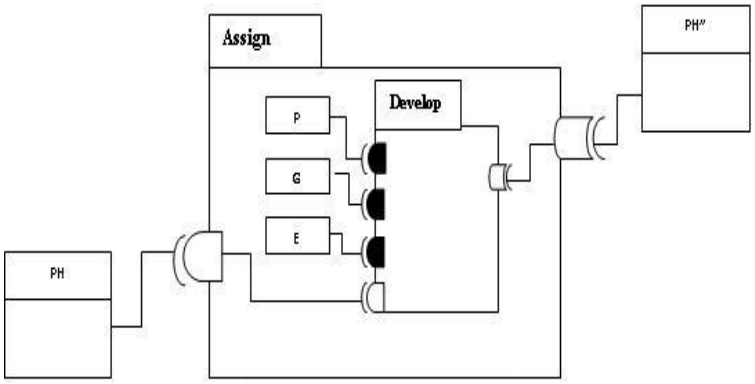


Figure 5.6. Représentation semi formelle des composants assign et devlop

4.1.4.2. Phylogenèse

Iterate (C, Assign(PH, Select(FM, Reproduce(RM,PH,P,E))))

C est une condition, S est un ensemble de modèles (e.g. des génomes d'individus), FM un modèle décrivant la fonction d'adaptation, et RM est un modèle décrivant les caractéristiques de la reproduction.

Les figures 5.7, 5.8 et 5.9 représentent une schématisation du processus POEtic phylogénétique ou la figure 5.7 montre le processus phylogénétique sous forme de iterat(assign(PH,X)) où X est représenté dans la figure 5.8 sous la forme, assign(PH, Select(FM,Y)).

La figure 5.9 montre en détails la formule select(FM,Y) ou Y représente la formule Reproduce(PH,RM,P,E)

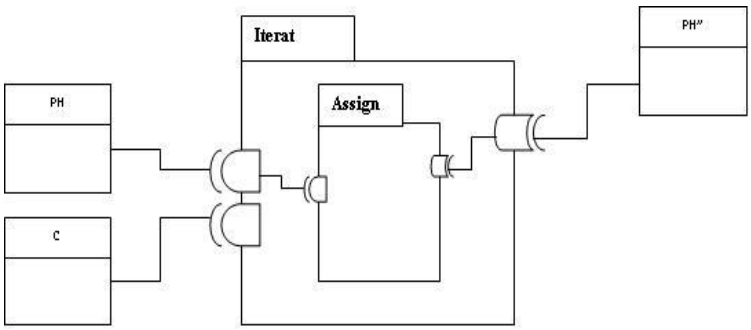


Figure 5.7. Représentation semi formelle des composants iterat et assign de la phylogenèse

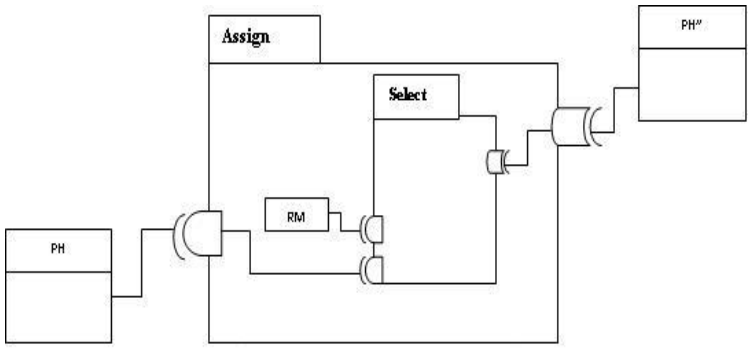


Figure 5.8. Représentation semi formelle des composants assign et select

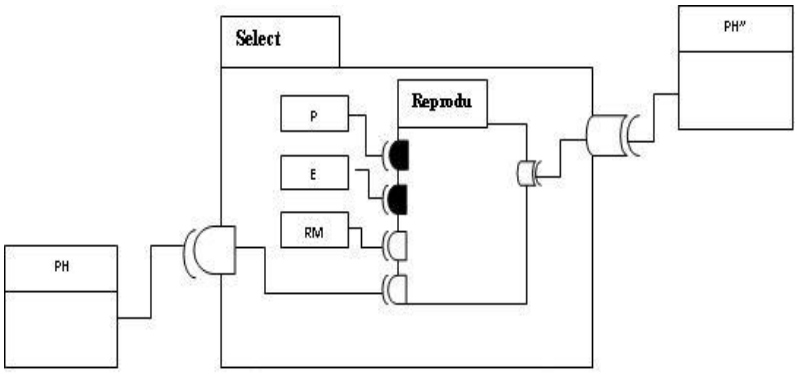


Figure 5.9. Représentation semi formelle des composants select et reproduce

4.1.4.3. Epigenèse

Pour représenter le processus d'apprentissage, on utilise l'architecture suivante :

```
(  Assign(M, Null),
   Iterate (C,(
     Iterate(SC, Assign(M, Develop(D, M, P, E))),
    Assign(D, Adjust(Interpret(M,IDM,P',E'), D, P'', E''))
                )
             )
)
```

Signifiant qu'un modèle M est construit de sa description D, puis interprété et le résultat de l'interprétation est utilisé pour ajuster sa description. Le processus se répète jusqu'à ce qu'une certaine condition C soit satisfaite. Null est la valeur nulle et SC une condition utilisée pour arrêter la construction du modèle M à chaque itération du bloc le plus externe.

Dans la figure 5.10 nous représentons le processus épigénétique d'une manière générale, nous présenterons dans les figures 5.11 ,5.12 et 5.13 les détails de chaque composant.

La figure 5.11 représente en détails la partie en rouge dans la formule ci-dessous

```
(  Assign(M, Null),
   Iterate (C,(
     Iterate(SC, Assign(M, Develop(D, M, P, E))),
    Assign(D, Adjust(Interpret(M,IDM,P',E'), D, P'', E''))
                )
             )
)
```

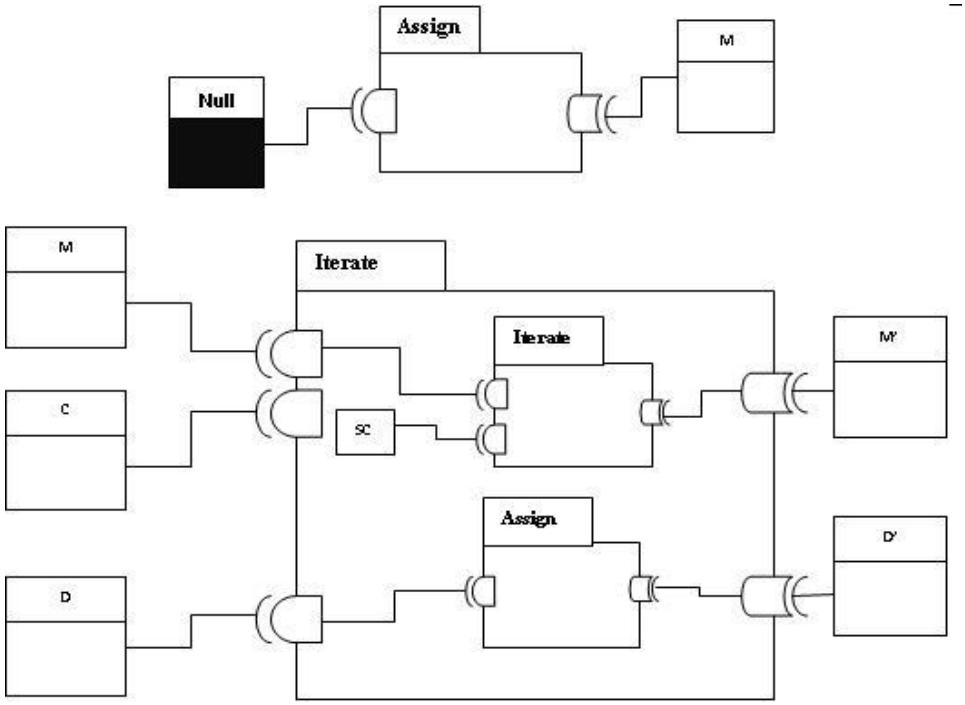


Figure 5.10. Représentation graphique du processus épigénétique

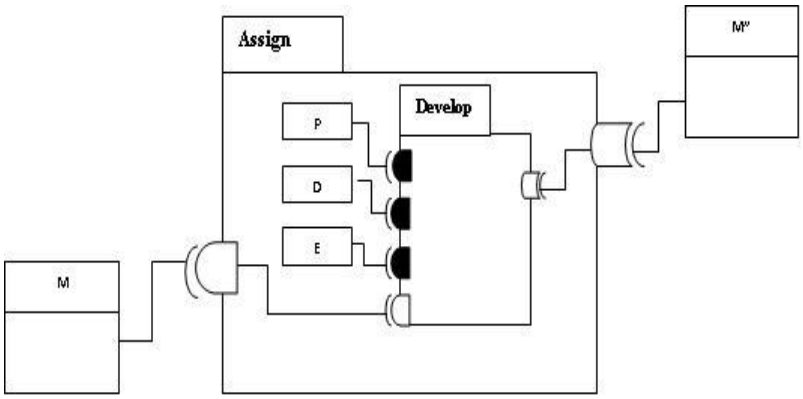


Figure 5.11. Représentation semi formelle des composants assign et develop

Le processus épigénétique est décomposé en deux parties, la partie création et la partie dite apprentissage. Les figures 5.12 et 5.13 représentent la partie apprentissage dans un processus épigénétique.

```
(  Assign(M, Null),
   Iterate (C,(
     Iterate(SC, Assign(M, Develop(D, M, P, E))),
    Assign(D, Adjust(X, D, P'', E''))
                )
             )
)
```

Où X represente Interpret(M,IDM,P',E')

La figure 5.12 représente la partie en rouge dans la formule ci-dessus

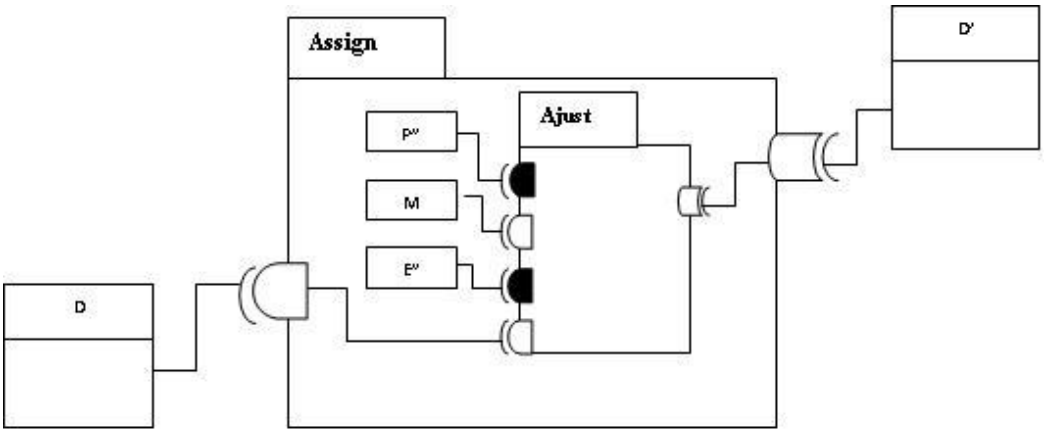


Figure 5.12. Représentation semi formelle des composants assign et Ajust

La figure 5.13 quand à elle, représente le résultat que donne le processus épigénétique qui est un modèle D dit descripteur et un modèle M qui est notre modèle final. On peut donner comme exemple d'un processus epegenetquie un réseau de neurones à reconnaissance de formes où M représente le modèle finale s'il est reconnu ou pas et D représente l'évolution des pois du réseau.

```
(  Assign(M, Null),
   Iterate (C,(
     Iterate(SC, Assign(M, Develop(D, M, P, E))),
    Assign(D, Adjust(Interpret(M,IDM,P',E'), D, P'', E''))
                )
              )
)
```

La figure 5.13 représente la partie en rouge dans la formule ci-dessus.

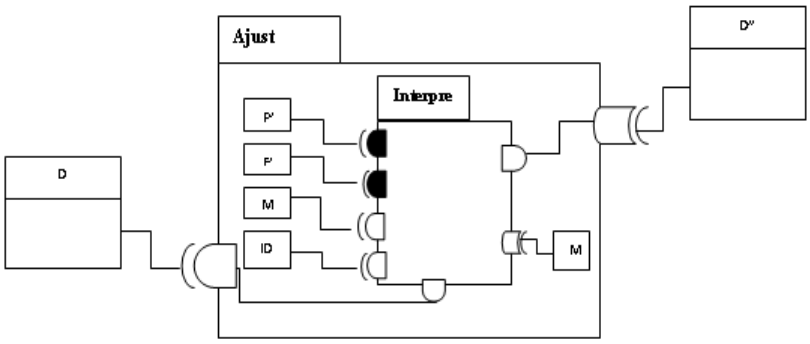


Figure 5.13. Représentation semi formelle des composants Ajust et Interpret

Apres avoir décrit les processus de l'approche POEtic sous forme de composant de base nous avons pu montrer l'aspect fonctionnelle de chacune des branches de l'approche ainsi nous

pouvant décrire n'importe quelle système en utilisant les composants de base. Mais notre démarche reste insuffisante ; en effet :

- La modélisation d'un système sous forme de composant de base ne permet pas de décrire le rôle de chaque modèle.
- La schématisation d'un système sous forme de composants de base ne permet pas de décrire la structure de chaque modèle

La section suivante va combler les insuffisances décrites plus haut en exposant la partie structurelle de cette démarche

4.2. Description de la structure d'un système bio-inspiré

Définir la structure d'un système bio-inspirée revient à définir un certain nombre d'aspect qui façonnent chacun à sa manière ce système. Nos investigation nous on mené à extraire six critères qui définissent la structure d'un système bio-inspiré.
Nous décrivons ci-après l'ensemble des critères que nous proposons.

4.2.1. Rôle

Un modèle peut jouer deux rôles possibles dans les unités de transformation où il est impliqué. Un rôle d'individu ou un rôle d'espèce. De plus, un même modèle peut jouer simultanément un rôle d'individu dans une transformation et un rôle d'espèce dans une autre.

Prenant l'exemple du cas du voyageur de commerce résolu par un système bio-inspiré basé sur le concept des colonies de fourmis, l'unité de transformation de développement qui permet la création de trajectoire prend comme entrée le modèle du problème concret modélisé sous forme de diagramme UML où les nœuds et les arcs constituent le corps de notre modèle. Ici on peut percevoir l'ensemble des sommets du graphe comme étant une population et par conséquent le modèle ici joue le rôle de population. La transformation de développement pour le modèle du voyageur de commerce donne comme résultat un ensemble de trajectoires possibles où on peut dire un ensemble de solutions possibles, ici le résultat est une trajectoire à la fois, on peut voir ce résultat comme étant un individu. Ici un même modèle peut jouer à la fois le rôle d'individu et de population.

Autre cas, l'exemple du Xor représenté par un réseau de neurone artificiel où la transformation d'ajustement fait évoluer le poids des nœuds du réseau de neurones, la transformation touche la classe lien du système de neurones. Cette dernière peut être considérée comme un modèle et l'ajustement ici transforme l'ensemble des nœuds qui peuvent être considérés comme des individus pour former des liens. Ici aussi, le modèle en entrée joue le rôle de population et donne comme résultat un individu qui et le résultat du Xor. Le modèle en sortite joue un rôle d'individu. Mais si on considère le problème du voyageur de commerce résolu par un algorithme génétique, lors de la transformation de reproduction, où on utilise le croisement, le modèle en entrée est une population (deux individus ou plus) qui vont être croisés pour donner une autre population. Ici les modèles jouent un rôle de population. Pour ce qui est de l'opération de mutation, elle est complètement différente de la transformation de mutation qui opère sur des modèles en tant qu'individu, le rôle du modèle est par conséquent l'individu.

4.2.2. Type de description

Un modèle peut être un génome, un phénotype ou toute autre forme de description. Les modèles génomes sont souvent codés en utilisant des symboles de bas niveau (e.g. une chaîne de bits). A l'opposé, les phénotypes sont plus abstraits. Dans les systèmes bio-inspirés, divers types de descriptions sont utilisées dans les modèles : liste de symboles, L-systems [Lin 88], programmes, réseaux de neurones, règles, modules de données, ... [Ada 98].

La description d'un individu ou d'une population peut varier d'un système bio-inspiré à un autre. Prenons l'exemple d'un système évolutionnaire où les individus peuvent être représentés par des chromosomes sous forme d'un ensemble de bit. De même dans le cas des réseaux de neurone artificiel pour la reconnaissance des lettres de l'alphabet on peut représenter les individus sous forme de matrice représentant les pixels d'un caractère. Dans un système immunitaire artificiel visant la détection des intrusions, un individu est représenté sous la forme d'un graphe. Dans la programmation génétique les individus sont représentés sous forme d'arbre. Les modèles peuvent être implémentés sous forme de circuits électroniques ou stockés dans une certaine mémoire.

Quelque soit leur type de description, tous les modèles sont interprétables. L'unité architecturale d'interprétation est capable de propager des activations des neurones d'entrée aux neurones de sortie du réseau neuronal, d'inférer les règles d'un ensemble de règles, d'exécuter un programme, ... Les autres unités architecturales peuvent aussi être vues comme des interpréteurs de modèles par diverses manières : extraction d'éléments, comparaison, addition, ...

4.2.3. Elément/Ensemble

Le modèle peut être un élément simple comme il peut être un ensemble d'éléments semblables.

Selon le niveau d'abstraction ou la hiérarchie d'abstraction on peut considérer un modèle comme étant un élément d'un autre modèle et à chaque fois qu'on monte dans l'abstraction on va aller vers un seul élément.

Un modèle peut être un ensemble d'éléments dans un certain niveau d'abstraction et peut être un seul élément dans un autre niveau.

L'exemple le plus concret est celui du voyageur de commerce résolu par un système bio-inspiré basé sur le concept des colonie de fourmis, l'unité de transformation de développement qui permet la création de trajectoire prends comme entrée le modèle UML qui est constitué de plusieurs classes parmi ces dernières on note la classe nœud et la classe arc qui sont considérés comme un ensemble d'éléments, la transformation de développement va générer des trajectoires en transformant le modèle global et à partir des informations se trouvant dans les classes arc et nœud, c'est-à-dire les classes arc et nœud sont des éléments dans un ensemble qui constitue le modèle.

4.2.4. Granularité

Indique le plus petit élément pouvant être transformé. Les modèles peuvent être d'une granularité fine, moyenne, forte, etc. Lorsqu'on utilise la phylogenèse pour ajuster un réseau de neurones, le grain est le poids attaché à chaque connexion. Dans d'autres cas, le grain peut être un symbole, une règle, une instruction, une fonction dans un programme, etc. Le grain le plus fin étant le bit.
La granularité peut être variable d'une unité de transformation à une autre, si on prend l'unité de sélection dans un système à base d'algorithme génétique la granularité ici est grande puisque c'est l'individu lui-même qui va être considérer comme grain. Dans le même exemple, si on prend l'unité de transformation de reproduction avec la mutation, le grain est plus fins il devient un bit ou plusieurs selon le type de description de l'individu.

4.2.5. Altérabilité

Elle définit la facilité avec laquelle un modèle peut être altéré. Les modèles peuvent être hautement altérables lorsqu'ils sont stockés en mémoire. Ils sont moins altérables (ou juste reconfigurables) lorsqu'ils sont implémentés dans un circuit électronique.

L'altérabilité ici joue un rôle primordial pour discriminer les systèmes hardware et software. De plus l'altérabilité peut être manuelle, partiellement ou totalement automatique.

4.2.6. Composition

Un modèle peut être simple ou composé, un modèle composé peut être décomposé en sous modèles et transformations.

L'or du processus de transformation qui est un processus itératif le principe est la décomposition, le système dans sa totalité est décomposé en un ensemble d'unités de transformation (section 4.4). La description fonctionnelle d'un système bio-inspiré par les unités de transformation est vue comme un modèle qui va être décomposé au fur est à mesure. Les processus peuvent être caractérisés par les expressions fonctionnelles composées de modèle et d'unités architecturales, et peuvent être comparés en termes de présence/absence d'unités architecturales, leurs imbrications (i.e. profondeur de l'arbre d'expression) et le séquencement dans un bloc. Un attribut de qualité général, concernant le degré d'enchevêtrement des processus principaux du système, peut être déterminé. Un tel attribut peut par exemple s'écrire : les processus P1 et P2 sont fortement/faiblement couplés.

L'expression fonctionnelle est suffisamment précise et formelle pour supporter une analyse automatique. Dans certains cas, une unité architecturale peut être utilisée récursivement comme un modèle dans une transformation de haut niveau. Ceci permet, par exemple, de déterminer les caractéristiques des unités architecturales par un processus phylogénétique d'un niveau supérieur. Un autre critère est la nature automatique, semi-automatique ou manuelle des processus. Dans certains systèmes, un processus ontogénétique existe mais il est exécuté par un être humain.

Le système peut être caractérisé vis à vis de l'environnement où il opère, par deux critères : stimuli et adaptabilité. Si l'environnement change et le système réagit à ce changement, alors on dit que le système est adaptable. Un degré important d'adaptabilité est souhaitable.

Par conséquent, l'adaptabilité est une caractéristique hautement discriminative. Les stimuli sont des événements générés par l'environnement et perçus ou non par le système. Un système peut percevoir un évènement sans réagir à ce dernier.

Dans la section suivante nous allons discuter du faite que les processus biologiques peuvent être en rapport et peuvent être un apport les unes par rapport aux autres :

4.3. Relation entre processus biologiques

Dans la section 4.4 nous avons caractérisé les processus biologiques par les expressions fonctionnelles suivantes :

Ontogenèse

Iterate (C, Assign(Ph, Develop(G,Ph, P, E)))

Phylogenèse

Iterate (C, Assign(S, Select(FM, Reproduce(RM,S,P,E))))

Epigenèse

```
(   Assign(M, Null),
   Iterate (C,(
                Iterate(SC, Assign(M, Develop(D, M, P, E))),
               Assign(D, Adjust(Interpret(M,IDM,P',E'), D, P'', E''))
                )
           )
)
```

Si on analyse de près ces expressions, on peut déduire des ressemblances. Alors que l'ontogenèse fait évoluer un phénotype (Ph) en utilisant une représentation de bas niveau (le génome G), la phylogenèse fait évoluer un ensemble d'individus (S) moyennant une fonction d'adaptation (FM) et des opérations non déterministes. Si on considère cet ensemble (S), en faisant abstraction des individus qui le composent, comme un seul phénotype, on constate que la phylogenèse opère un développement aussi avec un génome sous la forme d'une fonction d'adaptation. Ce fait est perceptible si on considère une colonie de fourmis, chaque individu de la colonie évolue par ontogenèse, mais la colonie dans son ensemble évolue par phylogenèse. La colonie peut être considérée dans son ensemble comme un seul individu.

En effet, les fourmis ailées bougent leur ailes pour aérer la nid (c'est la fonction de respiration), les fourmis soldats défendent le nid contre les attaques (c'est la fonction de défense, système immunitaire), la reine produit les oeufs (elle assure la fonction de reproduction), d'autres réparent le nid, ... A l'opposé, un animal peut être vu comme une

collection importante de cellules plus ou moins semblables, à quelques exceptions près, ces cellules meurent et sont régénérées continuellement. On peut, dans ce cas, y voir un processus phylogénétique particulier.

De son côté, l'épigenèse dote l'individu (M) d'une multitude de propriétés. Ces propriétés sont comportementales et facilement altérables. On peut voir, dans l'épigenèse, une ontogenèse particulière qui développe des propriétés faciles à altérer, le génome étant les connaissances supportées par la société où vit l'individu, qui ajustent la description (D) de (M). De même, l'ontogenèse ressemble à une épigenèse qui octroie des propriétés difficiles à altérer à l'individu.

Dans le processus d'apprentissage, on remarque aussi l'existence d'une forme de phylogenèse où les individus sont des solutions possibles d'un problème donné, qu'on améliore et sélectionne jusqu'à arriver à une solution convenable.
De ce qui précède on peut conclure que les trois processus ont des ressemblances mais diffèrent sur quelques points :

- L'altérabilité du support utilisé
- Le niveau d'abstraction
- La fréquence de déroulement des cycles des processus dans le temps
- L'importance de l'intervention de l'environnement dans le déroulement du processus

Le schéma de la figure 5.14, résume nos propos.

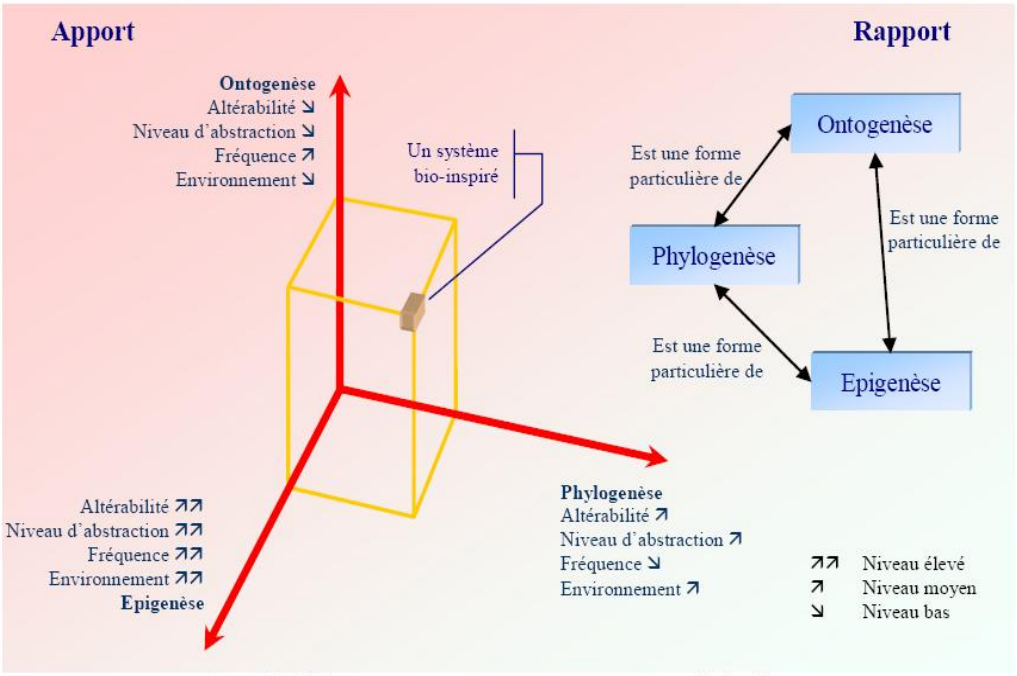


Figure 5.14. Apport et rapport entre processus biologiques

Face à un environnement changeant et contraignant, les organismes biologiques sont parvenus à une forme d'organisation où ils sont façonnés par trois processus hautement

complémentaires et robustes. L'ontogenèse donne à l'individu une possibilité de grandir, se réparer et s'adapter aux changements peu fréquents. La phylogenèse donne à une espèce la possibilité de faire face aux changements rares et non prévus. L'épigenèse permet à l'individu de supporter les changements fréquents et profonds.

Nous souhaitons décrire une multitude de systèmes bio-inspirés avec beaucoup de variantes, le problème qui se pose dans ce cas la est la multitude de données et de paramètres, l'utilisation du principe de transformation de modèle est un choix très judicieux car elle repousse toutes ces différences dans les modèles, les modèles peuvent encapsuler les données et paramètres et nous donné une image plus clair et plus concise des systèmes que nous étudions ou nous développons. De même la MDA par sa qualité de préconisation du savoir faire nous permet d'avoir plus de détails dans l'aspect comportemental d'un système en cour de fonctionnement.

La modélisation dans notre contexte vise la représentation du système en cours de fonctionnement et ne prend pas en compte les plateformes d'implémentation ou de déploiement. Représenter un système en cours de fonctionnement ne signifie pas la description de son comportement dynamique, mais plutôt la description de son état à un moment donné durant la progression de traitement.

La transformation de modèles est usuellement effectuée dans le cadre de développement d'une application ou sa maintenance (transformation dans le sens inverse). Nous avons vu que le processus MDA consiste à transformer les modèles PIMS aux modèles PSMs, puis générer le code source de l'application. Principalement, le processus de développement est composé d'un ensemble d'étapes (analyse, conception, implémentation, etc.) et MDA préconise l'utilisation des modèles et transformations des modèles pour passer d'une étape à une autre en profitant des avantages des modèles et leur transformation.

Similairement à un processus de développement de logiciel, le fonctionnement d'un système bio-inspiré peut être vu comme une succession d'étapes. Chaque étape correspond à un état particulier du système. Nous parlerons donc plus d'étapes, mais plutôt d'états dans lesquels le système peut se trouver durant son fonctionnement. Nous désignons par l'état du système l'ensemble de valeurs permettant d'expliquer son évolution pendant son fonctionnement. L'état initial du système correspond au début de son fonctionnement tandis que l'état final correspond à la fin de son fonctionnement. La transformation consiste à faire transiter le système d'un état à un autre jusqu'à qu'il atteint l'état final. La figure 14 résume ce principe.

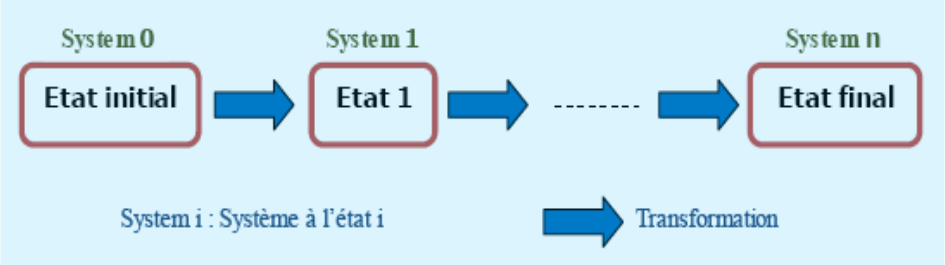


Figure 5.15. Évolution du système durant son fonctionnement

Si nous partons de ce principe, nous devons fournir à l'entrée de la transformation un modèle qui représente le système à un état donné durant son fonctionnement et le résultat devrait être un modèle qui représente le même système mais dans l'état suivant. Pour décrire le fonctionnement complet du système nous somme donc obligés d'effectuer une succession de transformations. Un système qui passe par n+1 états successifs à travers son fonctionnement nécessite l'exécution de la transformation n fois pour arriver à l'état final.

D'après ce qui précède nous avons été amené à créer des patterns dits de transformation qu'on va expliquer dans la section suivante.

4.4. Les patterns de transformation

Nous allons décrire dans cette section les patterns de transformations appliqués dans les systèmes bio-inspirés décrits au chapitre 3.

4.4.1. transformation de développement

La figure qui va suivre représente la modélisation en UML 2.0 du pattern de la transformation de développement, qui a besoin en entrée d'un modèle UML et fournit un nouveau modèle en sortie (transformation horizontale) les modèles en entrée et en sortie appartiennent au même méta modèle. On a utilisé le pattern stratégie pour la transformation du développement. La transformation de développement permet la création de nouveau individus.

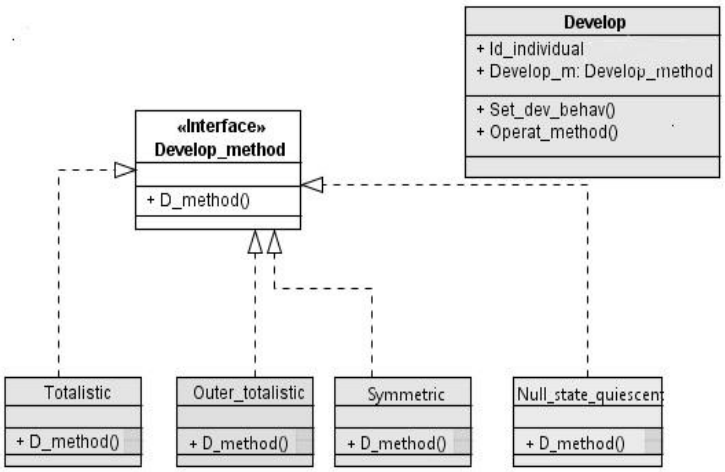


Figure 5.16. Vue du méta modèle de la développement

La classe Develop possède deux attributs:
Id_individual: qui identifie un individu dans une population.
Develop_m: cet attribut est utilisé dans la méthode Set_dev_behav pour décrire le comportement du développement.
Operat_method : est une méthode utilisée pour encapsuler le comportement du développement
D_method: c'est la méthode utilisée pour chaque type de développement (total, symétrique…), elle définie une règle pour chaque type de développement.

4.4.2. transformation de sélection

La figure ci-dessous représente la modélisation en UML 2.0 du pattern de la sélection, qui a besoin en entrée d'un modèle de population et fournit une nouvelle population en sortie. On a utilisé le pattern stratégie pour la transformation de la sélection.

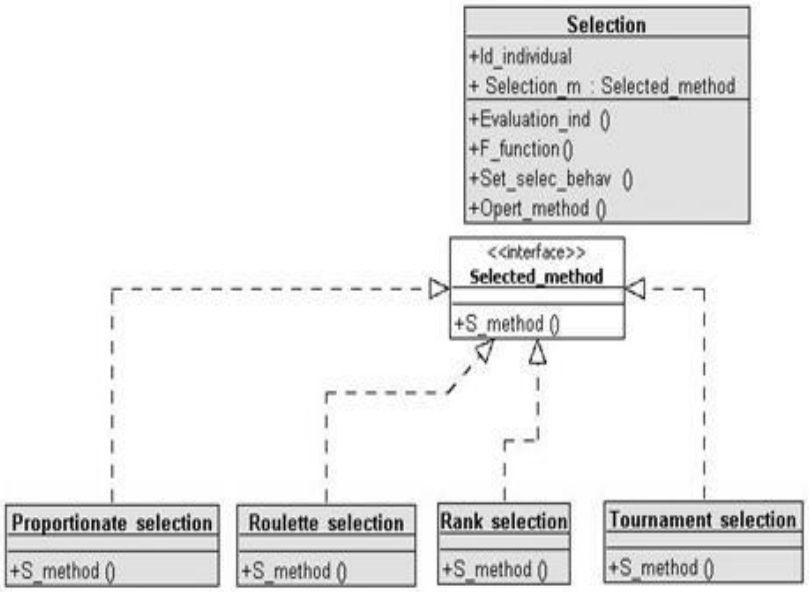


Figure 5.17. Vue du méta modèle de la sélection

La classe sélection possède deux attributs:
Id_individual: qui identifie un individu dans une population.
Selection_m: cet attribut est utilisé dans la méthode Set_selec_behav pour décrire le comportement de la sélection.
Operat_method : est une méthode utilisée pour encapsuler le comportement de la sélection
S_method: c'est la méthode utilisée pour chaque type de sélection (roulette, rank…), elle définie un algorithme pour chaque type de sélection.

4.4.3. transformation d'ajustement

La figure 5.18 montre la représentation en UML 2.0 du pattern d'ajustement qui accepte en entrée un modèle d'individu et fournie en sortie un nouveau modèle d'individu. On a utilisé le pattern stratégie pour modéliser la transformation d'ajustement. La transformation d'ajustement permet un changement ou une évolution des individus selon une démarche différente de celle de la mutation et du croisement.

La classe ajustement a deux attributs:
Id_individual : qui identifie un individu
Adjust_m: il est utilisé comme paramètre dans la méthode Set_adj_behav Set_adj_behav : cette méthode décrit le comportement de l'ajustement Operat_method: cette méthode est utilisée pour encapsuler le comportement de l'ajustement et utilise A_method pour chaque type d'ajustement (metropolis, retropropagation of error…).

A_method : représente un algorithme pour chaque ajustement.

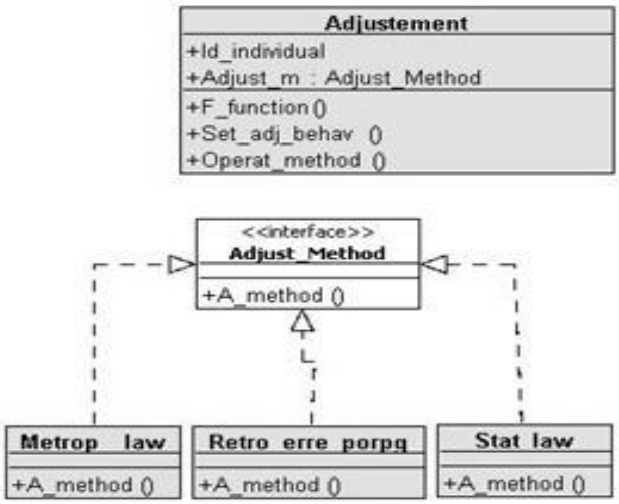


Figure 5.18. Vue du méta modèle de l'ajustement

4.4.4. transformation de reproduction

La figure ci-dessous montre la représentation en UML 2.0 du pattern de reproduction qui accepte en entrée un modèle d'individu et fournie en sortie un nouveau modèle d'individu. On a utilisé le pattern stratégie pour représenter la transformation de reproduction.

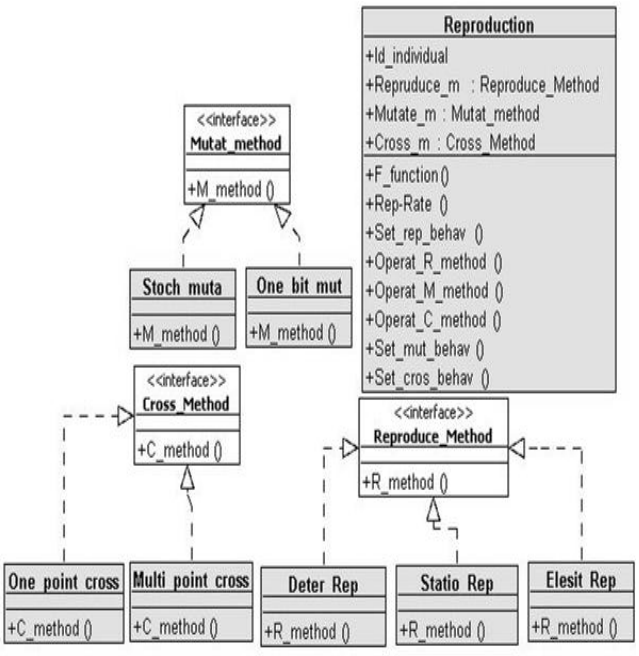


Figure 5.19. Vue du méta modèle de la reproduction

La classe reproduction possède quatre attributs :

Id_individual : qui identifie un individu
reproduce_m: il est utilisé comme paramètre dans la méthode Set_rep_behav Mutate_m et Cross_m : ces deux attributs sont utilisés respectivement dans les méthodes Set_Mut_behav et Set_cros_behav
Set_rep_behav: cette méthode décrit le comportement de la reproduction. Set_mut_behav et Set_cros_behav , ces méthodes sont utilisées respectivement pour décrire le comportement de la mutation et le croisement.
Operat_R_method, Operat_M_method et Operat_C_method sont utilisées respectivement pour encapsuler les comportements de la reproduction, la mutation et le croisement.
R_method : représente un algorithme pour chaque reproduction.
C_method : représente un algorithme pour chaque croisement.
M_method : représente un algorithme pour chaque mutation.

4.4.5. Transformation d'interprétation

La transformation d'interprétation prends en entrée un ou plusieurs modèles UML et produit en sortie un modèle UML (les modèles en entré et en sortie appartiennent au même meta modèle). Cette transformation produit un résultat qui nous permet de voir si notre systèmes s'adapte bien avec l'environnement où il opère ou pas

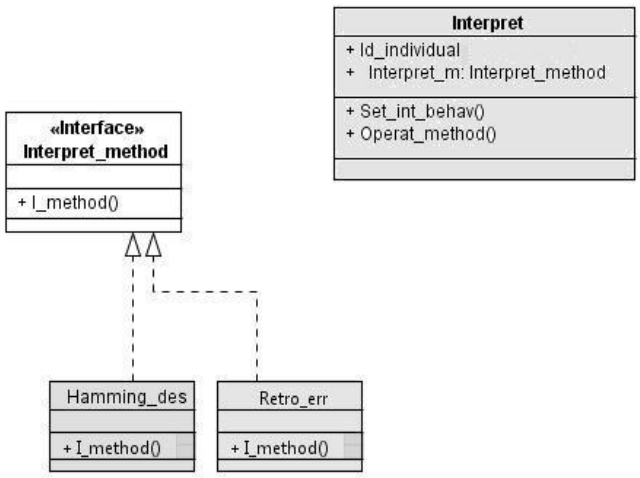


Figure 5.20. Vue du meta modele de l'interpretation

La classe Inerpret possède deux attributs et deux méthodes:
Id_individual: qui identifie un individu dans une population.
Interpret_m: cet attribut est utilisé dans la méthode Set_int_behav pour décrire le comportement de l'interprétation.
Operat_method : est une méthode utilisée pour encapsuler le comportement de l'interpretation.
D_method: c'est la méthode utilisée pour chaque type d'interprétation (calcule d'erreurs, calcule d'affinité…),

5. Conclusion

Nous avons proposé une nouvelle approche pour classer les systèmes bio-inspirés qui repose à la fois sur l'aspect structurel et l'aspect comportemental.

Il est à noter qu'une bonne classification des systèmes bio-inspirés est très utile pour le travail de conception de systèmes et d'amélioration des approches. Comme ces approches visent, en grande partie, la résolution des problèmes d'adaptation et d'évolution, nous avons été amenés à proposer des critères de classification et de comparaison qui permettent de situer l'apport de chaque approche, en précisant celui de chaque critère pour l'évolution.

A l'issue de ce chapitre, nous pouvons tirer les conclusions suivantes :

- Il existe une grande diversité dans les systèmes bio-inspirés.
- Les trois processus Poetic qui façonnent les organismes biologiques ne sont, en fait, que des processus complémentaires de l'évolution. Chacun utilise des concepts et des mécanismes spécifiques pour doter l'organisme, ou l'espèce, d'une aptitude à faire face à un environnement changeant et contraignant.
- Un système bio-inspiré peut être caractérisé vis à vis de l'environnement où il opère, par deux critères : stimuli et adaptabilité
- L'étude des systèmes bio-inspirés en se basant sur l'ingénierie des modèles nous a permit d'extraire des Template ou en peut dire des patterns de transformation permettant la conception et la réalisation d'une manière très claire une grande variété de systèmes bio-inspirés.

Dans le chapitre suivant nous allons utiliser les concepts vue dans ce chapitre pour classer quelques systèmes bio-inspirés.

6. Références

[Ada 98] C.Adami, Introduction To Artificial Life, Springer-Verlag New York, Inc., 1998.

[Ast 00] J.C.Astor, C.Adami, A Developmental Model For The Evolution Of Artificial Neural Networks, Artificial Life, 6(3) :189–218, 2000.

[Bac 97] T.Back, U.Hammel,H.P.Schwefel, Evolutionary Computation : Comments On The History And Current State, Ieee Transactions On Evolutionary Computation, 1(1) :3–17, Avril 1997.

[Bor 00] S.Bornholdt,T.Rohlf, Topological Evolution Of Dynamical Networks : Global Criticality From Local Dynamics, Physical Review Letters, 84(26) :6114–6117, Juin 2000.

[Deg 93] H. De Garis,Growing An Artificial Brain With A Million Neural Net Modules Inside A Trillion Cell Cellular Automaton Machine, Dans Proc. Of The Fourth International Symposium On Micro Machine And Computer Science, Pages 211– 214, 1993.

[Deg 97] H.De Garis, L.Kang, Q.He, Z.Pan, M.Ootani & E.Ronald, Million Module Neural Systems Evolution : The Next Step In Atr's Billion Neuron Artificial Brain,Dans Proceedings Of Evolution Artificielle 97 (Ea'97), Volume 10, Pages 231–243, 1997.

[Fer 95] J.Ferber, Les Systèmes Multi-Agents, Vers Une Intelligence Collective, Intereditions, Paris, 1995.

[Flo 98] D.Floreano & J.Urzelai, Evolution And Learning In Autonomous Robotics Agents, Dans D. Mange & M. Tomassini, Editeurs, Bio-Inspired Computing Machines, Chapitre 12, Pages 317–346. Ppur, Lausanne, 1998.

[For 96] S.Forrest, Genetic Algorithms, Acm Computing Surveys, Vol. 28, Issue 1, March 1996, Pp. 77-80.

[Goo 00] C.S.Goodman & B.C.Coughlin,The Evolution Of Evo-Devo Biology,Proceedings Of The National Academy Of Sciences Of The Usa, 97(9) :4424– 4425, Avril 2000.

[Gru 92] F. Gruau. Genetic Systems Of Boolean Networks With A Cell Rewriting Developmental Process. Dans D. Whitley & S.D. Schaffer, Editeurs, Combination Of Genetic Algorithms And Neural Networks, Pages 55–74. Ieee Computer Society Press, Los Alamitos, Ca, 1992.

[Had 01] P.C. Haddow, G. Tufte & P. Van Remortel,Shrinking The Genotype : L-Systems For Ehw?,Dans Ices '01 : Proceedings Of The 4th International Conference On Evolvable Systems : From Biology To Hardware, Pages 128–139. Springer-Verlag, 2001.

[Heu 94] J. C. Heudin, La Vie Artificielle, Edition Hermes, 1994.

[Hol 75] J.H.Holland, Adaptation In Natural And Artificial Systems, The University Of Michigan Press, Ann Arbor, Mi, 1975.

[Hol 99] P.W.H.Holland,The Future Of Evolutionary Developmental Biology,Nature, 402 :41–44, Décembre 1999.

[Jai 96] A.K.Jain, Artificial Neural Networks: A Tutorial, Ieee Computer, March 1996, Pp 31-44.

[Ken 95] J. Kennedy & R.Eberhardt, Particle Swarm Optimization , Dans Proc, Ieee International Conference On Neural Networks, Volume 4, Pages 1942– 1948, 1995.

[Kit 90] H.Kitano,Designing Neural Networks Using Genetic Algorithms With Graph Generation System, Complex Systems, 4(4) :461–476, 1990.

[Koz 92] J.R.Koza, Genetic Programming : On The Programming Of Computers By Means Of Natural Selection, Mit Press, Cambridge, Ma, 1992.

[Lin 68] A.Lindenmayer, Mathematical Models For Cellular Interaction In Development, Parts I And Ii, Journal Of Theoretical Biology, 18 :280–315, 1968.

[Lin 88] A.Lindenmayer And P.Prusinkiewiscz, Developmental Models Of Multicellular Organisms: A Computer Graphics Perspective, Addison-Wesley, 1988.

[Man00] D. Mange, M. Sipper, A. Stauffer & G. Tempesti, Towards Robust Integrated Circuits : The Embryonics Approach, Proceedings Of The Ieee, 88(4) :516–541, Avril 2000.

[Man98] D. Mange & M. Tomassini, Bio-Inspired Computing Machines : Towards Novel Computational Architectures, Presses Polytechniques & Universitaires Romandes, Lausanne, Switzerland, 1998.

[Mes 03] B. Mesot, E. Sanchez, C.-A. Pena & A. Perez-Uribe, Sos++ : Finding Smart Behaviors Using Learning And Evolution, Dans R.K. Standish M.A. Bedau &

H.A. Abbass, Editeurs, Proceedings Of The Eighth International Conference On Artificial Life, Pages 264–273, Cambridge, Mass., 2003.

[Mes 06] D.Meslati, Mage : Une Approche Ontogénétique De L'évolution Dans Les Systèmes Logiciels Critiques Et Embarqués. Thèse De Doctorat D'état, Université De Annaba, Février 2006.

[Mil 04] J.F.Miller,Evolving A Self-Repairing, Self-Regulating, French Flag Organism, Gecco 2004, Volume 1 De Lecture Notes In Computer Science, Pages 129–139, Berlin, Heidelberg, 2004. Springer.

[Moo01] S.W.Moon & S.G. Kong, Block-Based Neural Networks ,Ieee Transactions On Neural Networks, 12(2) :307–317, Mars 2001.

[Ozt 04] C. Ozturkeri & M. Capcarrere, Emergent Robustness And Self-Repair Through Developmental Cellular Systems, Proc. Ninth International Conference On The Simulation And Synthesis Of Living Systems (Alife9), Pages 31–26, Cambridge, Massachusetts, Usa, 2004. The Mit Press.

[Per 99] A.Perez-Uribe,Structure-Adaptable Digital Neural Networks, Phd Thesis, Swiss Federal Institute Of Technologiy-Lausanne, Epfl, 1999.

[Rec 73] I.Rechenberg, Evolutionsstrategie : Optimierung Technischer Systeme Nach Prinzipien Der Biologischen Evolution, Frommann-Holzboog, Stuttgart, Germany, 1973.

[Rog 03] D. Roggen, D.Floreano & C. Mattiussi,A Morphogenetic Evolutionary System : Phylogenesis Of The Poetic Tissue, Proceedings Of The Fifth International Conference On Evolvable Systems (Ices 2003), Pages 153–164, Berlin, 2003. Springer.

[Seg 98] R. Segev & E. Ben-Jacob,From Neurons To Brain : Adaptive Self-Wiring Of Neurons, Journal Of Complex Systems, 1 :67–78, 1998.

[Sip 97] M. Sipper & Al, A Phylogenetic, Ontogenetic, And Epigenetic View Of Bio-Inspired Hardware Systems, Ieee Transactions On Evolutionary Computation, Vol. 1, No. 1, April 1997, Pp 83-97.

[Tem97] G.Tempesti, D.Mange & A.Stauffer,A Robust Multiplexer-Based Fpga Inspired By Biological Systems, Journal Of Systems Architecture : Special Issue On Dependable Parallel Computer Systems, 40(10) :719–733, Septembre 1997.

[Teu 01] C.Teuscher & E. Sanchez, Self-Organizing Topology Evolution Of Turing Neural Networks, Proceedings Of The International Conference On Artificial Neural Networks (Icann2001), Volume 2130 De Lecture Notes In Computer Science, Pages 820–826, Berlin, Heidelberg, 2001. Springer-Verlag.

[Tho 05] Y.Thoma, Tissu Numérique Cellulaire A Routage Et Configuration Dynamiques, Thèse Epfl, No 3226, 2005

[Uni 04] Taxonomie, Encyclopædia Universalis En Ligne, Mars 2004.

[Van 09] T.L.Van Zyl , E.M.Ehlers, A Need For Biologically Inspired Architectural Description: The Agent Ontogenesis Case, 10th Pacific Rim International Conference On Multi-Agents (Prima 2007), Bangkok (Thailan), 21-23 November, 2007, Lecture Notes In Computer Science, Springer, Vol. 5044, Agent Computing And Multi-Agent Systems, Pages 146-157, (2009).

[Wey07] D.Weyns, A.Omicini, J.Odell, Environment As A First Class Abstraction In

Multiagent Systems, Autonomous Agents And Multi-Agent Systems, Vol. 14, N°1, Springer, (2007) 5-30.

[Yao 99] X.Yao,Evolving Artificial Neural Networks, Proceedings Of The Ieee, 87(9) :1423–1447, Septembre 1999.

CHAPITRE 6 EXEMPLE DE CARACTERISATION DE SYSTEMES BIO-INSPIRES

1. Introduction

Une bonne démarche, permet de définir les méthodes qui guident les processus de développement et d'évolution d'un système au cours de son exécution. Comme nous l'avons vu dans le chapitre précédent notre approche permet de spécifier les systèmes bio-inspirés à la fois sur un aspect structurel et un aspect comportemental. Nous nous intéressons dans ce chapitre à deux aspects, dans un premier temps nous proposons une démonstration de notre démarche dans laquelle on modélise des systèmes bio-inspirés vue sur plusieurs dimensions, et dans un second temps, nous montrons l'intérêt de cette démarche pour la classification.

2. Caractérisation des approches bio-inspires

Nous allons donner dans cette section une caractérisation selon notre approche des familles de systèmes bio-inspirés les plus en vue, que nous groupant en quatre parties.
Dans la première nous caractérisons les approches évolutives. Dans la seconde on s'intéresse aux approches immunitaires. Dans la troisième, on caractérise les approches connexionnistes et dans la dernière nous nous intéressons aux systèmes hybrides.

2.1. Les Approches évolutives

Comme nous l'avons dit dans le chapitre trois il existe plusieurs types d'algorithmes évolutifs qui diffèrent généralement dans le choix de la représentation génétique et des opérateurs. Le choix optimal d'un algorithme, dépend des propriétés du problème considéré. En d'autres termes, il n'y a pas un algorithme meilleur que d'autres [Eib 03, Mic 96].

2.1.1. Les algorithmes génétiques

Ce type d'algorithmes utilise généralement la représentation binaire pour décrire les individus et met le point sur l'opération de croisement [Hol 75]. La programmation génétique fait parti aussi de la classe des algorithmes génétiques et elle utilise souvent la représentation en arbre pour représenté les individus surtout dans le domaine de la construction de circuits électroniques [Koz 92]. Une autre catégorie des algorithmes génétique est la programmation

évolutive qui utilise directement les paramètres qui définissent un phénotype et par une légère perturbation (i ;e. mutation)selon la distribution de Gauss fait évoluer la population d'individus. La programmation évolutive utilise souvent la sélection basée sur le tournoi et n'utilise pas de croisement [Fog 66].

Ce sont des approches purement phylogénétiques. Elles sont représentées par la formule suivante :

Iterate(C, assign (S, Select (SM, Reproduce (OM , S',P),P')))

Ou C: est la condition de convergence exemple en optimisation la satisfaction de la fonction objective
S : c'est le modèle des solutions qui satisfont notre problème
S' :c'est le modèle de toutes les solutions
SM : est le modèle de sélection ici on peut avoir le modèle de la sélection proportionnelle, la sélection par remplacement général, la sélection basée sur le tournoi, la Sélection basée sur les classes tronquées
OM : est le modèle de la reproduction et il peut être de deux sortes soit mutation soit croisement et chacune de ses opérations a son propre modèle. Ici c'est plutôt l'opération de croisement qui est utilisé.
P : contient les paramètres de la transformation, par exemple le type de représentation des individus qui est de type binaire pour les algorithmes génétiques et en arbre pour la programmation génétique
P' : contient les paramètres de la sélection ici c'est la taille des populations
En se qui concerne la granularité, c'est le génome qui est le grain le plus petit les modèles de reproduction sont hautement altérable. Pour les modèles de sélections, elles peuvent être manuellement altérables, tous les modèles jouent un rôle d'individu.

2.1.2. L'approche Island

Ce type d'algorithmes maintien un ensemble de populations qui évolue en parallèle et échange des individus à chaque génération. L'idée ici est de laisser l'évolution se poursuivre dans différentes parcours et dans différente populations en maintenant la diversité dans les populations et en exploitant les synergies potentielles par un changement d'individus entre les populations [Whi 98]. Le modèle Island est approprié pour une implémentation parallèle pour un problème linéairement séparable ou les meilleures solutions découvertes dans les différentes populations peuvent être recombinées. Les paramètres essentiels dans ce modèle sont le nombre de population et le nombre et la fréquence de migration des individus à travers les populations [Dar 08].

L'approche Island est un modèle aussi purement phylogénétique mais la différence avec les algorithmes génétique c'est la migration des individus à travers les populations. Ce type est représenté en deux étapes la première est la sélection des individus à migrer et la deuxième est la migration des éléments vers une autre population.

Iterat(C,GetEleme(Assign(S,Select(SM,Repreduce(RM,S',P,E),P'))) ||……||……||
GetEleme(Assign(Sn,Select(SMn,Repreduce(RMn,Sn',P,E),P'))))

Iterat(C,AddElem(In,Assign(S,Select(SM,Repreduce(RM,S',P,E),P'))) ||…..||……||
AddElem(I,Assign(Sn,Select(SMn,Repreduce(RMn,Sn',P,E),P'))))

Ou C: est la condition de convergence exemple en optimisation la satisfaction de la fonction objective
S : c'est le modèle des solutions qui satisfont notre problème
S' :c'est le modèle de toutes les solutions
SM : est le modèle de sélection ici on peut avoir le modèle de la sélection proportionnelle, la sélection par remplacement général, la sélection basée sur le tournoi et la Sélection basée sur les classes tronquées
OM : est le modèle de la reproduction et il peut être de deux sorte soit mutation soit croisement et chacune de ses opérations a son propre modèle ici c'est plutôt l'opération de croisement qui est utilisé.
P : contient les paramètres de la transformation, par exemple le type de représentation des individus qui est de type binaire pour les algorithmes génétiques et en arbre pour la programmation génétique

Chaque population peut être considérée comme un individu et on fait évoluer cet individu. Le rôle des modèles change, ils peuvent être des individus si on prend chaque ils séparément et vue comme un ensemble d'individus si on prend toutes les ils en considération. Le génome a le grain le plus petit et les modèles de reproduction sont hautement altérables. Pour les modèles de sélections, ils peuvent être manuellement altérables.
In, I : sont les éléments qui ont migré d'une population à une autre « I, In sont des sous ensembles de S, Sn respectivement »
L'unité architecturale GetEleme permet de retirer plusieurs éléments d'un ensemble d'éléments.

2.1.3. L'évolution d'états stable

Dans ce type d'algorithmes les individus qui obtiennent la mauvaise valeur de fitness sont remplacés par les descendants des individus qui ont obtenu les meilleures valeurs de fitness [Sys 89]. L'idée ici est de maintenir à l'intérieur de la population les meilleures solutions trouvées et d'ajouter toujours les meilleures.

Du point de vu processus, cette approche est une phylogenèse qui s'écrit :

Iterate(C, assign (S, Select (SM, Reproduce (OM , S',P),P')))

Ou C: est la condition de convergence, par exemple en optimisation la satisfaction de la fonction objective.
S : c'est le modèle des solutions qui satisfont notre problème.
S' :c'est le modèle de toutes les solutions.

SM : est le modèle de sélection ici on peut avoir le modèle de la sélection proportionnelle, la sélection par remplacement général, la sélection basée sur le tournoi et la sélection basée sur les classes tronquées
OM : est le modèle de la reproduction et il peut être de deux sorte soit mutation soit croisement et chacune de ses opérations a son propre modèle ici c'est plutôt l'opération de croisement qui est utilisé.
P : contient les paramètres de la transformation, par exemple, le type de représentation des individus qui est binaire pour les algorithmes génétiques et en arbre pour la programmation génétique
P' : contient les paramètres de la sélection comme la taille de la population.
En ce qui concerne la granularité, c'est le génome qui a le grain le plus petit les modèles de reproduction sont hautement altérables. Pour les modèles de sélections, ils peuvent être manuellement altérables et tous les modèles jouent un rôle d'individus.
Ce type d'algorithme n'est pas adaptable parce que si l'environnement change, les meilleurs individus de la population précédente ne sont pas forcement les meilleurs dans la population actuelle.

2.1.4. Le recuit simulé

C'est une fonction d'optimisation qui se base sur une perturbation aléatoire des candidats et une décision sur une certaine probabilité de retenir les solutions qui ont été engendrés après l'opération de mutation. Le recuit simulé prend son inspiration du processus de refroidissement progressif du métal. Dans la simulation le métal est le candidat « une des solutions » ou ces paramètres sont aléatoirement initialisés [Kir83]. Ce candidat va subir une mutation et si son énergie « équivalente à l'inverse de la valeur de fitness » est basse que celle de l'étape précédente le nouveau candidat remplace l'ancien. L'opération s'arrête quand la température du recuit est proche de zéro. La différence entre les algorithmes génétiques et le recuit simulé est que le premier opère sur une population d'individus et l'autre sur un seul individu.

L'évolution se fait par ajustement d'un ensemble de solutions ici le processus est légèrement différent des autres.

Iterate(C, assign (S, Select (SM, Adjust (OM , S',P),P')))

C: est la condition de convergence est la température.
S : c'est le modèle des solutions qui satisfont notre problème.
S' :c'est le modèle de toutes les solutions.
SM : est le modèle de sélection ici on utilise la méthode de Metropolis pour faire la sélection.
OM : est le modèle d'ajustement selon un coefficient de refroidissement
P' : contient les paramètres de la sélection ici c'est la taille des populations
La granularité concerne ici l'individu qui est généralement représenté en binaire, les modèles sont hautement altérables et totalement automatiques.

2.1.5. Les populations basées sur l'apprentissage incrémental

Ce type opère sur une seule chaîne de caractères génétiques qui représente la totalité des individus de la population. L'algorithme suppose que les génotypes de tous les individus ont la même taille et sont utilisés en codage binaire. Pi étant la probabilité de trouvée un 1 à la position i du génotype, généralement à l'initialisation de la population tous les Pi sont à 0,5. On sélectionne les individus qui ont la meilleure valeur de fitness pour la reproduction et on recalcule les nouveaux Pi. Une nouvelle population est créée par échantillonnage. On continue ainsi jusqu'à ce que l'on atteigne un critère d'arrêt prédéterminé [Bal 95].

C'est un ajustement d'une population

Iterate(C, assign (S, Select (SM, Adjust (OM , S',P),P')))

SM : est le modèle de sélection qui repose sur une loi de probabilité comme la méthode de Metropolis.
OM : est le mode d'ajustement on utilise ici une loi d'échantillonnage pour crée de nouvel individus comme la loi de Bernouli.
P : le type de représentation des individus est binaire et tous les génomes ont la même taille.
P' : ce sont les paramètres de la sélection on utilise ici une loi d'échantillonnage pour crée de nouvel individus la probabilité pi qui représente la probabilité de trouvée un chiffre un à la position i du génotype.
La différence entre les populations basées sur l'apprentissage incrémental et le recuit simulé réside dans les modèles d'ajustements et le critère d'arrêt.

Dans la section qui suit nous allons représenter le problème du voyageur de commerce et sa représentation sous forme de modèle et transformation de modèle.

2.1.6. cas du voyageur de commerce

Le problème du voyageur de commerce (TSM) consiste, étant donné un ensemble de villes séparées par des distances données, à trouver le plus court chemin qui relie toutes les villes. Plus concrètement ca consiste en la recherche d'un trajet minimal permettant à un voyageur de visiter n villes. En règle générale on cherche à minimiser le temps de parcours total ou la distance totale parcourue.

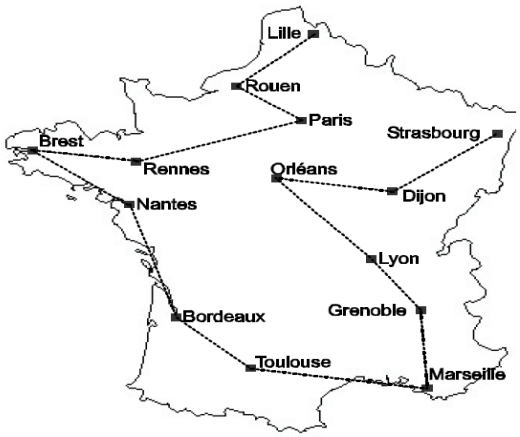


Figure 6.1. Représentation du problème du voyageur de commerce

On se donne n villes ainsi que les distances (longueur ou temps) entre chaque ville. On peut représenter le problème sous forme d'un graphe et d'une matrice représentant le graphe (les villes sont indicées de 0 à
n -1) :

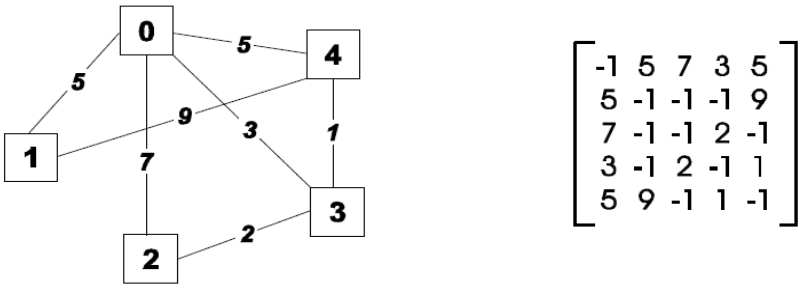


Figure 6.2. Représentation mathématique du voyageur de commerce

De façon générale, on peut considérer des graphes où certains arcs (chemins) n'existent pas. Dans la matrice représentant le graphe, les arcs inexistants ont une distance strictement négative (-1). Par ailleurs, on ne considère pas les arcs reliant une ville à elle même.
Le voyageur de commerce devant revenir dans la ville de départ, on peut représenter une "tournée" par un vecteur de dimension n, contenant la liste des villes visitées dans l'ordre :

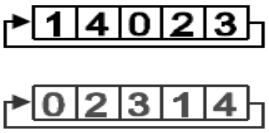


Figure 6.3. Représentation d'une tournée

Une tournée est définie à une permutation près. Une fois la ville de départ fixée, elle est par contre unique.
La représentation du problème de voyageur de commerce en utilisant un algorithme génétique s'appuient sur un principe de sélection des individus d'une population qui présentent des caractéristiques se rapprochant au mieux de ce que l'on recherche; cette population évoluant par ailleurs selon des critères d'évolution génétique à choisir. Dans le contexte du problème du voyageur de commerce, un individu est une tournée, un chemin et une population un ensemble de tournées. Il s'agit maintenant de définir un critère de sélection ainsi que des règles d'évolution de la population.

Sachant que l'on a à un instant donné k une population Pk constituée de p individus, comment générer une population Pk+1 à l'instant suivant toujours constituée de p individus mais qui présente de meilleurs caractéristiques. Il y a donc deux étapes : la génération d'une nouvelle population et la sélection des "meilleurs" individus.
On peut représenter la résolution du problème du voyageur de commerce en utilisant les algorithmes génétiques par le modèle dans la figure 6.4

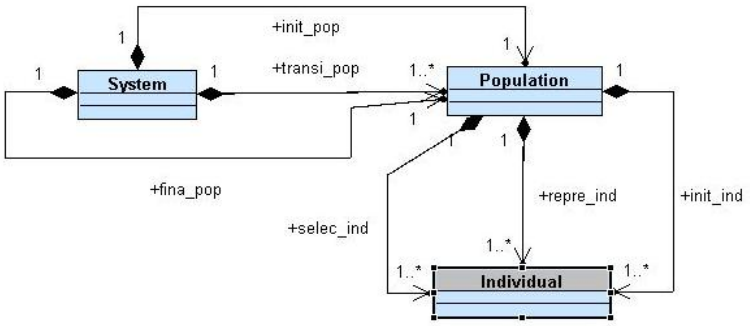


Figure 6.4. Représentation du modèle du voyageur de commerce (TSM)

Ce modèle est composé de trois classes :
La classe System représente le système évolutionnaire et contient un attribut nommé name dont le type est une chaîne de caractères pour indiquer le nom du système «dans notre cas l'attribut name indique voyageur de commerce ». La classe system spécifie que chaque système contient une population d'entrée, une population de sortie et peut contenir une ou plusieurs population intermédiaires. Chaque population est composée de plusieurs individus. Le système peut contenir à un instant k une population d'individus.
La classe population contient des informations sur la population en question
Id_population : cet attribut défini si c'est une population initiale ou finale ou intermédiaire
Seize : cet attribut défini la taille de la population ou on peut dire le nombre d'individus qui constitue une population
End_pop : cette méthode définie la population finale et par déduction, il n'y a plus d'évolution.
La classe individual contient des informations sur les individus de notre système
Id_Individual :cet attribut défini le vecteur qui représente une tournée
Eval_ind : cet attribut représente le graphe de distance concernant une tournée
F_function est la fonction d'adaptation qui prend comme paramètre l'attribut Eval_ind, Il s'agit de caractériser l'adaptation d'un individu au critère visé, on peut simplement utiliser la distance de la tournée comme fonction d'adaptation d'un individu, étant entendu que plus cette distance est petite meilleure est son adaptation.

La population initiale est générée aléatoirement par le système, la prochaine étape consiste à faire évoluer cette population en utilisant la mutation et le croisement après cette étape vient la sélection des individus pour la reproduction.
Pour la réalisation de ce scenario deux transformations sont appliqué sur le modèle de la figure 6.4
On applique la transformation de reproduction en utilisant la mutation et le croisement. Ici la transformation de reproduction va agir sur la classe individual du modèle selon la description fonctionnelle suivante :

$$Reproduc(M,R,P) \rightarrow M'$$

M : représente le modèle dans la figure 6.4

R : représente le paramètre de reproduction comme le montre la figure 8 il existe trois type de reproduction la mutation le croisement et la reproduction pour une nouvelle population
P : représente la méthode de reproduction si c'est une mutation On génère un nouvel individu à partir d'un individu I = (i0; i1; …in-1) en permutant aléatoirement deux éléments (k; l) : I = (i0; …; ik; …il; …in-1) →J = (i0; …; il; …ik; …in-1) le résultat de cette transformation est une novelle instanciation du modèle présenté dans la figure 6.4, ce dernier contiens de nouvel individus qui sont injecter dans la classe population.

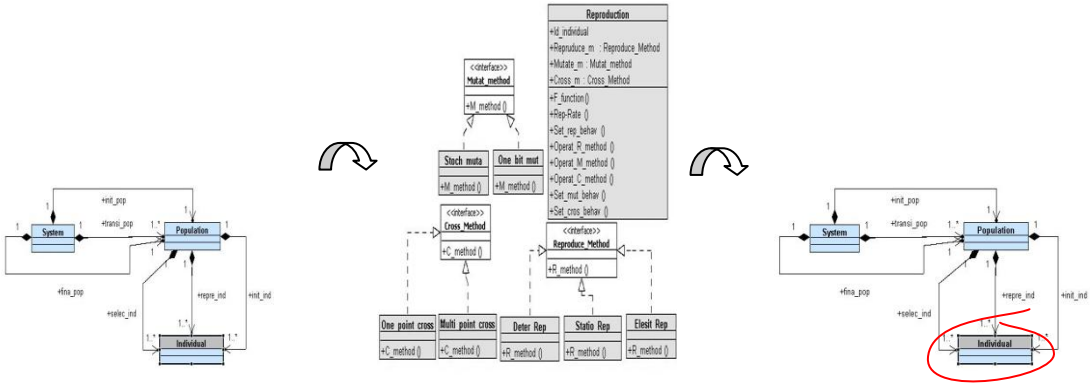


Figure 6.5. Représentation de la transformation de reproduction du modèle TSM

Le rôle : le rôle que joue le modèle de la figure 6.4 dans la transformation de reproduction est un rôle d'individu parce que la transformation agit sur la classe individu
Type de description : dans le cas du voyageur de commerce la description d'un individu est sous forme de vecteurs un vecteur représente une tournée figure 6.3.
Elément/ensemble : le modèle représentant le voyageur de commerce est un ensemble d'éléments qui sont les classes de ce modèle, il est donc constituer de trois éléments : system qui est un élément qui identifie notre système une seule instance est possible, le deuxième élément étant la population qui instancie une population d'entrée, une population de sortie et plusieurs populations intermédiaires, en dernier viens la classe individual qui constitue un individu
Granularité dans le cas de la transformation de reproduction la granularité est variante, si on considère la mutation et le croisement la granularité ici est un élément du vecteur qui constitue un individu mais si on prend la reproduction pour la sélection la granularité ici est tout l'individu, on a donc une granularité fine pour la mutation et le croisement et une granularité grosse pour la reproduction de sélection.

La prochaine étape consiste à sélectionner des individus dans le but de reproduire une nouvelle population. Le modèle de sortie de la transformation de reproduction devient un modèle d'entré pour la transformation de sélection selon la description fonctionnelle suivante :

$$Select(M',P) \rightarrow M''$$

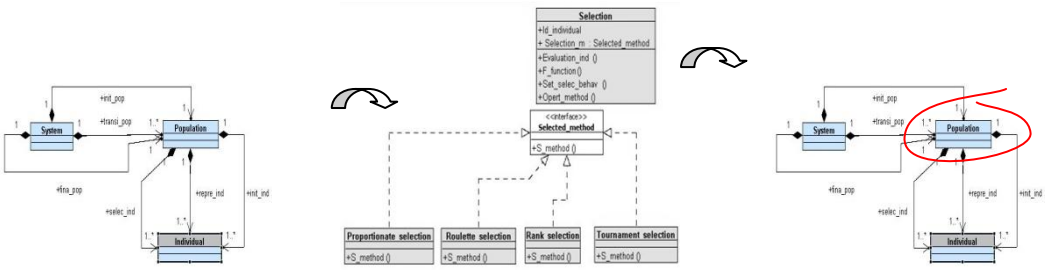


Figure 6.6. Représentation de la transformation de sélection du modèle TSM

M' : représente le modèle qui résulte après la transformation de reproduction
P : représente la méthode de sélection comme le montre la figure 6.6 il existe quatre méthode de sélection la roulette, tournois…
La transformation de sélection va agir sur la classe population du modèle de la figure 6.4
Le rôle : le rôle que joue le modèle de la figure 6.4 dans la transformation de sélection est un rôle d'espèce parce que la transformation agit sur la classe population
Type de description : dans le cas du voyageur de commerce la description d'une population est sous forme d'ensemble de vecteurs, un vecteur représente une tournée (figure 6.3).
Elément/ensemble : le modèle représentant le voyageur de commerce est le même que dans la transformation de reproduction.
Granularité dans le cas de la transformation de sélection la granularité est grosse, car ici c'est tout un individu qu'on va transférer.

L'étape qui suit la sélection et la reproduction des éléments sélectionnés en vu de générer une nouvelle population. Le modèle de sortie de la transformation de sélection devient un modèle d'entré pour la transformation de reproduction selon la description fonctionnelle suivante :

$$\text{Reproduc}(M'',R,P) \rightarrow MP$$

M'' : représente le modèle qui résulte de la transformation de sélection.
R : représente le paramètre de reproduction où il existe trois type de reproduction la mutation le croisement et la reproduction pour une nouvelle population
P : représente la méthode de reproduction pour le remplacement ici on utilise le remplacement stationnaire qui consiste à remplacer les pires individus parmi les parents

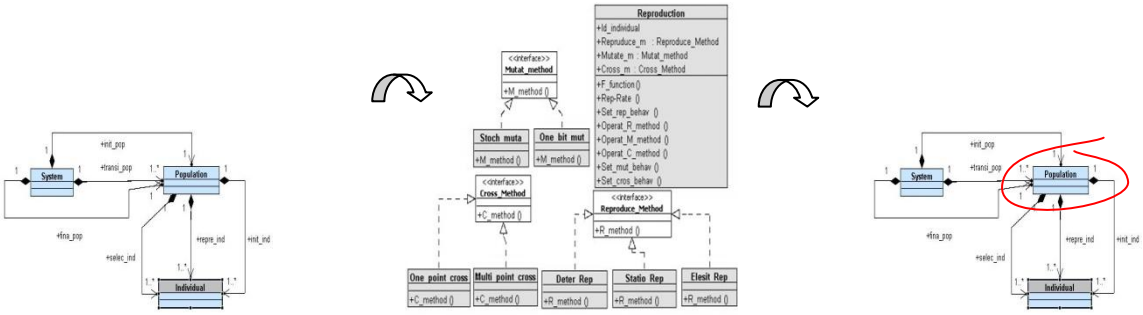


Figure 6.7. Représentation de la transformation de remplacement du modèle TSM

Le rôle : le rôle que joue le modèle de la figure 6.4 dans la transformation de reproduction pour le remplacement est un rôle d'espèce parce que la transformation agit sur la classe population
Type de description : dans le cas du voyageur de commerce la description d'une population est sous forme d'ensemble de vecteurs, un vecteur représente une tournée figure 6.3.
Elément/ensemble : le modèle représentant le voyageur de commerce est le même que dans la transformation de reproduction.
Granularité dans le cas de la transformation de remplacement pour la reproduction la granularité est grosse, car ici c'est tout un individu qu'on va transformer.

La dernière transformation est une transformation secondaire et qui a pour but de faire une itération de tous ces composants, la description fonctionnelle de cette transformation est la suivante :

Iterate(C,MS)→MS'

C : représente le critère d'arrêt de l'ensemble des transformations, si les individus de la population n-1 sont les mêmes de la population n il n ya plus d'évolution est cette population est considéré comme la population finale.
MS :c'est la composition des modèles et transformation vu précédemment.

Altérabilité : si la solution du voyageur de commerce est implémenter en software l'altérabilité est facile mais si elle est implémenter en hardware l'altérabilité devient difficile.
Composition : le système de voyageur de commerce dans sa globalité est composé du modèle à faire évoluer par le bais des transformations et de trois composant de transformation qui sont la reproduction, la sélection et la reproduction pour le remplacement. Les transformations vont être itérer et sur cet ordre reproduction sélection reproduction jusqu'à ce que il n ya plus d'évolution.

2.2. Les Approches immunitaires

Les systèmes immunitaires artificiels étant inspirés des systèmes immunitaires naturels. Ils apportent, d'un point de vue informatique, des propriétés telles que la reconnaissance, la discrimination, la mémorisation, l'apprentissage, l'auto-organisation, l'adaptation, la robustesse et l'évolutivité [Dar 08, Fre 09].
Les trois sections suivantes caractérisent respectivement la sélection négative, la sélection clonale et le réseau immunitaire.

2.2.1. La sélection négative

Dans les systèmes immunitaires artificiels, le processus de la sélection négative permet la génération de récepteurs capables de reconnaître le non-soi. Identiquement au thymus, seules les cellules ne présentant pas un antigène du non-soi et étant tolérantes envers celles du soi sont conservées.

[Tim 08,Dec 02] proposent un algorithme de la sélection négative basé sur la génération d'un ensemble de récepteurs et sur le calcul de l'affinité de chacun d'entre eux avec les récepteurs des éléments du soi.
Le processus de la sélection négative s'écrit :

Iterate(C, assign (M, Select (SM, Interpret (IM , M,P,E),P',E)))

C : représente la condition d'arrêt des transformations, autrement dit jusqu'à ce que tous les détecteurs soient comparés.
M : représente le modèle de détecteurs est c'est une population de cellules anticorps.
SM représente le modèle de sélection selon lequel vont être sélectionnés les bons détecteurs
IM : représente le modèle d'interprétation qui compare les affinités entre antigène du soi et anticorps.
E : le modèle de l'environnement et il représente les antigènes du soi et du non soi.
P' : paramètre de sélection, il correspond au seuil d'affinité.

Le processus de la sélection négative peut être vu comme une sorte de processus phylogénétique mais ici il n'y a pas d'évolution mais on parle plutôt de raffinement de la population.
Ici le modèle M représente une population d'individus qu'on appelle ici des cellules ou des anticorps et le modèle M peut être vue comme un ensemble d'éléments semblable qui sont des anticorps.
La même chose peut être dite sur le modèle de l'environnement E mais au lieu d'avoir des anticorps on a des antigènes.
Les cellules peuvent être représentées par des vecteurs de nombres binaires ou réels et la granularité ici est forte parce qu'aucune transformation n'est permise pour les cellules. C'est la population des anticorps qui change à chaque fois

2.2.2. La sélection clonale

En complément du rôle de la sélection négative, la sélection clonale est une approche utilisée pour expliquer comment une réponse immunitaire est activée lorsqu'une cellule B reconnaît la forme d'un antigène du non-soi.
La sélection clonale a pour objectif de produire une population de cellules B. Ces dernières sont composées, à leur surface, de récepteurs (anticorps). Seuls les anticorps obtenant une affinité supérieure à un seuil donné (mesure de la similarité entre deux récepteurs [Wat 04]) suffisant avec un antigène sont dupliqués. C'est la phase d'expansion clonale. Elle permet de produire une population de cellules suffisamment conséquente pour lutter contre l'antigène. En parallèle de cette reproduction, les cellules sélectionnées seront soumises à un processus d'hyper mutation somatique afin que les récepteurs obtiennent une meilleure affinité avec l'antigène.
Certaines des cellules ayant les plus grandes affinités avec l'antigène deviendront des cellules mémoire de telle sorte que si le système "rencontre" le même antigène, la création d'anticorps adaptés soit plus rapide.

Le processus de sélection clonale s'apparente à un processus phylogénétique parce que c'est une évolution d'une population d'anticorps en utilisant le clonage

Iterate(C, assign (M, Interpret (IM, Reproduce (OM , M,P),E)))

C : représente la condition d'arrêt et c'est quand tous les antigènes soient présentés au système.
M : le modèle représentant la population des anticorps et vers la fin nous aurons deux sortes d'anticorps des cellules mémoires et des cellules simples.
E : le modèle de l'environnement et il représente les antigènes.
IM : représente le modèle d'interprétation selon l'affinité entres antigènes et anticorps.
P' : paramètre de sélection ici c'est le seuil d'affinité.
OM : le modèle de reproduction. C'est un modèle composé qui peut être décomposé en deux modèles : le modèle de mutation et le modèle de clonage.
P : paramètre de reproduction et de mutation, seuls les meilleurs anticorps seront clonés.

La sélection clonale est un processus phylogénétique qui s'appuis sur la mutation et le clonage des individus ayant la meilleure affinité avec les antigènes qui représentent le problème à résoudre.
Ici le modèle M représente une population d'individus qu'on appelle cellules ou anticorps.
Le modèle M peut être vu comme un ensemble d'éléments semblables que sont les anticorps et vers la fin on a deux sortes d'anticorps ou de cellules : des cellules mémoires et des cellules simples
Le modèle de l'environnement E représente une population d'individus qu'on appelle ici des antigènes et ce sont des individus semblables.
Le modèle OM est un modèle composé qui peut être décomposé en deux modèles plus simples qui sont le modèle de mutation et le modèle de clonage.
Les cellules peuvent être représentées par des vecteurs de nombres binaires ou réels et la granularité ici est fine parce que la transformation la plus petite concerne une cellule et c'est un élément du vecteur qui la représente qui est le grain le plus fin.

2.2.3. réseaux immunitaires

La sélection clonale est le moyen utilisé par le système immunitaire pour vaincre des antigènes du non-soi, tandis que la sélection négative lui permet de se protéger contre des antigènes du soi. Le réseau immunitaire quant-à-lui définit la façon dont les cellules réagissent entre elles dans le système immunitaire. Cette théorie du réseau immunitaire, a été proposée par Jerne en 1974 [Fre 09].
La formation du réseau immunitaire est un processus complexe. Avec le processus de la sélection négative, les cellules générées permettent la reconnaissance du non-soi. Cependant, il peut arriver que les cellules se lient également à d'autres cellules pour former un réseau.
Cette formation du réseau est rendu possible par l'existence d'un second récepteur sur les anticorps (appelé idiotope) qui ne peut se lier qu'à un récepteur d'un anticorps. Ce réseau est régulé en permanence. Lorsqu'il n'y a pas d'antigène, le réseau est dans un état stable. En revanche, lorsqu'un antigène est présent, ce n'est plus une seule cellule qui réagit, mais l'ensemble du réseau qui se retrouve mis à contribution. Lorsqu'une cellule du réseau se lie à

un antigène, elle et ses voisines (notion des k nearest neighbor décrit dans [Wat 04]) deviennent actives et réagissent vis-à-vis de l'antigène.
La réaction des cellules est celle du processus de la sélection clonale. Ceci implique que les cellules ayant été dupliquées, sont à leurs tours intégrées au réseau immunitaire à condition qu'elles puissent se lier.

Le processus du réseau immunitaire s'apparente à un enchevêtrement de processus phylogénétique et ontogénétique, il s'écrit :

Iterate (C,(
Iterate(CC, assign (M, Select (SM, Reproduce (OM , M,P),P',E))),
,Assign(M, Develop(D, M, P))))

On peut le diviser en deux parties :

La partie création des cellules:

Iterate(CC, assign (M, Select (SM, Reproduce (OM , M,P),P',E)))

Cette partie est la sélection clonale.
CC : représente la condition d'arrêt et c'est quand tous les antigènes seront présentés au système.
M le modèle représentant la population des anticorps et vers la fin nous aurons deux sortes d'anticorps des cellules mémoires et des cellules simples.
E : le modèle de l'environnement et il représente les antigènes.
SM : représente le modèle de sélection selon l'affinité entres antigènes et anticorps.
P' : paramètre de sélection ici c'est le seuil d'affinité.
OM : modèle de reproduction ici c'est un modèle composé qui peut être décomposé en deux modèles, le modèle de mutation et le modèle de clonage
P : paramètre de reproduction et de mutation seule les meilleurs anticorps seront clonés.

Dans ce processus, le modèle M représente une population d'individus qu'on appelle ici des cellules ou des anticorps
Et le modèle M peut être vu comme un ensemble d'éléments semblables qui sont les anticorps et vers la fin on a deux sortes d'anticorps ou de cellules : des cellules mémoires et des cellules simples.
Le modèle de l'environnement E représente une population d'individus qu'on appelle ici des antigènes.
Le modèle OM est un modèle composé qui peut être décomposé en deux modèles plus simples qui sont le modèle de mutation et le modèle de clonage
Les cellules peuvent être représentées par des vecteurs de nombres binaires ou réels et la granularité ici est fine parce que la transformation la plus petite concerne une cellule et c'est un élément du vecteur qui la représente qui est le grain le plus fin.

Pour la partie liaison des cellules, elle s'écrit :

Assign(M, Develop(D, M, P))

D : représente le modèle de développement du réseau immunitaire ; il permet de définir comment les cellules doivent se lier pour former le réseau.
P : paramètre de transformation du réseau ; les cellules se lie à un seul voisin.
M : le modèle résultant de la sélection clonale.

Le processus du réseau immunitaire s'apparente à un enchevêtrement phylogénétique et ontogénétique qui provoque à chaque itération un changement dans la topologie du réseau (ajout, suppression de liaison, ajout, suppression de cellules).
Le modèle M représente une population d'individus qui se nomme anticorps et qui va se développer pour former un seul individu qui est le réseau immunitaire. Le modèle M joue alors le rôle de population dans la transformation Develop et le rôle d'individu dans la transformation Assign.
La granularité ici est moyenne parce que c'est la topologie du réseau qui change en modifiant à la fois les cellules et les liaisons. Les cellules et leurs liaisons peuvent être représentées par des vecteurs de nombres binaires ou réels
Le modèle de l'environnement E représente une population d'individus qu'on appel ici des antigènes.

2.2.4. Cas de CLONCLAS (CLONal CLASsification)

CLONCLAS (pour CLONal CLASsification) est l'adaptation de la sélection clonale pour la reconnaissance des formes. Le système est basé sur une approche de reconnaissance par prototypes et utilise la sélection clonale pour faire son apprentissage.
En entrée nous avons des images binaires de chiffres imprimés, un exemple est donné à la figure 6.8. Nous supposons donc qu'une étape de préparation de données à été effectuée, à l'issue de laquelle les chiffres ont été segmentés et mis sous forme binaire.

Figure 6.8.images binaires de chiffres

Pour chaque classe de chiffres (0-9) nous avons un certain nombre d'exemples d'entraînements. Le but de la phase d'apprentissage est de trouver pour chaque classe le modèle ou prototype, qui est l'image qui représente le mieux les exemples de la classe. Pour classer une forme inconnue, elle reçoit la classe du modèle qui lui ressemble le plus, en utilisant une certaine mesure de similarité.
Pour pouvoir utiliser les principes de la sélection clonale dans le système décrit, les auteurs de CLONCLAS ont défini les correspondances suivantes [Whi 03]:
Antigène : représente la classe pour laquelle nous voulons calculer le modèle, chaque antigène étant traité indépendamment des autres. Concrètement l'antigène $_i Ag$ est l'ensemble des exemples d'entraînement de la classe $_i Cla$. Chaque exemple est un vecteur de **len** bits, dont chaque bit correspond à un pixel de l'image du caractère (voir figure 6.9).

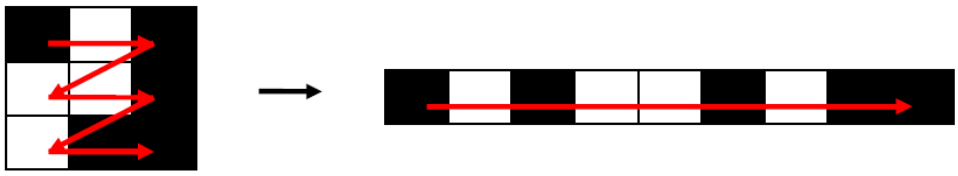


Figure 6.9. Image binaire → vecteur

Anticorps : représente une solution possible pour le problème courant. Si le système est confronté à l'antigène ${}_{i}Ag$, chaque anticorps ${}_{j}Ab$ représente un modèle possible pour la classe ${}_{i}Cla$. Un anticorps est un vecteur binaire de **len** bits, dont chaque bit est un pixel de l'image du modèle.

Affinité anticorps–antigène : l'affinité entre un anticorps ${}_{j}Ab$ et un antigène ${}_{i}Ag$ indique le degré de similarité entre le modèle représenté par l'anticorps ${}_{j}Ab$ et le meilleur modèle possible pour ${}_{i}Ag$. Plus l'affinité est grande et plus ${}_{j}Ab$ a de chances d'être le meilleur modèle de ${}_{i}Ag$.

Cellule mémoire : la cellule mémoire ${}_{i}Abm$ représente le meilleur modèle trouvé pour la classe ${}_{i}Cla$.

CLONCLAS va servir à optimiser l'affinité des modèles (cellules mémoire) de chaque classe. On peut représenter le système CLONCLAS par le modèle suivant :

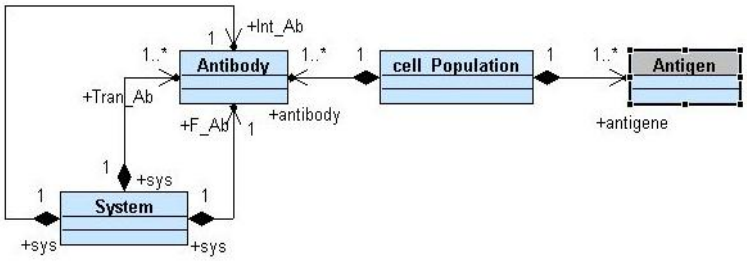


Figure 6.10. Représentation du modèle CLONCLAS

Ce modèle est composé de quatre classes :

La classe system qui définie notre système avec un attribut Id_sys qui contient le nom du système, dans notre cas CLONCLAS.

La classe Antibody, définie les anticorps, elle possède trois attributs, Id_ab qui identifie un anticorps et type, qui définie le type de l'anticorps, le dernier attribut étant Aff qui possède l'affinité entre un anticorps et un antigène.

La classe cell_population spécifie les populations de cellule deux type de population existe, la population antigen-anticorps et la population d'anticorps pour la mutation. Deux attribut définissent cette classe, Id_pop qui définie une population et type qui définie le type de la population.

La classe antigen définie un antigène elle possède deux attribut Id_ag qui définie un antigène et Id_class qui définie la classe d'un antigène.

On va tout d'abord générer aléatoirement une population initiale d'anticorps, **Ab**. Cette population est divisée entre : cellules mémoire **Abm** (1 par antigène), et population réservoir **Abr**. Créer un ensemble d'antigènes **Ag** à partir des exemples d'entraînement

La première phase consiste à calculer l'affinité entre les anticorps de la population et un antigène bien définie. Pour cela on va utiliser la transformation d'interprétation selon la description fonctionnelle suivante :

Interpret(MC,I,Ag)→MC

MC : la transformation interpret prend comme modèle d'entré le modèle représentant le système CLONCLAS représenté dans la figure 6.10.
I : représente la manière d'interprétation dans le cas de CLONCLAS on utilise la distance de hamming pour calculer l'affinité entre les individus.
Ag : représente l'instance d'un antigène pour qui on va calculer l'affinité

Comme le montre la figure 6.11 la transformation d'interprétation va agir sur les classes antigen et antibody

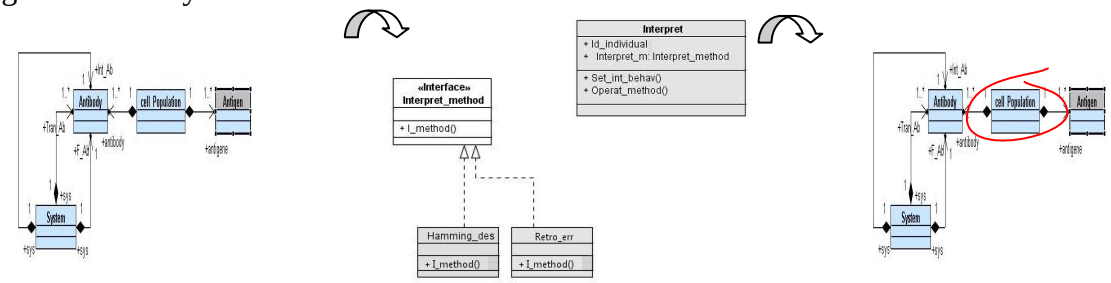


Figure 6.11. Représentation de la transformation d'interprétation de CLONCLAS

Le rôle : le calcule de l'affinité va nous dévoiler si un ensemble d'anticorps correspondent à un antigène. Le rôle que joue le modèle dans ce cas là est un rôle d'espèce parce que la transformation agit sur la classe cell_population.
Type de description : dans le cas de CLONCLAS pour la reconnaissance des formes, la description d'un antigène et d'un anticorps est sous forme d'un vecteur binaire de **len** bits, dont chaque bit est un pixel de l'image du modèle comme représenté dans la figure 6.9.
Elément/ensemble : le modèle représentant le système CLONCLAS est un ensemble d'éléments qui sont les classes de ce modèle, il est donc constituer de quatre éléments : system qui est un élément qui identifie notre système une seule instance est possible, le deuxième élément étant l'antigène, le troisième élimant étant la classe antibody qui instancie un ensemble d'anticorps. Puis vient la classe cell population qui constitue Une population d'anticorps et d'antigènes.
Granularité la transformation d'interprétation va définir de nouvelles informations sur les anticorps et les antigènes, ici la granularité est grande puisque il y a création d'information.
Composition : le modèle de CLONCLAS est composé d'une population de cellule qui peut être décomposé en deux catégorie : les antigènes et les anticorps. L'anticorps lui-même peut être décomposé en deux partie : cellules mémoire **Abm** (1 par antigène), et population réservoir **Abr**

La phase à suivre sera la sélection pour le clonage et la mutation. Pour l'opération de sélection on utilise la transformation de sélection selon la représentation fonctionnelle ci-dessous :

Select(MC, S)→ MC'

MC : ce modèle est le résultat de la transformation de la figure 6.11.
S: représente la manière de sélection dans le cas de CLONCLAS on utilise la sélection de **n** anticorps avec la meilleure affinité.

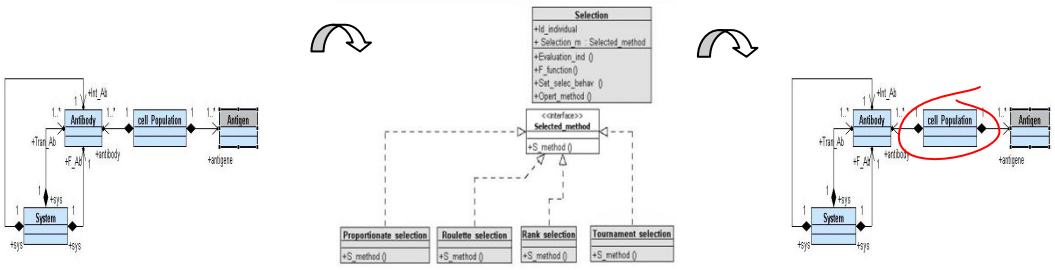


Figure 6.12. Représentation de la transformation de sélection de CLONCLAS

Le rôle : le rôle que joue le modèle de la figure 6.12 dans la transformation de sélection est un rôle d'espèce parce que la transformation agit sur la classe cell_population.
Granularité dans le cas de la transformation de sélection la granularité est grosse, car ici c'est tout un individu qu'on va sélectionner.

Apres la sélection vient la procédure de clonage et de mutation. Pour ca on utilise la transformation de reproduction. La description fonctionnelle de cette transformation est la suivante :

Reproduce(MC',R,P)→ MC''

MC' : définie le modèle représenté dans la figure 6.10 mais après avoir subi deux transformations l'interprétation et la sélection
R : représente le paramètre de reproduction comme le montre la figure 6.13 il existe trois type de reproduction la mutation le croisement et la reproduction pour une nouvelle population. On utilise pour le cas de cette transformation deux type de reproduction, la mutation et la reproduction pour une nouvelle population.
P : représente la méthode de mutation ici on utilise la mutation à 1 bit et la méthode de reproduction pour une nouvelle population.

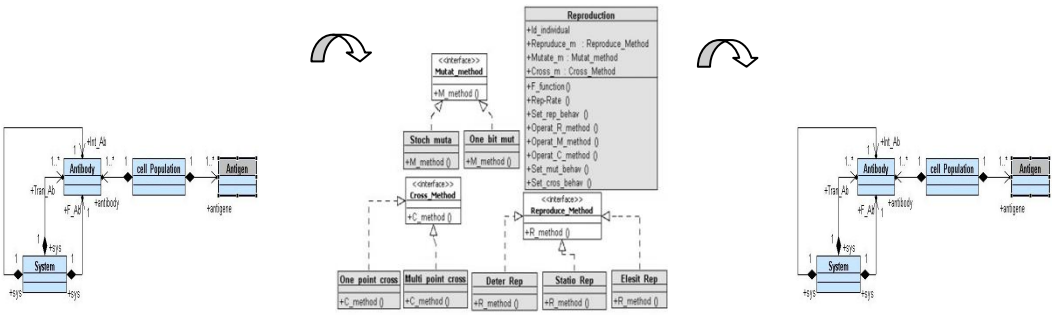


Figure 6.13. Représentation de la transformation reproduce de CLONCLAS.

Le rôle : le rôle que joue le modèle dans ce cas de figure est un rôle d'espèce parce que la transformation agit sur la classe cell_population.
Granularité la transformation de reproduction va définir de nouveaux individus qui sont les anticorps et une nouvelle population, ici la granularité est variable.

Après avoir cloner et muter les anticorps dans une nouvelle population. On va procéder au calcule de l'affinité afin de changer la population initial. Pour cette raison une transformation d'interprétation est nécessaire. La description fonctionnelle de cette transformation est la suivante :

$$Interpret(MC'',I,Ag) \rightarrow MC''$$

MC'' : la transformation interpret prend comme modèle d'entré le modèle représentant le système CLONCLAS représenté dans la figure 6.10.
I : représente la manière d'interprétation dans le cas de CLONCLAS on utilise la distance de hamming pour calculer l'affinité entre les individus.
Ag : représente l'instance d'un antigène pour qui on va calculer l'affinité

La dernière étape dans le système CLONCALS est le remplacement des anticorps par les nouvelles. La transformation de reproduction est utilisée dans cette étape. La description fonctionnelle de cette transformation est la suivante :

$$Reproduce(MC'',R,P) \rightarrow MC$$

MC'' : définie le modèle représenté dans la figure 6.10 après la transformation d'interprétation.
R : représente le paramètre de reproduction. On utilise pour le cas de cette transformation la reproduction pour une nouvelle population.
P : représente la méthode de reproduction pour une nouvelle population.
Toutes les transformations qu'on a vues plus haut représentent une identification pour un seul antigène dans le système CLONCLAS. Pour que tous les antigènes soient vérifiés, on a besoin d'une transformation d'itération. La description fonctionnelle de cette transformation est la suivante :

$$Iterate\ (C,M\ ) \rightarrow M'$$

M : représente le résultat de toutes les transformations vues plus haut.
C : un conteur qui est décrémenté à chaque fois que nous traitons un antigène. La transformation d'itération s'arrête l'ors qu'il n ya plus d'antigène dans la population de cellule.
Composition : le système CLONCLAS dans sa globalité est composé du modèle à faire évoluer par le biais des transformations et de trois composants de transformation qui sont la reproduction, la sélection et l'interprétation. Les transformations vont être itérer et sur cet

ordre interprétation sélection reproduction interprétation reproduction jusqu'à ce que il n ya plus d'antigène.

2.3. Les Approches Connexionnistes

Le principe de fonctionnement de ces approches peut être énoncé comme suit. Nous disposons initialement d'une base de connaissances constituée de couples de données entrées / sorties et nous souhaitons utiliser cette base de données pour "entraîner" un algorithme informatique à reproduire les associations constatées entre les entrées et les sorties de l'échantillon. Un bon exemple est donné par le diagnostic médical. Des médecins disposent d'une importante base de données de symptômes auxquels sont associés différents diagnostics qui ont été validés. Réaliser un réseau de neurones reproduisant le diagnostic peut être effectué en utilisant les couples symptômes diagnostics de cette base de données en tant qu'échantillon d'apprentissage [Jea 06]. Nous considérons dans ce qui suit trois approches.

2.3.1. Le perceptron

Dans le modèle de base, le perceptron est formé des éléments suivants :

- Des unités sensitives (S-units) sensibles à des stimulations physiques diverses (son, lumière …). Les unités sensitives simples renvoient +1 si le signal d'entrée dépasse un seuil θ, sinon elles renvoient 0.
- Des unités d'association (A-units) qui disposent de connexions entrantes et sortantes et génèrent un signal de sortie en fonction des entrées. Les unités d'association simples renvoient +1 si la somme algébrique des signaux d'entrée dépasse un certain seuil θ, sinon elles renvoient 0.
- Des unités de réponse (R-units) qui disposent de connexions entrante et génèrent un signal en dehors du réseau. Les unités de réponse simples renvoient +1 si la somme des signaux d'entrées est strictement positive, elles renvoient -1 si elle est négative. Si elle est égale à 0 la sortie est indéterminée.
- Une matrice d'interaction qui définit des coefficients de couplage entre différentes unités (poids synaptiques).

Dans le perceptron simple les différents types d'unités sont organisés en couches. Les seules connexions possibles vont de S vers A, puis de A vers R. les coefficients de couplage de S vers A sont fixes, ceux de A vers R peuvent varier dans le temps. Il n'y a qu'une seule unité de réponse et la vitesse de transmission est identique pour toutes les unités.

Le perceptron est susceptible d'apprentissage, c'est-à-dire qu'on peut le faire passer par une phase au cours de laquelle les poids synaptiques sont progressivement modifiés pour que la sortie du réseau corresponde à la sortie désirée. Plus généralement, l'apprentissage est défini comme :
Le mécanisme par lequel les paramètres libres d'un réseau de neurones sont adaptés à travers un processus de stimulation par l'environnement dans le quel le réseau est intégré. Le type d'apprentissage est déterminé par la façon dont les changements de paramètres sont mis en œuvre.

On a donc un système à travers lequel les stimulations externes induisent des transformations internes modifiant les réponses à l'environnement [Jea 06].

Notre perceptron ici s'apparente clairement à une épigenèse.

Iterate(SC, Assign(M, Develop(D, M, P)))

SC : la condition qui arrête l'opération de création
D : le modèle qui décrit le développement du réseau de neurone il peut être manuel, semi-automatique ou automatique (comme l'utilisation de L-systems)
P : paramètres de développement ; au maximum une seule couche de neurones.
Apres la création du réseau vient ensuite la partie apprentissage qui est donné par la formule :

Iterate(C,Assign(M, Adjust(AJ, Interpret(I,M,P',E'), P'', E')))

C : la condition où on juge que notre réseau a fini l'apprentissage
E' : le modèle qui contient les solutions qui doivent être fournis par le réseau (les résultats désirés)
M : le modèle qui défini le réseau de neurones résultant du processus de création (un vecteur)
AJ : le modèle selon lequel M va être corrigé pour s'adapter au problème que doit résoudre le réseau de neurones et il est donné par la formule 1.

$$w_{ij}(t+1) = w_{ij}(t) + \eta(d_j * s_j)s_i \qquad (1)$$

Où $w_{ij}(t)$ poids de la connexion entre les neurones i et j à l'instant t
η pas d'apprentissage, situé entre 0 et 1.
d_j sortie désirée pour l'entrée courante
s_j sortie obtenue pour l'entrée courante
s_i signal transmis par le neurone i.
I : le modèle qui permet de décrire si notre réseau diverge ou converge selon la formule 2.

$$\sum_{i=1}^{n} s_i \, w_i + s_0 w_0 \qquad (2)$$

Où s_0 w_0 sont respectivement le signal du biais et son poids

Ici tous les modèles sont représentés par des vecteurs de nombres réels
Le modèle M peut être vu comme un vecteur à deux dimensions contenant les poids de connexions entre les neurones, E' peut aussi être vu comme un vecteur qui contient les valeurs qui devront être fournies par le réseau. Pour ce qui est de la granularité, le grain est une case d'un vecteur où sont stockés les poids synaptiques. Tous les modèles ici sont simples parce que la représentation utilisée est de type vecteur. On peut voir le modèle M comme étant une espèce lors du processus de création. Une espèce qui contient des individus (les neurones), mais dans la partie apprentissage le modèle M est vu comme un individu qui est en train d'évoluer alors on peut conclure que le modèle M joue le rôle d'espèce dans la partie création et le rôle d'individu dans la partie apprentissage.

2.3.2. Perceptron multicouches

Pour un réseau de neurone de type perceptron multicouches, seuls les paramètres de développement changent. Le processus correspondant s'écrit :

Iterate(SC, Assign(M, Develop(D, M, P, E)))

SC : la condition qui arrête l'opération de création.
D : le modèle qui décrit le développement du réseau de neurones. Il peut être manuel, semi-automatique ou automatique (comme l'utilisation de L-systems).
P : paramètres de développement avec plusieurs couches de neurones.
E : stimuli de l'environnement qui représente la définition des poids synaptique
Après la création du réseau vient ensuite la partie apprentissage qui est donnée par la formule :

Iterate(C,Assign(M, Adjust(AJ, Interpret(I,M,P',E'), P'', E')))

L'apprentissage s'appuie sur la méthode de retro propagation de l'erreur.
C : la condition où on juge que notre réseau a fini l'apprentissage.
E' : le modèle qui contient les solutions qui doivent être fournies par le réseau (les résultats désirés).
M : le modèle qui définit le réseau de neurones résultant du processus de création (un vecteur).
AJ : le modèle selon lequel M va être corrigé pour s'adapter au problème que doit résoudre le réseau de neurones et il est donné par les formules 3 et 4. Une pour la couche de sortie et une pour la ou les couches cachées :

$$w_{ho}(t+1) = w_{ho}(t) + \Delta w_{ho} \text{ où } \Delta w_{ho} = \eta * \delta_o * s_h \quad (3)$$

$$w_{ih}(t+1) = w_{ih}(t) + \Delta w_{ih} \text{ où } \Delta w_{ih} = \eta * \delta_h * s_i \quad (4)$$

I : le modèle qui permet de décrire si notre réseau diverge ou converge selon la formule 5.

$$E = 1/2 \sum_{o=1}^{o} (d_0 - s_0)^2 \quad (5)$$

Si l'erreur respecte la condition C l'apprentissage est fini sinon on doit calculer le signal d'erreur sur la couche de sortie selon la formule 6

$$\delta_o = (d_o - s_o) * s_o * (1 - s_o) \quad (6)$$

P'': paramètre d'ajustement qui permettent de faire la correction selon la formule (7) qui permet de calculer le signal d'erreur sur la couche cachée.

$$\delta_h = s_h * (1 - s_h) * \sum_{o=1}^{o} \delta_o * w_{ho} \quad (7)$$

P’ : les paramètres d’interprétation qui permet le calcul du signal de chaque neurone caché. Pour cela on utilise la formule 8

$$p_i = \sum w_{ji} + s_j \qquad (8)$$

Ici tous les modèles sont représentés par des vecteurs de nombres réels
Le modèle M peut etre vue comme un vecteur à deux dimensions contenant les poids de connexions entre les neurones, E’ peut aussi être vue comme un vecteur qui contient les valeurs devront être fournies par le réseau. Pour ce qui est de la granularité, le grain est une case d’un vecteur où sont stockés les poids synaptiques. Tous les modèles ici sont simples parce que la représentation utilisée est de type vecteur. On peut voir le modèle M comme étant une espèce lors du processus de création. Une espèce qui contient des individus (les neurones), mais dans la partie apprentissage le modèle M est vu comme un individu qui est en train d’évoluer. On peut conclure que le modèle M joue le rôle d’espèce dans la partie création et le rôle d’individu dans la partie apprentissage.
Les modèles I et AJ sont des modèles composés qui peuvent être décomposés en modèles plus simples.

2.3.3. Cas de l’afficheur à sept segments (perceptron multicouches)

Les perceptrons multicouches sont couramment utilisés pour la reconnaissance de forme. Pour illustrer la mise en œuvre de celui-ci nous allons décrire un réseau apte à reconnaitre les chiffres sur un afficheur sept segments.
Les afficheurs sept segments sont utilisés dans les calculatrices, les ascenseurs ou les tableaux de commandes. Ils utilisent sept éléments d’affichage pour symboliser les dix chiffres figure 17.

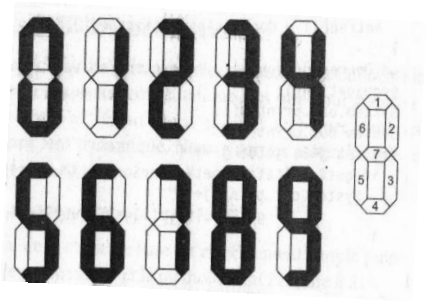


Figure 6.14. Un afficheur à sept segments

C’est là un problème simple pour un perceptron multicouche. On a besoin de sept neurones d’entrée, et de dix neurones de sortie figure 6.15.

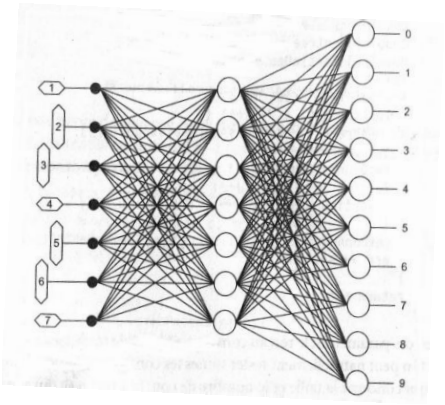


Figure 6.15. Réseau de reconnaissance de l'afficheur à sept segments

On peut représenter la résolution du problème d'un afficheur à sept segments en utilisant un perceptron multi couche par le modèle dans la figure 6.16

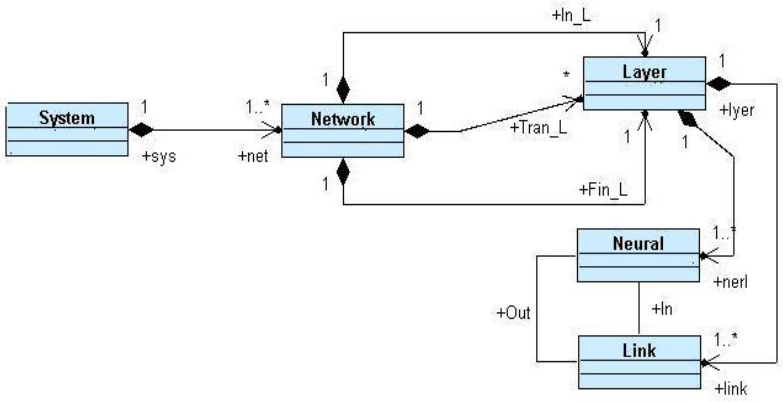


Figure 6.16. Représentation du modèle d'un afficheur à sept segments

Ce modèle est composé de cinq classes :
La classe System représente le système afficheur sept segments et contient un attribut nommé name dont le type est une chaîne de caractères pour indiquer le nom du système «dans notre cas l'attribut name indique afficheur sept segments ». La classe system décrit que chaque système contient un réseau qui décrit le problème, dans notre cas il décrit la solution d'un afficheur à sept segments. La classe Network contient des informations sur le réseau en question
La classe layer contient un attribut nomé Id_l qui est un identifiant pour chaque couche dans le réseau, cette classe comporte aussi un attribut nomé Nb_n qui détermine le nombre de neurone appartenant à une couche.
La classe neural contient des informations sur les neurones qui constitue le reseau, un attribut Id_neural qui identifie un neurone et une méthode activation qui spécifie si un neurone est actif ou non. La classe link encapsule des informations consternant les liaisons des différents neurones du réseau, un identifiant de liaison Id_link et un paramètre w spécifiant les poids de liaisons entre deux neurones.

Le réseau de neurone est construis en définissant le nombre des couches (couche d'entrée, couche de sortie et une ou plusieurs couches caché), à l'intérieur de chaque couche on définie le nombre de neurone et leur liaison avec d'autres couches du réseau.

Ce qui a été dit plus haut définie la première étape de la construction de notre réseau, cette étape est réalisée on utilisant la transformation de développement qui est appliqué au modèle de la figure 6.16

On applique la transformation de développement on utilisant une méthode de génération de couche de neurone et de liaison. Ici la transformation de développement va agir sur le modèle selon la description fonctionnelle suivante :

$$Devlop(M,D,P) \rightarrow M'$$

M : représente le modèle dans la figure 6.16
D : représente le paramètre de développement comme le montre la figure 6.15 il existe trois couches dans le système (une couche d'entrée, une couche caché et une couche de sortie), chaque couche est composée de neurones et de liaisons.
La couche d'entrée est composée de sept neurones et de quarante neuf liaisons entre les neurones de couche d'entrée et les neurones de la couche cachée.
La couche cachée est composé de sept neurones et de cent dix neuf liaisons, quarante neuf entre la couche cachée et la couche d'entrée et quatre vingt dix entre la couche caché et la couche de sortie.
Les neurones d'entrées définissent chaque segment et les neurones de sortie définissent les chiffres.

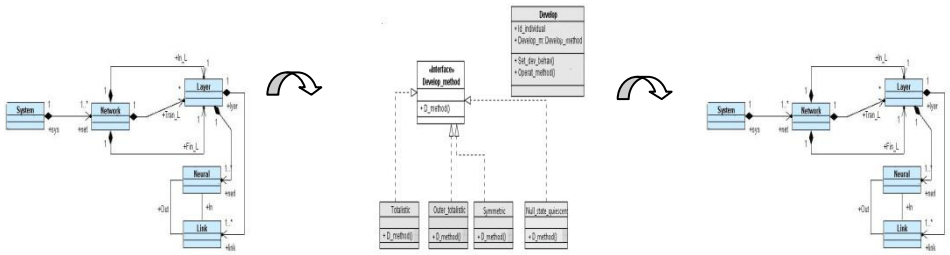


Figure 6.17. Représentation de la transformation de développement d'un afficheur à sept segments

Le rôle : le rôle que joue le modèle de la figure 6.17 dans la transformation de développement est un rôle d'espèce parce que la transformation agit à la fois sur la classe layer, neural et link. L'instanciation de ces classes génère des individus de type neurone et liaison et d'un point de vue plus abstrait, des individus de types couche.
Type de description : dans le cas du perceptron multicouche pour l'afficheur à sept segments la description d'une entrée est sous forme de tableau à deux dimentions. La première dimension représente la valeur à reconnaitre, la seconde contient les sept bits de l'afficheur dans l'ordre représenté dans la figure 6.15. Un bit à 0 signifie que le segment est éteint, un bit à 1 signifie qu'il est allumé.
Elément/ensemble : le modèle représentant l'afficheur à sept segments est un ensemble d'éléments qui sont les classes de ce modèle, il est donc constituer de cinq éléments : system qui est un élément qui identifie notre système une seule instance est possible, le deuxième élément étant le réseau, une seule instance est possible aussi, le troisième élément étant la classe couche qui instancie une couche d'entrée, une couche de sortie est une couche

intermédiaire. Puis vien les deux autres classe, neurone et lien qui constituent Les deux éléments les plus important dans le cas du perceptron.
Granularité dans le cas de la transformation de développement la granularité consiste à définir la topologie du réseau en créant des neurones et des lien la granularité est fine si on modifie la topologie du réseau par l'ajout ou la suppression d'une liaison, et la granularité est considéré comme moyenne si on modifie la topologie du réseau en ajoutant ou en supprimant un neurone et elle est considérée comme grosse si on rajoute ou on supprime une couche dans le but de modifier la topologie du réseau.
Composition : notre modèle de l'afficheur à sept segments est composée d'un réseau qui peut être décomposé en trois couches et ces dernières à leur tour peuvent être décomposées en neurones et liaisons. L'élément le plus simple dans un perceptron multicouche étant le neurone et la liaison.

Comme l'étape de développement et une étape itérative l'utilisation de la transformation d'itération est nécessaire afin d'achever l'étape de la représentation de l'afficheur à sept segments
La description fonctionnelle décrivant l'itération du développement du réseau est la suivante :

$$\text{Iterate}(C,M') \rightarrow MN'$$

C : représente le critère d'arrêt et la fin de la création du réseau, si on considéré qu'une transformation de développement définie les couche du réseau et une autre définie les nœuds et une dernière définie les liaisons. La condition serais un compteur initialisé à trois et décrémenté à chaque transformation.
M' : représente le modèle à faire itérer et il est représenté dans la figure 6.17, où M'= Devlop(M,D,P)

La prochaine phase consiste à préparer le réseau pour l'apprentissage. Le modèle va donc subir la transformation d'ajustement qui va agir sur les classe neurone et liaison du modèle de la figure 6.16. Ci-dessous l'expression fonctionnelle définissant la transformation d'ajustement pour le modèle d'afficheur de sept segments

$$\text{Ajust}(MN',Aj) \rightarrow MN''$$

MN' : représente l'évolution du modèle de la figure 6.16 après la transformation du développement
Aj : représente le paramètre de transformation que va subir le modèle MN' et qui est encapsulé dans la transformation ajust. Dans le cas d'un perceptron multicouche on utilise la méthode de propagation en avant.
La figure 6.18 montre la transformation du modèle MN' par le biais de la transformation ajust qui va agir sur les classes neural et link.

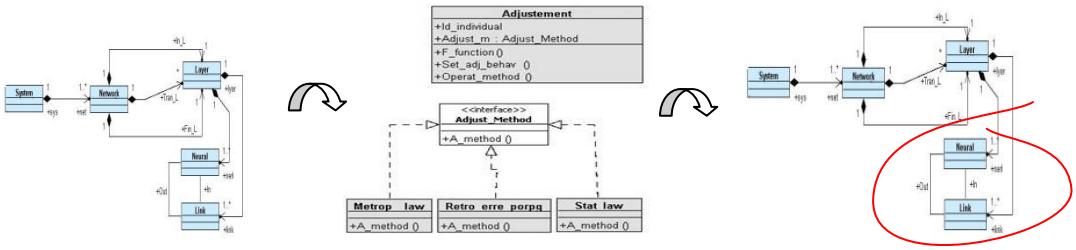


Figure 6.18. Représentation de la transformation d'ajustement d'un afficheur à sept segments

Le rôle : le rôle que joue le modèle de la figure 6.18 dans la transformation d'justement est un rôle d'espèce parce que la transformation agit sur les classes link et neural. La transformation va agir sur ces deux classes et modifier légèrement les paramètres de chaque neurone et chaque liaison.
Type de description : la description de cette transformation qui va agir sur le neurone (son état) et la liaison(son poids) et par conséquent la description ici et de type réel.
Granularité dans le cas de la transformation d'ajustement la granularité est fine, car ici on va perturber légèrement les poids de liaison.

La prochaine phase constitue la reconnaissance c'est-à-dire propagé chacun des dix vecteurs d'entrée et afficher les neurones dont le signal de sortie est supérieur à un seuil d'acceptation. Pour cela on utilise la transformation d'interprétation. Le modèle de sortie de la transformation d'ajustement devient un modèle d'entré pour la transformation d'interprétation selon la description fonctionnelle suivante :

$$\text{Interpret}(MN'',I) \rightarrow MN''$$

MN'' : représente le modèle qui résulte de la transformation d'ajustement.
I : représente la méthode d'interprétation ici en utilise la retro propagation de l'erreur

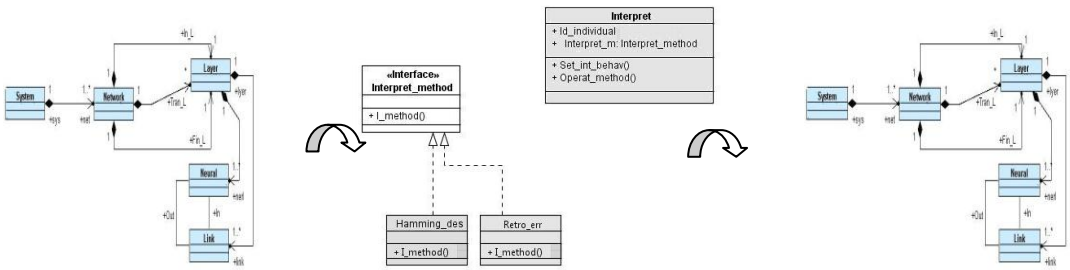


Figure 6.19. Représentation de la transformation d'interprétation d'un afficheur à sept segments

Le rôle : même role que dans la transformation d'ajustement.
Type de description : la description de l'erreur est sous forme d'un nombre réel.
Granularité la granularité ici est fine vue que c'est une légère perturbation de l'erreur calculé dans la phase d'apprentissage.
Composition : le système d'affichage à sept segments dans sa globalité est composé du modèle qui représente un réseau de neurone à faire développer par le biais des transformations et de trois composant de transformation qui sont le développement, l'ajustement et l'interprétation.

2.4. Les approches bio-inspirés hybrides

Le section suivante est dédié à quelque système bio-inspiré hybride

2.4.1. Les modèles objet adaptatifs

Un modèle objet adaptatif est un système qui représente les classes, les attributs, les relations et les comportements sous forme de métamodèles [Yod 02]. Les utilisateurs changent les métamodèles pour refléter les changements qui touchent le domaine considéré et ces changements altèrent le comportement du système directement. Les métamodèles sont stockés dans une base de données et le système n'est qu'un interpréteur de ces métamodèles. Dès qu'une métadonnée est changée, le système en tient immédiatement compte.
Cette approche s'apparente à une simple interprétation qui peut être décrite par l'expression fonctionnelle suivante :

Iterate (C, Interpret(MD, I, P, E))

MD est le métamodèle du système qui est interprété en présence de données I et une sémantique d'interprétation P. E étant le modèle de stimuli. La granularité du métamodèle est variable car l'utilisateur peut altérer n'importe quelle information (attribut, classe, méthode, ...) qui s'y trouve. Le modèle P est altérable, ce qui permet de changer le style de l'interprétation elle-même. La description des métamodèles est faite moyennant un ensemble de constructions (dites patterns) pour déclarer dynamiquement un type d'objet, une propriété ou même une stratégie qui consiste en un ensemble de règles [Mes 06].

2.4.2. L'approche Rainbow

C'est une approche qui a été proposée par D. Garlan et B. Schmerl de l'université Carnegie Mellon [Gar 02]. Elle traite de l'adaptation dynamique de l'architecture des systèmes logiciels à base de composants en faisant le parallèle de cette adaptation avec le processus de cicatrisation. Les auteurs proposent d'extérioriser l'adaptation en utilisant des modèles externes au système afin d'identifier et résoudre les problèmes qui peuvent l'affecter durant son exécution. La figure 1 illustre le principe de l'approche.

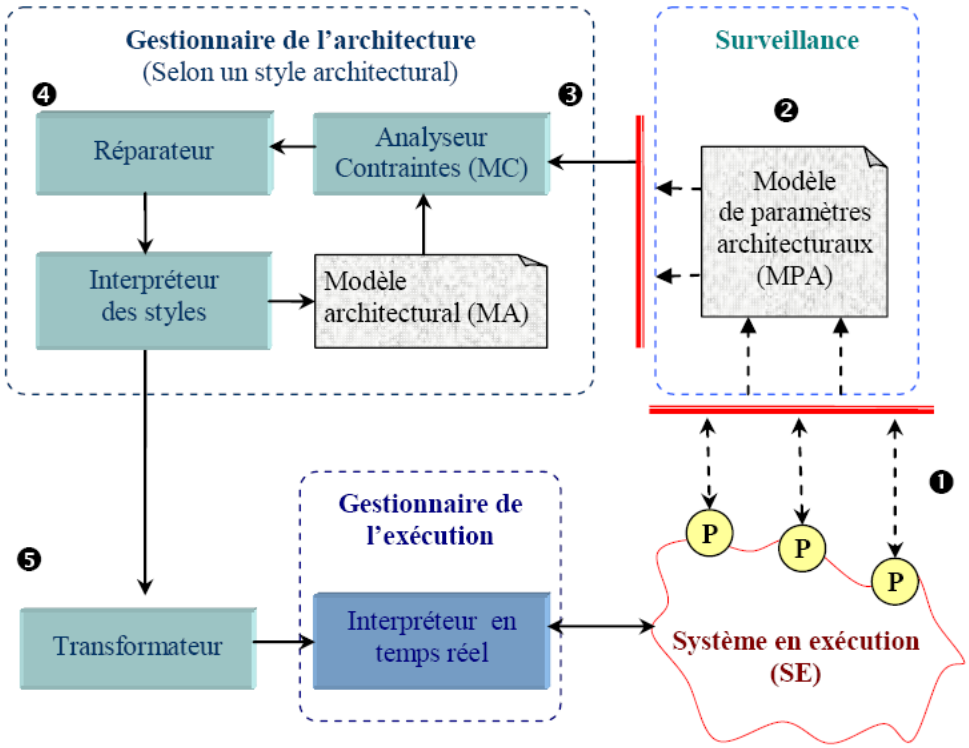


Figure 6.20. Principe de l'approche Rainbow [Mes 06]

L'approche prône le scénario suivant :

- Le système est observé durant son exécution. Cette dernière est génératrice de propriétés qui reflètent son état (indiquées par P dans la figure 1). Le temps de réponse d'un service est un exemple de propriété.
- Extraction, à partir des propriétés P, d'une forme plus abstraite (modèle MPA) en relation avec l'architecture (par exemple, le taux d'occupation d'un composant serveur).
- Déterminer si les nouvelles propriétés respectent les contraintes imposées sur le style, en utilisant le modèle de contraintes (MC).
- En cas de violation, effectuer les réparations
- Appliquer les nouveaux changements sur le système en exécution.

Bien que les auteurs utilisent le terme d'auto-cicatrisation (self-healing), qui devrait correspondre à une ontogénèse, l'approche s'apparente plutôt à une épigenèse dont l'expression fonctionnelle est :

Iterate(C, (
Iterate(SC, Assign(SE, Develop(MA, SE, P, E))),
Assign(MA, Adjust(Interpret(SE,IDM,P',E'), MA, P'', E''))
)
)

Le système SE est interprété avec des données IDM selon la sémantique donnée dans P' et en présence de stimuli de l'environnement E'. Cette interprétation produit un modèle de propriétés architecturales MPA qui est utilisé avec les contraintes architecturales pour ajuster le modèle architectural MA. Ce dernier est à nouveau utilisé pour changer le système SE dynamiquement (changer les connexions et activer/désactiver des composants). SC est une condition qui devient vraie une fois que l'opération d'altération du système SE terminée.
C est vraie tant que le système est opérationnel. MA et MC sont appelés style architectural, ils peuvent être altérés manuellement. La granularité de changement correspond aux composants et aux connecteurs. MA est décrit par un graphe avec des annotations [Mes 06].

2.4.3. L'approche Poetic

Le but du projet Poetic est le développement d'un substrat de calcul optimisé pour l'implémentation des systèmes digitaux qui supportent les processus Poetic [Poe 05].
Concrètement, les auteurs ont développé une cellule électronique en utilisant les circuits logiques programmables de type FPGA (Field Programmable Gate Array). Les cellules peuvent être connectées avec d'autres cellules pour former un tissu. Le tissu entre en interaction avec l'environnement à travers des capteurs spatialement distribués et des incitateurs, pour :

- Se développer et adapter ses fonctionnalités, par les processus phylogénétique, ontogénétique et épigénétique, à un environnement dynamique et partiellement imprédictible.
- Se réparer les parties endommagées par l'usure ou des facteurs environnementaux afin de survivre et maintenir ses fonctionnalités.

Les tissus Poetic sont structurés en trois couches: une couche génotype (GL), une couche phénotype (PL) et une couche qui fait le mapping entre les deux (ML) [Tem 02]. Le processus phylogénétique agit sur le matériel génétique de la couche GL (chaque cellule est dotée d'un génome). La couche ML se charge du mapping de la couche GL vers la couche PL, et supporte la différentiation de la cellule, sa croissance et son autoréparation. La couche PL supporte les fonctionnalités de chaque cellule et la connexion d'une multitude de cellules détermine les fonctionnalités du tissu (voir figure 6.21).

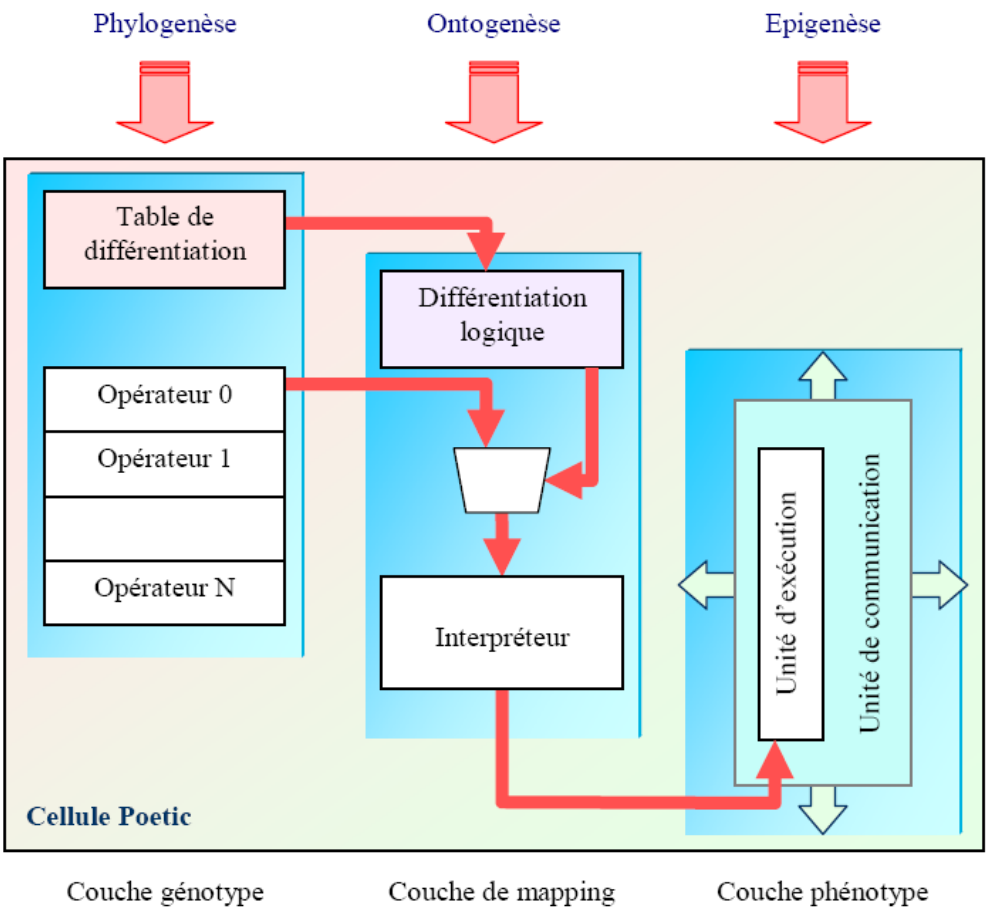


Figure 6.21. Les couches composant une cellule Poetic [Mes 06]

Le génome de chaque cellule (i.e. couche GL) contient un ensemble d'opérateurs et une table de différentiation utilisée pour déterminer le phénotype à développer pour la cellule.
Ce génome peut être impliqué dans une phylogenèse, mais cette dernière n'est pas supportée par la cellule.

Le processus ontogénétique est automatique et totalement supporté par la cellule. Il garantit l'autoréparation et la duplication de la couche PL en fonction de la couche GL. La granularité de la couche GL correspond aux chaînes de bits de la table de différentiation qui peuvent être altérées extérieurement en utilisant des microcontrôleurs. La couche PL a une description variable car elle peut être un neurone, un compteur, un additionneur, etc. Par conséquent, la granularité et l'altérabilité sont variables et dépendent de l'objectif du système construit par les cellules. Les stimuli de l'environnement vont de la détection des détériorations de la cellule à la capture d'évènements externes par des capteurs distribués.

On peut caractériser l'ontogenèse dans l'approche Poetic par l'expression fonctionnelle suivante :

Iterate (C, Assign(PL, Develop(GL,PL, P, E)))

Où PL et GL sont les couches phénotype et génotype, P les paramètres du développement (différentiation logique et l'interpréteur) et E les stimuli externes (senseurs et détecteurs de

détérioration). PL contient, après développement, une unité d'exécution et une unité de communication qui couvrent les fonctionnalités de la cellule. GL contient la table de différentiation et les opérateurs de base.
C est une condition qui indique si la cellule est opérationnelle ou pas. En cas de détérioration la condition C devient fausse.

2.4.4. Mage

Mage est une nouvelle vision de l'évolution qui considère les changements qu'un système logiciel subit, dès les premiers moments de son existence, comme sa dimension ontogénétique. Mage est un paradigme bio-inspiré, partant des mécanismes de la génétique, il modélise l'ontogenèse d'un système logiciel sous la forme d'un génome embarqué, dont le rôle consiste à façonner continuellement le système en fonction des changements anticipés et non anticipés [Mes 06]. Mage est la première mise en œuvre de l'ontogenèse dans le génie logiciel.
Un modèle Mage est composé de deux parties un génome et un phénotype. Le phénotype représente l'équivalent d'un système logiciel classique. Le génome est une collection de gènes qui façonnent continuellement le phénotype [Mes 06].

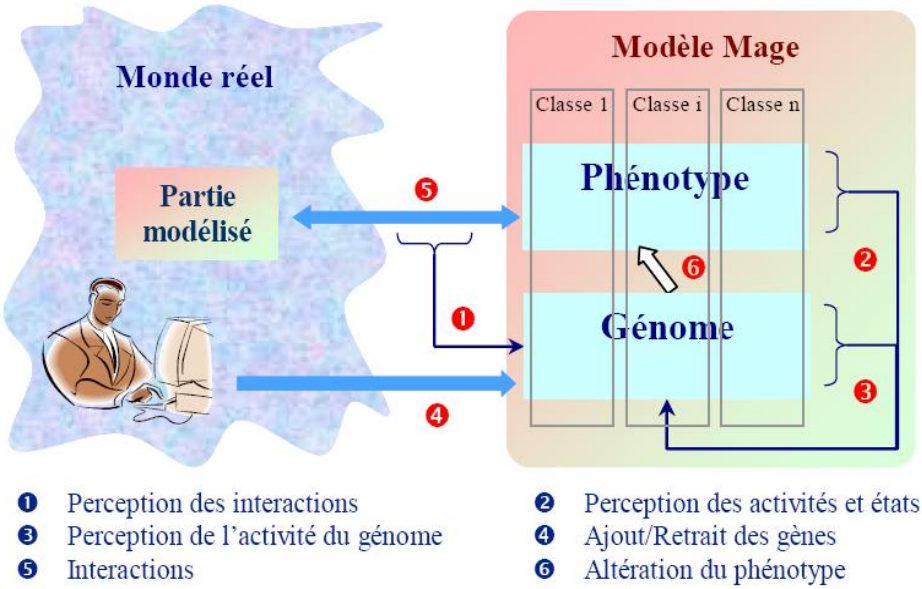


Figure 6.22. Représentation de Mage

Un système Mage peut être caractérisé par deux expressions fonctionnelles. La première est valable avant la fin de l'embryogenèse et la deuxième est valable pendant la phase d'évolution continue, une fois le phénotype devenu actif.
Durant la phase de l'embryogenèse l'expression fonctionnelle est :

Iterate (C, Assign (Ph, develop (G, Ph, P, E)))

L'expression signifie que le système (G et Ph) subit un développement itératif selon la description donnée dans G (le génome) pour produire Ph (le phénotype qui est initialement vide) jusqu'à ce que la condition C devienne vraie. C correspond à la fin de la construction de

la méthode main() du système et le début de l'exécution de celle-ci. P est un modèle qui représente les paramètres de la transformation, il exprime la sémantique associée à l'exécution du génome. Par exemple, un gène exécuté devient passif, un gène actif vérifie continuellement sa condition de déclenchement, etc. Le modèle E renferme les stimuli de l'environnement, à ce niveau de développement, les seuls stimuli sont ceux générés par le génome lui même. Par exemple, l'existence d'un composant, l'activation/désactivation d'un gène, etc. Le stimulus qui mettra fin à la phase de l'embryogenèse correspond à l'exécution du gène qui active la fonction main() (i.e. exécute un Run sur le composant représentant la fonction main()).
Durant la phase d'évolution continue, l'expression fonctionnelle est :

```
Iterate (C', ( Assign(Ph, develop(G, Ph, P, E)) ||
Interpret(Ph,IDM,P',E')
)
)
```

L'expression signifie que le système développé selon le génome G (i.e. develop(G, Ph, P, E)) est interprété parallèlement selon le modèle de données d'entrées IDM (e.g. échange de valeurs avec l'extérieur). Cette interprétation met à jour le phénotype (i.e. les valeurs des composants) ainsi que les modèles de stimuli E et E'. L'interprétation du phénotype se fait selon le modèle P' qui spécifie la sémantique d'exécution des composants universels et du phénotype. La condition C' correspond à la fin de fonction main(), c'est à dire l'arrêt de toute activité du phénotype.
Rôle. Un modèle Mage joue le rôle d'un individu.
Type de description. Il y a plusieurs descriptions dans Mage. Le génome est décrit sous forme de chromosomes qui sont, à leur tour, des collections de quadruplets : (état, condition de déclenchement, type, informations complémentaires). Le phénotype est décrit uniformément moyennant le composant universel. Les modèles d'interprétation du génome et du phénotype sont complexes et représentés par la sémantique associée à chaque type de gènes et chaque type de composants ainsi que leurs combinaisons.
Elément/ensemble. Un système Mage est un ensemble de classes qui partionnent le génome et le phénotype. Cependant, cette caractéristique n'est pas utilisée à des fins de phylogenèse car chaque classe assure une fonctionnalité différente.
Granularité. La granularité du phénotype est variable car il peut représenter des données, des instructions mais aussi des graphes, des classes, etc. La granularité du génome est le gène, il est composé de chaînes symboliques (i.e. condition de déclenchement, type, ...).
Altérabilité. Le phénotype est altérable dans sa totalité et à tout moment. Le génome dans Mage est prévu pour subir une évolution par phylogenèse. Il est hautement altérable, mais dans l'état actuel de notre travail, il est changé manuellement.
Composition. Un système Mage est une juxtaposition de deux transformations qui ont lieu au même moment. La première est l'ontogénèse, la seconde est l'interprétation qui s'apparente partiellement à une épigenèse [Mes 06].

3. Discussion

Apres avoir analysé de prés les quatre catégories des systèmes bio-inspirés (systèmes évolutionnaires, système immunitaire, système connexionniste systèmes hybrides) on peut déduire des ressemblances. Alors qu'un système à base de système immunitaire artificielle (CLONCLAS) permet à des anticorps d'être capable de reconnaitre des antigènes à travers les transformations de sélection reproduction et interprétation, les systèmes à base de réseau de neurones quant à eux (afficheur à sept segments) permettent de reconnaitre des objets en entrés en utilisant les transformations d'ajustement et d'interprétation. Les systèmes suivent une démarche itérative basé sur la transformation, d'un point de vue abstrait toutes les transformations se valle (elles appartiennent au même metameta modèle), on peut voire ces systèmes comme une suite de transformation.

Pour ce qui est de la sélection clonale (CLONCLAS), c'est un processus phylogénétique particulier, la différence avec les approches évolutives réside dans les cellules mémoires. Dans les systèmes évolutionnaires tous les individus se valent mais dans la sélection clonale nous avons plusieurs types d'individus, les anticorps et les antigènes, aussi il ya deux catégorie de cellules (anticorps) : cellules simples et cellules mémoires. C'est ce qui donne à l'approche de sélection clonale l'aspect d'apprentissage et par conséquent l'aspect épigénétique.

D'un point de vue comportemental, les réseaux de neurones (afficheur à sept segments) se divisent en deux parties la partie création qui s'apparente toujours à un processus ontogénétique et la partie apprentissage qui peut être soit un processus épigénétique soit un processus phylogénétique. Le processus de création dans les réseaux de neurones suit l'axe ontogénétique c-à-d qu'on peut le comparer à un processus de développement cellulaire. La deuxième phase dans un système connexionniste étant l'apprentissage elle est définie par deux transformations, l'ajustement et l'interprétation.

De ce qui a été dit plus haut ont peut dire qu'il ya des similitudes mais il ya une certaine différence sur quelque points.

- Les systèmes procédant avec un processus d'interprétation ont une grande interaction avec l'environnement.
- Les systèmes procédant avec un processus d'ajustement on un niveau d'abstraction plus élevé que les systèmes procédant avec des processus de sélection et de reproduction.
- Le type et le rôle de la population dans les processus constituant le système sont différents.

Cette démarche nous a permis comme nous l'avons mentionné dans la section 3 du chapitre précédant d'aidé à la création de nouveau système bio-inspirés par le biais d'un nouveau outil qui se base sur les patterns de transformation cette outils qui est en cours de développement nous permettra la création et la simulation de modèles basé sur les systèmes bio-inspirés.

Nous donnerons dans la section qui suis un descriptif d'un simulateur universel que nous alimentons avec des modèles (structure, comportement, environnement) et qui s'exécute et produit des modèles résultat.

4. Description de notre simulateur

Notre simulateur est basé sur l'idée principale que les modèles sont des entités de première classe". Par conséquent, et puisque tout est modèle, les transformations sont aussi considérées comme des modèles.
Notre simulateur doit nous permettre de maitre en point des modèles bio-inspirés ie définir la structure de chaque système à travers des outils et produire un modèle de ce dernier, il doit aussi nous permettre de définir un certain nombres d'environnements où les systèmes peuvent opérer , l'autre fonctionnalité du simulateur est de permettre de voir le comportement et l'évolution de chaque système bio-inspirés crée à travers une série de transformations

Dans cette partie nous donnerons des diagrammes qui décrivent notre simulateur.

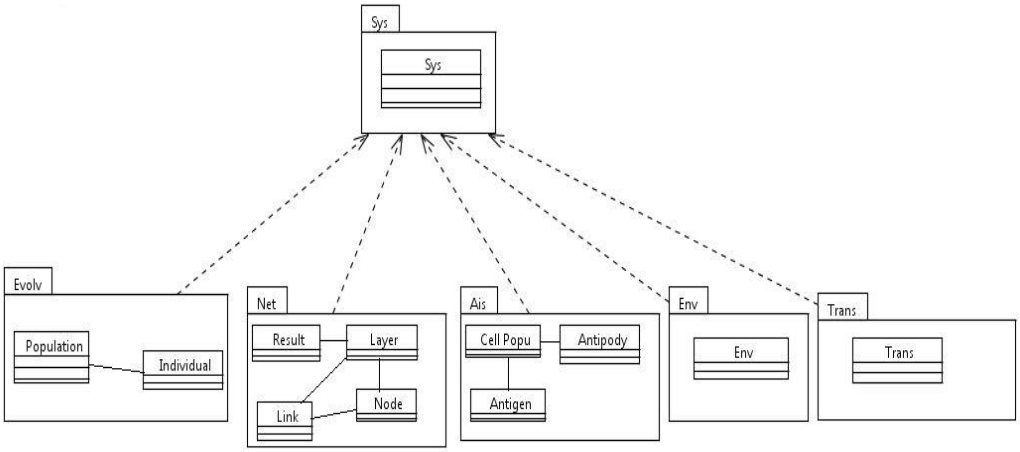


Figure6.23. Description du meta modele notre simulateur

Notre simulateur est crée pour s'inscrire dans une approche MDA. Sa syntaxe abstraite a été décrite comme un meta-modèle.

La figure 6.23 représente le meta modèle de notre simulateur qui est constitué de cinq packages :
Package Sys : décrit le type de notre système que se soit un système connexionniste ou évolutionnaire ou autres, le système peut aussi être une transformation parce que elle est considérer comme un modèle.
Package Evolv : ce package encapsule les systèmes modélisés par une approche évolutionnaire, il contient deux classes : La classe Individual contient des informations sur les individus de notre système, La classe population contient des informations sur la population en question.
Package Net : ce package encapsule les systemes modélisés par une approche connexionniste, il contiens quatre classes : La classe Result contient des informations sur le réseau en question, La classe node contient des informations sur les neurones qui constitue le réseau, La classe layer contient des informations sur le nombre de couche dans le réseau et

comment il sont constitué, La classe link encapsule des informations consternant les liaisons des différents neurones du réseau.

Package Ais : ce package encapsule les systemes modélisés par une approche immunitaire artificielle il contient trois classes La classe Antibody, définie les anticorps, La classe cell_population spécifie les populations de cellule deux type de population existe, la population antigen-anticorps et la population d'anticorps pour la mutation, La classe Antigen définie un antigène.

Package Env : dans ce package on peut définir des modèles d'environnements pour différentes situations de nos systèmes.

Package Trans : ce package s'occupe de la création des transformations nécessaire à l'évolution de nos systèmes.

Nous détaillerons dans ce qui suit le package Tarns

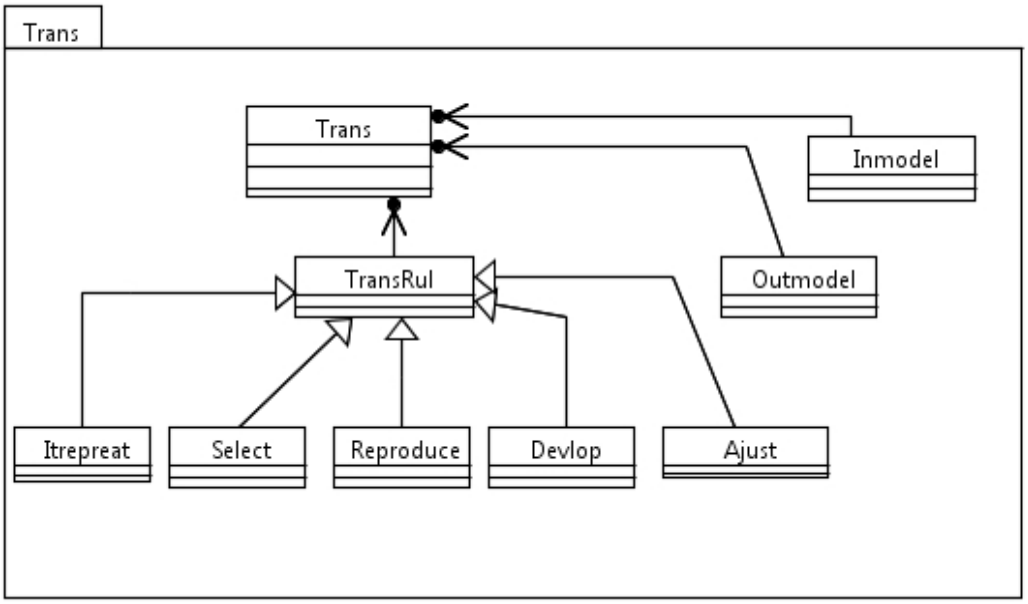


Figure 6.24. Description du Package Trans

Une transformation doit impérativement avoir un modèle en entré et obtenir un modèle résultat.

Dans le cas de notre simulateur le modèle en entré et le modèle en sortie doivent appartenir au même meta modèle. Il existe plusieurs types de règles de transformation selon la manière dont elles sont appelées et le type de résultat qu'elles retournent. Le résultat d'une règle est un ensemble d'éléments de modèles prédéfini (Outmodel).

Comme nous l'avons vu plus haut il existe deux types de modèles utilisés par les règles de transformation. Les Inmodel, d'une part, utilisés pour la détection d'un ensemble d'éléments particulier dans le modèle source et permettant le déclenchement d'une transformation. Il s'agit en fait, d'un ensemble de types issu du meta-modèle cible. Les Outmodel, d'autre part, sont définis comme des ensembles de types issus du meta-modèle cible. Ils représentent les éléments de modèles résultant de l'application d'une règle de transformation. Afin d'obtenir la correspondance entre éléments du modèle source et Inmodel d'une règle, on définit des filtres.

Ces filtres sont décrits à l'aide d'expressions booléennes OCL et permettent de guider le choix des éléments à mettre en correspondance avec le modèle en entrée de la règle.
Les classes héritant de la classe TransRule on été définit en utilisant le pattern stratégique pour encapsulé les différents algorithmes permettant la transformation de touts système bio-inspirés.

5. Conclusion

Dans ce chapitre, nous avons donné un aperçu sur la démarche de modélisation et d'implémentation d'un système bio-inspirés et nous avons montré comment celui-ci évolue à travers un enchainement de modèles et de transformations. A travers cette démarche nous avons pu mettre en évidence les points de différences et les points de similitude.
Nous avons terminé ce chapitre par une représentation brève d'une conception d'un simulateur universel de systèmes bio-inspirés, le travail est en cour afin de maitre au point un prototype.

6. Références

[Bal95] S.Baluja, R.Caruana, Removing The Genetics From The Standard Genetic Algorithm, Proceedings Of The Twelfth International Conference On Machine Learning, Pages 38–46. Morgan Kaufmann, San Mateo, Ca, 1995.

[Dar 08] F.Dario,C.Mattiussi, Bio-Inspired Artificial Intelligence Theories, Methods, And Technologies, The Mit Press Cambridge, Massachusetts London, England, 2008

[Dec 02] L.DeCastro, , J.Timmis, Artificial Immune Systems : A Novel Paradigm To Pattern Recognition, University Of Paisley, 67–84, 2002.

[Dor 99] M.Dorigo, G.Dicaro, Ant Colony Optimization Meta-Heuristic, In D.Corne, M.Dorigo, & F.Glover, Editors, New Ideas In Optimization, Pages 11-32, Mcgraw-Hill, 1999.

[Eib03] A.Eiben, J.Smith, Introduction To Evolutionary Computing. Springer-Verlag, Berlin, 2003.

[Fog66] L.Fogel, A.Owens, Artificial Intelligence Through Simulated Evolution. Wiley, New York, 1966.

[Fre 09] A.Frederic, Un Système Tutoriel Intelligent A Base De Système Immunitaire Artificiel, Ecole Nationale D'ingénieurs De Brest Rapport De Stage De Master Recherche En Informatique 2009

[Gar 02] D.Garlan And B.Schmerl, Model-Based Adaptation For Self-Healing Systems, Woss'02, Charleston, Usa, Nov. 2002..

[Hol75] J.Holland, Adaptation In Natural And Artificial Systems, University Of Michigan Press, Ann Arbor, 1975.

[Jea 06] R.Jean, Les Reseaux Neuronaux Chapitre 7 Livre Edition Vuibert 2006.

[Kir83] S.Kirkpatrick, C.Gelatt, Optimization By Simulated Annealing, Journal of Science, 220:671–680, 1983.

[Koz92] J.Koza, Genetic Programming: On The Programming Of Computers By Means Of Natural Selection, Mit Press, Cambridge, Ma, 1992.

[Mes 06] D.Meslati, Mage : Une Approche Ontogénétique De L'évolution Dans Les Systèmes Logiciels Critiques Et Embarqués. Thèse De Doctorat D'état, Université De Annaba, Février 2006.

[Mic96] Z.Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, 3rd Edition, Springer-Verlag, Berlin, 1996.

[Poe 05] Http://Www.Poetictissue.Org.

[Sys89] G.Syswerda, Uniform Crossover In Genetic Algorithms. In Proceedings Of The Third International Conference On Genetic Algorithms, San Mateo, Ca, 1989.

[Tem 02] G.Tempesti, A Poetic Architecture For Bio-Inspired Hardware, 8th Int. Conf. On The Simulation And Synthesis Of Living Systems, Sydney, Australia, Dec. 2002, Mit Press, Cambridge, 2002.

[Tim 08] J.Timmis, A.Hone, Theoretical Advances In Artificial Immune Systems. Volume 403, 11–32 2008.

[Wat 04] A.Watkins, J.Timmis, Artificial Immune Recognition System (Airs) :An Immune-Inspired Supervised Machine Learning Algorithm, Journal of Genetic Programming And Evolvable Machines Volume 5, 291–317, 2004.

[Whi 03] J.A.White, S.M.Garrett, Improved pattern recognition with artificial clonal Selection, In Proceedings of ICARIS, 2003.

[Whi98] D.Whitley, S.Rana, The Island Model Genetic Algorithm: On Separability, Population Size And Convergence, Journal Of Computing And Information Technology, 7(1):33–47, 1998.

[Yod 02] J.W.Yoder & R.Johnson, Implementing Business Rules With Adaptive Object Models : Business Rules Approach, Prentice Hall, 2002

CONCLUSION ET PERSPECTIVES

Conclusion et Perspectives

Trouver une classification n'est pas une tâche facile, vu qu'il faut atteindre partiellement certains objectifs (cerner le domaine de la taxonomie, déterminer les caractéristiques communes, déterminer les caractéristiques unificateurs, etc.) avant de parvenir à une classification. Ceci n'est pas un dilemme, mais plutôt un processus incrémental où on classe pour atteindre partiellement certains objectifs puis on raffine la classification. Ce processus conduira à l'amélioration des approches bio-inspirées et, à terme, promouvra une synergie interdisciplinaire qui correspond mieux à leur nature. Dans cette perspective, nous avons entrepris le travail présenté dans cette thèse, en ayant comme objectif la contribution par une meilleure compréhension du problème et ses facettes, par l'évaluation et l'étude de l'approche POE ainsi que la proposition d'une nouvelle approche de caractérisation. Dans ce qui suit, nous donnons une conclusion de ce travail, nous énumérons ses contributions puis nous indiquons les directions de recherches qu'il nous semble bénéfique d'investir.

Conclusion

Le rêve de créer une machine dotée d'une forme d'intelligence est présent depuis fort longtemps dans l'imagination humaine. Des scientifiques ont mené des recherches en s'inspirant des organismes biologiques dont l'objectif est la création des systèmes artificiels possédant des capacités d'adaptation d'apprentissage et de d'autoréparation proches de celles de l'être humain. Ces recherches ont abouti à l'apparition de plusieurs approches bio-inspirées. Dans ce travail nous avons été amenés à proposer une nouvelle façon de voir et de modéliser les systemes bio-inspirés, et nous avons pu tirer les conclusions suivantes :

- Il existe une grande diversité dans les systèmes bio-inspirés.
- Les trois processus POEtic qui façonnent les organismes biologiques ne sont, en fait, que des processus complémentaires de l'évolution. Chacun utilise des concepts et des mécanismes spécifiques pour doter l'organisme, ou l'espèce, d'une aptitude à faire face à un environnement changeant et contraignant.
- Un système bio-inspiré peut être caractérisé vis à vis de l'environnement où il opère, par deux critères : stimuli et adaptabilité.
- Contrairement à l'utilisation classique de MDA, dans le contexte de notre travail, la description par les modèles ne concerne pas le processus de développement mais le fonctionnement lui même d'un système bio-inspiré. Cette application non usuelle de l'approche MDA dans le domaine de la modélisation engendre le problème de la nécessité d'adapter l'utilisation classique de MDA pour arriver à réaliser notre

objectif. Ainsi, le choix de cette approche de représentation nous semble être très approprié pour décrire le fonctionnement d'un système bio-inspiré.

- L'étude des systèmes bio-inspirés en se basant sur l'ingénierie des modèles nous a permit d'extraire des template ou patterns de transformation permettant la conception et la réalisation d'une manière très claire une grande variété de systèmes bio-inspirés.

Dans cette thèse, nous nous sommes concentrés sur la modélisation des approches bio-inspirées. Nous avons constaté que ces dernières présentent des facteurs communs (rôle, type de description, granularité, etc.). L'utilisation des patterns de transformation présente des avantages indéniables quant à l'étude et l'évolution des systèmes bio-inspirés, et offrent des concepts et mécanismes puissants qui nous semblent être les prémisses de l'ingénierie du logiciel de demain.

Résumé des contributions

Nous estimons que ce travail nous a permis d'apporter certains points de vue, de définir et d'en proposer de nouveaux concepts, de nouveaux mécanismes. Comme point de départ de nos contributions, nous avons présenté de façon synthétique les approches bio-inspirées les plus en vu.

Nous avons identifié divers critères qui permettent la caractérisation des approches bio-inspirées aussi bien des systèmes logiciels que matériels. Ces critères et la caractérisation à base d'expressions architecturales fonctionnelles, nous ont permis de mettre en relation les différents processus des systèmes biologiques: ontogenèse, phylogenèse et épigenèse, et de situer le rôle de chacun.

Nous avons proposé une représentation semi formelle des expressions architecturales qui permettent de clarifier le processus de fonctionnement de tout système bio-inspiré.

Nous avons proposé des patterns dits patterns de transformation à base des expressions architecturales pour faire la conception de n'importe quel système bio-inspiré et cela en respectant le paradigme MDA.

Notre travail a posé les jalons d'une nouvelle démarche, cependant, elle gagnerait à être enrichie et testée sur d'autres systèmes bio-inspirés.

Perspectives de recherche

De nombreuses issues peuvent s'inscrire comme perspectives futures au travail présenté dans cette thèse. Nous pensons que la réalisation d'un simulateur de systèmes bio-inspirés que nous alimentons avec des modèles (structure, comportement, environnement) et qui s'exécute et produit des modèles résultat peut être bénéfique. L'ambition ultime serait d'arriver à un simulateur qui donnerait le même résultat que le système bio-inspiré réel (quelque soit ce dernier).

A un niveau moins abstrait, la majorité des recherches dans le domaine de l'ingénierie des logiciels visent le processus de développement des logiciels. Ce courant de recherche est motivé par les besoins réels des entreprises. Ces besoins se résument par la productivité, la pérennité du savoir-faire et la prise en compte des plateformes d'exécution.

Nos investigations nous ont montré que les langages de transformations actuellement utilisés sont des langages déclaratifs et impératifs à la fois. En fait, ils ne fournissent pas de facilités pour les transformations qui visent le développement des logiciels seulement mais, en plus, ils peuvent être utilisés, pour décrire le fonctionnement des systèmes. Les expressions impératives permettent la description des traitements extrêmement complexes tandis que les expressions déclaratives ignorent certains détails des transformations.

Ces langages peuvent être améliorés dans ce contexte en introduisant des techniques flexibles permettant l'intégration des concepts utiles pour la description de tous les types de traitement qui peuvent être effectué par les systèmes.

REFERENCES GLOBALES

[Ada 98] C.Adami, Introduction to Artificial Life, AI and Society Journal, Springer-Verlag V13, New-York Etas-Unis, 1998, pp450-451.

[Aic 03] P.Aickelin, P.Bentley, S.Cayzer, J.Kim, J.McLeod, Danger Theory: The Link between AIS and IDS?, 2nd International Conference on Artificial Immune Systems, Napier University, Edinburgh, UK, 1-3 Septembre 2003, pp 147-155.

[Ast 00] J.C.Astor, C. Adami, A Developmental Model For The Evolution Of Artificial Neural Networks, Artificial Life, 6(3) :189–218, 2000.

[Atl 06] ATLAS, Complex data management in distributed systems, http://www.sciences. univnantes. fr/lina/ATLAS/, 2006.

[Aza 00] F. Azam Biologically Inspired Modular Neural Networks. PhD thesis, Faculty of the Virginia Polytechnic Institute and State University USA, May 2000.

[Bac 97] T.Back, U.Hammel, H.P.Schwefel, Evolutionary Computation : Comments On The History And Current State, Ieee Transactions On Evolutionary Computation, 1(1) :3–17, Avril 1997.

[Bal 95] S.Baluja, R.Caruana, Removing the genetics from the standard genetic algorithm, Proceedings of the Twelfth International Conference on Machine Learning, San Mateo, CA Etas-Unis, 22 Mai 1995,pp38-46.

[Bar 05] F. Barbier, UML 2 ET MDE Ingénierie des modèles avec études de cas, Livre édition Dunod, 2005.

[Bel 08] A.Belaïd, Implémentation d'une méthode de classification non supervisée de données à l'aide d'algorithmes génétiques évolutionnaires, Mémoire d'ingénieur en informatique INI Algerie 2008.

[Bel 92] E. Belhaire, Contribution à la réalisation électronique de Réseaux de Neurones Formels : Intégration Analogique d'une machine de Boltzmann, Thèse de doctorat, Université de Paris XI Orsay Paris France, Février 1992.

[Ben 07] I.Benkermi, Modèle et algorithme d'ordonnancement pour architectures reconfigurable dynamiquement. Thèse de doctorat, Université de Rennes 1 Rennes France, Janvier 2007.

[Bie 10] M.Biehl, Literature Study on Model Transformations, Journal of Royal Institute of Technology, Tech. Rep, July 2010.

[Bla 05] X.Blanc, MDA en Action ingénierie logicielle guidée par les modèles, livre éditions Eyrolles, 2005.

[Bla 96] F.Blayo et M.Verleysen, Les réseaux de neurons artificiels, Presses Universitaires de France, 1ère édition Que Sais-je No 3042, France, 1996.

[Bor 00] S.Bornholdt, T.Rohlf, Topological Evolution Of Dynamical Networks : Global Criticality From Local Dynamics, Physical Review Letters, 84(26) :6114–6117, Juin 2000.

[Bud 04] F.Budinsky, S.D.Merks, Eclipse Modeling Framework, livre edition Addison Wesley, The Eclipse series, 2004.

[Buf 02] E. Buffetaut, cuvier le découvreur du monde disparus, Belin pour la science, Paris, Octobre 2002.

[Bur 91] G. Burel, réseaux de neurones en traitement d'images: Des Modèles Théoriques aux Applications Industrielles. Thèse de doctorat, Université de Bretagne Occidentale, Décembre 1991.

[Cam 00] G.Camazine, S.Deneubourg, J.Franks, N.Sneyd, J.Theraulaz, and E.Bonabeau, Self-Organization in Biological Systems, Livre édition Princeton University Press, New Jersey Etas-Unis, 2000.

[Car 06] B.Caroline, Modélisation mathématique de la réponse lymphocytaire T spécifique à une infection virale, Thèse de Doctorat, l Ecole Nationale Supérieure des Mines de Saint-Etienne et de l'Université Jean Monnet de Saint-Etienne France, le14 avril 2006.

[Cay 05] S.Cayzer, J.Smith, A.R.Marshall, T.Kovacs, What Have Gene Libraries Done for AIS? , 4th International Conference on Artificial Immune Systems, Banff, Canada, 14-17 Aout 2005.

[Cay 06] S.Cayzer & J.Smith, Gene Libraries: Coverage, efficiency and diversity, 6th Conference of Adaptation in Artificial and Biological Systems University of Bristol, Bristol, Angleterre, 03-06 Avril 2006.

[Cou 07] P.Coulibaly, F.Anctil, B.Bobée, Prévision hydrologique par réseaux de neurones artificiels : état de l'art, Revue canadienne de génie civil. Vol. 26, 1999, pp. 293-304.

[Cza 06] K.Czarnecki, S.Helsen, Feature-based survey of model transformation approaches, IBM Systems Journal, V45, N 3, 2006, pp621-645.

[Dar 08] F.Dario, C.Mattiussi, Bio-Inspired Artificial Intelligence Theories, Methods and Technologies, Livre edition The MIT Press Cambridge, Massachusetts, London 2008.

[Das 06] D.Dasgupta, Advances in Artificial Immune Systems, IEEE Computational Intelligence Magazine, Novembre 2006, pp40-49.

[Das 99] D.Dasgupta, S. Forrest, Artificial Immune Systems in Industrial Applications, the Second International Conference on Intelligent Processing and Manufacturing of Materials (IPMM), Honolulu, Etas-Unis, 10-15 Juillet 1999.

[Dec 00] L.N.De Castro, F.J.Von Zuben, Artificial Immune Systems: Part II A Survey of Applications, Technical Report RT DCA 02/00, FEEC/UNICAMP, Brazil, 2000.

[Dec 02] L.DeCastro, , J.Timmis, Artificial Immune Systems : A Novel Paradigm To Pattern Recognition, University Of Paisley, 67–84, 2002.

[Dec 02a] L.N.De Castro, F.J.Von Zuben, Learning and Optimization Using the Clonal Selection Principle, the Special Issue on Artificial Immune Systems of the journal IEEE Transactions on Evolutionary Computation, Vol. 6, N°3, Juin 2002.

[Dec 02b] L.N.De Castro, J.Timmis, Artificial Immune Systems: A Novel Approach to Pattern Recognition, In L Alonso J Corchado and C Fyfe, livre edition Artificial Neural Networks in Pattern Recognition, University of Paisley, Ecosse, Janvier 2002, pp67-84.

[Dec 03] L.N.De Castro, J.Timmis, Artificial Immune Systems as a Novel Soft Computing Paradigm. In the Soft Computing Journal, V7, Issue 7, Juillet 2003.

[Dec 99] L.N.de Castro, F.J.Von Zuben, Artificial Immune Systems: Part I Basic Theory and Applications. Technical Report – RT DCA 01/99, FEEC/UNICAMP, Brazil, 1999.

[Deg 93] H. De Garis,Growing An Artificial Brain With A Million Neural Net Modules Inside A Trillion Cell Cellular Automaton Machine, Dans Proc. Of The Fourth International Symposium On Micro Machine And Computer Science, Pages 211– 214, 1993.

[Deg 97] H.De Garis, L.Kang, Q.He, Z.Pan, M.Ootani & E.Ronald, Million Module Neural Systems Evolution: The Next Step In Atr's Billion Neuron Artificial Brain, In Proceedings Of Evolution Artificielle 97 (Ea'97), Volume 10, Pages 231–243, 1997.

[Der 04] G. Dreyfus, J.M. Martinez, M. Samuelides, M. B. Gordon, F. Badran, S. Thiria, L. Hérault, Réseaux de neurones, méthodologie et applications, sous la direction de Gérard Dreyfus, Livre 2ème édition, Eyrolles, avril 2004.

[Dia 09] S.Diaw, R.Lbath, B.Coulette, Etat de l'art sur le développement logicie basé sur les transformations de modèles, Université de Toulouse France, 2009.

[Dob 66] T.Dobzhansky, L'homme en évolution, Edition Flammarion, Paris 1966.

[Dor 99] M.Dorigo, G.Dicaro, Ant Colony Optimization Meta-Heuristic, In D.Corne, M.Dorigo, & F.Glover, Editors, New Ideas In Optimization, Pages 11-32, Mcgraw-Hill, 1999.

[Dre 03] J.Dreo, A.Perowski, P.Siarry, E.Taillard, Métaheuristiques pour l'optimisation difficile, Livre Edition Eyrolles 2003.

[Dre 05] J. Dréo, A. Petrowski, P. Siarry, E.Taillard, Métaheuristiques pour l'optimisation difficile, livre Editions Eyrolles, 2005.

[Dub 98] D. Duboule, A. S. Wilkins, The evolution of bricolage, Trends in Genetics Journal, Vol14, Fevrier 1998, pp54-59.

[Eib 03] A.E.Eiben, J.E.Smith, Introduction to Evolutionary Computing. Livre edition Natural Computing Series Springer-Verlag, Berlin Allemagne, 2003.

[Fer 95] J.Ferber, Les Systèmes Multi-Agents, Vers Une Intelligence Collective, Intereditions, Paris, 1995.

[Flo 98] D.Floreano & J.Urzelai, Evolution And Learning In Autonomous Robotics Agents, Dans D. Mange & M. Tomassini, Editeurs, Bio-Inspired Computing Machines, Chapitre 12, Pages 317–346. Ppur, Lausanne, 1998.

[Fog 66] L.J.Fogel, A.J.Owens, M.J.Walsh, Artificial Intelligence through Simulated Evolution, Livre édition Wiley, New York Etas-Unis, 1966.

[For 96] S.Forrest, Genetic Algorithms, Acm Computing Surveys, Vol. 28, Issue 1, March 1996, Pp. 77-80.

[Fre 09] A.Frederic, Un Système Tutoriel Intelligent A Base De Système Immunitaire Artificiel, Ecole Nationale D'ingénieurs De Brest Rapport De Stage De Master Recherche En Informatique 2009

[Gar 02] D.Garlan And B.Schmerl, Model-Based Adaptation For Self-Healing Systems, Woss'02, Charleston, Usa, Nov. 2002..

[Gat 07] M.L. Gatet, Intégration de réseaux de neurones pour la Télémétrie Laser. Thèse de doctorat, Université de Toulouse France, Septembre 2007.

[Geh 02] W.Gehring, M.Affolter et T.Burglin, Homeodomain Protiens, Annual Reviews of Earth and Planetary Sciences, 2002, pp487-523.

[Gil 03] S.F.Gilbert, Developmental biology, Livre 7eme edition, Sinauer associates Inc. Publishers, 2003.

[Gol 89] D.E.Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Livre edition Addison-Wesley, Redwood City, CA Etas-Unis,1989.

[Goo 00] C.S.Goodman & B.C.Coughlin, The Evolution Of Evo-Devo Biology,Proceedings Of The National Academy Of Sciences Of The Usa, 97(9) :4424– 4425, Avril 2000.

[Gou 91] S.J.Gould, La vie est belle les surprises de l'évolution, livre édition science, Paris, 1991.

[Gru 92] F. Gruau. Genetic Systems Of Boolean Networks With A Cell Rewriting Developmental Process. Dans D. Whitley & S.D. Schaffer, Editeurs, Combination Of Genetic Algorithms And Neural Networks, Pages 55–74. Ieee Computer Society Press, Los Alamitos, Ca, 1992.

[Gut 94] H.A.Gutowitz, C.G.Langton, Methods for designing Cellular Automata with Interesting Behavior, 1994.

[Had 01] P.C. Haddow, G. Tufte & P. Van Remortel, Shrinking The Genotype : L-Systems For Ehw?,Dans Ices '01 : Proceedings Of The 4th International Conference On Evolvable Systems : From Biology To Hardware, Pages 128–139. Springer-Verlag, 2001.

[Har 01] F.Harold, The Way of the Cell: Molecules, Organisms, and the Order of Life, Oxford University Press, Oxford, 2001.

[Has 95] M.H. Hassoum, Fundamentals of artificial neural networks. MIT Press, Cambridge Angleterre, 1995.

[Hay 99] B.Hayes, experimental Lamarckism, American scientist Journal, Vol. 87, N°6, November-December, 1999, pp494-498.

[Heu 94] J. C. Heudin, La Vie Artificielle, Edition Hermes, 1994.

[Heu 98] J.C.Heudin, L'évolution au bord du chaos, Livre édition Hermès, Paris, 1998.

[Hol 75] J.H.Holland, Adaptation in Natural and Artificial Systems, Livre édition University of Michigan Press, Ann Arbor, Etas-Unis, 1975.

[Hol 99] P.W.H.Holland, The Future Of Evolutionary Developmental Biology,Nature, 402 :41–44, Décembre 1999.

[Hub 08] P. Huber, The model transformation language jungle - an evaluation and extension of existing approaches, Memoire Master, Universite de Technische Vienne Autriche, May 2008.

[Jai 96] A.K.Jain, Artificial Neural Networks: A Tutorial, Ieee Computer, March 1996, Pp 31-44.

[Jea 06] R.Jean, Les Reseaux Neuronaux Chapitre 7 Livre Edition Vuibert 2006.

[Jou 05] F.Jouault et I.Kurtev, Transforming Models with ATL, In Satellite Events at the Models 2005 Conférence, Proceedings of the Model Transformations in Practice Workshop, volume 3844 de Lecture Notes in Computer Science Springer, Montego Bay Jamaica, 2005.

[Kaa 02] G. Kaati, L.O.Bygren et S.Edvinsson, cardiovascular and diabetes mortality determined by nutrition during parents and grandparents slow growth period, european journal of human genetics, London, 2002, pp682-688.

[Ken 95] J. Kennedy & R.Eberhardt, Particle Swarm Optimization, Dans Proc, Ieee International Conference On Neural Networks, Volume 4, Pages 1942– 1948, 1995.

[Kir 83] S.Kirkpatrick, C.D.Gelatt, M.P.Vecchi, Optimization by simulated annealing. Journal of Science, 1983, pp671-680.

[Kit 90] H.Kitano,Designing Neural Networks Using Genetic Algorithms With Graph Generation System, Complex Systems, 4(4) :461–476, 1990.

[Koh 87] T. Kohonen, Self-organization and associative memory, Livre 2eme Édition Springer-verlag, Berlin, 1987.

[Koz 92] J.R.Koza, Genetic Programming: On the Programming of Computers by Means of Natural Selection, Livre edition MIT Press, Cambridge, MA Etas-Unis, 1992.

[Lam 09] M.L.Lamali, Approches évolutives pour la classification non supervisée de données, Mémoire de fin d'études pour l'obtention du diplôme d'ingénieur d'état en informatique, Ecole national Supérieure d'Informatique (ESI) Oued-Smar, Alger 2009.

[Lew 99] B.Lewin, Genes VII, Livre Oxford University Press, Oxford, 1999

[Lin 68] A.Lindenmayer, Mathematical Models For Cellular Interaction In Development, Parts I And Ii, Journal Of Theoretical Biology, 18:280–315, 1968.

[Lin 88] A.Lindenmayer And P.Prusinkiewiscz, Developmental Models Of Multicellular Organisms: A Computer Graphics Perspective, Addison-Wesley, 1988.

[Lin 91] F.Lints, Génétique, 3eme édition, Lavoisier Technique et Documentation, 1991.

[Man 00] D. Mange, M. Sipper, A. Stauffer & G. Tempesti, Towards Robust Integrated Circuits: The Embryonics Approach, Proceedings Of The Ieee, 88(4) :516–541, Avril 2000.

[Man 98] D. Mange & M. Tomassini, Bio-Inspired Computing Machines : Towards Novel Computational Architectures, Presses Polytechniques Et Universitaires Romandes, Lausanne, Switzerland, 1998.

[Mat 02] S.Matzinger, The Danger Model: A renewed sense of self Science, Journal of Science, 2002, pp301-304.

[May 82] E.Mayr, speciation and macroevolution, Journal of evolution, 1982, pp1119-1132.

[Mcc 43] W.S.Mcculloh, W.Pitt, A logical calculus of the ideas immanent in nervous activity, Bulletin of Mathematical Biophysics, 1943.

[Med 07] M.Medjoudj, Approche de Partitionnement et Ordonnancement à base de Systèmes Immunitaires Artificiels Mémoire de fin d'études pour l'obtention du diplôme d'ingénieur d'état en informatique Institut National de formation en Informatique (I.N.I) Oued-Smar, Alger 2007.

[Men 06] T.Mens K.Czarnecki, P.Van Gorp, Applying a model transformation taxonomy to graph transformation technology, in Proceedings of the International Workshop on Graph and Model Transformation, V152, Tallinn Estonia, mars 2006, pp143-159.

[Mes 03] B. Mesot, E. Sanchez, C.-A. Pena & A. Perez-Uribe, Sos++ : Finding Smart Behaviors Using Learning And Evolution, Dans R.K. Standish M.A. Bedau & H.A. Abbass, Editeurs, Proceedings Of The Eighth International Conference On Artificial Life, Pages 264–273, Cambridge, Mass., 2003.

[Mes 06] D.Meslati, Mage : Une Approche Ontogénétique De L'évolution Dans Les Systèmes Logiciels Critiques Et Embarqués. Thèse De Doctorat D'état, Université De Annaba, Février 2006.

[Mia 10] MIA-Software. MIA-Transformation and MIA-Generation website . http://www.mia-software.com.

[Mic 03] E.Michel, Les neurones et créativité, revue littéraire Lieux d'être, n° 36, automne, Paris, 2003.

[Mic 96] Z.Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, Livre 3rd edition, Springer-Verlag, Berlin 1996.

[Mil 04] J.F.Miller,Evolving A Self-Repairing, Self-Regulating, French Flag Organism, Gecco 2004, Volume 1 De Lecture Notes In Computer Science, Pages 129–139, Berlin, Heidelberg, 2004. Springer.

[Min 75] M.Minsky, A framework for Representing Knowledge. In: The Psychology of Computer Vision, Livre edition Wiston, MacGraw-Hill, New York Etat-Unis, 1975.

[Mof 10] http://www.omg.org/mof/

[Moo 01] S.W.Moon & S.G. Kong, Block-Based Neural Networks, Ieee Transactions On Neural Networks, 12(2) :307–317, Mars 2001.

[Mou 05] I. Mounier, X. Blanc, UML2 Pour Les Développeurs, livre édition Eyrolles, 2005.

[Nyd 01] U.Nydegger, Hematologie la cellule souche d'ici a la et retour, Forum medical suisse N° 51/52, décembre 2001

[Ocl 05] Object Constraint Language Version 2.0, the official homepage. http://www.omg.org/spec/OCL/2.0/, 2005.

[Ozt 04] C. Ozturkeri & M. Capcarrere, Emergent Robustness And Self-Repair Through Developmental Cellular Systems, Proc. Ninth International Conference On The Simulation And Synthesis Of Living Systems (Alife9), Pages 31–26, Cambridge, Massachusetts, Usa, 2004. The Mit Press.

[Per 99] A.Perez-Uribe, Structure-Adaptable Digital Neural Networks, Phd Thesis, Swiss Federal Institute Of Technologiy-Lausanne, Epfl, 1999.

[Poe 05] Http://Www.Poetictissue.Org.

[Pre 96] L.Prechelt, A quantitative study of experimental evaluations of neural networklearning algorithms: current research practice, The Official Journal of the International Neural Network Society, European Neural Network Society & Japanese Neural Network Society V9, Avril 1996, pp457-462.

[Rec 73] I.Rechenberg, Evolutions strategie : Optimierung Technischer Systeme Nach Prinzipien Der Biologischen Evolution, Frommann-Holzboog, Stuttgart, Germany, 1973.

[Ren 00] J.P.Rennard support de cours les automates cellulaires **http://www.rennard.org/alife** 2000.

[Rid 96] M. Ridley, Evolution, Livre 2eme edition, Blackwell scientific publications Ltd, Oxford, 1996.

[Rog 03] D. Roggen, D.Floreano & C. Mattiussi,A Morphogenetic Evolutionary System : Phylogenesis Of The Poetic Tissue, Proceedings Of The Fifth International Conference On Evolvable Systems (Ices 2003), Pages 153–164, Berlin, 2003. Springer.

[Ros 59] F. Rosenblatt, Principles of Neurodynamics, Livre édition Spartan Books, New York Etas-Unis, 1959.

[San 98] F.Santos Osório, un système hybride neuro-symbolique pour l'apprentissage automatique constructif, Thèse de doctorat, Institut national polytechnique de Grenoble France, février 1998.

[Sas 00] T.Sasaki et M.Tokoro, comparison between lamarckian and darwinian evolution on a model using neural networks and genetic algorithms, Journal of knowledge and information systems, London, juin 2000, pp201-222.

[Sch 92] N.Schraudolph, R.K.Belew, Dynamic parameter encoding for genetic algorithms, éditeur Machine Learning, 1992.

[Seg 98] R.Segev, E.Ben Jacob from neurons to brain: adaptive self writing of neurons, journal of complex systems, 1998, pp67-78.

[Sip 97] M. Sipper & Al, A Phylogenetic, Ontogenetic, And Epigenetic View Of Bio-Inspired Hardware Systems, Ieee Transactions On Evolutionary Computation, Vol. 1, No. 1, April 1997, Pp 83-97.

[Sol 00] R.Soley, Model Driven Architecture (MDA), Draft 3.2. Object Management Group, Inc, 27 novembre 2000.

[Sor 95] P. Sordino, F.Van Der Hoeven et D. Duboule, Hox Gene Expression in Teleost Fins and the Origin of Vertebrate Digits . Journal of Nature, juin 1995, pp678–681.

[Ste 04] S.Stepney, R.E.Smith, J.Timmis and A.M.Tyrrell., Towards a Conceptual Framework for Artificial Immune Systems, In the proceedings of the Third International Conference on Artificial Immune Systems, Catania, Italy,13-16 Sptembre, 2004.

[Sys 89] G.Syswerda, Uniform crossover in genetic algorithms. In Proceedings of the Third International Conference on Genetic Algorithms, Morgan Kaufmann, San Mateo, CA Etas-Unis, 04-07 Juin 1989, pp2-9.

[Tem 02] G.Tempesti, A Poetic Architecture For Bio-Inspired Hardware, 8th Int. Conf. On The Simulation And Synthesis Of Living Systems, Sydney, Australia, Dec. 2002, Mit Press, Cambridge, 2002.

[Tem97] G.Tempesti, D.Mange & A.Stauffer,A Robust Multiplexer-Based Fpga Inspired By Biological Systems, Journal Of Systems Architecture : Special Issue On Dependable Parallel Computer Systems, 40(10) :719–733, Septembre 1997.

[Teu 01] C.Teuscher & E. Sanchez, Self-Organizing Topology Evolution Of Turing Neural Networks, Proceedings Of The International Conference On Artificial Neural Networks (Icann2001), Volume 2130 De Lecture Notes In Computer Science, Pages 820–826, Berlin, Heidelberg, 2001. Springer-Verlag.

[Tho 05] Y.Thoma, Tissu Numérique Cellulaire A Routage Et Configuration Dynamiques, Thèse Epfl, No 3226, 2005

[Tim 05] J.Timmis, M.Neal, Once more unto the breach: Towards artificial homeostasis, Recent Developments in Biologically Inspired Computing, chapter in Book Recent Developments in Biologically Inspired Computing edition L N De Castro and F J Von Zuben, 2005, pp340-365.

[Tim 08] J.Timmis, A.Hone, Theoretical Advances In Artificial Immune Systems. Volume 403, 11–32 2008.

[Tor 97] J.M.Torres-Moreno, Apprentissage et généralisation par des réseaux de neurones : étude de nouveaux algorithmes constructifs. Thèse de doctorat, Institut national polytechnique de Grenoble, France, septembre 1997.

[Uni 04] Taxonomie, Encyclopædia Universalis En Ligne, Mars 2004.

[Van 09] T.L.Van Zyl , E.M.Ehlers, A Need For Biologically Inspired Architectural Description: The Agent Ontogenesis Case, 10th Pacific Rim International Conference On Multi-Agents (Prima 2007), Bangkok (Thailan), 21-23 November, 2007, Lecture Notes In Computer Science, Springer, Vol. 5044, Agent Computing And Multi-Agent Systems, Pages 146-157, (2009).

[Vir 01] B.Virole, Résaux de neurones et Psychométrie. Editions du Centre de Psychologie Appliquée, Juin 2001.

[Wag 96] G.P.Wagner, L.Altenberg, Complex adaptations and the evolution of evolvability. Journal of Evolution, 1996, pp967-976.

[Wat 04] A.Watkins, J.Timmis, Artificial Immune Recognition System (Airs) :An Immune-Inspired Supervised Machine Learning Algorithm, Journal of Genetic Programming And Evolvable Machines Volume 5, 291–317, 2004.

[Wat 53] J.Watson, F.Crick, A structure for deoxyribose nucleic acid , Journal of Nature, Avril 1953, pp737-738.

[Wei 91] S.M.Weiss, C.A.Kulikowski, Computer systems that learn, Morgan Kaufmann Publishers Inc., San Mateo, Californie Etas-Unis, 1991.

[Wes 03] A.Wesley, A.Kleppe, MDA Explained: The Model Driven Architecture: Practice and Promise, Livre edition Wesley, 2003.

[Wey 07] D.Weyns, A.Omicini, J.Odell, Environment As A First Class Abstraction In Multiagent Systems, Autonomous Agents And Multi-Agent Systems, Vol. 14, N°1, Springer, (2007) 5-30.

[Whi 03] A.White, M.Simon, Improved Pattern Recognition with Artificial Clonal Selection? , 2ed International Conference on Artificial Immune Systems (ICARIS), Napier University, Edinburgh, UK, 1-3 Septembre 2003.

[Whi 98] D.Whitley, S.Rana, R.B.Heckendorn, The island model genetic algorithm: On separability, population size and convergence, Journal of Computing and Information Technology, 1998, pp33-47.

[Wol 02] S.Wolfram, A New Kind of Science, Livre édition Wolfram Media, 2002.

[Yao 99] X.Yao, Evolving Artificial Neural Networks, Proceedings Of The Ieee, 87(9) :1423–1447, Septembre 1999.

[Yod 02] J.W.Yoder & R.Johnson, Implementing Business Rules With Adaptive Object Models : Business Rules Approach, Prentice Hall, 2002

[Zem 03] M.R.Zemouri, Contribution à la surveillance des systèmes de production à l'aide des réseaux de neurones dynamiques : Application à la e-maintenance, Thèse de doctorat, Université de Franche-Comté, Novembre 2003.