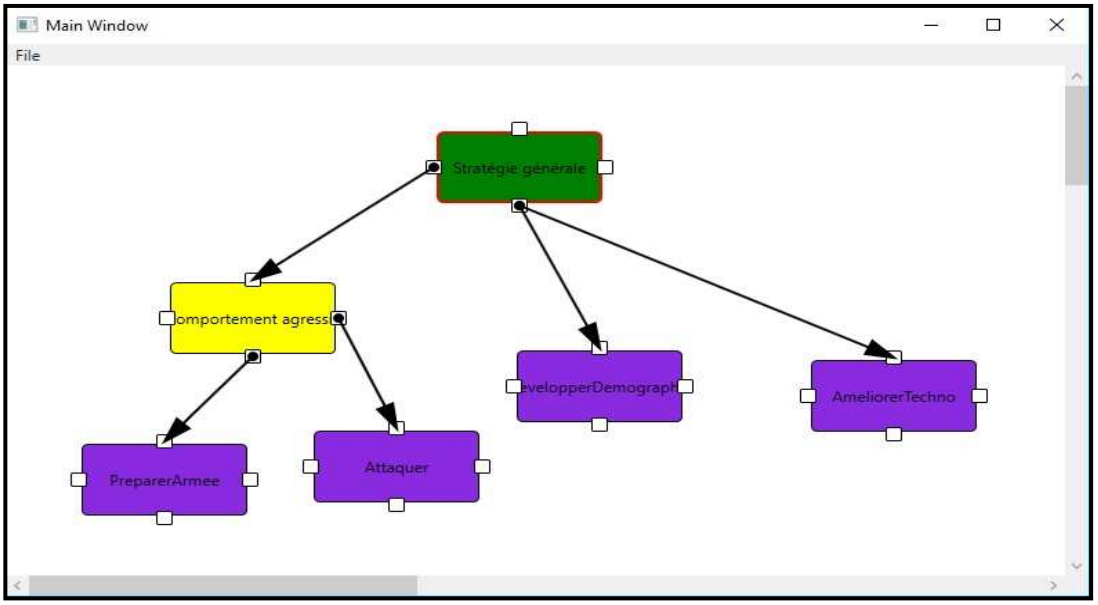
Vers une simplification de la conception de comportements stratégiques pour les opposants dans les jeux vidéo de stratégie

Juliette Lemaitre

Par **Juliette LEMAITRE**

Vers une simplification de la conception de comportements stratégiques pour les opposants dans les jeux vidéo de stratégie

Thèse présentée pour l'obtention du grade de Docteur de l'UTC



Soutenue le 21 mars 2017
Spécialité : Informatique et Sciences et Technologies de l'Information et des Systèmes : Unité de recherche Heudyasic (UMR-7253)

D2343

Manuscrit de thèse
pour l'obtention du grade de Docteur de l'Université de Technologie de Compiègne

Spécialité : Informatique et Sciences et Technologies de l'Information et des Systèmes

Vers une simplification de la conception de comportements stratégiques pour les opposants dans les jeux vidéo de stratégie

Soutenue le 21 Mars 2017

Juliette LEMAITRE

Jury composé de :

M. Pascal ESTRAILLIER *Rapporteur*	Professeur, L3i, Université de La Rochelle
M. René MANDIAU *Rapporteur*	Professeur, LAMIH-CNRS, Université de Valencienne
M. Jean-Paul BARTHES *Examinateur*	Professeur émérite, Heudiasyc, Université de Technologie de Compiègne
M. Vincent CORRUBLE *Examinateur*	Maître de conférences, LIP6, Université Pierre et Marie Curie
M. Stéphane NATKIN *Examinateur*	Professeur, CEDRIC, Conservatoire National des Arts et Métiers
M. Florian RICHOUX *Examinateur*	Maître de conférences, LS2N, Université de Nantes
M. Nicolas SABOURET *Examinateur*	Professeur, LIMSI-CNRS, Université Paris Sud
Mme Domitile LOURDEAUX *Directrice*	Maître de conférences, Heudiasyc, Université de Technologie de Compiègne

A mon papa qui est parti trop tôt.
Je t'aime Papa.

Abstract

This PhD thesis addresses the topic of creating artificial intelligence (AI) to control high-level decision-making in strategy games. This kind of game offers complex environments that require the manipulation of a large number of resources by choosing actions depending on long-term goals. This AI design is not simple because it is about providing to the player a playful and interesting experience. Hence, the aim is not to create unbeatable behaviors, but rather to display several personality traits allowing the player to face diverse opponents. Its creation involves game designers who are responsible of defining several strategies according to the experience they want to provide to the player, and game developers who implement those strategies to put them into the game. The collaboration between them requires many exchanges and development iterations to obtain a result corresponding to game designers' expectations. The objective of this PhD thesis is to improve and simplify the creation of strategical behaviors by proposing a strategy model intelligible to game designers and that can be interfaced easily with developers' work. For game designers, a strategy model has been created to allow them to express rules guiding the choice of goals and their allocated resources. These rules make it possible for game designers to express which goal to choose according to the context but also to choose several of them and give them relative importance in order to influence the resource distribution. To improve intelligibility we use a graphical model inspired from finite state machines and behavior trees. Our proposition also includes a strategy engine which executes the strategies created with the model. This execution produces directives that are represented by a list of selected strategical goals and the resources that have been allocated according to the importance and needs of each goal. These directives are intended for a tactical module in charge of their application. The developers are then responsible for the implementation of this tactical module. Our solution enables game designers to directly design the strategical level of an AI and therefore facilitates their cooperation with game developers and simplifies the entire creation process of the AI.

Résumé

Cette thèse aborde la problématique de la création d'intelligences artificielles (IA) contrôlant la prise de décision haut-niveau dans les jeux de stratégie. Ce type de jeux propose des environnements complexes nécessitant de manipuler de nombreuses ressources en faisant des choix d'actions dépendant d'objectifs à long terme. La conception de ces IA n'est pas simple car il s'agit de fournir une expérience pour le joueur qui soit divertissante et intéressante à jouer. Ainsi, le but n'est pas d'obtenir des comportements d'IA imbattables, mais plutôt de refléter différents traits de personnalités permettant au joueur d'être confronté à des adversaires diversifiés. Leur conception fait intervenir des game designers qui vont définir les différentes stratégies en fonction de l'expérience qu'ils souhaitent créer pour le joueur, et des développeurs qui programment et intègrent ces stratégies au jeu. La collaboration entre eux nécessite de nombreux échanges et itérations de développement pour obtenir un résultat qui correspond aux attentes des designers. L'objectif de cette thèse est de proposer une solution de modélisation de stratégies accessible aux game designers en vue d'améliorer et de simplifier la création de comportements stratégiques. Notre proposition prend la forme d'un moteur stratégique choisissant des objectifs à long terme et vient se placer au dessus d'un module tactique qui gère l'application concrète de ces objectifs. La solution proposée n'impose pas de méthode pour résoudre ces objectifs et laisse libre le fonctionnement du module tactique. Le moteur est couplé à un modèle de stratégie permettant à l'utilisateur d'exprimer des règles permettant au moteur de choisir les objectifs et de leur allouer des ressources. Ces règles permettent d'exprimer le choix d'objectifs en fonction du contexte, mais également d'en choisir plusieurs en parallèle et de leur donner des importances relatives afin d'influencer la répartition des ressources. Pour améliorer l'intelligibilité nous utilisons un modèle graphique inspiré des machine à états finis et des behavior trees. Les stratégies créées à l'aide de notre modèle sont ensuite exécutées par le moteur de stratégie pour produire des directives qui sont données au module tactique. Ces directives se présentent sous la forme d'objectifs stratégiques et de ressources qui leur sont allouées en fonction de leurs besoins et de l'importance relative qui leur a été donnée. Le module stratégique permet donc de rendre accessible la conception du niveau stratégique d'une IA contrôlant un adversaire dans un jeu de stratégie.

Remerciements

Je tiens tout d'abord à remercier l'entreprise MASA Group pour m'avoir donné l'opportunité de faire cette thèse sur un sujet qui m'a énormément plu, pour les rencontres que j'ai pu y faire et également pour l'environnement de travail chaleureux et le soutien durant ces trois années. Je les remercie également, ainsi que l'ANRT, pour le financement.

Des remerciements particuliers à Caroline CHOPINAUD à l'origine de cette thèse pour m'avoir épaulée du début à la fin, et pour son encadrement bienveillant. Merci également à Domitile LOURDEAUX qui a acceptée de diriger cette thèse et m'a aidée à découvrir la recherche académique et ses rouages, que je ne connaissais alors pas du tout.

Je remercie également tous les membres de mon jury qui m'ont fait le plaisir de bien vouloir lire mes travaux et assister à ma soutenance. Je les remercie pour leur temps, leur intérêt et leurs questions très intéressantes. Je remercie en particulier mes rapporteurs, M. Pascal ESTRAILLIER et M. René MANDIAU pour avoir pris le temps d'étudier ce manuscrit en détail et pour leurs retours très enrichissants.

Je tiens à remercier certaines personnes de MASA Group que j'ai eu la chance de rencontrer pendant cette thèse. Tout d'abord James pour son aide à la rédaction de mes articles en anglais et dont la curiosité et l'intérêt rafraichissant pour mes recherches a pu raviver ma motivation pour continuer celles-ci. Un grand merci également à Clodéric pour m'avoir encouragée à faire du bénévolat à nucl.ai et à postuler à la bourse IGDA m'ayant permis d'aller d'assister à la GDC, ce fut des étapes décisives pour la suite de mon parcours professionnel et je ne serais surement pas où je suis désormais sans ses conseils précieux.

Je ne peux évidemment pas oublier de remercier mes fidèles co-thésardes, Lucile, Melody et Lauriane, qui m'ont permis de partager les moments difficiles et de me sentir moins seule durant les périodes de doute. Aucun mot ne me permettra de vous dire assez merci malheureusement, mais en partenaires de galère vous savez mieux que quiconque l'importance que peuvent avoir les co-thésards. Je suis heureuse d'avoir pu vous rencontrer pendant cette aventure ! Et Melody... merci pour la bonne humeur :)

Durant cette thèse il y a bien entendu eu des périodes plus difficiles et pour arriver jusqu'au bout j'ai eu la chance de me sentir très entourée grâce à une famille et une belle-famille très présentes et aimantes. Elles m'ont permis de mettre en perspective cette thèse et de croire en ma capacité à la terminer avec succès. Un merci un peu particulier à mes beaux-parents qui ont eu "la chance" de m'héberger pendant les derniers mois et notamment pendant la période assez

intense du dernier mois de rédaction. Et bien entendu un grand merci à Steph pour m'avoir soutenue et avoir cru en moi et en mes capacités à aller jusqu'au bout malgré mes doutes et mes complaintes incessantes !

Table des matières

Abstract v

Résumé vii

1 Introduction 1

2 La création d'un jeu vidéo 7

2.1 L'expérience du joueur . . . 7

2.1.1 Le niveau de difficulté . . . 8

2.1.2 La multiplicité de l'expérience . . . 10

2.1.3 Conclusion . . . 13

2.2 L'expérience du game designer . . . 13

2.2.1 L'assistance du game designer . . . 13

2.2.2 Les contraintes de l'assistance en IA . . . 15

2.2.3 Conclusion . . . 16

2.3 Les jeux de stratégie . . . 16

2.3.1 La définition d'un jeu 4X . . . 16

2.3.2 Les défis pour la recherche . . . 18

2.3.3 Conclusion . . . 19

2.4 Bilan du chapitre . . . 19

3 La définition commune d'une stratégie 21

3.1 La stratégie prend place dans un système... . . . 22

3.1.1 ... De taille importante . . . 22

3.1.2 ... Avec des ressources internes communes . . . 25

3.1.3 ... Qui se place dans un environnement dynamique . . . 26

3.1.4 Conclusion . . . 27

3.2 La mise en place d'une stratégie . . . 28

3.2.1 Les informations transmises par la stratégie . . . 28

3.2.2 L'adaptation de la stratégie . . . 30

3.2.3 Conclusion . . . 33

3.3 Bilan du chapitre . . . 33

4 La stratégie dans les jeux vidéo 35
4.1 Le contexte des jeux de stratégie . . . 36
4.2 La décomposition du problème . . . 37
4.2.1 La décomposition hiérarchique . . . 37
4.2.2 La décomposition fonctionnelle . . . 40
4.2.3 Conclusion . . . 43
4.3 L'adaptabilité des objectifs . . . 44
4.3.1 L'adaptation de la sélection . . . 44
4.3.2 La remise en cause de la sélection . . . 46
4.3.3 Conclusion . . . 47
4.4 L'allocation des ressources . . . 48
4.5 Bilan du chapitre . . . 50

5 Méthode de création et d'exécution d'un raisonnement stratégique 53
5.1 La vue d'ensemble . . . 54
5.2 Le modèle . . . 55
5.2.1 La décomposition . . . 56
5.2.2 L'adaptation . . . 58
5.2.3 La répartition des ressources . . . 62
5.2.4 Conclusion . . . 67
5.3 Le moteur . . . 67
5.3.1 La sélection des objectifs . . . 67
5.3.2 La répartition des ressources . . . 70
5.3.3 Conclusion . . . 78
5.4 Bilan du chapitre . . . 78

6 Evaluation 81
6.1 L'implémentation . . . 81
6.1.1 Le moteur de stratégie dans le jeu . . . 82
6.1.2 L'outil . . . 88
6.2 Le test utilisateur . . . 92
6.2.1 Déroulement . . . 92
6.2.2 Présentation du questionnaire . . . 92
6.2.3 Résultats . . . 93
6.2.4 Discussion . . . 95
6.3 L'évaluation . . . 96
6.3.1 La méthodologie . . . 97
6.3.2 Les critères d'évaluation . . . 97
6.3.3 Déroulement . . . 98
6.3.4 Présentation du questionnaire . . . 99

6.3.5 Résultats . . . 101
6.3.6 Discussion . . . 103
6.4 Bilan du chapitre . . . 104

7 Conclusion Générale 105
7.1 Résumé de la proposition . . . 106
7.1.1 Le contexte . . . 106
7.1.2 La construction de la stratégie . . . 106
7.1.3 L'évaluation du modèle . . . 108
7.2 Limites et Perspectives . . . 108
7.2.1 Les évolutions du modèle et du moteur . . . 108
7.2.2 Les modules annexes . . . 111
7.2.3 La place du game designer . . . 112
7.3 Bilan . . . 112

A Publications 117

B UML 119

C Exemples de fichiers 121
C.1 Exemple de stratégie en JSON . . . 121
C.2 Exemple de fichier XML . . . 123

D Test utilisateur 125
D.1 Questionnaire . . . 125
D.2 Réponses . . . 126

E Evaluation 135
E.1 Explications fournies . . . 135
E.1.1 Evaluation et jeu . . . 135
E.1.2 XML . . . 137
E.1.3 Modèle et éditeur de stratégie . . . 139
E.2 Détails des résultats . . . 141
E.2.1 Questions générales . . . 141
E.2.2 SUS . . . 143
E.2.3 Expressivité . . . 143

Bibliographie 145

Liste des figures

2.1 Éléments du modèle de jeu de Cook (2007) 9
2.2 Résultats de l'évaluation de Nielsen et al. (2014) 10
2.3 MDA (Hunicke et al., 2004) 11
2.4 Le cube des joueurs de Bartle (Bartle, 2005) 12

3.1 Les niveaux de la prise de décision selon Beaufre (1964) et Fiévet (1992) 24

4.1 Exemples d'architectures IA ayant participé à des compétitions Starcraft (Ontanón et al., 2013) 41
4.2 Les différents modules qui composent EISBot (McCoy and Mateas, 2008) 42

5.1 Organisation des modules 54
5.2 Exemple d'une hiérarchie de comportements 57
5.3 Les états de notre exemple de comportement logique 58
5.4 Les transitions sur l'état du contexte de notre exemple de comportement logique 60
5.5 Les transitions sur le résultat du combat 60
5.6 Les transitions sur les résultats enrichis 61
5.7 Les états enrichis permettent d'envoyer des messages au comportement supérieur 62
5.8 L'utilisation de l'importance dans la répartition des ressources 64
5.9 La hiérarchie de ressources 65
5.10 La mise à jour de la stratégie 70
5.11 Schématisation de l'allocation des ressources dans un comportement parallèle . . 75

6.1 L'interface du jeu développé pour les évaluations 84
6.2 La stratégie sous forme de graphe affichée dans l'outil développé 89
6.3 Édition d'un état dans un comportement logique 90
6.4 Une FSM affichée dans l'outil développé 91
6.5 Un comportement parallèle affiché dans l'outil développé 91
6.6 Répartition des participants dans les groupes de test 99
6.7 Affirmations du formulaire SUS 100
6.8 Résultats des tests de student sur le questionnaire SUS 102
6.9 Résultats des tests de student sur le respect des directives 102

Liste des tables

5.1 Exemple de comportement parallèle 66

6.1 Résultats des questions associées à la transparence 94
6.2 Résultats des questions associées à l'expressivité 95

Chapitre 1

Introduction

Cette thèse s'intéresse à la création des intelligences artificielles (IA) contrôlant des adversaires dans les jeux vidéo de stratégie à visée commerciale. Pour cela nous nous intéressons à leur place au sein du jeu et à la manière dont elles sont conçues et développées dans le processus de création du jeu. Les jeux sont souvent utilisés en recherche comme un environnement de test. Par exemple l'entreprise Deepmind a récemment utilisé le jeu de Go pour démontrer les capacités du deep learning avec AlphaGo, première IA à battre l'un des meilleurs joueurs de Go. Cependant les résultats obtenus ne sont pas adaptés pour des jeux commerciaux : Fan Hui, considéré comme le meilleur joueur d'Europe de Go et battu cinq fois de suite par AlphaGo, a expliqué par la suite s'être senti face à un mur lorsqu'il jouait[1]. En effet, les jeux de stratégie comme le jeu de Go résident sur une exploitation des faiblesses et erreurs de l'adversaire, ainsi que sur du bluff sur ses propres capacités et intentions. Face à une IA experte et n'étant pas sensible au bluff, le jeu perd donc de son intérêt pour le joueur puisque la victoire semble impossible. Lors de la création d'un jeu commercial, c'est l'ensemble du jeu qui est réfléchi par les game designers pour rendre l'expérience vécue par le joueur divertissante. Les IA doivent donc être créées pour participer à cette expérience, et nous proposons alors aux game designers d'exprimer des stratégies qui viennent guider les prises de décision des IA de façon à répondre aux besoins de l'expérience.

L'expérience du jeu

L'objectif d'un jeu commercial est de proposer une expérience divertissante pour le joueur. En effet un jeu est par définition une expérience autotélique, c'est-à-dire possédant une motivation intrinsèque : le joueur ne joue que pour le plaisir de jouer et ne jouera pas si l'expérience proposée ne procure pas de plaisir. Il est difficile de comprendre les éléments qui permettent de créer une telle expérience, mais plusieurs études ont essayé de les définir, de rationaliser ce qui est appelé le *fun* et de comprendre ce qui plaît et attire les joueurs. La plus célèbre de ces études est certainement celle de Nakamura and Csikszentmihalyi (2002) qui expliquent le plaisir

[1] http://www.nature.com/news/go-players-react-to-computer-defeat-1.19255 (accédé le 6 février 2017)

par un état maximal de concentration et d'engagement dans l'activité effectuée qu'ils nomment le *flow*. Les auteurs se sont dans un premier temps intéressés à la motivation intrinsèque et le domaine des jeux a donc servi à réaliser les premières évaluations. Bien qu'ensuite étendue aux activités à motivation extrinsèque, le flow reste une référence lorsque le sujet du fun dans les jeux est abordé. Le flow est souvent expliqué par un niveau de difficulté bien équilibré : ni trop dur pour éviter de décourager le joueur, ni trop facile pour ne pas l'ennuyer. On retrouve dans cette notion l'idée d'une partie dont on ne peut pas connaître le résultat avant la fin : la victoire comme la défaite doivent être des options possibles. D'autres études ont également essayé de comprendre ce qui se cache derrière le fun, et ce qui en ressort est la définition de différents types de plaisirs, ne parlant donc plus du fun comme d'une idée unique mais comme l'assemblage de plusieurs composantes. Ces études mettent également en avant le fait que les préférences en terme de fun ne sont pas les mêmes pour tout le monde et que les jeux associent en général plusieurs types de fun en fonction de l'audience visée. Le jeu des mimes est par exemple décrit comme rassemblant du challenge, du social et de l'expression par Hunicke et al. (2004).

Le rôle d'un game designer

Lors du développement d'un jeu, ce sont les game designers qui sont les architectes de l'expérience proposée au joueur et qui vont décider des aspects importants du jeu. Les IA qui contrôlent les adversaires étant là pour contribuer à cette expérience, ce sont les game designers qui sont également responsables de leur conception, de façon à créer un ensemble cohérent. Cependant, le game designer délègue la création concrète du jeu tout en gardant un rôle de superviseur afin de garantir l'obtention de l'expérience désirée. Concernant l'implémentation des IA, ce sont donc des développeurs qui en ont la charge et il est nécessaire que les game designers puissent transmettre efficacement leurs idées aux développeurs afin d'obtenir le résultat désiré. Actuellement la majorité de ces idées sont expliquées par oral ou écrit sans outil ni format approprié. Les machines à états finis, bien que rarement utilisées lors de l'implémentation, le sont parfois pour communiquer les idées du game designer car elles proposent un modèle facile à comprendre et à prendre en main. La qualité de la communication est essentielle car une mauvaise compréhension peut engendrer de nombreuses itérations de développement et de test ; or le temps est une ressource précieuse lors du développement d'un jeu, il est donc important de ne pas en perdre inutilement.

Les jeux 4X

Pour cette étude, nous nous intéressons aux jeux vidéo de stratégie dits *4X* qui proposent des environnements complexes rendant d'autant plus difficile la communication évoquée ci-dessus et dans lesquels la notion de stratégie est centrale et particulièrement riche. Dans ces jeux, le joueur possède une civilisation qu'il va devoir développer. Cette civilisation est en compétition avec des adversaires évoluant dans le même environnement, dont le joueur va devoir tenir compte et avec lesquels ils va devoir interagir. Le nommage *4X* provient de quatre mots décrivant les

grands mécanismes qui sont au centre de ce type de jeu : l'exploration, l'exploitation, l'expansion et l'extermination. L'exploration reflète le fait que la carte qui constitue l'environnement du jeu est partiellement visible, le joueur doit utiliser des unités de sa civilisation en les déplaçant pour découvrir les endroits inconnus et observer ce qui s'y trouve. L'exploitation correspond à l'utilisation des ressources à disposition dans le but de réaliser l'expansion. Cette dernière peut être géographique, via la construction de ressources physiques qui déterminent le territoire de la civilisation, mais elle peut également être de nombreuses autres natures : religieuse, culturelle, politique... L'extermination reflète l'aspect militaire de ce type de jeux, en effet les conditions de victoire les plus courantes sont l'acquisition d'un territoire assez grand et la destruction de l'ensemble des adversaires. Cependant nous préférons associer l'extermination aux conditions de victoire qui sont souvent plus variées et correspondent à atteindre un certain stade de domination par l'une des expansions précédemment évoquées, montrant ainsi la diversité du champ d'action de ces jeux.

La gestion des ressources est centrale dans ces jeux car les ressources déterminent les actions qui peuvent être effectuées. Ces univers proposent un nombre important d'actions possibles, un univers virtuel complexe et un nombre de ressources qui devient au cours du jeu rapidement grand. Toutes ces caractéristiques associées se traduisent par une grande liberté d'action proposée au joueur. Cette dernière permet au joueur d'exprimer sa créativité et sa personnalité. En effet au cours du jeu le joueur devra choisir parmi les actions que les ressources qu'ils possèdent lui permettent de faire, et ces actions lui permettront d'acquérir de nouvelles ressources. Certaines de ces ressources sont généralistes, c'est-à-dire qu'elles peuvent servir pour de nombreuses actions différentes, et d'autres sont plus spécialisées et les acquérir implique d'avoir fait des choix sur les domaines à privilégier. Afin d'être dominant dans l'un des domaines et de pouvoir ainsi remporter la partie, il est donc nécessaire de faire des concessions et de renoncer à être performant sur d'autres domaines. Les IA proposées par ce type de jeux reflètent la complexité de ces décisions : au delà du choix du niveau de difficulté plusieurs personnalités sont proposées. Civilization, jeu emblématique du type 4X, propose par exemple de se confronter à Gandhi, qui a une approche diplomatique pacifique, ou encore à Attila, qui à l'inverse aura une tendance très agressive. Cette liberté d'action se traduit également par de longues parties de plusieurs heures rendant difficiles à mettre en place des parties multi-joueurs et rendant d'autant plus importante la qualité des IA fournissant des adversaires lors des parties solo.

La problématique

Cette thèse CIFRE a été réalisée dans l'entreprise MASA Group dont le cœur de métier est de rendre la conception de l'IA accessible à tous. Cette thèse a été proposée afin de répondre au besoin d'une nouvelle méthode de conception d'IA destinée à un raisonnement stratégique. Pour répondre à cette demande nous avons décidé de nous intéresser plus particulièrement aux IA contrôlant les adversaires dans les jeux de stratégie dits 4X. L'objectif de cette thèse est donc de proposer une méthode permettant d'améliorer leur conception. Pour cela nous proposons de

réintroduire le game designer au coeur de la création des IA car actuellement la conception et l'implémentation des IA sont réalisées par des personnes différentes : les game designers et les développeurs. Cela pose problème car la communication est source de pertes d'information et de malentendus. En fournissant un contrôle direct pour les game designers sur l'implémentation des IA, nous souhaitons réduire la perte de temps liée à ces erreurs de communication.

Comme l'explique Isla (2005), il n'est pas nécessaire de fournir aux game designers le contrôle sur l'intégralité de l'IA et leur permettre de donner des indications haut-niveau sur la prise de décision suffit. Nous avons donc cherché à définir la nature de ces indications haut-niveau et nous nous sommes intéressés au découpage hiérarchique de la prise de décision et plus précisément à ce qui compose le niveau le plus haut : la stratégie. Pour cela nous avons effectué un état de l'art sur la stratégie dans son utilisation extérieure aux jeux, et en particulier dans le domaine militaire et celui de l'entreprise, puis sur son utilisation dans les jeux de stratégie.

Un game designer ne possède pas toujours de compétences en programmation, il est donc nécessaire de leur proposer un modèle qui soit **intelligible** pour qu'il puisse le comprendre et le manipuler. L'intelligibilité dans le cadre de la création d'une IA est définie par Isla (2008) par la transparence, c'est-à-dire que le lien entre l'IA créée et le résultat observé doit se faire intuitivement pour le game designer : il doit pouvoir prédire et comprendre les résultats obtenus et doit être capable d'effectuer les modifications nécessaires pour changer le comportement si celui observé n'est pas celui voulu. Nous avons alors réfléchi à la manière de présenter les stratégies aux game designers afin qu'ils puissent les comprendre et les manipuler avec une formation minimale. Afin que l'intervention d'un développeur ne soit pas nécessaire, nous avons également dû faire en sorte que le modèle soit **interprétable** informatiquement et nous avons donc réfléchi à la manière dont une stratégie serait exécutée au niveau algorithmique. A cela vient s'ajouter le besoin d'**expressivité** qui provient du besoin de limiter ici aussi l'intervention d'un développeur. Pour cela nous devons permettre au game designer de définir des stratégies diversifiées et riches en informations qui permettent d'altérer de différentes manières le comportement produit. De cette manière il pourra apporter les modifications souhaitées par le biais de notre modèle sans avoir besoin de faire appel à un développeur.

Nous pouvons synthétiser notre problématique de la manière suivante : nous souhaitons proposer une méthode de création d'IA pour les jeux de stratégie dits 4X qui réintègre le game designer dans le processus en lui fournissant un modèle haut-niveau qui est à la fois **intelligible**, **interprétable** et **expressif**.

La proposition

Une stratégie s'applique généralement sur des organisations de grande taille qui présentent une décomposition hiérarchique. Une stratégie permet alors d'apporter une cohérence à la prise de décision distribuée qui en découle. La gestion des ressources entre les différentes décompositions et la gestion de l'adaptation au contexte hautement dynamique ont été déterminées comme des besoins critiques sur lesquels la stratégie agit. A partir de ces éléments et en nous inspi-

rant de modèles graphiques d'IA nous avons créé un modèle permettant aux game designers de définir une stratégie composée de paramètres répondant à ces besoins. Nous avons alors abouti à la définition du modèle de stratégie présenté dans ce manuscrit qui utilise le découpage hiérarchique pour simplifier les problèmes à résoudre et pour permettre de raisonner sur des domaines différents en parallèle. Pour rendre cette structure hiérarchique adaptable au contexte dynamique, nous utilisons les machines à états finis (FSM) accompagnés d'un système de messages. Pour répartir les ressources dans cette structure hiérarchique nous proposons une définition de paramètres venant influencer à la fois les quantités et la nature des ressources allouées.

Le modèle fourni au game designer lui permet de définir une prise de décision stratégique, il est donc nécessaire de le faire fonctionner conjointement avec des modules complémentaires implémentés par les développeurs d'IA et s'occupant de la prise de décision à un niveau plus bas. Nous avons donc créé un moteur de stratégie qui utilise une stratégie définie par notre modèle pour prendre des décisions. Nous avons également défini l'interaction nécessaire entre notre module stratégique et les modules IA complémentaires afin de leur transmettre les décisions mais également pour récupérer les informations nécessaires à la prise de décision.

L'organisation du manuscrit

Le chapitre 2 qui suit cette introduction revient sur l'objectif de divertissement d'un jeu ainsi que sur le rôle du game designer. Le chapitre se poursuit par une étude des méthodes qui peuvent être mises en place pour améliorer l'implication du game designer dans la création d'un jeu vidéo et plus particulièrement dans celle des IA. Ce chapitre se finit par la description des jeux 4X et de leur pertinence pour notre problématique.

Le chapitre 3 aborde la question de la stratégie et de sa définition. Pour ce chapitre nous avons étudié la littérature existante en dehors du domaine des jeux vidéo pour établir la définition de la stratégie dans le langage courant. Dans un premier temps nous étudions les éléments qui la composent ainsi que le contexte dans lequel elle se place. Dans un deuxième temps nous cherchons à formaliser l'objectif qu'elle sert et comment ses éléments sont organisés pour répondre à cet objectif.

Dans le chapitre 4 nous nous intéressons plus précisément à la question de la stratégie dans le domaine des jeux vidéo. Le domaine de la recherche et le domaine de l'industrie sont étudiés pour en voir les différences et avoir une vue complète du travail réalisé sur la stratégie dans les jeux vidéo.

La proposition de cette thèse est présentée dans le chapitre 5. La première partie de ce chapitre présente les différentes composantes d'une stratégie dans notre modèle : une vue générale de la stratégie, puis les paramètres permettant d'influencer l'adaptation de la stratégie en fonction du contexte et enfin ceux permettant d'influencer la répartition des ressources. La deuxième partie présente le moteur de stratégie et de quelle manière chaque paramètre est utilisé au cours du jeu pour réaliser la stratégie. On y retrouve également la description des interactions

nécessaires avec les autres modules.

Le chapitre 6 présente les évaluations réalisées dans le cadre de cette thèse dans le but d'évaluer le modèle créé. Dans un premier temps les implémentations nécessaires à la réalisation de ces évaluations sont décrites : un outil permettant de créer une stratégie ainsi qu'un petit jeu de stratégie permettant de les mettre en oeuvre. Un test utilisateur puis une évaluation ont été réalisés : le déroulement de chacun est décrit, puis les résultats sont présentés et discutés.

Enfin dans le chapitre 7 nous résumons brièvement la proposition présentée dans ce manuscrit puis nous en étudions les limites et perspectives.

Chapitre 2

La création d'un jeu vidéo

Le marché du jeu n'existe que grâce à l'envie des joueurs, en effet rien n'oblige l'achat d'un jeu. Le jeu est un produit consommé par un utilisateur et qui répond à un besoin. Mais le jeu a de particulier qu'il répond à un besoin de divertissement, celui-ci étant difficilement définissable et quantifiable. La seule raison qui va pousser une personne à acheter un jeu est donc le plaisir qu'il va en retirer et la valeur qu'il accorde à ce plaisir. Dans un premier temps, nous allons donc voir plus en détail l'expérience vécue par le joueur lors de l'utilisation d'un jeu et comment la rendre divertissante. Nous y ferons référence sous le terme de *fun* qui est fréquemment employé dans la littérature pour caractériser l'expérience recherchée d'un jeu. Nous allons ensuite nous intéresser à comment améliorer la participation du game designer dans la création des IA. Pour ce deuxième point nous allons essayer de comprendre son rôle et les problèmes qu'il peut rencontrer lors du développement d'un jeu vidéo et nous allons explorer les premiers travaux qui ont déjà cherché à faciliter sa tâche. Dans un dernier temps nous allons présenter les jeux 4X qui seront notre environnement de recherche : nous allons les définir, comprendre leurs caractéristiques et décrire ce qui les rend pertinents pour notre question de recherche.

2.1 L'expérience du joueur

Pour cette première section nous nous intéressons au plaisir ressenti par un joueur lors de l'utilisation d'un jeu, c'est-à-dire à l'expérience qu'il construit pendant cette utilisation. Nous utilisons ici le même angle d'approche que Triclot (2011) et Pettersson (2013) lorsqu'ils décrivent respectivement le jeu comme une expérience ou une activité. Lors du travail de création d'un jeu la finalité est de rendre l'expérience produite la plus divertissante possible.

Pour cela une des particularités du jeu est de créer un univers alternatif, dans le cas du jeu vidéo cet univers est retransmis via l'ordinateur. Les personnages non-joueurs font partie intégrante de cet univers et viennent y contribuer. Ces personnages dirigés par une intelligence artificielle peuvent prendre différents rôles dans ce monde : ils peuvent avoir un rôle neutre dont le but est simplement de peupler l'environnement ou de déclencher des événements, ils peuvent apporter une aide aux joueurs, ou bien ils peuvent être des obstacles à la progression du joueur.

Tous ces types de personnages viennent contribuer au fun pour améliorer l'expérience du joueur. Lors de leur création il est donc important de prendre en compte l'expérience dans son ensemble.

Le niveau de difficulté de la partie est une composante reconnue du fun et dans laquelle les IA jouent un rôle primordial. Nous allons donc voir dans un premier temps quelques travaux qui se sont intéressés à cet aspect. Puis nous allons voir des travaux qui ont montré une vision plus complète du fun en essayant d'en détailler les différentes facettes.

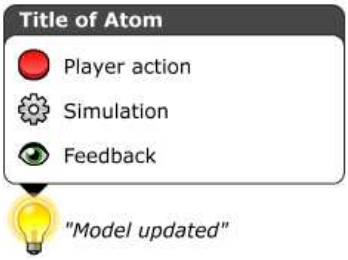
2.1.1 Le niveau de difficulté

Un des points souvent relevés dans la qualité de l'expérience du joueur est la notion d'apprentissage. C'est notamment le point central relevé par Koster (2004) dans son livre "A Theory of Fun". Jouer serait la recherche d'une performance parfaite, d'une maîtrise complète que le joueur atteindrait par la découverte du jeu, en apprenant les différents mécanismes et en perfectionnant à la fois sa connaissance du jeu mais également les contrôles ou gestes à effectuer. On retrouve cette vision dans les premiers jeux vidéo créés dans les universités par des "hackers". Par exemple *SpaceWar*[1] nécessitait de comprendre le principe d'attraction exercée par une étoile puis de le maîtriser pour attaquer les autres joueurs (Triclot, 2011). Le gagnant était alors la personne utilisant le mieux le mécanisme et l'expérience était valorisée.

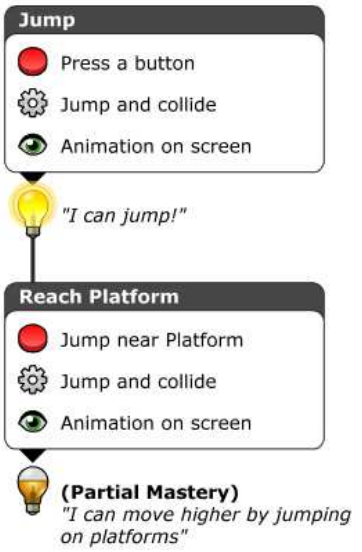
Cook (2007) a voulu formaliser le game design d'un jeu en se concentrant sur l'apprentissage suivi par le joueur. Il a ainsi souhaité fournir un outil d'aide au game design en partant de l'hypothèse que le joueur ne s'amuse que lorsqu'il a quelque chose à découvrir et à apprendre. Il construit ainsi une carte des compétences que le joueur peut découvrir et apprendre au cours du jeu. Chaque compétence est décrite par l'action que doit effectuer le joueur, la réaction provoquée dans le moteur de jeu, et le feedback retourné au joueur afin qu'il comprenne ce qu'il vient de découvrir, comme le montre la figure 2.1a. Ces compétences sont alors liées par l'ordre dans lequel elles sont découvertes. Par exemple dans la figure 2.1b on peut voir que le joueur doit d'abord apprendre qu'il peut faire sauter le personnage qu'il contrôle avant de pouvoir apprendre qu'il peut atteindre les plateformes en hauteur grâce à la première compétence. Ces découvertes peuvent être dans plusieurs états durant le jeu : soit non découverte, soit découverte et utilisée (avec plusieurs phases de maîtrise de la compétence associée), soit découverte mais abandonnée. L'abandon d'une compétence entraîne l'impossibilité de découvrir les compétences qui en découle comme le montre la figure 2.1c. Le modèle peut alors être utilisé pour tester et diagnostiquer un jeu en étudiant quelles compétences sont délaissées par le joueur et celles qui ne sont jamais découvertes. On peut alors vérifier que la progression dans la découverte des compétences est fluide, et que les compétences les plus difficiles à trouver et maîtriser ne bloque pas la découverte d'une trop grande partie de la carte.

Les IA opposantes sont des personnages centraux dans cette notion d'apprentissage puisque suivant la qualité de leurs décisions, l'apprentissage sera plus au moins difficile pour le joueur. La sélection d'un niveau de difficulté par le joueur en début de jeu vient souvent impacter leur

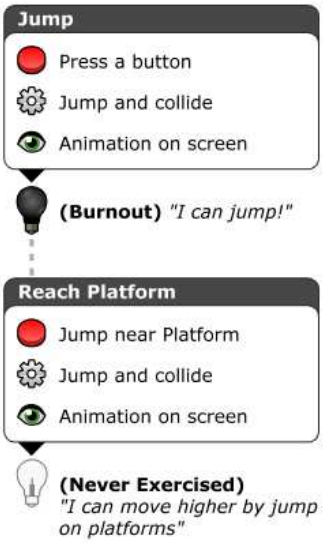
[1] http://museum.mit.edu/150/25 (accédé le 6 février 2017)



(a) Description d'une compétence



(b) Lien de découverte entre deux compétences



(c) Lien mort entre deux compétences

FIGURE 2.1: Éléments du modèle de jeu de Cook (2007)

prise de décision. Cependant plusieurs études se sont intéressées à la modulation automatique du niveau de difficulté afin de l'adapter à l'évolution des compétences du joueur tout au long du jeu. Andrade et al. (2005) couple par exemple un système d'apprentissage par renforcement pour construire une importante collection de comportements avec un mécanisme de sélection d'action qui utilise cette collection afin de choisir un comportement au niveau approprié au joueur.

Cependant il serait réducteur de résumer le fun à la simple notion d'apprentissage, et si le fun peut être amené de manière différente, l'IA peut certainement y contribuer de manière différente également.

2.1.2 La multiplicité de l'expérience

Nous allons voir dans cette partie les travaux qui ont voulu étendre cette notion de fun à d'autres aspects que la difficulté. Anthony et al. (2014) évoquent par exemple la "beauté" dans un jeu sans pour autant la définir. Au contraire, ils appuient le manque de définition en précisant l'impossibilité de créer une heuristique pour obtenir cette beauté. Cependant cette étude part de l'hypothèse que les comportements doivent être réalistes, dans le sens où ils doivent paraître humains, or cette hypothèse est réfutée par les évaluations qui ont été réalisées par Nielsen et al. (2014). Ces derniers ont en effet réalisé des évaluations comparant plusieurs méthodes de création d'IA en fonction de plusieurs aspects.

Les méthodes comparées sont les FSM, un algorithme de recherche MinMax, un algorithme Monte Carlo Tree Search (MCTS), une méthode développée par les auteurs inspirée de travaux sur les champs de potentiel (PF) et un algorithme NEAT (Neural Evolution of Augmenting Technologies). Afin de réaliser les évaluations, des volontaires ont dû jouer plusieurs parties contre les IA créées à partir de ces méthodes. Il leur était ensuite demandé de donner trois notes pour chacune : l'intensité du challenge proposé par l'IA, le degré de similarité avec le comportement d'un joueur humain, et le plaisir ressenti durant le jeu. Les résultats obtenus pour ces trois notes ainsi que le taux de réussite de l'IA sont ceux présentés sur la figure 2.2. Ce qui nous intéressent ici ce n'est pas les résultats individuels de chaque méthode mais l'absence de corrélation entre les trois notes. Nous pouvons notamment voir que la technique la plus appréciée était les FSM, avec lesquelles on obtenait pourtant le comportement le moins ressemblant à un comportement humain. L'hypothèse qu'un comportement fun doit être réaliste est donc fausse.

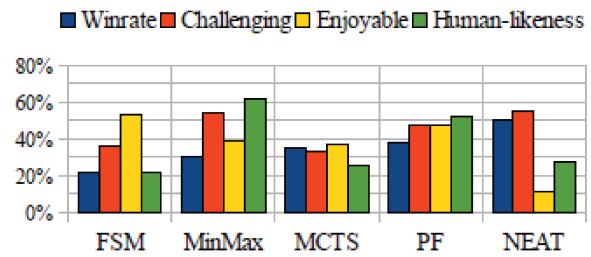


FIGURE 2.2: Résultats de l'évaluation de Nielsen et al. (2014)

Si les résultats n'excluent pas l'influence de la difficulté proposée ou du réalisme sur le

plaisir ressenti par le joueur, ils supposent en revanche que d'autres paramètres entrent en considération. Plusieurs autres études ont cherché à expliquer davantage ces paramètres qui rendent un jeu divertissant. Nous en présentons ici deux qui proposent des approches différentes mais complémentaires : le formalisme MDA proposé par Hunicke et al. (2004) et notamment les différentes catégories d'expérience du joueur qu'ils ont définies, et les catégories de joueurs définies par Bartle (2005). Ces deux études mettent en lumière un point repris par toutes les études sur le fun dont nous avons eu connaissance : la multiplicité de l'expérience.

L' "esthétique" dans le formalisme MDA

Le formalisme MDA décrit par Hunicke et al. (2004) définit trois niveaux selon lesquels on peut observer un jeu : la mécanique, la dynamique et l'esthétique. Chaque niveau se situe plus ou moins proche du joueur tel que décrit sur la figure 2.3. La mécanique d'un jeu correspond à la vision du designer, c'est le niveau le plus éloigné du joueur. On y retrouve les contrôles élémentaires donnés au joueur. L'aspect esthétique est le plus proche du joueur, il correspond à l'expérience vécue par le joueur pendant la partie. La dynamique d'un jeu se situe entre les deux, elle fait le lien en exprimant l'utilisation des actions associée au sens qui est donné par le joueur lors de son utilisation.

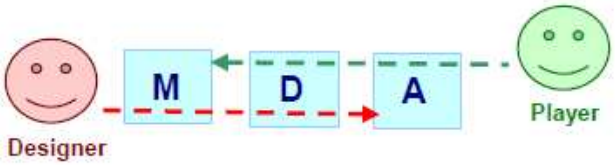


FIGURE 2.3: MDA (Hunicke et al., 2004)

Les auteurs ont décrit plus précisément le dimension esthétique en essayant de catégoriser les différentes expériences qui peuvent être recherchées par le joueur. Ces différentes catégories ne sont pas exclusives, au contraire un jeu est en général considéré comme faisant partie de plusieurs catégories. Huit catégories ont été mises en évidence :

la sensation Le jeu est vu comme un plaisir des sens

la "fantasy" Le jeu est vu comme un univers imaginaire

la narration Le jeu est vu comme une histoire

le challenge Le jeu est vu comme une course d'obstacles

la camaraderie Le jeu est vu comme un espace d'échange social

la découverte Le jeu est vu comme un univers inexploré

l'expression Le jeu est vu comme une découverte de soi

la soumission Le jeu est vu comme passe-temps

Ces catégories représentent des plaisirs différents qui peuvent être procurés au joueur par l'action de jouer.

Les différents joueurs selon Bartle

Bartle (2005) décrit une autre approche du fun : au lieu de se concentrer sur l'expérience, il se concentre sur le joueur en utilisant un questionnaire pour essayer de les classifier suivant l'utilisation qu'ils font d'un MUD (multi-user dungeon)[2]. Il aboutit ainsi à une classification des joueurs en quatre groupes suivant deux axes : un axe interaction/action et un axe autres joueurs/monde. Les quatre groupes obtenus sont les suivants :

- les "sociaux" interagissent avec les autres joueurs,
- les "tueurs" agissent sur les autres joueurs,
- les "explorateurs" interagissent avec le monde,
- et ceux qui sont à la recherche du meilleur résultat ("achievers") agissent sur le monde.

Par la suite un axe a été rajouté : l'axe implicite/explicite. Les groupes ont alors été entièrement renommés pour intégrer cette nouvelle notion, montant le nombre de catégories de joueurs à huit. On peut alors montrer l'espace des profils possibles dans un cube tel que la figure 2.4 le montre.

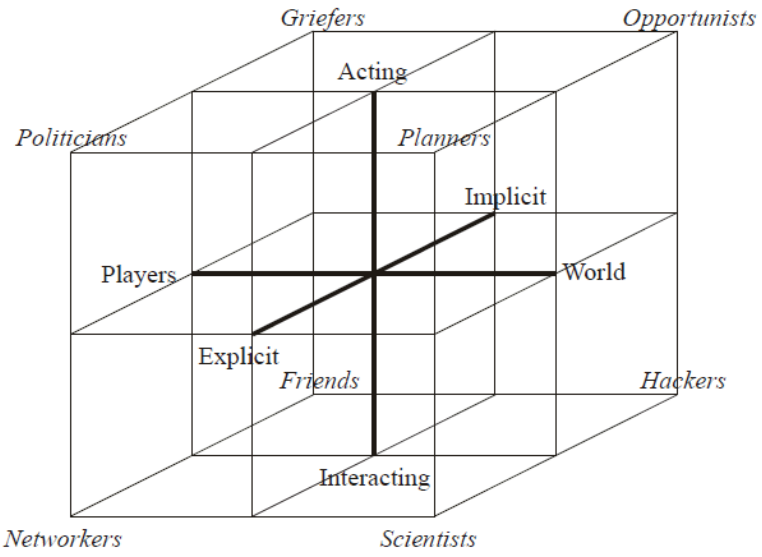


FIGURE 2.4: Le cube des joueurs de Bartle (Bartle, 2005)

Bartle (2005) suggère également que les joueurs ont tendance à changer de profil au fil de la découverte du jeu et de l'apprentissage. Cela signifie que non seulement les joueurs n'ont pas les mêmes attentes de l'expérience de jeu, mais qu'en plus ces attentes évoluent également pour un même joueur au cours de l'utilisation du jeu.

[2]Un MUD est un jeu de rôle textuel en ligne auquel plusieurs joueurs peuvent participer. Le but est d'explorer, de tuer des monstres et de réaliser des quêtes.

2.1.3 Conclusion

Les classifications présentées par Hunicke et al. (2004) et Bartle (2005) mettent en avant la multiplicité des expériences de jeu de part la nature du jeu qui reflètent également des envies différentes selon les joueurs. Si le niveau de difficulté doit pouvoir être ajusté au joueur, l'expérience proposée par un jeu vidéo va au-delà de simplement proposer un challenge. Un jeu sera imaginé en fonction des joueurs visés, et il sera d'autant plus difficile de le créer que l'expérience est évolutive comme l'a montré Bartle (2005). Ce n'est donc pas une expérience qui est proposée au joueur, mais plusieurs expériences. Aucune formalisation du fun ne semble égaler les connaissances qu'un game designer accumule par expérience. Cette définition floue du fun et de l'expérience de jeu explique également en partie la difficulté à communiquer les envies d'un game designer aux développeurs, il semble donc indispensable de donner au game designer un accès direct à l'implémentation des IA. Nous allons donc désormais nous intéresser de plus près à son rôle et aux recherches qui ont étudié comment assister le design d'un jeu.

2.2 L'expérience du game designer

Le game designer est l'architecte de l'expérience du joueur et imagine le jeu qui répond à cette expérience avec une vue globale. Cependant le game designer divise la création en plusieurs tâches qu'il délègue, comme la création des IA qui est assignée à des développeurs, parfois spécialisés, et c'est au game designer de veiller à ce que le jeu créé réalise effectivement cette expérience. Nous séparons ici la conception qui est la définition théorique du jeu effectuée en majorité par les game designer, et la création qui est la réalisation informatique du jeu qui a été imaginé en phase de conception et qui est donc déléguée à une équipe de création. Isla (2005) témoigne des problématiques rencontrées lors de la mise en place de l'IA d'un jeu commercialisé à large échelle dans le contexte de la création d'Halo II. Selon lui, l'IA doit faciliter l'histoire qui se construit dans l'esprit du joueur. En effet le joueur va avoir tendance à personnifier une IA en lui associant des intentions derrière ses prises de décisions, dans le but de construire une histoire autour de ses différents personnages. Cette histoire qui émerge des comportements des IA doit répondre à l'expérience globale fournie par le jeu, il est donc essentiel que les IA d'un jeu ne soient pas considérées de manière indépendante mais en relation avec les autres parties du jeu qui doivent fonctionner ensemble pour créer l'expérience voulue. C'est dans cette optique que le game designer supervise la création des IA.

2.2.1 L'assistance du game designer

Pour garantir la réponse des IA aux besoins de l'expérence de jeu, le game designer doit réussir à retransmettre l'univers global du jeu et ce que les IA peuvent y apporter. Cependant, les demandes du game designer sont souvent communiquées par des discussions et des échanges écrits sans véritable formalisme adapté destiné à faciliter cette communication. Ceux parfois

utilisés sont les FSM[3] et les Behavior Trees[4](BT), des formalismes graphiques et simples à prendre en main mais qui peuvent rapidement être limités par leur expressivité. C'est cette communication que nous souhaitons remplacer dans le cadre de comportements complexes, du moins partiellement, en fournissant un contrôle direct aux game designers.

Isla (2005) précise, au sujet de son expérience sur Halo II, qu'il n'est pas intéressant pour le designer de définir la totalité de ce que doit faire une IA car ce serait trop complexe. Il est en revanche intéressant de lui donner un contrôle haut-niveau sur la direction générale, par exemple en ordonnant à l'IA d'agir de façon agressive ou de façon peureuse. On retrouve l'idée déjà exploitée par Prigent et al. (2009) et Booth et al. (2015) d'un découpage hiérarchique dont seulement le contrôle supérieur serait donné aux designers. Cela nécessite d'interfacer le module fournie au game designers avec des modules IA plus bas-niveau dont le développement seraient laissé aux développeurs.

Hecker (2008) a imaginé permettre la création des IA par un logiciel de type *photoshop* qui formalise l'IA et simplifie sa création et manipulation. Pour cela il distingue la structure du style. La première, la structure, est la représentation informatique que prend l'IA, elle doit permettre des calculs complexes, une utilisation riche pour le raisonnement. Le deuxième, le style, doit proposer une grande expressivité pour les utilisateurs afin qu'ils ne soient pas limités par l'outil dans leur création. Il fait ainsi le parallèle avec les logiciels graphiques qui proposent de nombreux outils qui accélèrent la création de contenu. Cette approche met en lumière l'importance de la liberté de deux points de vue qui peuvent s'opposer : celui du développeur qui va souhaiter pouvoir manipuler les données le plus librement possible, et celui de l'utilisateur qui va souhaiter pouvoir exprimer toutes ses envies par le biais de l'outil.

Des outils permettant de simplifier la création des jeux en général existent déjà. Ils sont très répandus dans la création graphique avec de nombreux moteurs de jeu incluant des éditeurs permettant de créer les environnements graphiques ainsi que des moteurs physiques permettant de faire fonctionner ces environnements (e.g. *Unity*[5], Unreal Engine[6]). Il existe également des magasins virtuels proposant souvent des plugins permettant d'ajouter des outils au produit de base pouvant avoir de nouvelles fonctionnalités. C'est le cas par exemple de l'AssetStore[7] de Unity qui en plus de fournir des modèles 3D, des animations ou encore des sons, proposent également des scripts qui viennent compléter les fonctionnalités de Unity selon les besoins de chacun. Lorsque l'on s'intéresse à ceux répondant à des besoins en IA, on y trouve des frameworks destinés à des développeurs, et des interfaces graphiques généralistes permettant de définir des règles simples de transition entre deux comportements s'inspirant souvent des BT et des FSM. A cela s'ajoutent des outils dédiés à des problèmes spécifiques tels que les dialogues et le pathfinding. Nous n'avons pas trouvé d'outils s'intéressant spécifiquement à la création de comportements

[3] `http://aigamedev.com/open/article/fsm-reusable/` (accédé le 6 février 2017)

[4] `http://aigamedev.com/open/article/bt-overview/` (accédé le 6 février 2017)

[5] `https://unity3d.com/fr` (accédé le 6 février 2017)

[6] `https://www.unrealengine.com/` (accédé le 6 février 2017)

[7] `https://www.assetstore.unity3d.com/en/` (accédé le 6 février 2017)

stratégiques complexes.

2.2.2 Les contraintes de l'assistance en IA

Lors de la création d'IA, le contrôle reste un point important dans l'industrie du jeu vidéo. Malgré de nombreuses recherches sur la génération automatique de contenu et l'apprentissage, l'industrie du jeu reste assez réfractaire à ce type de méthode à moins qu'elle ne soit au centre des mécaniques de jeu. Minecraft[8] utilise par exemple la génération procédurale pour un monde où l'exploration et la construction infinie sont les mécaniques centrales du jeu. Black and White[9] utilise des techniques d'apprentissage pour permettre au joueur de dompter une bête particulièrement mises en avant lors de la présentation du jeu (Evans, 2002). Les jeux se complexifient de plus en plus et offrent des environnements avec davantage de détails et de possibilités pour le joueur. Avoir davantage de liberté devient désormais la norme. Il va donc devenir essentiel pour l'industrie du jeu de se munir d'outils facilitant la création et le contrôle de jeux toujours plus complexes puisque les outils actuels ne sont pas adaptés (Lucas et al., 2012).

Afin de garder le contrôle souhaité, Isla (2005) modère les bénéfices de l'enrichissement la prise de décision d'une IA. En effet, il note de nombreux points négatifs à ajouter toujours plus d'informations et de règles : le temps de calcul nécessaire mais également la difficulté à comprendre le résultat, à la fois lors de la création pour débugger mais également pour le joueur qui va essayer de comprendre les raisons de la prise de décision. Ces dernières sont dues au fait qu'un être humain est limité en quantité d'information pour prendre une décision par les capacités du cerveau, alors qu'en comparaison un ordinateur va pouvoir utiliser un nombre bien plus important d'informations pour choisir les actions à effectuer. Cela sert parfois à contrebalancer l'instinct humain par une puissance de calcul supérieure, mais il n'est pas utile d'utiliser trop d'information puisque le but final est que le joueur puisse lire dans les actions d'un opposant un raisonnement logique afin de construire une histoire.

La transparence définie par Isla (2008) détaille cette double lecture, par le designer et par le joueur. Le joueur doit après avoir vu une action être capable de trouver une explication à cette prise de décision car il risque sinon d'assumer que l'IA a des avantages telles que des connaissances supplémentaires et que la compétition n'est pas juste. Il doit également pouvoir anticiper un comportement ennemi en construisant dans son esprit un modèle du raisonnement de l'IA, de cette manière il a le sentiment d'évoluer dans sa maîtrise du jeu comme nous l'avons vu dans la partie 2.1.1. Le designer doit lui être capable en observant une action de comprendre les étapes qui ont amené à cette décision, prédire également une décision en fonction du contexte, être capable de savoir comment modifier un comportement et doit pouvoir reconnaître une erreur et la résoudre. Nous retrouvons ces différents besoins avec Presnell et al. (2007) qui affirment que pour créer efficacement des comportements il faut que les experts puissent créer, contrôler

[8] `https://minecraft.net/` (accédé le 6 février 2017)

[9] `https://web.archive.org/web/20160322115729/http://www.lionhead.com/games/black-white/` (accédé le 6 février 2017)

et modifier ces comportements. Ils identifient deux principaux challenges pour permettre de leur donner cet accès : la création des règles de décision par des non-développeurs (notamment la question de la programmation textuelle ou visuelle) et l'analyse des comportements (permettre, réduire, voire supprimer les nombreuses itérations de "tester et réparer").

2.2.3 Conclusion

Lors de la création d'un jeu, le game designer est le créateur de l'expérience proposée au joueur. C'est lui qui supervise le travail afin de vérifier que l'expérience proposée est bien celle voulue. Cette délégation du travail peut créer une perte de temps importante due à des boucles de test par le designer puis de modifications par le développeur, or le temps est une contrainte importante lors de la création d'un jeu. Un contrôle direct du game designer sur l'implémentation de l'IA permettrait de gagner considérablement du temps. Un contrôle haut-niveau permettant de définir une direction générale est suffisant pour un game designer. Celui-ci doit répondre à certains besoins :

- une **accessibilité** à des non-développeurs,
- une **expressivité** importante pour pouvoir créer les IA désirés,
- une **transparence** pour pouvoir faire le lien entre les comportements créés et les comportements observés lors de leur utilisation,
- une transparence également pour pouvoir apporter des modifications lors de l'observation d'un comportement inadapté ou inattendu.

2.3 Les jeux de stratégie

Hecker (2008) recommande, en raison de la diversité du domaine de l'IA, de s'intéresser à l'analyse d'un genre d'IA en particulier pour essayer de le formaliser. Etant donné notre intérêt pour le niveau stratégique nous nous somme portés sur les jeux de stratégie. Si les jeux de stratégie en temps-réel (RTS) apparaissent dans de nombreuses études, il nous a semblé plus pertinent de nous intéresser aux jeux de stratégie au tour par tour(TBS) car la complexité des univers et de l'espace de réflexion y est démultipliée. Nous avons abordé en particulier les jeux dits *4X* dont nous proposons par la suite une définition afin de présenter ce qui caractérise ce type de jeux et les points intéressants pour notre problématique.

2.3.1 La définition d'un jeu 4X

Les jeux emblématiques du genre 4X sont les jeux de la série *Civilization*[10]. Un jeu 4X oppose plusieurs joueurs qui vont devoir développer une civilisation plus rapidement que leurs adversaires. Le jeu se déroule sur une carte possédant un *brouillard de guerre*, expression utilisée

[10] https://civilization.com/ (accédé le 6 février 2017)

pour désigner la vue partielle du monde virtuel obtenue par le placement des unités sur la carte. Chaque endroit de la carte est alors soit inconnu (jamais observé), connu mais invisible (déjà observé mais aucune unité ne le voit au moment courant) ou visible (une unité est placée de façon à voir l'endroit au moment courant). Les 4X sont des jeux de stratégie au tour par tour, ce qui signifie que les joueurs jouent chacun leur tour et ont le temps qu'ils souhaitent pour choisir ses actions. Chaque tour consiste à faire bouger et agir ses unités. La carte est découpée en tuiles et le déplacement d'une unité sur un tour est limité par le nombre de tuiles et parfois par la nature des tuiles (e.g. forêt, montagne). Chaque unité peut en général effectuer une action dans le monde qui peut être principalement la récolte d'une ressource ou l'attaque d'un adversaire, mais chaque jeu ajoute ses spécificités. En plus des unités le joueur peut souvent régler plusieurs paramètres de sa civilisation dont les plus courantes concernent les productions en ressources (il faut souvent faire des choix sur les ressources que l'on souhaite obtenir à chaque tour) ou le déblocage de ressources ou d'actions par la recherche (qui nécessite également de choisir entre plusieurs options). Il peut également lui-même effectuer certaines actions notamment au niveau des relations avec les autres civilisations : proposer une trêve, proposer un cadeau... Ces actions nécessitent souvent l'utilisation de ressources.

L'objectif assez vague de *développer* une civilisation est volontaire, il illustre la liberté offerte par ce genre de jeu : on peut choisir quel aspect on souhaite développer. Un aspect reste central et a priori transversal à toute manière de jouer, c'est celui de développer la civilisation au niveau de la taille de la population et de la superficie sur laquelle s'étend le territoire du joueur. Cette liberté de jeu est souvent encouragée par de multiples conditions possibles de victoire. Par exemple, il existe dans le jeu *Civilization V* cinq conditions qui, si l'une est atteinte, permettent de gagner : la victoire par domination (qui nécessite de développer sa puissance militaire et de faire la guerre), la victoire par la science (qui nécessite de débloquer de nombreuses recherches), la victoire par la culture (qui nécessite de diversifier les politiques menées au sein de sa civilisation), la victoire par diplomatie (qui nécessite d'entretenir de bonnes relations diplomatiques avec l'ensemble des autres civilisations) et la victoire par le temps (qui nécessitent d'accumuler beaucoup de points de diverses manières). L'intérêt ici est que les conditions de victoire soient très différentes afin qu'elles s'atteignent par des styles de jeu très différents. Cela permet au joueur d'exprimer ses préférences et sa personnalité. **?** explorent les styles de jeu en proposant une modélisation d'archétypes de style de prise de décision. Ceux-ci sont représentés par un ensemble d'utilités sur des actions qui semblent caractériser les styles de jeux. Ces utilités sont ensuite utilisées pour créer les IA avec un algorithme évolutionniste.

Afin de mieux comprendre les piliers des jeux 4X, on peut observer ce que signifient les différents 'X' :

eXploration Le brouillard de guerre introduit de l'incertitude dans la prise de décision et met en avant le besoin de récolter des informations sur l'environnement et sur les adversaires.

eXpansion Les différents buts du jeu correspondent à un besoin de s'étendre qui peut se

traduire de plusieurs manières : géographiquement mais aussi scientifiquement, culturellement, religieusement...

eXploitation Afin de s'étendre le joueur doit avoir une bonne gestion des ressources puisque l'expansion est centrée autour de la récolte et l'utilisation des ressources. Les mécaniques du jeu sont centrées sur l'utilisation des ressources actuellement possédées pour en obtenir de nouvelles.

eXtermination Le jeu est une compétition et il s'agit d'être meilleur que les autres. L'aspect militaire est presque toujours présent mais il n'est pas le seul qui peut être utilisé. Ici c'est plus la compétition que nous retiendrons, qui peut se faire sur plusieurs fronts.

2.3.2 Les défis pour la recherche

Les défis proposés par les 4X sont assez proches de ceux proposés par les RTS. Les défis que proposent ces derniers ont été mis en avant par Buro (2003). Selon lui les RTS proposent des environnements complexes présentant de nombreux challenges intéressants pour la recherche. Il en énumère six : la gestion des ressources, la prise de décision dans un contexte incertain, le raisonnement spatial et temporel, la collaboration, la modélisation de l'adversaire et la planification en temps-réel. Depuis, de nombreuses recherches ont utilisé les RTS comme environnement de test pour des travaux abordant ces questions de recherche. Des compétitions Starcraft ont également fait leur apparition dans les conférences CIG et AIIDE. Ontanón et al. (2013) pointent certaines différences entre les RTS et les jeux de stratégie de plateau tels que les échecs présentant ainsi les contextes de recherche pour lesquels ils peuvent être plus intéressants. Nous retrouvons le temps-réel ainsi que l'incertitude, qu'il découpe en deux catégories : l'incertitude provenant de la connaissance partielle de l'environnement et celle provenant des actions stochastiques. A cela il ajoute l'aspect simultané des actions du joueur avec celles de l'adversaire, qui peut se rapprocher de l'aspect temps-réel ; et la complexité en terme de taille de l'environnement et du nombre d'actions disponibles. Dans le cas des 4X nous retenons les challenges suivants :

- la gestion des ressources,
- l'incertitude liée à la connaissance partielle de l'environnement,
- la modélisation des adversaires et l'incertitude liée aux interactions avec ces adversaires,
- le raisonnement spatial et temporel,
- et la collaboration.

Dans nos travaux nous allons nous intéresser plus particulièrement à la gestion des ressources et en partie à l'incertitude.

2.3.3 Conclusion

Les 4X que nous avons choisi pour nos travaux fournissent des environnements complexes dans lesquels il n'existe pas de comportement optimal notamment à cause de l'incertitude mais qui permettent de mettre en place des IA variées utilisant des styles de jeux différents. La gestion des ressources est un challenge central qui associée à la diversité des actions possibles rend le travail de communication du game designer particulièrement difficile. Cet environnement se prête également bien à la division hiérarchique de la prise de décision qui permet alors de fournir au game designer un contrôle partiel.

2.4 Bilan du chapitre

Un jeu a pour objectif de fournir une expérience divertissante au joueur à laquelle les IA vont participer. Chaque jeu va pouvoir proposer une expérience différente car il existe de nombreuses façons de créer le divertissement et chaque joueur va avoir ses propres préférences. Lors de la création d'un jeu vidéo, c'est le game designer qui est responsable de la définition de l'expérience qui va être proposée et qui va vérifier que le jeu créé correspond à cette expérience grâce à une vue d'ensemble des différentes composantes du jeu. Cependant pour créer le jeu il doit déléguer la réalisation de chaque partie et garder un rôle de superviseur qui lui permet de maintenir sa vision d'ensemble. La communication des besoins aux personnes réalisant chaque partie peut s'avérer compliquée. C'est le cas par exemple des développeurs de l'IA auxquels le game designer doit expliquer comment les IA doivent participer à l'expérience de jeu en fonction des situations qu'elles vont créer. Peu de formalismes permettent actuellement de rendre cette communication efficace et beaucoup de temps est alors perdu en raison de nombreuses boucles de développement et tests. Pour limiter cette perte de temps, nous proposons de fournir un contrôle direct au game designer sur la conception des IA. En lui fournissant un contrôle uniquement au niveau stratégique nous simplifions sa tâche tout en lui permettant l'accès aux aspects de l'IA qui l'intéressent et en laissant la réalisation de l'IA plus bas-niveau aux développeurs. Afin de réaliser nos travaux, nous avons décidé de nous intéresser aux jeux 4X qui offrent des environnements riches nous permettant de nous questionner sur la manière de rendre accessible une prise de décision a priori très complexe. Pour cela nous allons explorer davantage ce qui constitue une prise de décision stratégique en étudiant dans le chapitre 3 la définition d'une stratégie dans le langage courant puis dans le chapire 4 la signification de la stratégie dans les jeux de stratégie.

Chapitre 3

La définition commune d'une stratégie

Dans le chapitre précédent nous avons établi le besoin de proposer aux game designers un outil leur donnant la possibilité de définir le niveau stratégique de l'IA. Nous avons également établi que nous utiliserions les jeux 4X comme environnement de recherche. Dans ces jeux, l'IA va devoir contrôler une civilisation dans le but de la développer. Il nous a semblé intéressant de s'inspirer du sens commun donné au mot *stratégie* et de voir comment il peut s'appliquer dans ce cadre-là.

L'étymologie du mot *stratégie* renvoie au commandement d'une armée. Cette notion de diriger se retrouve dans l'utilisation actuelle du mot qui s'applique désormais à des domaines autres que militaire. Il est par exemple courant d'entendre parler de stratégie politique, économique ou encore de communication. Cette banalisation de l'utilisation du mot *stratégie* entraîne un flou dans sa définition qui est pointé du doigt par Lorino and Tarondeau (2006).

Dans le dictionnaire TLFi (Trésor de la langue française informatisé) nous retrouvons plusieurs définitions montrant cette absence de consensus[1]. La première reflète l'origine militaire du mot puisqu'elle se restreint à ce domaine : "Art d'organiser et de conduire un ensemble d'opérations militaires prévisionnelles et de coordonner l'action des forces armées sur le théâtre des opérations jusqu'au moment où elles sont en contact avec l'ennemi". Une autre définition, plus large, s'approche davantage de l'utilisation actuelle du mot : "Ensemble d'actions coordonnées, d'opérations habiles, de manœuvres en vue d'atteindre un but précis". De nombreuses recherches et outils abordent la question de l'analyse de l'environnement préalable aux décisions stratégiques mais il existe pourtant un manque flagrant de formalisation de la stratégie elle-même, comme le font remarquer C. Hambrick and W. Fredrickson (2005).

Notre objectif dans cette partie est de mettre en évidence une base commune aux différentes utilisations du mot *stratégie* et de mieux comprendre ce qui définit une stratégie dans l'opinion commune afin de l'appliquer au domaine du jeu vidéo de stratégie. En effet, il n'existe pas à

[1] `http://www.cnrtl.fr/definition/strat%C3%A9gie` (accédé le 6 février 2017)

notre connaissance de définition de stratégie dans le domaine du jeu, mais il existe cependant de nombreux ouvrages sur la stratégie militaire et la stratégie d'entreprise.

Dans une première partie nous allons nous intéresser au contexte dans lequel prend place la stratégie : la stratégie se définit au niveau d'une organisation, que ce soit l'entreprise ou bien l'armée. Nous allons alors essayer de définir les caractéristiques de ces organisations et les raisons qui amènent à la mise en place d'une stratégie. Dans un deuxième temps nous allons essayer de comprendre comment la stratégie est mise en place et répond aux besoins découverts dans la première partie.

3.1 La stratégie prend place dans un système...

Dans cette partie nous allons étudier les caractéristiques des organisations qui utilisent une stratégie et les problèmes pour la prise de décision qui leur sont spécifiques de façon à mieux comprendre d'où provient le besoin de stratégie.

Nous nous intéressons à deux types d'organisation : l'armée et l'entreprise. En ce qui concerne la stratégie d'entreprise, nous étudions la stratégie définie pour toute l'entreprise. La stratégie marketing ou la stratégie économique par exemple, correspondent à des sous-parties de la stratégie que nous n'aborderons pas en particulier. En ce qui concerne la stratégie militaire, elle est souvent évoquée comme faisant partie d'une stratégie plus grande qui englobe tous les fronts sur lesquels deux Etats peuvent être en compétition (politique, économique, diplomatique et militaire). Lorsque nous y ferons référence nous utiliserons le terme de *stratégie totale* proposé par Beaufre (1964). Bien que ce soit celle-ci qui nous intéresse le plus, les ouvrages abordent plus souvent l'aspect spécifique de la stratégie militaire.

Que ce soit pour la stratégie d'entreprise, totale ou militaire, nous retrouvons l'idée d'un **système**, que l'on définit comme un ensemble de personnes faisant partie d'une même organisation et fonctionnant ensemble pour le bien de cette organisation. Ces systèmes sont de **grande taille** et doivent s'organiser pour apporter une cohérence aux actions de chacun. Le bon fonctionnement du système requiert de savoir gérer les **ressources communes** du système qui peuvent parfois être utiles de plusieurs manières, ce qui nécessite de prendre des décisions. En plus de cette organisation interne, le système se situe également au milieu d'un **environnement dynamique** qui peut venir apporter des contraintes ou des opportunités. Les sections qui suivent explorent plus en détail chacun de ces aspects.

3.1.1 ... De taille importante

Un système est donc composé de plusieurs membres qui doivent fonctionner ensemble. Les actions individuelles doivent donc se compléter et ne pas s'opposer, c'est-à-dire qu'elles doivent être coordonnées. Le rôle d'une stratégie est d'assurer cette coordination comme on peut le voir dans les deux définitions du TLFi données au début de ce chapitre.

En effet, plus l'organisation est grande, plus il y aura de décisions à prendre. Afin de répartir le travail de décision et les responsabilités, les décisions sont en général distribuées. Un nombre

important de membres induit donc une prise de décision dispersée pour laquelle la coordination a du mal à être mise en place. Définir une stratégie claire est donc d'autant plus important que le nombre de membres est grand.

Afin d'obtenir une cohérence dans les prises de décision il est nécessaire d'avoir certaines règles partagées guidant la décision qui font apparaître des schémas de décisions observables au niveau du système (Mintzberg, 1978). La stratégie a alors pour rôle de garantir l'unité, symbolisant le travail commun, et la cohérence, évoquant les schémas de décisions (Fiévet, 1992).

Afin de faciliter la mise en place de cette coordination, deux éléments sont récurrents dans les ouvrages : la structuration du système de façon **hiérarchique** ainsi que l'importance de la **communication** entre les différentes parties du système.

Le découpage hiérarchique

Lorsqu'une personne se trouve face à un problème nouveau, elle aura tendance à découper ce problème en sous-problèmes afin de simplifier la réflexion et d'essayer de retrouver des schémas et problèmes déjà rencontrés (Mintzberg, 1976). On retrouve ce découpage hiérarchique dans l'organisation de nombreuses entreprises : celles-ci s'organisent en plusieurs services, chacun appliquant son expertise dans son domaine. Ces services peuvent employer de nombreuses personnes et être eux-mêmes découpés en plusieurs sous-services possédant une expertise plus spécifique.

De la même manière, Beaufre (1964) met en avant le découpage hiérarchique de la stratégie totale en stratégies spécialisées tel que le montre la figure 3.1. Chaque stratégie spécialisée se voit assigner une importance relative aux autres par la stratégie totale. Beaufre (1964) distingue notamment deux types de stratégie totale : la stratégie directe qui considère la stratégie militaire comme le moyen principal d'exécuter la stratégie et qui lui assignera donc une importance élevée relativement aux autres stratégies spécialisées, et la stratégie indirecte qui au contraire met les autres domaines davantage en avant, assignant donc un importance faible à la stratégie militaire.

Les niveaux définis par Fiévet (1992) représentés également sur la figure 3.1, montrent une confrontation progressive du "souhaitable" avec le "possible". Le niveau politique (correspondant à la stratégie totale) détermine une stratégie correspondant au "souhaitable". Cette stratégie est ensuite confrontée une première fois au réel par le niveau tactique (correspondant aux stratégies spécialisées) en définissant des objectifs répondant à la stratégie et adaptés à l'environnement observé. Une deuxième confrontation est ensuite effectuée par le niveau opérationnel qui va venir répondre aux objectifs des stratégies spécialisées en utilisant les moyens disponibles : c'est l'allocation des ressources aux actions de manière à répondre concrètement aux objectifs définis par les niveaux supérieurs.

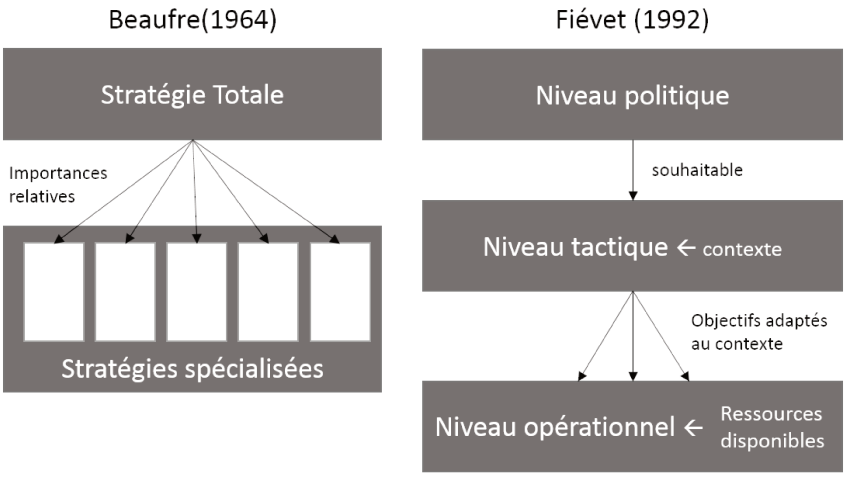


FIGURE 3.1: Les niveaux de la prise de décision selon Beaufre (1964) et Fiévet (1992)

La communication

Diviser le système en sous-systèmes permet donc d'aborder un sous-problème avec une vision simplifiée. Cependant prendre une décision sur un sous-problème sans tenir compte de l'ensemble serait une erreur : une décision ne peut être prise qu'avec une vue complète du problème. C. Hambrick and W. Fredrickson (2005) effectuent une comparaison avec la peinture : un peintre ne décide pas individuellement des couleurs, des formes, des pinceaux utilisés, mais voit une œuvre dans son ensemble, et c'est la combinaison des différents éléments qui en fait la qualité. De la même manière les décisions d'un système doivent être faites avec une vue d'ensemble, et celle-ci est donnée via la stratégie puisqu'elle est définie au niveau le plus haut de la hiérarchie.

Afin de permettre la cohérence des décisions, les niveaux doivent communiquer entre eux car ils sont interdépendants (Fiévet, 1992). La communication doit se faire de manière descendante par le biais de la stratégie afin que des informations pertinentes à propos du problème global soient transmises aux niveaux inférieurs. Ainsi un sous-système n'a pas besoin de connaître le problème entier, mais simplement les informations que la stratégie aura sélectionnées pour lui. Mais la communication doit également se faire de manière ascendante afin d'adapter les décisions au résultats obtenus. Cet aspect est exploré plus en détail dans la section 3.2.2.

Conclusion

Les stratégies sont donc utilisées par des groupes cherchant à donner une cohérence à leurs actions. Plus les groupes ont de membres plus il est difficile de garantir cette cohérence. Pour cela ces groupes mettent en place des systèmes hiérarchiques dont le plus haut niveau a la responsabilité d'établir une stratégie pour donner des indications aux niveaux inférieurs afin d'orienter les décisions pour qu'elles suivent une ligne directrice. La communication est primordiale pour que les décisions soient prises en fonction de l'état du système global, à la fois de manière descendante par le biais de la diffusion de la stratégie pour que chaque sous-système ait une vi-

sion d'ensemble du problème, mais également de manière ascendante pour que les informations diffusées par la stratégie soient à jour des actions entreprises et des résultats obtenus.

3.1.2 ... Avec des ressources internes communes

En divisant en plusieurs parties le système, la problématique de la gestion des ressources apparaît. En effet, l'organisation possède des ressources communes qu'elle va devoir répartir entre les différents sous-problèmes. Ces ressources incluent les membres de l'organisation ainsi que ses différentes possessions. Après avoir reçu les demandes transmises par la stratégie, les niveaux inférieurs doivent les confronter à l'environnement mais également aux ressources pour déterminer les capacités de l'organisation à y répondre. Si certaines ressources sont spécialisées et sont difficilement transférables, d'autres, par exemple l'argent, peuvent être utilisées à différentes fins suivant la tâche à laquelle elles sont affectées.

Foch a défini trois grands principes de la guerre encore utilisés par l'armée aujourd'hui (Lorino and Tarondeau, 2006) qui font écho à la gestion des ressources :

- la concentration des forces incite à choisir un domaine prédominant pour y assigner une majeure partie voire la totalité des ressources plutôt que de les disperser,
- l'économie des moyens recommande de former avec ses ressources un système dynamique et adaptable,
- et enfin la liberté d'action incite à rester dans des situations offrant un choix d'actions le plus important possible.

Hart (1962) nuance le principe de concentration des forces en affirmant que c'est un idéal hors d'atteinte et dangereux. En effet pendant une bataille, il est important de répartir ses ressources pour distraire l'ennemi en lui donnant de fausses pistes afin qu'il ne puisse pas concentrer ses propres ressources en défense et anticiper l'attaque principale.

Selon Mintzberg (1978) les stratégies sont parfois suggérées et décidées en fonction des ressources disponibles. Si le principe de liberté d'action peut sembler plus éloigné de la gestion des ressources, il faut voir ici les ressources disponibles comme une variable importante dans la détermination des actions possibles. Il est donc nécessaire lors des prises de décision de tenir compte des ressources à garder afin de favoriser cette liberté d'action. Cela permet ainsi de savoir quelles ressources doivent être protégées, lesquelles doivent être sacrifiées, ou bien encore lesquelles manquent et doivent être obtenues.

Nous retenons ici le rôle centrale des ressources dans la prise de décision car elles viennent contraindre les possibilités d'actions de l'organisation. Les ressources à utilisations diverses et les différentes parties de l'organisation provenant de sa division hiérarchique impliquent de faire des choix sur la répartition des ressources. Cette répartition est déterminée par la stratégie qui possède la vue globale de l'organisation dans son contexte.

3.1.3 ... Qui se place dans un environnement dynamique

Les systèmes ayant recours à une stratégie sont situés au sein d'un environnement dont le contrôle leur échappe. Dans le cas de l'entreprise, celle-ci se situe au sein d'un environnement commercial mais également d'un pays qui va venir imposer certaines contraintes (e.g. les lois, les contraintes géographiques). Suivant son champ d'application il peut aussi être nécessaire de prendre en compte l'environnement au niveau mondial. En ce qui concerne le domaine militaire, l'Etat se place dans l'environnement mondial en fonction de ses relations démocratiques avec les autres Etats et géographiques avec les limites de son territoire et ce qui l'entoure.

L'évolution de l'environnement est imprévisible. Dans le domaine de la guerre, de nombreux stratèges font référence dans leurs ouvrages au caractère aléatoire de l'environnement et à son influence sur la stratégie (Fiévet, 1992; Hart, 1962) : Moltke l'associe à l'imprévisibilité de l'ennemi mais De Gaulle l'étend également aux moyens, au terrain et au climat (Fiévet, 1992). L'imprévisibilité sur le terrain provient du *brouillard de guerre*, terme déjà utilisé dans le domaine militaire avant de faire son apparition dans les jeux, qui désigne la connaissance partielle du terrain et de ce qui s'y passe pendant le combat (García, 2012). Au niveau de l'entreprise on retrouve également cette imprévisibilité avec les cinq forces extérieures de Porter qui sont des forces venant influencer les décisions stratégiques (Ocler, 2002). Ces forces extérieures sont prises en compte lors de l'établissement d'une stratégie et peuvent également venir la contredire pendant son exécution, obligeant ainsi à effectuer un changement de stratégie.

Les autres systèmes

Du point de vue militaire comme du point de vue de l'entreprise, l'environnement comprend notamment d'autres systèmes qui sont souvent à l'origine de nombreux stimuli importants. Les relations de compétition sont les plus fréquemment évoquées puisque la situation de danger dans laquelle elles placent le système fixent alors un objectif de la stratégie qui est de protéger ses membres avec le maintien de l'activité de l'organisation et l'empêchement de sa disparition. D'autres ouvrages définissent les relations avec les autres acteurs de manière plus large. Hart (1962) évoque par exemple les autres Etats comme des éléments extérieurs qui peuvent être des outils à la réalisation d'un objectif, ne les plaçant pas comme des adversaires mais plutôt comme des variables de l'environnement. Fiévet (1992) évoque également des relations d'évitement et de coopération entre les différents acteurs.

Dans un environnement de guerre, les autres acteurs correspondent par exemple aux armées opposantes, mais également aux autres nations pouvant être impactées par la guerre : les alliés, les voisins... Pour une entreprise, les autres acteurs sont multiples :

- les autres entreprises proposant des services ou produits similaires qui sont alors des concurrents,
- les autres entreprises évoluant dans un domaine annexe qui peuvent alors être des fournisseurs ou des partenaires,

- les clients et clients potentiels.

De plus, il est nécessaire de prendre en compte la possibilité de l'arrivée de nouveaux acteurs.

Conclusion

Les environnements dans lesquels se situent les systèmes évoluent de manière imprévisible. Cela peut provenir de la méconnaissance du terrain, des évènements intrinsèques au terrain (e.g. la météo) ou bien de l'influence des autres systèmes que l'on considère comme faisant partie de cet environnement. A cela il faut ajouter la complexité des relations entre systèmes qui peut être difficile à observer. En effet les autres systèmes possèdent eux-mêmes des stratégies et des méthodes de prise de décision inconnues de notre système qui rend leurs actions imprévisibles. L'ensemble de ces paramètres rend l'opération de création d'une stratégie périlleuse et induit la nécessité d'une observation pertinente et d'une grande capacité d'adaptation.

3.1.4 Conclusion

Nous avons donc déterminé qu'une stratégie est mise en place par un système lorsqu'il comporte de nombreuses personnes travaillant ensemble pour le bien du système, lorsque des ressources doivent être partagées entre les différentes actions entreprises et lorsqu'il se place dans un environnement dynamique. Nous avons ainsi pu aborder les problèmes spécifiques que ces caractéristiques causaient et de quelle manière une stratégie permettait d'y répondre. Une quantité de personnes importante nécessite par exemple de découper le système en sous-systèmes afin de simplifier les problèmes auxquels il répond mais nécessite alors une méthode permettant de garder une cohérence dans la prise de décision distribuée qui en résulte. Pour obtenir cette cohérence, la communication est primordiale. Une stratégie est alors mise en place afin de transmettre des informations sur la vision commune que doit partager le système. Les ressources partagées du système peut créer des conflits lors de leur utilisation par les sous-systèmes. La stratégie peut alors apporter une solution en aidant à déterminer la réponse qui sera bénéfique pour l'objectif global du système. Enfin, le besoin de stabilité de la prise de décision entre en conflit avec l'environnement dynamique dans lequel se situe le système (Mintzberg, 1978). Le rôle de la stratégie est alors de trouver un équilibre pour faire fonctionner le système tout en s'adaptant aux contraintes et opportunités qui apparaissent dans l'environnement.

En résumé une stratégie doit fournir des renseignements au sujet de l'objectif global et de ce que chaque sous-système peut y apporter, elle doit permettre de répartir les ressources en fonction des demandes et de l'objectif global, et elle doit être capable de s'adapter à l'évolution du contexte lorsque c'est nécessaire. Nous distinguons ici l'*environnement* qui concerne ce qui est extérieur au système, et le *contexte* qui y ajoute l'état actuel du système qui se compose de l'état de la stratégie et des ressources disponibles.

3.2 La mise en place d'une stratégie

Nous avons déterminé les caractéristiques d'un système nécessitant une stratégie et les raisons de ce besoin. Nous souhaitons maintenant comprendre sa mise en place. Dans une première partie nous allons nous intéresser à la forme de la stratégie et aux **informations** qu'elle doit véhiculer, et dans un deuxième temps nous allons nous intéresser à son **évolution** dans le temps et aux mécanismes qui peuvent être mis en place pour faire face au dynamisme de l'environnement.

3.2.1 Les informations transmises par la stratégie

Dans la section 3.1.1 nous avons mis en évidence qu'une stratégie est mise en place dans un système découpé hiérarchiquement. La stratégie est alors utilisée pour communiquer des informations sur la direction globale du système aux sous-systèmes qui travaillent sur une sous-partie du problème global mais doivent y répondre en prenant en compte l'ensemble du problème. Lorsque Hart (1962) décrit le fonctionnement de la stratégie totale, qui a donc à sa disposition l'ensemble des ressources de l'Etat, il évoque la répartition de rôles et de puissances[2]. Le rôle d'un sous-système reflète la nature de la tâche à effectuer demandée par le système supérieur ; quant à la puissance d'un sous-système, elle dépend des moyens donnés pour effectuer la tâche, autrement dit les ressources allouées. Nous allons dans les parties suivantes nous intéresser d'abord à l'allocation des ressources puis à la nature de la tâche.

L'allocation des ressources

L'allocation des ressources comme nous l'avons vu dans la partie 3.1.2 est au cœur de la prise de décision stratégique. C'est d'ailleurs l'un des trois éléments de la stratégie communs à la stratégie militaire et la stratégie d'entreprise définis par Ocler (2002)[3]. La division d'un système intermédiaire en plusieurs sous-systèmes nécessite en effet de répartir les ressources. Cette répartition se fait suivant les besoins de chaque sous-système mais il peut arriver qu'il y ait des conflits. En effet si certaines ressources spécialisées sont dédiées à certains domaines, d'autres plus génériques, notamment les ressources monétaires, peuvent être utilisées à des fins différentes. Afin de réaliser ce partage de ressources, la stratégie peut fournir des informations supplémentaires qui permettent de déterminer quel sous-système va avoir la priorité.

Les systèmes intermédiaires vont avoir la tâche de répartir les ressources entre leurs différents sous-systèmes en combinant leurs besoins et la stratégie. La première étape est donc de connaître les besoins pour détecter les conflits puis dans un deuxième temps d'utiliser la stratégie pour les régler. Les informations supplémentaires fournies par la stratégie pour régler les conflits portent différents noms : *importance* pour Beaufre (1964) ou *priorité* pour Fiévet (1992) par exemple. Chaque sous-système va se voir affecter un paramètre contenant cette information. Les ouvrages

[2] Il appelle la stratégie totale la *grande stratégie*.

[3] Les trois points communs sont : un objectif à atteindre, la notion de *pilotage stratégique* et l'allocation des ressources.

ne détaillent pas l'utilisation de ces informations, nous pouvons simplement supposer qu'elles servent à créer une hiérarchie dans les sous-systèmes permettant de résoudre les conflits dans la répartition des ressources. Nous y ferons par la suite référence avec le terme *importance*.

Beaufre (1964) définit la politique comme le choix de buts et de volumes de moyens et la stratégie comme le choix des moyens et de leurs combinaisons. En ce qui nous concerne nous ne faisons pas de séparation entre politique et stratégie et nous retenons de l'association des deux les besoins suivants :

- le choix de buts correspond à la définition des sous-systèmes et des tâches qui leur sont affectées que nous aborderons dans la section suivante,
- le choix de volumes de moyens fait écho à la notion d'importance que nous venons d'aborder,
- le choix des moyens ne parle plus de la quantité des ressources mais de leur nature,
- enfin le choix des combinaisons note le besoin de prendre en compte la manière dont les ressources s'assemblent.

On peut donc voir que l'importance ne permet de répondre qu'à une partie de ces informations, il faut également prendre en compte la nature des ressources et leurs combinaisons.

La définition des tâches

Chaque sous-système va avoir une tâche qui lui sera affectée lui permettant de déterminer la nature de ce qu'il doit faire. Cette tâche lui permet de participer à l'effort commun pour le bien du système. Mais que signifie "le bien du système" ? Plusieurs ouvrages parlent d'un but, de l'objectif d'un système, pour lequel l'ensemble des membres travaillerait (García, 2012). Cependant une stratégie n'est pas un objectif, elle y répond, c'est-à-dire qu'elle exprime le fonctionnement que doit adopter le système pour l'atteindre (C. Hambrick and W. Fredrickson, 2005; Rumelt, 2011). Selon Mintzberg (1976) cet objectif peut répondre à un problème ou bien à un opportunité. Rumelt (2011) précise cependant qu'il correspondra dans tous les cas à un challenge, c'est-à-dire que la solution n'est pas triviale et que la réussite n'est pas assurée. La stratégie fixe alors une méthode de résolution d'un objectif, mais elle n'est pas définie par optimisation mais davantage par choix arbitraire qui peut être guidé par l'instinct ou par la personnalité.

L'environnement étant dynamique (voir section 3.1.3) il est nécessaire que ce fonctionnement soit assez souple pour s'adapter. En ce qui concerne la forme que doit prendre la tâche demandée à un sous-système, Fiévet (1992) recommande de transmettre les ordres d'un niveau au niveau inférieur sous forme d'objectif mais sans imposer de méthode de résolution afin de laisser une liberté permettant des initiatives et l'adaptation au contexte. C'est le résultat qui est ensuite analysé et comparé aux attentes par le niveau supérieur.

Conclusion

Dans cette partie nous avons pu voir que deux éléments fondamentaux devaient être transmis par la stratégie. La décomposition hiérarchique nécessite de donner à chaque sous-système des informations sur la tâche qu'il doit accomplir. En renseignant celle-ci sous la forme d'un objectif sans imposer de méthode de résolution, cela permet de garder l'adaptabilité nécessaire à l'environnement dynamique. Il est également nécessaire de fournir des renseignements pour l'allocation des ressources. Nous en avons identifié trois : l'importance qui va influencer les quantités de ressources, des informations sur la nature des ressources et des informations sur l'association des ressources.

3.2.2 L'adaptation de la stratégie

Nous avons précédemment évoqué l'impossibilité de définir une stratégie optimale en raison de l'imprévisibilité trop grande qui entoure la prise de décision. Cette imprévisibilité entraîne un besoin fondamental de réactivité et d'adaptation de la stratégie. Nous avons déjà défini que l'utilisation d'objectifs permet l'adaptation de la réalisation de la stratégie. Nous nous interrogeons désormais sur l'adaptation de la stratégie elle-même, c'est-à-dire sur la définition des objectifs.

Plusieurs auteurs insistent sur le risque élevé d'échec puisque la tâche de définir les objectifs incombe au stratège et comme nous l'avons déjà évoqué les décisions sont alors en partie guidées par l'instinct ou la personnalité. Il n'existe pas de règles précises permettant de déterminer les bonnes décisions stratégiques, simplement des outils permettant d'aider en fournissant une analyse la plus complète possible du contexte, mais c'est au stratège de faire ses choix selon ses propres critères, ses propres règles (Fiévet, 1992). Aristote compare en quelque sorte le stratège à un artiste lorsqu'il définit la composante stratégique comme un mélange de *techné*, qui signifie l'art créatif, et la *tuché*, qui signifie le hasard (Ocler, 2002).

> "La victoire n'est que le fruit d'une supputation exacte." Tzu

Le stratège correspond en ce qui nous concerne au game designer. Aucune formalisation du travail du stratège n'existe, et puisque le game designer ne peut pas être présent pendant le déroulement d'une partie de jeu pour adapter la stratégie lorsque cela est nécessaire, il doit envisager les modifications qui peuvent survenir. Pour cela nous allons dans un premier temps revenir sur les caractéristiques du système qui peuvent expliquer un besoin de remise en cause de la stratégie, puis nous allons nous penchés sur les évocations de la planification dans les ouvrages existants.

La remise en cause de la stratégie

Plusieurs circonstances peuvent justifier de revoir une stratégie, d'un point de vue général on peut dire qu'il est nécessaire de confronter les ressources et possibilités du système avec les menaces

et opportunités qui apparaissent dans l'environnement (Chaffee, 1985). Une modification des uns ou des autres peuvent nécessiter un changement de stratégie.

Les menaces et opportunités sont des stimuli extérieurs, c'est-à-dire qu'ils proviennent de l'environnement qui entoure notre système. Pour percevoir ces stimuli, le système doit repérer les différences entre l'état perçu et l'état attendu et ils peuvent être négatifs (crise) ou positifs (opportunité) (Mintzberg, 1976). Ils peuvent se présenter sous la forme d'un seul stimulus important ou bien de nombreux stimuli plus faibles qui se cumulent jusqu'à atteindre un seuil critique car nous avons tendance à réagir par palier et non en continu (Mintzberg, 1978). Il faut donc que le changement soit assez important pour être perçu comme significatif. Afin de réagir correctement au stimulus, il est nécessaire de récolter des informations : comprendre les raisons qui l'ont provoqué, comprendre ce qu'il implique et ce qu'il permet (Mintzberg, 1976). On retrouve dans les raisons de ces stimuli :

- l'environnement imprévisible en lui-même :
  - par des événements (e.g. événements météorologiques),
  - par la découverte du terrain (e.g. nouvelles ressources, point d'observation sur l'ennemi),
- et les autres acteurs :
  - par leurs actions sur le système (e.g. attaque, proposition d'alliance),
  - par leurs actions sur le reste de l'environnement.

A ces stimuli extérieurs viennent s'ajouter des stimuli intérieurs qui peuvent être déclenchés pour deux raisons : une modification des ressources ou actions disponibles qui peut venir modifier les possibilités s'offrant au système, et les résultats obtenus par la mise en place d'une stratégie. La modification des ressources peut survenir dans les quantités, mais elle peut également survenir dans la nature des ressources, par l'apparition d'une nouvelle technologie par exemple. Dans le domaine militaire, ce peut être à l'origine de changements importants dans la nature du combat (Beaufre, 1964; Fiévet, 1992). Les stimuli provenant des résultats obtenus peuvent être positifs comme négatifs. Ils proviennent d'une comparaison ce qui était attendu de la mise en œuvre de la stratégie avec la manière dont elle s'est réellement déroulée.

Les stimuli qui vont venir remettre en cause la stratégie sont donc de différentes natures. Ils peuvent être externes ou internes à la stratégie. Ils peuvent soit modifier directement les possibilités du système soit modifier ses connaissances. Si le changement est suffisant, il peut justifier un changement au niveau stratégique.

La planification stratégique

Mintzberg note l'aspect contradictoire de la formule "planification stratégique" (Fiévet, 1992) : d'un côté la planification fait écho à un plan statique ne prévoyant pas le besoin d'adaptation nécessaires pendant son exécution, et de l'autre la stratégie se met en place justement dans un

système ayant besoin de gérer un environnement imprévisible nécessitant une remise en cause régulière du plan en cours.

Pour anticiper l'échec d'un plan, on peut considérer les actions souvent stochastiques et prévoir des alternatives suivant le résultat obtenu. Mais ces alternatives peuvent également se situer au niveau d'un sous-plan plus complet. L'importance de possibles alternatives en cas d'échec est un point mis en avant par de nombreux stratèges : Hart (1962) cite notamment Bourcet[4] ("Tout plan de campagne doit avoir plusieurs branches et doit avoir été si bien médité que l'une et l'autre de ces dites branches ne peut manquer de conduire au succès"), Napoléon Bonaparte ("[il] cherchait toujours à faire son thème de deux façons") et Sherman[5] ("placer l'ennemi sur les cornes d'un dilemme"). Il définit par ailleurs huit axiomes pour la création d'une stratégie parmi lesquels "Adoptez une ligne d'opération procurant des objectifs alternatifs" et "Assurez-vous de la souplesse, à la fois, du plan et du dispositif qui doivent pouvoir s'adapter aux circonstances". Changer pour une alternative peut se faire à la suite d'un échec total ou partiel de la solution actuelle ou bien à l'apparition d'une opportunité rendant une alternative plus intéressante que la solution actuelle.

De façon plus générale, l'ensemble des stimuli évoqués précédemment peuvent justifier une remise en cause de la stratégie et la définition des stimuli pertinents et de leurs effets sur la stratégie peut être décrit. Cependant la représentation sous forme de plans n'est pas adapté car il n'y a pas systématique une notion d'ordre car la majorité des stimuli ne sont pas associés à des actions et peuvent survenir à tout moment.

Ces remises en cause de la stratégie se heurte de plus à un *momentum bureaucratique*, autrement dit un besoin de stabilité bureaucratique Mintzberg (1978). Pour cela il suggère que la stratégie alterne entre les périodes de changements pour s'adapter à la situation courante de l'environnement dynamique et les périodes "plates" pour stabiliser l'état de la stratégie et en observer le résultat.

Conclusion

Nous avons donc défini les différents types de stimuli qui peuvent justifier une remise en cause de la stratégie courante :

- les stimuli externes :
  - l'environnement imprévisible en lui-même :
    - par des événements (e.g. événements météorologiques),
    - par la découverte du terrain (e.g. nouvelles ressources, point d'observation sur l'ennemi),
  - et les autres acteurs :
    - par leurs actions sur le système (e.g. attaque, proposition d'alliance),

[4]Pierre Joseph de Bourcet(1700-1780) Lieutenant-général

[5]William Tecumseh Sherman(1820-1891) Général de l'Armée (armée américaine)

    - par leurs actions sur le reste de l'environnement,

- et les stimuli externes :
    - par la modification des ressources disponibles,
    - par des retours sur la mise en place de la stratégie actuelle.

Cependant la définition de la prise en compte de ces stimuli dans la stratégie ne présente pas beaucoup de pistes de formalisation dans les ouvrages étudiés qui s'en remettent à l'utilisation d'un stratège. Ce dernier a pour rôle de définir la stratégie puisque la réussite de celle-ci est trop incertaine pour que des règles de construction puissent être spécifiées.

La seule technique évoquée est la planification arborescente mais celle-ci ne permet pas de répondre entièrement au besoin d'adaptation.

3.2.3 Conclusion

Une stratégie permet de répondre à un objectif global d'un système. Pour distribuer la réalisation de cet objectif dans le système découpé hiérarchiquement, nous avons défini deux informations que la stratégie doit permettre de communiquer aux sous-systèmes : la définition de la tâche que doit effectuer le sous-système et des informations servant à l'allocation des ressources.

En définissant les tâches sous la forme d'objectifs, cela permet de garder une flexibilité permettant l'adaptation nécessaire pour la réalisation d'une stratégie. Pour réaliser ces objectifs les sous-systèmes vont se voir assigner des ressources, et pour déterminer les ressources qu'ils vont recevoir trois informations sont nécessaires : la quantité de ressources qui peuvent lui être allouées, la nature des ressources, ainsi que la combinaison des ressources.

Ces informations, objectifs et ressources, doivent pouvoir évoluer pour s'adapter aux stimuli qui viennent remettre en cause la stratégie courante. Ces stimuli peuvent provenir du monde environnant, des autres systèmes ou bien du système lui-même. Ils peuvent être négatifs, c'est-à-dire que la stratégie actuelle n'est plus adaptée, ou positifs, c'est-à-dire qu'une autre stratégie serait plus efficace. Cependant aucune formalisation de la prise en compte de ces stimuli n'a pu être extraite des ouvrages étudiés.

3.3 Bilan du chapitre

La stratégie est là pour guider la prise de décision dans un système de grande taille situé dans un environnement dynamique. En effet un système possédant de nombreux membres est souvent découpé de façon **hiérarchique** en sous-systèmes afin de simplifier l'objectif poursuivi par le système. Cette division implique le besoin d'une communication de qualité afin de maintenir une cohérence dans les actions effectuées. Les ressources possédées par le système doivent également être réparties dans les divisions du système selon les besoins de chacun mais en répondant également à l'objectif global du système.

Afin de garantir la cohérence de la prise de décision distribuée, ces systèmes utilisent une stratégie. Celle-ci doit fournir deux types d'éléments : les tâches à réaliser par chaque sous-systèmes et la répartition des ressources notamment en situation de conflit. De plus l'environnement dynamique dans lequel se situe le système vient mettre à mal les décisions prises en provoquant des changements de contexte imprévisibles. La stratégie doit donc être suffisamment flexible pour s'adapter à ces changements. Afin de répondre à ce besoin de flexibilité, la stratégie définit les tâches sous forme d'**objectifs** permettant ainsi au sous-système d'adapter la méthode de résolution au contexte et aux moyens disponibles. En ce qui concerne la répartition des ressources, la stratégie n'alloue pas les ressources mais renseignent des paramètres pour l'orienter. Ces paramètres contiennent l'**importance** de chaque sous-système dans la résolution de l'objectif commun afin d'orienter les volumes de ressources allouées, mais également des informations sur **la nature et les combinaisons des ressources** allouées.

Cette stratégie doit être **adaptative** pour répondre à l'évolution du contexte. Les stimuli qui peuvent venir perturber notre stratégie peuvent être de différentes natures que nous avons classifier en deux grandes catégories : externes et internes. La première est composé des informations provenant du monde environnant qui parviennent à la suite d'un évènement comme la météo ou par l'exploration du terrain, et des évènements provoqués par les autres acteurs. La deuxième catégorie contient quant à elle la modification des ressources disponibles et les retours de la mise en place de la stratégie courante. La prise en compte de l'influence de ces stimuli sur la stratégie n'a pas été davantage formalisée, dans les ouvrages étudiés les stimuli était pris en compte par les stratèges qui conçoivent les stratégies. Les limites des propositions venant de la définition commune de la stratégie apparaissent : une stratégie mise en place dans le monde réel pourra toujours voir des personnes s'atteler à la création d'une nouvelle stratégie en cas de situation imprévue, or dans notre cas d'application nous ne souhaitons pas d'intervention humaine lors de l'apparition d'une situation non gérée par la stratégie actuelle.

Dans le chapitre suivant, nous allons nous intéresser à la mise en place d'une stratégie dans les jeux de stratégie et nous allons alors pouvoir d'une part comparer les formats de stratégie avec celui formalisé dans ce chapitre et d'autre part explorer les solutions qui y ont été mises en place pour permettre l'adaptation nécessaire.

Chapitre 4

La stratégie dans les jeux vidéo

Dans le chapitre 2 nous avons défini que l'objectif d'un jeu vidéo est de fournir au joueur une expérience divertissante dont le game designer est l'architecte. Lors du développement du jeu le game designer conçoit puis délègue la réalisation et perd ainsi le contrôle direct en gardant uniquement un rôle de superviseur. Le game designer doit garantir que le jeu créé produit bien l'expérience désirée et doit pour cela communiquer des informations pertinentes et suffisantes à l'équipe réalisant le jeu. Les jeux de stratégie proposent un environnement riche qui rend très difficile le contrôle sur leur création avec les outils traditionnels : c'est le mur de l'*authoring* décrit dans Lucas et al. (2012). Ils décrivent en effet une courbe représentant l'effort de création nécessaire qui monte exponentiellement avec la diversité et le dynamisme d'un jeu. Nous avons vu que dans ce type de jeu, les IA qui contrôlent les opposants jouent un rôle primordial pour la qualité de l'expérience mais qu'il n'existe pas de formalisme adapté permettant une communication de qualité entre les game designers et les développeurs IA. Nous avons donc décidé de remettre entre les mains du game designer la création de l'expérience de jeu en lui proposant de définir la prise de décision stratégique des IA par le biais d'un modèle de stratégie accessible sans connaissances de programmation.

Afin de créer notre modèle, nous nous sommes intéressés dans un premier temps au sens commun que l'on donne au mot *stratégie*. Nous souhaitons dans ce chapitre comparer ce qui ressort de ce sens commun et la forme que prend la stratégie dans les méthodes utilisées pour les IA de jeux de stratégie. Nous nous intéressons en particulier aux contrôles que le game designer peut avoir sur la mise en place de stratégies. Pour cela nous allons nous intéresser aux propositions de la recherche académique mais également aux solutions utilisées par l'industrie. Les travaux de recherche s'intéressant aux jeux de stratégie sont abondants depuis que Buro (2003) a mis en avant leur intérêt pour la recherche en IA détaillé dans la section 2.3.2. Nous pouvons notamment en retrouver une partie dans les comptes-rendus réalisés par Ontanón et al. (2013) et Lara-Cabrera et al. (2013). Du côté de l'industrie, il est parfois plus compliqué de trouver des références mais les séries AI Game Programming Wisdom (Rabin, 2002, 2004, 2006, 2008) et plus récemment Game AI Pro (Rabin, 2013, 2015) qui ont comme objectif de mettre à disposition de tous les connaissances en IA dans les jeux, permettent d'avoir une vision de ce

qui est mis en place dans l'industrie et des problèmes qui y sont abordés.

Dans une première partie nous comparerons le contexte d'une IA dans un jeu de stratégie aux caractéristiques d'un système utilisant une stratégie telles que définies dans la partie 3.1. Dans la partie 4.2, nous étudierons plus en détail l'une de ces caractéristiques : la décomposition du système en sous-systèmes. Nous verrons ensuite comment ces systèmes adaptent la stratégie selon deux axes : le choix des objectifs avec la section 4.3 et l'allocation des ressources avec la section 4.4. Pour chaque partie, nous verrons comment le game designer peut venir influencer la prise de décision.

4.1 Le contexte des jeux de stratégie

Nous avons vu dans la partie 3.1 les caractéristiques des systèmes pour lesquels l'usage d'une stratégie est bénéfique. Nous en avons défini trois :

- une taille importante,
- des ressources communes,
- un environnement dynamique.

Nous retrouvons bien ces caractéristiques dans les jeux de stratégie, dont les jeux 4X, où chaque joueur a pour objectif de gérer une organisation afin de la développer et de faire fructifier ses ressources. Si les joueurs débutent souvent avec une organisation de très petite taille, celle-ci évolue rapidement.

L'organisation possède des ressources de natures variées : des ressources qui peuvent se déplacer dans le monde virtuel (les unités), des ressources présents dans le monde virtuel mais qui ne peuvent pas se déplacer (les bâtiments), les autres ressources récoltées dans l'environnement ou créées par l'organisation, et des ressources virtuelles plus symboliques comme le bonheur des habitants ou la puissance diplomatique. Ces ressources vont pouvoir être utilisées à plusieurs fins et être bénéfiques à différents domaines. Nous avons donc bien affaire à un système pour lequel des choix doivent être effectués au sujet de la répartition des ressources.

Nous retrouvons également l'environnement dynamique par le biais de deux mécaniques de jeu : le brouillard de guerre dont nous avons déjà parlé dans le chapitre 3 et la compétition avec d'autres organisations. L'organisation contrôlée par les joueurs se place ainsi dans un environnement inconnu qu'ils perçoivent partiellement et doivent découvrir, et la découverte progressive implique une évolution des connaissances de l'environnement. De plus les joueurs évoluent dans un monde virtuel qu'ils partagent avec d'autres joueurs contre lesquels ils se trouvent en compétition et dont le comportement va venir modifier l'environnement. On peut cependant noter que le jeu propose un environnement plus connu que pour les systèmes étudiés dans le chapitre 3. En effet, la liste des ressources, des actions, des évènements qui peuvent apparaître dans le jeu est prédéfinie et connue des joueurs. Bien que le jeu possède une incertitude indiscutable par le nombre important d'associations entre ces différents éléments, elle est donc plus modérée que dans le monde réel.

4.2 La décomposition du problème

Nous avons pu observer dans la partie 3.1 qu'un système de taille importante simplifiait sa prise de décision en se décomposant en une structure hiérarchique. Cela permet d'obtenir des problèmes simplifiés pour lesquels il est notamment plus facile de faire des analogies avec des problèmes déjà rencontrés. Les problèmes abordés dans un jeu de stratégie sont trop complexes et doivent également être divisés en problèmes plus simples (Dahlbom and Niklasson, 2006). Dans les architectures des IA, la décomposition se présente sous plusieurs formes que l'on peut répartir en deux catégories : hiérarchique et fonctionnelle.

La décomposition hiérarchique fait écho à celle du chapitre précédent. Elle symbolise une forme d'autorité : les modules vont venir influencer le fonctionnement des modules inférieurs hiérarchiquement à travers des directives. Les décisions sont alors prises par les modules les plus haut-niveau en premier puis les résultats de ces décisions sont passés aux modules du niveau inférieur qui prennent à leur tour leur décision, et ainsi de suite jusqu'aux modules les plus bas-niveau. On observe aussi une précision plus accrue du problème en descendant la hiérarchie : les informations nécessaires sont plus spécifiques et les décisions prises s'appliquent à un domaine plus restreint.

A l'inverse dans la décomposition que nous appelons fonctionnelle, il n'y a pas d'ascendance entre les modules qui partagent des informations soit par l'envoi direct de messages soit par le biais d'un *blackboard*[1]. Les problèmes abordés par chaque module sont disjoints et de tailles relativement similaires.

Ces deux types de décomposition ne sont pas exclusifs mais complémentaires, ils peuvent se retrouver dans un même système. Dans la suite nous allons mettre en évidence l'emploi de ces deux formes de décomposition dans les solutions actuellement proposées afin de les confronter aux contraintes des jeux 4X et à leur utilisation par un game designer.

4.2.1 La décomposition hiérarchique

Nous allons voir plusieurs différences fondamentales entre les architectures existantes et essayer de voir lesquelles nous semblent répondre de façon plus pertinente à notre problématique. Nous allons également étudier leur composition et voir quelles sont leurs limites.

La décomposition des ressources ou des objectifs

On retrouve deux types de décomposition :

- celles qui sont centrées sur les ressources,
- celles centrées sur les objectifs.

[1] Un *blackboard* est une structure partagée accessible en lecture et en écriture par plusieurs modules permettant ainsi de partager des informations.

Galvao Madeira (2007) a choisi la première pour construire son approche STRADA en s'inspirant de la structure militaire pour créer des ensemble d'unités avec plusieurs niveaux de commandement. Nous avons mis en avant le besoin de flexibilité des jeux 4X, qui peut s'appliquer sur les actions mises en place, comme le propose Galvao Madeira (2007), mais également sur les ressources allouées. Il semble cependant que les groupes d'unités dans STRADA soient fixes. Une telle structure rend difficile la possibilité de faire varier la quantité de ressources allouées à chaque tâche, et nécessite un niveau de coordination supplémentaire complexe.

A l'inverse une structure hiérarchique des objectifs poursuivis guide la décision, et l'allocation des ressources s'effectue dans un deuxième temps en fonction des besoins des objectifs sélectionnés. Cette structure supporte une prise de décision flexible sur les actions mises en place, sans contraindre la répartition des ressources qui peut alors s'adapter aux décisions. Nous allons donc nous intéresser davantage aux travaux qui utilisent une décomposition des objectifs poursuivis.

Les décompositions fixes et les décompositions variables

Certains travaux proposent une décomposition en plusieurs niveaux prédéfinis. Ceux-ci sont souvent inspirés de la décomposition militaire bien connue en niveaux stratégique, tactique et opérationnel présentée dans la section 3.1.1. C'est le cas du *Multi-Tiered AI Framework* (MTAIF) proposé par Kent (2004) et de l'architecture ADAPTA de Bergsma (2008). Cependant nous avons déjà déterminé que dans cette thèse nous nous intéressons uniquement à la décision stratégique et que nous souhaitons la simplifier en lui apportant également une structure hiérarchique. Ces approches n'utilisent pas de hiérarchie à l'intérieur de leur niveau stratégique, elles ne nous intéressent donc pas dans cette partie.

Sharma et al. (2007) propose une décomposition hiérarchique dont le principe consiste en plusieurs niveaux composés d'un unique module qui font le lien entre le niveau qui leur est directement supérieur et le niveau qui leur est directement inférieur. Chaque niveau reçoit un objectif pour lequel il construit un plan de sous-objectifs à l'aide d'une base d'actions et d'un planificateur. Le niveau va alors effectuer le plan en fournissant le sous-objectif à réaliser au niveau inférieur. La propagation se fait jusqu'au niveau le plus bas qui crée alors un plan d'actions directement exécutables dans le monde virtuel du jeu. L'architecture ne permet pas d'envisager la parallélisation d'actions, pourtant indispensable à la mise en place d'une stratégie, à cause de sa hiérarchie linéaire (chaque module possède un seul module directement supérieur et un seul module directement inférieur). De plus elle suppose que tous les objectifs pouvant être sélectionnés nécessitent un nombre identique de niveaux de décomposition puisque le nombre de modules est prédéfini. Cette hypothèse ne semble présenter aucune justification et limite inutilement la flexibilité du système.

Nous allons nous intéresser aux travaux proposant une décomposition des objectifs plus flexible, notamment en n'imposant aucune contrainte de profondeur de hiérarchie.

Les architectures proposant une décomposition variable des objectifs

Les architectures qui évoluent dans leur profondeur ou largeur en fonction des besoins utilisent en général une base de méthodes, correspondant à des façons de résoudre un objectif, dans laquelle ils vont chercher pour créer une hiérarchie d'objectifs. Ces méthodes se composent de sous-objectifs, nécessitant la création d'un nouveau niveau hiérarchique, et d'actions applicables dans le monde virtuel, qui signifient que le niveau le plus bas est atteint. Plusieurs formes de méthodes existent :

- Johnson (2006) propose d'utiliser les FSM pour représenter la résolution d'un objectif. Les états du FSM correspondent alors à des sous-objectifs ou à des actions applicables dans le monde virtuel. Les transitions permettent de passer d'un état à l'autre en fonction de conditions sur l'état du monde et/ou du résultat retourné par l'état du FSM (succès ou échec). Lorsqu'aucune transition n'est déclenchée à la fin de la réalisation d'un état, le résultat retourné par l'état sélectionné est remonté dans la hiérarchie.

- Hinrichs and Forbus (2007) et Laagland (2008) utilisent les *Hierarchical Task Network* (HTN). La résolution d'un objectif est décrite par un *task network* tel qu'expliqué par Erol et al. (1994). Un objectif est décrit par un ensemble de tâches qui doivent toutes être effectuées pour atteindre l'objectif. Un ensemble de contraintes vient également définir comment exécuter les tâches. Dans ces contraintes nous retrouvons notamment des contraintes d'ordre entre les tâches et des contraintes sur l'état du monde.

Le *case-based planning* (CBP) utilisé par le système DARMOK (Mehta et al., 2008; Virmani et al., 2008), de même que le *dynamic scripting* (DS) comme les HTN, définit plusieurs plans pour atteindre un objectif. Afin de choisir lequel choisir, les méthodes utilisées sont différentes :

- Les HTN utilisent un ensemble de préconditions qui doivent être vraies pour permettre d'exécuter le plan. Les plans sont définis dans un ordre qui détermine celui choisi si plusieurs ont leurs pré-conditions satisfaites.

- Le CBP associe un contexte d'application à chaque plan et c'est celui dont se rapproche le plus le contexte courant qui est choisi. Des conditions doivent également rester satisfaites pendant l'application du plan, ce sont les *alive-conditions*.

- Pour le DS, un poids est associé à chaque méthode et évolue en fonction de la réussite de son utilisation. Ce poids vient ensuite influencer la probabilité que la méthode soit choisie. Ainsi plus une méthode est efficace, plus elle aura de chance d'être choisie. Il faut cependant noter que ce choix n'est pas effectué au moment où l'objectif est choisi, mais de façon régulière. Lorsque l'objectif est choisi, une seule méthode déjà sélectionnée est disponible.

Nous avons donc des objectifs qui peuvent être résolus avec plusieurs méthodes décomposées en sous-objectifs. Les FSM telles que Johnson (2006) les utilise ont l'avantage de regrouper

les différentes méthodes pour résoudre l'objectif et la manière de les choisir. Les méthodes ne sont pas découpées mais imbriquées les unes aux autres pour les représenter ensemble et permettre la sélection d'un sous-objectif. Ce formalisme présente également l'avantage de proposer un contrôle graphique facile à prendre en main. Les *task networks* permettent également une représentation graphique mais nécessite d'en avoir un par méthode et d'y ajouter des préconditions. En revanche ils permettent l'exécution en parallèle de plusieurs états, ce qui n'est pas permis dans les FSM telles que les utilise Johnson (2006).

4.2.2 La décomposition fonctionnelle

La décomposition fonctionnelle propose de découper le problème en sous-problèmes disjoints qui sont étudiés en parallèle et échangent éventuellement des informations. Elle apparait dans les ouvrages présentés dans le chapitre 3, cependant elle y est toujours associée à un découpage hiérarchique. Nous la retrouvons pas exemple dans les stratégies spécialisées de Beaufre présentées dans la section 3.1.1. Celles-ci abordent des domaines différents : la politique, l'économie ou le militaire par exemple.

Nous retrouvons également dans les environnements de jeu proposés par les jeux 4X ces différents domaines de développement comme nous l'avons présenté dans la partie 2.3.1. La liberté d'action qui en découle a inspiré une décomposition de la prise de décision en modules thématiques. Nous allons détailler quelques exemples de décomposition fonctionnelle et voir de quelles manières les modules peuvent être définis.

La décomposition fonctionnelle par domaine

La décomposition fonctionnelle est en générale spécifique au jeu abordé et aux différents domaines qui le composent. Dans son architecture pour les jeux 4X, Bergsma (2008) propose une décomposition du niveau tactique en modules inspirée de la signification des quatre 'X'[2], cependant il précise que ce n'est qu'un exemple afin de développer davantage l'un des modules tactiques (l'extermination) pour illustrer le modèle. Dans les RTS, l'aspect militaire est une composante importante puisque pour gagner la seule possibilité est de détruire les unités adverses, mais pour cela il faut également créer ses ressources et l'ordre dans lequel le joueur les crée est un problème important, appelé le *build order*, puisqu'il va définir l'évolution de la composition de l'armée au cours de la partie.

Les deux compétitions StarCraft[3] qui ont lieu dans les conférences CIG et AIIDE sont source de nombreux travaux. Churchill et al. (2016) présentent les architectures de sept participants qui sont représentées sur la figure 4.1. Chaque rectangle correspond à un module dont la tâche est brièvement expliquée par le label, les rectangle noirs sont ceux communiquant directement avec le jeu. Les flèches pointillées représentent des transmissions d'information alors que les flèches pleines indiquent un contrôle, autrement dit la capacité à commander la réalisation d'une

[2] eXpansion, eXploration, eXploitation et eXtermination

[3] `http://us.blizzard.com/en-us/games/sc/` (accédé le 6 février 2017)

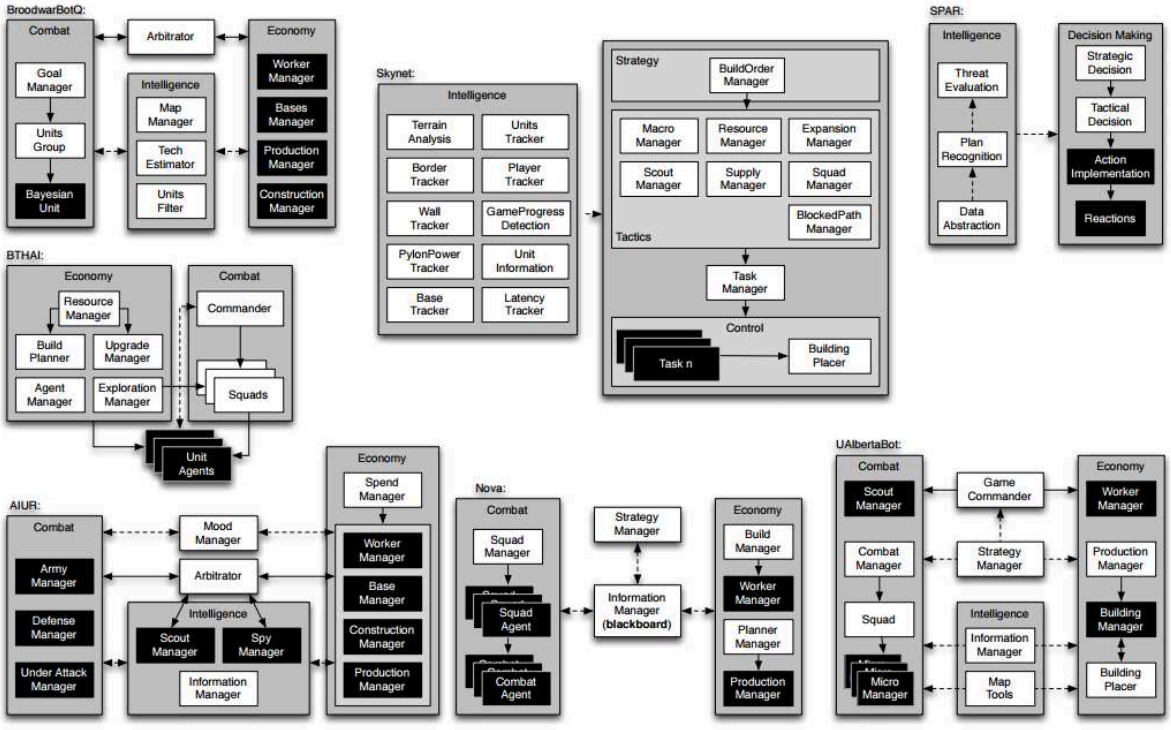


FIGURE 4.1: Exemples d'architectures IA ayant participé à des compétitions Starcraft (Ontanón et al., 2013)

tâche spécifique à l'autre module. On peut observer que toutes les architectures présentent une décomposition en plusieurs modules. On peut également y voir la variété des structures qui mêlent hiérarchie et thématiques. On y retrouve cependant le plus souvent deux modules haut-niveau : un module *Economy* correspondant à la création et à l'amélioration des ressources, et un module *Combat* qui s'occupe de l'aspect militaire. A ces deux modules s'ajoute parfois un module *Intelligence* correspondant à la récolte d'informations. EISBot est un système présenté par McCoy and Mateas (2008) destiné à jouer à un autre RTS : Wargus[4]. On retrouve dans son architecture (figure 4.2) un module militaire (TM), un module de récolte d'information (RM), mais l'économie est séparée dans deux modules (PM et IM). A ces modules viennent s'ajouter un cinquième, un module stratégique (SM) venant former une hiérarchie.

Si l'on trouve des similitudes entre les architectures lorsqu'elles sont créées pour un même type de jeu, comme le montrent les architectures présentées dans Churchill et al. (2016) et McCoy and Mateas (2008), quelques différences apparaissent également et nous pouvons supposer qu'elles seraient plus nombreuses sur un jeu 4X puisque l'environnement est plus complexe. Cette décomposition sous forme de modules thématiques symbolise des objectifs disjoints de domaine différent qui sont donc étudiés séparément. Le découpage dépend donc de la perception du problème et de l'environnement dans lequel il se pose par le concepteur de l'architecture. L'architecture dépend donc du jeu mais également du concepteur.

[4] `http://wargus.sourceforge.net/index.shtml` (accédé le 6 février 2017)

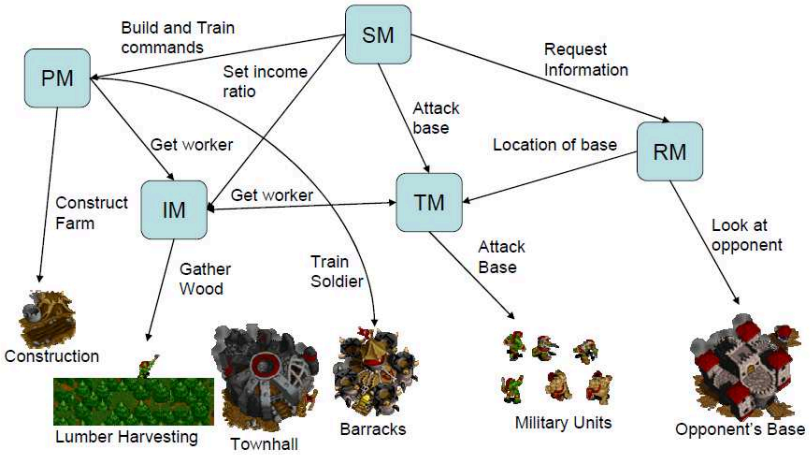


FIGURE 4.2: Les différents modules qui composent EISBot (McCoy and Mateas, 2008)

La décomposition fonctionnelle par type d'IA

D'autres approches mettent en avant une autre variable jouant dans la définition des différents modules : le type d'IA souhaité.

Young and Hawes (2012) s'inspirent par exemple des framework motivationnels qui consistent à définir des *drives*. Chaque drive définit :

- un état de motivation qui exprime un état désiré du système,
- une méthode qui permet de mesurer à quel degré le désir est respecté,
- un seuil au delà duquel on considère que le désir n'est plus respecté,
- et un ensemble d'objectifs permettant de satisfaire de nouveau le désir lorsque le seuil est atteint (un seul suffit et doit être choisi).

Un système possède plusieurs drives indépendants correspondant à différents états désirés par le système et c'est pourquoi nous avons classé cette architecture dans cette partie, cependant Young and Hawes (2012) ne donnent pas d'exemple de drive qui correspondrait à un jeu de stratégie.

Nieves (2015) utilise plusieurs fonctions d'utilité dont la combinaison symbolise une personnalité. L'exemple proposé se concentre uniquement sur les déplacements spatiaux d'unités, et les fonctions viennent données une utilité aux tuiles qui composent le monde virtuel. On peut par exemple avoir une fonction d'utilité pour atteindre un emplacement clé (plus la tuile est proche de l'emplacement plus elle a une utilité élevée) et une autre pour éviter un ennemi (plus la tuile est près d'un ennemi moins son utilité est élevée). Les résultats des fonctions d'utilité sont ensuite normalisés puis combinés en les multipliant par un poids et en les additionnant. Ce sont les poids qui vont définir des personnalités d'IA, et l'auteur compare les utilités à des traits de personnalités. Par exemple l'utilité pour atteindre un but pourrait correspondre à une trait de personnalité "fonceur" alors que l'utilité d'évitement de l'ennemi correspond à un

trait de personnalité "prudent". Définir une personnalité revient alors à définir à quelle intensité chacun de ces traits s'expriment relativement aux autres. Une personnalité peureuse pourra par exemple donner au trait "prudent" un poids cinq fois plus important qu'au trait "fonceur".

Une décomposition fonctionnelle doit donc prendre en compte le jeu, une manière de percevoir le problème posé par le jeu, mais également le type d'IA que l'on souhaite définir. Notre objectif étant de mettre à disposition des game designers notre modèle de stratégie, une décomposition fonctionnelle semble adaptée à condition qu'elle soit modifiable par le game designer. En effet, puisqu'il en sera l'utilisateur, il est important que l'architecture reflète sa manière de percevoir le problème ; de plus c'est lui qui est le concepteur du type d'IA désirée.

Conclusion

La décomposition fonctionnelle consiste à créer plusieurs modules qui vont chacun être responsable d'un sous-problème auquel est confronté le système. Nous avons observé deux manières de réaliser cette décomposition : en plusieurs domaines ou bien en traits de personnalité. Chaque décomposition est dépendante de l'environnement dans lequel l'IA doit évoluer et du comportement souhaité pour l'IA, soit finalement dépendante du concepteur et de la manière dont il perçoit le rôle de l'IA. Cette décomposition est créée par le concepteur ce qui permet une meilleure intelligibilité mais induit une rigidité de la décomposition.

4.2.3 Conclusion

La décomposition du problème est indispensable dans le cadre des jeux de stratégie comme en témoignent les différents travaux présentés. En effet, les jeux de stratégie correspondant aux caractéristiques des systèmes étudiés dans le chapitre 3, la décomposition permet de diviser le problème en problèmes plus simples. Nous avons identifié deux types de décompositions : hiérarchique et fonctionnelle.

La **décomposition hiérarchique** permet de répondre aux besoins de simplification du problème tout en permettant un contrôle par les niveaux les plus hauts qui vont pouvoir indiquer des objectifs plus simples aux niveaux inférieurs. De plus les architectures existantes proposent des solutions flexibles permettant une adaptation au contexte en passant par des pré-conditions, des contextes d'application ou bien par le biais d'une FSM. Si l'utilisation de FSM présente la plus grande **adaptabilité** et des avantages d'**intelligibilité**, l'architecture actuelle ne propose pas la possibilité d'effectuer des sous-objectifs en parallèle.

Le besoin de parallélisme peut trouver une réponse dans la **décomposition fonctionnelle** qui propose un découpage fixe dépendant du jeu, de la vision du problème et du type d'IA désiré. Les architectures étudiées présentaient soit une décomposition dépendante de la vision du problème engendrant des modules abordant des domaines différents, soit une décomposition dépendante du type d'IA désiré symbolisée par un ensemble de traits de personnalité ou d'états désirés. Les avantages ici sont le **parallélisme** et la **personnalisation** du découpage, cependant cela entraîne un manque de flexibilité de la décomposition.

Ces deux types de décomposition ne sont pas exclusifs et chacun présente des avantages adaptés pour notre problématique. Nous allons donc dans notre proposition étudier la possibilité de combiner les deux approches.

4.3 L'adaptabilité des objectifs

Nous avons vu dans le chapitre 3 que le contexte dynamique rend indispensable l'adaptabilité d'une stratégie. Nous avons établi qu'une décomposition par les objectifs permet déjà une décomposition flexible favorable à l'adaptation. En effet, fournir un objectif sans imposer de méthode permet aux niveaux plus bas d'adapter leur fonctionnement au contexte observé. Nous souhaitons dans cette partie explorer plus en détail comment le choix des objectifs évolue. Nous souhaitons également voir de quelles manières un game designer peut venir influencer ce choix. Nous allons nous intéresser à l'adaptation sous deux angles : l'adaptation de la sélection et la remise en cause des choix.

4.3.1 L'adaptation de la sélection

La sélection d'un nouvel objectif ou d'une nouvelle méthode peut se faire grâce à différentes techniques. Nous allons explorer dans cette partie les caractéristiques de celles utilisées par les systèmes étudiés et nous allons nous intéresser plus particulièrement au degré de contrôle externe qu'elles proposent puisque nous souhaitons proposer au game designer d'exercer un contrôle sur cette sélection.

Fonctions d'analyse du contexte Les systèmes présentant une décomposition fonctionnelle dépendante du type d'IA désiré, que nous avons déjà présentés dans la section 4.2.2, proposent des méthodes composées de plusieurs fonctions d'évaluation du contexte permettant de personnaliser la perception. Cependant les systèmes diffèrent dans leur utilisation des résultats. Les fonctions utilisées par Young and Hawes (2012) correspondent au calcul du degré de satisfaction de situations désirées. Chacune est associée à un seuil au delà duquel la situation désirée est considérée comme non satisfaite déclenchant alors l'objectif de la satisfaire de nouveau. Nieves (2015) utilise quant à lui des fonctions d'utilité dont les résultats sont combinés pour la prise de décision. Ces fonctions d'utilité symbolisent des filtres de perception du contexte, c'est-à-dire plusieurs manières de l'analyser, auxquels Nieves (2015) propose d'associer des poids pour créer une perception unique qui les combinent.

Les deux techniques proposent un contrôle commun qui est la définition des fonctions d'évaluations du contexte qui nécessite des connaissances en programmation et qui ne rentre donc pas dans nos critères. Des paramètres plus accessibles y sont associés : ce sont la définition de seuils de déclenchement et de poids d'influence. Les premiers permettent de définir quand un contexte va justifier la sélection d'un objectif et les deuxièmes créent une perception unique du contexte permettant de comparer plusieurs options.

Le système de *drives* proposé par Young and Hawes (2012) pose problème par le manque d'adaptation au cours du temps. En effet aucune évolution des états désirés n'est prévue, le système supposant plutôt un désir de constance peu adapté pour le domaine des jeux 4X supposant une évolution constante de l'organisation dont l'IA a le contrôle.

Le système de fonctions d'utilité proposé par Nieves (2015) est très intéressant pour créer une perception du contexte personnalisée. L'exemple proposé est assez bas-niveau, son adaptation à un niveau stratégique reviendrait à utiliser le calcul pour noter différents objectifs ou différentes méthodes. L'inconvénient de cette méthode est la transparence du résultat : les fonctions d'utilité peuvent être opaques pour le game designer, et la combinaison difficile à maitriser (les implications de la modification d'un poids sont difficiles à prévoir, et la difficulté augmente avec le nombre de fonctions d'utilité et le nombre de paramètres). L'auteur note d'ailleurs que le travail pour ajuster les poids peut être long et contraignant car nécessitant de nombreux tests.

Aléatoire pondéré et apprentissage Le *dynamic scripting* s'adapte grâce à l'évaluation de l'exécution. Pour cela deux fonctions sont féfinies : une première fonction qui évalue le résultat obtenu par une méthode pour répondre à un objectif et une seconde fonction qui prend en paramètre la note résultant de cette évaluation pour modifier le poids associé à la méthode pour cet objectif. La technique permet également d'avoir une évolution car une fonction est régulièrement appelée pour sélectionner aléatoirement la méthode à utiliser pour chaque objectif existant. Les poids viennent pondérer cette décision de façon à ce qu'une règle ayant été efficace précédemment ait davantage de probabilité d'être choisie, tout en gardant une part d'aléatoire. Cette technique n'est pas très adaptée à notre problématique pour deux raisons : tout d'abord la sélection d'une méthode ne prend absolument pas en compte le contexte, elle est notée de façon absolue sur sa réponse à un objectif en supposant qu'elle s'appliquera de la même manière quelque soit la situation ; deuxièmement elle n'est pas idéale pour proposer un contrôle à un game designer puisque tous les contrôles se font par des fonctions qui nécessiteraient donc des connaissances en programmation.

Comparaison de contextes Le CBP du système DARMOK (Mehta et al., 2008; Virmani et al., 2008) propose d'associer les méthodes répondant aux objectifs à des contextes d'application symbolisant le contexte idéal pour appliquer chaque méthode. La sélection d'une méthode se fait alors par comparaison du contexte courant à ces contextes d'application. Celle dont le contexte d'application est le plus proche du contexte courant du jeu est choisie. Cette technique de sélection garantit un résultat unique. Elle est donc appropriée pour sélectionner une méthode pour répondre à un objectif, mais pas pour sélectionner plusieurs objectifs, comme nous en avons besoin pour un 4X.

Pré-conditions et sélection Les techniques FSM, HTN et ABL (A Behavior Language) proposent d'utiliser des pré-conditions pour choisir les méthodes adaptées au contexte courant. Elles se présentent sous la forme d'états du monde partiels qui doivent être respectés pour

pouvoir sélectionner la méthode associée. Lorsque plusieurs méthodes sont réalisables, alors chaque technique possède un moyen de les départager.

Le langage ABL utilisé par Weber et al. (2010) dont le fonctionnement est décrit par Mateas and Stern (2004) ajoute aux pré-conditions une spécificité permettant de départager deux méthodes. L'algorithme regarde d'abord quelles méthodes ont leurs pré-conditions qui sont satisfaites. S'il y en a plusieurs il choisit celle(s) ayant la plus haute spécificité. Enfin si plusieurs méthodes ont leurs pré-conditions satisfaites et possèdent la plus haute spécificité alors le choix est fait aléatoirement.

Dans un HTN, les méthodes d'un objectif sont évaluées dans un ordre prédéterminé lors de la conception du HTN. Lorsque les pré-conditions d'une méthode sont satisfaites, la méthode est sélectionnée et les suivantes ne sont pas évaluées. La définition d'un ordre entre les méthodes empêche donc l'apparition de conflits. Dans les HTN les pré-conditions sont donc utilisées pour sélectionner une méthode, mais elles sont également utilisées pour sélectionner des objectifs. En effet, les *task networks* permettant de définir une méthode utilisent également des pré-conditions pour déterminer lorsqu'un sous-objectif peut être poursuivi. Ces pré-conditions correspondent à un état du monde partiel mais également à d'autres objectifs qui doivent avoir été réalisés avant. Cela permet de définir un ordre partiel entre les objectifs d'une méthode.

Dans les FSM nous retrouvons cette notion d'ordre entre les objectifs, cependant la complétion de l'objectif n'est pas obligatoire pour en sélectionner un nouveau. Cela implique donc d'interrompre l'objectif en cours, aspect que nous abordons dans la partie suivante.

4.3.2 La remise en cause de la sélection

Nous abordons dans cette partie l'interruption d'un objectif ou d'une méthode. En effet, mais une fois qu'une décision est prise, plusieurs raisons vues dans la section 3.2.2 peuvent justifier de la modifier. La première raison est interne à la stratégie et consiste à adapter la sélection aux résultats que celle-ci produit. Mais le contexte peut également faire apparaître des menaces ou des opportunités qui justifient la décision par un niveau supérieur de changer sa sélection, interrompant prématurément la réalisation de la sélection actuelle. Nous allons voir dans quelles conditions et de quelles manières ces interruptions peuvent être effectuées.

La propagation d'un résultat L'interruption d'un objectif peut être déclenchée par l'interruption de ses propres sous-objectifs. Il doit alors retourner des informations exprimant son résultat afin de permettre aux niveaux supérieurs d'agir en conséquence. Pour les méthodes présentant une hiérarchie alternant objectifs et méthodes, c'est-à-dire les HTN, le CBP et le DS, lorsqu'un objectif échoue, une autre méthode permettant de le réaliser est si possible sélectionnée, sinon l'échec est propagé au niveau supérieur. Lorsqu'un objectif réussi, la méthode permet de déterminer si un autre sous-objectif est nécessaire ou si la méthode est finie, signifiant sa réussite qui est propagée au niveau supérieur.

Les FSM de Johnson (2006) peuvent avoir sur leurs transitions des conditions concernant la réussite ou l'échec des états. Si aucune transition ne peut être franchie, le résultat du FSM

inférieur s'applique au FSM courant et est transmis au FSM supérieur.

Les résultats des systèmes précédents sont binaires : l'objectif a soit réussi soit échoué. Cela peut suffire pour prendre une décision, cependant certains systèmes proposent d'enrichir ce résultat. Johnson (2006) propose par exemple à un état qui échoue de proposer un remplacement. Si c'est le cas, une transition sans condition est ajoutée de l'état courant à l'état suggéré puis retirée une fois la transition effectuée. Ontañón and Ram (2011) propose quant à lui un résultat plus nuancé sous la forme d'un réel compris entre zéro et un, zéro symbolisant l'échec complet et un la réussite. Leur utilisation par les niveaux supérieurs n'est cependant pas détaillé.

Cette propagation d'information sur les résultats obtenus par la mise en place d'une méthode est importante pour enrichir la prise de décision comme nous l'avons vu dans la section 3.2.2.

Les conditions de maintien Le CBP (Mehta et al., 2008) propose des conditions qui doivent être satisfaites pendant l'exécution d'une méthode : ce sont les *alive conditions*. Elles se présentent comme des pré-conditions sous la forme d'un état du monde partiel. Si l'une d'entre elles n'est plus respectée, alors la méthode est abandonnée pour être remplacée. Le langage ABL propose la même technique et les appelle des *context conditions*.

La technique du *goal-driven autonomy* (GDA) utilisée par Jaidee et al. (2011) possède un fonctionnement similaire : elle consiste à vérifier si l'état du monde observé est celui attendu, et si ce n'est cas le cas à supprimer le plan courant pour en créer un nouveau. Ce sont donc des conditions qui sont vérifiées pendant l'exécution et qui viennent l'interrompre si elles ne sont pas satisfaites.

Ces conditions suggèrent que les méthodes possèdent des conditions d'échec ne dépendant pas de la réussite des sous-objectifs mais d'une situation nécessaire à leur application.

L'interruption par un niveau supérieur Les FSM, comme noté précédemment, permettent d'interrompre un sous-objectif pour le remplacer par un autre en définissant des conditions qui, lorsqu'elles sont vérifiées, vont déclencher un changement de sous-objectif. Ces conditions peuvent inclure une condition de fin pour l'objectif en cours, mais, si elles n'en contiennent pas, vont déclencher son interruption. Cela peut cependant poser problème si une action ne doit pas être interrompue. Johnson (2006) propose pour résoudre ce problème de permettre à une FSM de spécifier une propriété nommée "critique" qui, lorsqu'elle est vraie, signifie que la FSM ne doit pas être interrompue. Les transitions sont alors prises en compte mais réalisées uniquement lorsqu'il n'y a aucune FSM critique en cours.

4.3.3 Conclusion

L'adaptabilité au contexte est un point clé d'une IA dans les jeux de stratégie 4X. Le contexte changeant de façon imprédictible, il est nécessaire de choisir des solutions en fonction de celui-ci. Afin de mettre entre les mains du game designer le contrôle de cette adaptation, la définition de contextes ou de pré-conditions semblent les méthodes les plus accessibles. Les FSM et les *task*

networks repérés dans la partie précédente grâce à leur représentation graphique de la décomposition proposent la définition de conditions et répondent donc bien à ce besoin d'adaptabilité au contexte.

Il est également important pour une stratégie de pouvoir remettre en cause celle appliquée comme nous l'avons vu dans la section 3.2.2. Nous avons déterminé que cela peut venir des résultats de l'exécution de la stratégie ou bien d'opportunités ou de menaces importantes apparaissant dans le contexte. La plupart des systèmes proposent la propagation de la réussite ou de l'échec, mais le résultat peut également être enrichi comme le propose Johnson (2006) et Ontañón and Ram (2011). Cela permet en effet de fournir davantage d'information pour la prise de décision des niveaux supérieurs. Certains systèmes proposent également la possibilité d'interrompre un comportement indépendamment des résultats, c'est le cas des FSM qui permettent ainsi de gérer les opportunités et les menaces.

4.4 L'allocation des ressources

Nous avons également défini que la stratégie vient ajouter des renseignements qui vont guider la répartition des ressources entre les objectifs sélectionnés. Dans cette partie nous allons comparer les méthodes utilisées dans les systèmes étudiés. Comme dans la partie précédente, nous nous intéressons en particulier aux paramètres que le game designer pourrait utiliser pour avoir un contrôle sur le calcul de répartition des ressources.

Search Stanescu et al. (2014) utilise l'algorithme de recherche heuristique MinMax pour répondre aux objectifs envoyés par le niveau supérieur. Cette technique utilise une simulation des tâches assignées pour estimer leur taux de réponse aux demandes de la stratégie. Pour cela elle simule plusieurs allocations des ressources sur une sélection d'un sous-ensemble de tâches. Cette méthode n'est pas transparente pour le game designer qui ne pourra pas comprendre la décision et la modifier. Le seul point de contrôle se situe sur l'évaluation de la qualité d'une solution mais elle se fait par le biais d'une fonction nécessitant des compétences en programmation.

Algorithme glouton Miles et al. (2007) utilisent un algorithme glouton en calculant les ratio bénéfice/coût et en attribuant les ressources aux objectifs du ratio le plus haut au plus bas. Des contrôles pourraient être fait sur l'évaluation du bénéfice d'une allocation, mais celle-ci nécessiterait également des compétences en programmation.

Enchères Des systèmes d'enchères sont utilisés par Bergsma (2008) et Stanescu et al. (2014). Ceux-ci consistent pour chaque tâche à exécuter à un instant t, à générer une offre pour des ressources qui reflète le bénéfice qui sera reçu si les ressources lui sont affectées, Bergsma (2008) parle d'ailleurs de fonction d'utilité. L'une des difficultés pointées par Bergsma (2008) est d'unifier les offres faites par les différents modules puisque l'utilité est calculée par chaque module tactique. Il propose alors un module dont le rôle est d'unifier les offres en affectant un poids à

chaque module tactique qui viendra pondérer ses offres. Ils proposent d'utiliser un algorithme d'apprentissage pour générer automatiquement ces poids ou bien de les définir manuellement. Cette dernière option permet de fournir un contrôle au game designer mais nécessite plusieurs essais pour ajuster les poids.

Priorité des buts Dill (2006) propose d'assigner les ressources en fonction de priorités assignées aux différents objectifs. Pour cela il détermine un calcul de priorité de base dépendant de nombreux facteurs contextuels. Il détermine également un calcul de priorité courante qui tient compte des ressources allouées. Il prend pour exemple une attaque : sa priorité de base va dépendre des ennemis à attaquer (leur force, leur vie...) et la priorité courante va dépendre du ratio entre les ennemis attaqués et les soldats utilisés pour attaquer. L'allocation des ressources utilise alors dans un premier temps un algorithme glouton qui va assigner des ressources aux objectifs de la priorité de base la plus haute à celle la plus basse. Si des ressources sont assignées elles doivent permettre d'avoir une priorité courante au minimum égale à la priorité de base. Une deuxième phase va alors tester toutes les modifications possibles et les effectuer uniquement si elles permettent d'augmenter la somme de toutes les priorités courantes. Il faut noter que cet algorithme a l'avantage d'être anytime puisque la phase la plus longue est l'optimisation des allocations mais celle-ci peut éventuellement être interrompue si nécessaire et des allocations possibles pourront être fournies. L'inconvénient de cette méthode est la complexité de l'optimisation des priorités qui limite la transparence des résultats. Le calcul de priorité de base est transparent et s'adapte au contexte, mais les interactions entre les priorités courantes sont difficiles à maîtriser.

Politique d'investissement Souza et al. (2014) proposent quant à eux un système appelé PICFlex qui considère les ressources utilisées pour effectuer des actions comme des investissements. Des politiques d'investissement permettent d'attribuer des pourcentages d'investissement à différents domaines nécessitant donc un découpage thématique. Le système va donc réguler les demandes de ressources grâce à une politique d'investissement. Ils proposent plusieurs variantes montrant que le respect de la politique peut être plus ou moins strict : en plus des exemples extrêmes (n'acceptant jamais une demande excédant la politique ou acceptant toutes les demandes), ils proposent par exemple une méthode n'acceptant qu'une demande au delà de ce que permet la politique et n'autorisant un nouvel écart que lorsque le précédent a été "remboursé" (le pourcentage dépensé pour le domaine est repassé sous celui défini dans la politique). Ils proposent également de faire évoluer cette politique au cours du jeu en définissant manuellement plusieurs politiques et en les associant à des contextes à la manière du CBR. Les contextes proposés dans leur exemple sont cependant des évaluations des informations brutes sur l'état du jeu : paix, sous attaque (défenseur) ou attaque lancée (attaquant). Cette approche est très intéressante car elle permet une répartition des ressources personnalisée avec des simples pourcentages rendant sa définition très accessible.

Préférences Gordon (2006) utilise de simples priorités pour gérer les conflits de ressources entre des objectifs et n'approfondit pas leur fonctionnement, cependant un autre point intéressant est détaillé : les requêtes de ressources. En effet un objectif peut demander une ressource sous la forme d'un type de ressource et de deux listes de propriétés. La première permet de détailler les propriétés indispensables à la ressource pour permettre de répondre au besoin de l'objectif, la seconde représente des préférences, c'est-à-dire des propriétés qui ne sont pas indispensables pour son utilisation mais qui seront préférées si elles sont présentes. L'exemple donné est la requête d'un véhicule (type de ressource) de type voiture (indispensable) et de couleur rouge (optionnel) pour un objectif de déplacement. L'auteure ajoute également la possibilité de définir des préférences sur des nombres en proposant une représentation sous forme d'intervalles en précisant des limites fixes ou souples, les premières représentant une condition indispensable et les secondes optionnelle. Elle y ajoute la possibilité de définir si le nombre doit être le plus petit ou le plus grand possible. Les propriétés indispensables peuvent être définis par l'objectif lui-même puisque c'est l'objectif qui détermine ce dont il a besoin, cependant les préférences peuvent être définies par un game designer qui souhaite définir une personnalité pour l'IA.

Conclusion En ce qui concerne l'allocation des ressources, nous observons principalement que les modules définissent leurs besoins en ressources et que le module supérieur va chercher à y répondre en les attribuant du plus important au moins important. Pour permettre au game designer d'affecter des importances il est nécessaire d'utiliser une décomposition prédéfinie. Plusieurs travaux existants proposent des contrôles externes intéressants, dont certains peuvent être combinés.

Bergsma (2008) propose par exemple d'affecter un poids à chaque module tactique qui va venir influencer l'importance donnée à ses demandes de ressources. Dill (2006) propose quant à lui de définir des priorités de base dépendantes du contexte et des priorités courantes dépendantes des ressources allouées, cependant le fonctionnement de ces dernières peut sembler un peu trop complexe pour avoir un contrôle transparent dessus. Ces deux méthodes permettent donc d'influencer la priorité que vont avoir les différents domaines. Souza et al. (2014) propose également d'affecter des poids aux domaines cependant ceux-ci ne sont pas utilisé pour définir une priorité mais pour définir des pourcentages de répartition. Si les méthodes précédentes avaient davantage fonction à déterminer quel module serait servi en premier, cette méthode ne détermine pas un ordre mais des proportions. Enfin les derniers travaux que nous retenons sont ceux de Gordon (2006) qui propose d'enrichir les demandes en y distinguant des éléments indispensables et des éléments optionnels permettant d'ajouter une souplesse et une personnalisation de l'allocation des ressources.

4.5 Bilan du chapitre

On peut retrouver de nombreux points de similitudes entre les caractéristiques de la stratégie que nous avons explorées dans le chapitre 3 et la mise en place de prise de décision dans les jeux de

stratégie. Nous retrouvons en effet dans ces jeux un contexte particulièrement adapté à la mise en place d'une stratégie et présentant donc la même complexité de prise de décision. Celle-ci appelle à la décomposition qui peut prendre de nombreuses formes, à la fois hiérarchique ou fonctionnelle, et nous avons détaillé dans la partie 4.2 les décompositions proposées par les systèmes étudiés. La **décomposition hiérarchique** permet une adaptation des méthodes choisies pour répondre aux objectifs, alors que la décomposition fonctionnelle permet un parallélisme des problèmes à traiter. Les FSM et les *task networks* proposent une représentation graphique intéressante pour un contrôle accessible à tous, cependant les FSM ne proposent pas la parallélisation et les *task networks* ne permettent pas les choix entre plusieurs alternatives. La **décomposition fonctionnelle** est dépendante du jeu, de la perception que l'on se fait du problème posé par le jeu et de l'IA que l'on souhaite obtenir. Il est nécessaire que cette décomposition soit entièrement créer par la personne concevant l'IA, soit dans notre cas le game designer. Elle consiste à définir les différents domaines et, selon les systèmes étudiés, on peut leur donner des poids, ou définir des interactions. Les deux types de décomposition ne sont pas exclusifs et peuvent être combiner.

Nous avons déterminé dans le chapitre 3 que l'adaptabilité de la stratégie au contexte est indispensable. Dans ce chapitre, nous avons examiné l'**adaptation de la sélection** au contexte et la **remise en cause de la sélection** une fois qu'elle est faite. Pour le choix d'un objectif ou d'une méthode, les pré-conditions sont la technique la plus utilisée et la plus accessible. Les FSM et les *task network* proposent également de combiner les pré-conditions avec la notion d'ordre en reliant les objectifs entre eux. Pour la remise en cause de la sélection, celle-ci peut venir de l'objectif lui-même qui détermine le résultat de son exécution, ou bien d'un évènement extérieur faisant apparaitre une opportunité ou une menace. Le résultat de l'exécution est bien souvent un résultat binaire de succès ou d'échec, mais des travaux proposent d'y ajouter des informations ou de le nuancer pour enrichir la prise de décision. L'interruption de la sélection n'est possible que par les FSM. Johnson (2006) y ajoute tout de même des exceptions empêchant l'interruption lors de situations critiques.

Dans la dernière partie, nous nous sommes intéressés aux méthodes d'allocation des ressources et aux paramètres qui permettent de fournir un contrôle extérieur. Les systèmes étudiés utilisent des techniques très variées dont certaines peuvent être combinées. Nous retenons en particulier trois notions permettant un contrôle extérieur intelligible pour des personnes ne possédant pas de compétences en programmation. Ces trois notions ajoutent des informations sur plusieurs objectifs parallèles afin de guider l'allocation de ressources entre ces objectifs. La première notion est celle de **priorité** qui vient ordonner les objectifs afin que ceux ayant une plus grande priorité reçoive plus facilement ses requêtes. La deuxième notion est celle de **politique d'investissement** proposée par Souza et al. (2014) qui consiste à associer des pourcentages à chaque domaine afin de symboliser les quantités de ressources qu'ils peuvent recevoir de manière relative aux autres. Enfin la dernière notion est celle de **préférences** de Gordon (2006) qui propose plusieurs types de préférences qui viennent assouplir et personnaliser les requêtes de ressources.

Dans ce chapitre nous avons établi un état de l'art des études qui proposent des systèmes adaptés aux jeux de stratégie et nous nous sommes particulièrement intéressés à ceux qui permet-

tent un contrôle extérieur intelligible pour des personnes sans compétences de programmation. Nous avons mis en parallèle ces travaux avec l'utilisation commune du terme *stratégie* qui nous a permis d'identifier les points indispensables pour définir une bonne stratégie. Cet état de l'art a pour but de guider la conception d'un modèle de stratégie afin de rendre la conception d'IA stratégique transparente pour les game designers. Dans le chapitre suivant nous détaillerons notre proposition inspirée des travaux présentés dans ce chapitre.

Chapitre 5

Méthode de création et d'exécution d'un raisonnement stratégique

L'objectif de cette thèse est de permettre une création plus efficace des IA dans les jeux de stratégie commerciaux et en particulier pour les jeux 4X. Nous avons étudié la conception et la création d'un jeu et en particulier des IA dans le chapitre 2. Nous avons pu voir que les jeux commerciaux ont pour but de proposer une expérience divertissante aux joueurs et que les comportements des personnages non-joueurs jouent un rôle primordial dans cette expérience. Dans les jeux de stratégie nous nous intéressons aux adversaires contrôlés par une IA. C'est le game designer qui est l'architecte de l'expérience mais la création du jeu est ensuite déléguée à toute une équipe. Deux phases peuvent alors être distinguées : la phase de conception qui est la définition théorique du jeu, et la phase de création qui est la réalisation informatique du jeu imaginé en phase de conception. Le rôle du game designer est d'assurer que le jeu créé produit l'expérience imaginée en phase de conception.

Afin de simplifier le rôle du game designer, nous proposons de lui donner un contrôle direct partiel sur la création des IA. Pour cela nous avons déterminé qu'une séparation hiérarchique de la prise de décision est nécessaire afin de fournir un contrôle haut-niveau dit stratégique au game designer. Dans le chapitre 3 nous avons défini qu'une stratégie fixe des **objectifs haut-niveau** et donne des informations guidant l'**allocation des ressources** tout en s'adaptant au **contexte dynamique**. Dans ce chapitre nous allons présenter la proposition de cette thèse qui prend la forme d'une méthode qui regroupe les phases de conception et de création en permettant au game designer de représenter la stratégie imaginée de manière à ce qu'elle puisse être utilisée par les autres modules IA s'occupant des niveaux inférieurs de la prise de décision. La vue d'ensemble de cette méthode est présentée dans la partie 5.1, puis les deux composantes de la proposition que sont le modèle de stratégie et le moteur permettant de l'exécuter, sont détaillées dans les parties 5.2 et 5.3.

5.1 La vue d'ensemble

Notre méthode propose un contrôle au game designer sur le niveau stratégique de l'IA, ce qui suppose que les développeurs IA ont toujours la responsabilité de l'implémentation de la prise de décision inférieure, souvent découpée en raisonnement tactique et raisonnement opérationnel. Notre méthode doit donc fournir une interface accessible à des personnes ne possédant pas de compétences de programmation qui permette de définir un raisonnement stratégique, mais elle doit également fournir une interface de programmation applicative (API) permettant à une stratégie de fonctionner avec d'autres modules IA inférieurs hiérarchiquement dans la prise de décision.

Pour cela, nous avons divisé notre système représenté sur la figure 5.1 en deux parties. La première est un **modèle de stratégie** qui définit un formalisme destiné au game designer pour décrire une stratégie. Le terme *stratégie* sera par la suite utilisé pour faire référence à une stratégie respectant ce modèle. La deuxième partie est le **moteur de stratégie** qui permet, durant le jeu, d'utiliser une stratégie pour guider la prise de décision d'un des opposants. Lors de la définition de notre moteur nous avons dû par extension préciser les modalités de coopération entre notre moteur et un module IA externe que nous appelons module tactique, et cela sous la forme d'une API.

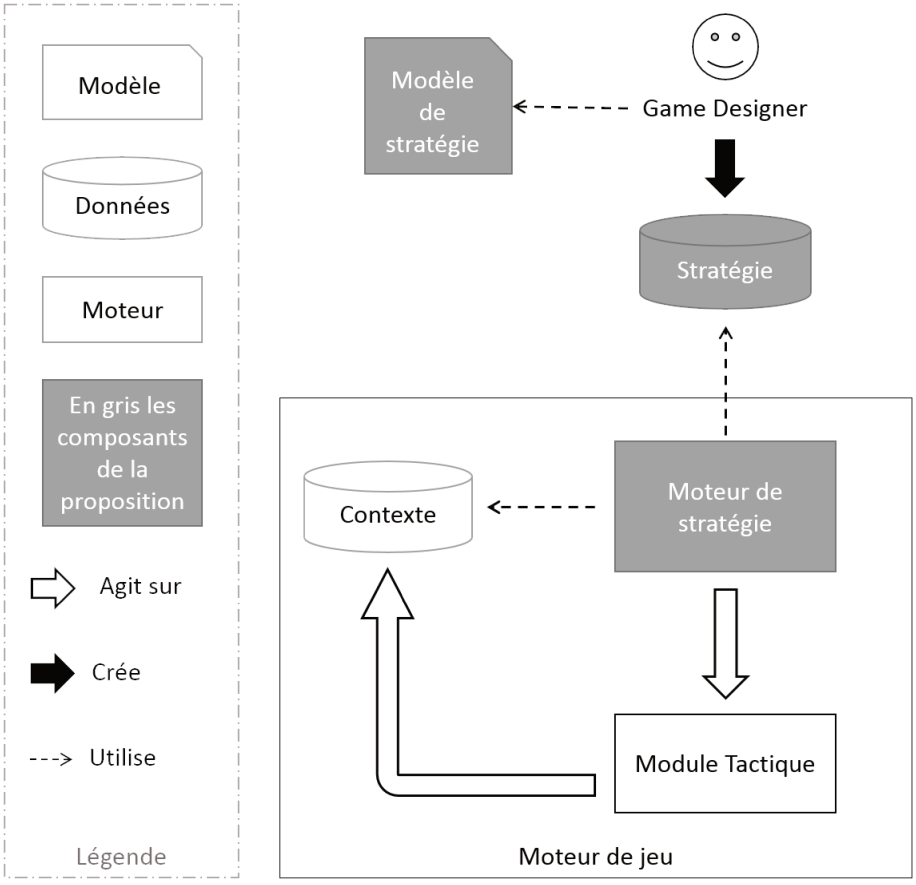


FIGURE 5.1: Organisation des modules

Notre objectif pour la définition de cette méthode est de proposer un ensemble **transparent** au sens décrit par Isla (2008) pour le game designer. Cela signifie qu'il doit être capable :

- de comprendre les étapes qui ont amené à une décision,
- de prédire également une décision en fonction du contexte,
- de savoir comment modifier un comportement,
- et de reconnaître une erreur et de la résoudre.

Afin de maximiser l'utilité de notre méthode, il est également nécessaire que le contrôle proposé soit suffisamment **expressif** pour que le game designer n'est pas besoin de l'intervention d'un développeur pour effectuer les modifications qu'il souhaite apporter. Cependant l'expressivité peut nuire à la transparence en surchargeant le game designer d'informations interdépendantes complexifiant ainsi la compréhension d'une stratégie.

La question de recherche qui en découle est la suivante : **Peut-on proposer un modèle de stratégie aux game designers qui soit suffisamment expressif pour leurs besoins mais qui reste suffisamment transparent pour être utilisé sans posséder de connaissances de programmation ?**

5.2 Le modèle

Dans cette section nous allons détailler le modèle de stratégie permettant aux game designers d'exprimer une stratégie destinée à une IA dirigeant un adversaire dans un jeu 4X. Afin de le définir nous nous sommes inspirés des utilisations de la stratégie dans le domaine courant ainsi que dans le domaine des jeux de stratégie. De son utilisation dans le domaine courant étudiée dans la partie 3, nous avons extrait deux aspects fondamentaux :

- La **cohérence** : une stratégie s'applique à un système complexe et permet de l'organiser en définissant comment les objectifs et les ressources doivent être répartis, impliquant une décomposition du système. Les décisions prises par le système présentent un ensemble cohérent poursuivant un même objectif.
- L'**adaptabilité** : une stratégie s'inscrit dans le temps et doit donc évoluer en fonction des résultats obtenus et de l'environnement dynamique dans lequel se situe le système.

Nous avons donc décidé d'utiliser ces aspects qui sont communs aux définitions de la stratégie pour définir les bases de notre modèle. Ce dernier doit donc permettre la définition de règles contrôlant l'adaptation du choix des objectifs et de la répartition des ressources en fonction du contexte et des résultats. Pour répondre à ces contraintes, nous allons nous inspirer des travaux étudiés dans le chapitre 4 et des solutions qu'ils proposent.

5.2.1 La décomposition

Une stratégie s'applique à un système complexe qui nécessite donc d'être décomposé. Nous avons pu distinguer deux types de décomposition dans les systèmes étudiés dans le chapitre 4 : hiérarchique et fonctionnelle. Nous avons en particulier retenu la décomposition hiérarchique par objectifs et la décomposition en modules thématiques. La première permet de définir comment répondre à un objectif en définissant un ensemble de sous-objectifs et quand chacun doit être sélectionné. La seconde permet de définir des prises de décision qui se font en parallèle. Les deux types de décompositions ne sont pas incompatibles c'est pourquoi nous avons décidé de les combiner pour pouvoir bénéficier des avantages de chacun.

Afin de combiner les deux approches, nous avons défini une stratégie comme une hiérarchie dont les composants sont des *comportements*. L'un des comportements est le point d'entrée de la stratégie, il se situe au niveau hiérarchique le plus haut, nous l'appelons *comportement racine*.

Nous pouvons ainsi formaliser une stratégie comme un tuple $\langle B, b_0, \Phi \rangle$ où :

- B est un ensemble fini de *comportements*,
- b_0 est le comportement *racine*,
- Φ est un ensemble fini de relations $b_x > b_y$ indiquant que b_x est un comportement *parent* de b_y.

Une stratégie n'est valide que si $\forall b_x \in B : x = 0 \vee \exists b_y > b_x \in \Phi$. Un comportement b_x se décompose alors en l'ensemble des comportements B_x tel que $\forall b_y \in B_x \exists b_x > b_y \in \Phi$.

Un comportement peut se décomposer en un ensemble de comportements parmi lesquels une sélection devra être faite, comme pour les hiérarchies d'objectifs que nous avons évoqués précédemment : nous l'appellerons *comportement logique*. Un comportement peut se décomposer en plusieurs comportements devant être réalisés en parallèle, comme pour la décomposition thématique : nous l'appellerons *comportement parallèle*. Un comportement possède donc un ensemble de comportements appartenant au niveau hiérarchique inférieur que nous appellerons les *sous-comportements* afin de clarifier l'ordre hiérarchique. Pour exprimer la relation inverse nous ferons référence au *comportement supérieur*. Un comportement ne possédant aucun sous-comportement correspond à un objectif qui sera ensuite transmis au module tactique lorsqu'il est sélectionné : nous l'appellerons comportement élémentaire. La figure 5.2 représente un exemple de stratégie possible avec lequel nous allons illustrer notre modèle dans la suite de cette partie. Les ronds symbolisent des comportements logiques, les carrés des comportements parallèles, et les losanges des comportements élémentaires.

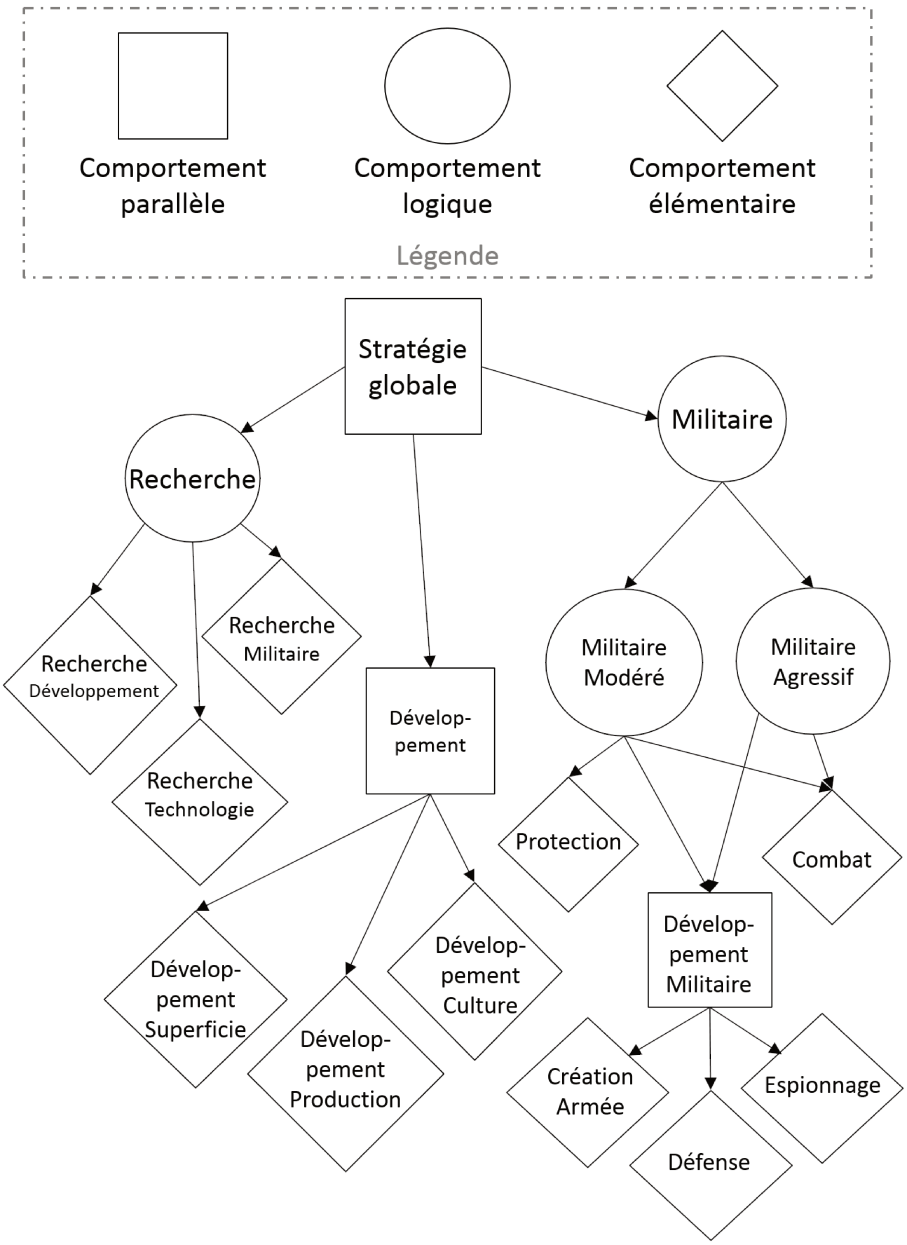


FIGURE 5.2: Exemple d'une hiérarchie de comportements

5.2.2 L'adaptation

Nous avons défini dans le chapitre 3 la nécessité de s'adapter à la fois au contexte dynamique et aux résultats obtenus par la réalisation de la stratégie courante. Dans la partie 4.3 nous avons étudié la capacité d'adaptation de plusieurs techniques appliquées à la décision stratégique dans les jeux de stratégie. La technique des FSM est celle qui présente la plus grande adaptabilité. En effet elle permet de s'adapter au résultat de la sélection actuelle et au contexte observé, mais son grand avantage par rapport aux autres techniques est sa capacité à interrompre le comportement en cours si la situation le justifie. De plus c'est une méthode bien connue dans le milieu du jeu vidéo qui a déjà été utilisée pour sa facilité de prise en main.

Une FSM est composée d'états et de transitions qui relient un état source à un état destination. Elle permet de sélectionner un état à la fois en définissant un état qui est initialement sélectionné et des conditions associées aux transitions qui spécifient lorsque la sélection doit passer de l'état source à l'état destination. On peut la formaliser par un tuple $\langle \Sigma, S, s_0, \Gamma \rangle$ où :

- Σ est un ensemble de symboles,
- S est un ensemble d'*états*,
- s_0 est l'état *initial*,
- $\Gamma : S \times \Sigma \rightarrow S$ est la fonction de transition.

Dans notre cas les états symbolisent les sous-comportements, et les conditions, comme défini précédemment, permettent d'indiquer lorsqu'un changement du contexte modifie la pertinence du sous-comportement actuellement sélectionné ou lorsque les résultats, et éventuellement le contexte, justifient la sélection de l'état destination. Par la suite nous allons développer l'exemple du comportement *Militaire Modéré* qui possèdent trois sous-comportements : *Développement Militaire*, *Protection* et *Combat* dans notre hiérarchie de comportements présentée sur la figure 5.2. Nous commençons donc en créant les états associés comme représentés sur la figure 5.3.

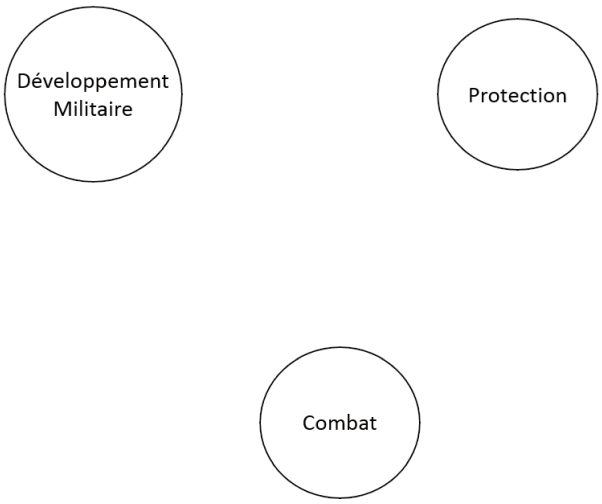


FIGURE 5.3: Les états de notre exemple de comportement logique

L'adaptation au contexte dynamique

L'observation du contexte peut s'effectuer par l'accès direct aux données brutes, mais il peut également être utile d'avoir accès à des données analysées (e.g. rapport de force militaire avec un adversaire). Dans ces travaux nous considérons que les informations proviennent d'un contexte possédant potentiellement les deux types de données puisque la nature des informations ne vient pas influencer le fonctionnement des FSM, nous nous intéressons uniquement à la forme que prennent ces données. Nous considérons que le contexte est un ensemble de paramètres $\langle n, v \rangle$ où n est le nom du paramètre et v sa valeur. Les valeurs peuvent être au choix des nombres ($\mathbb{Z}$ ou $\mathbb{R}$), des chaînes de caractères, ou bien des booléens. Pour exprimer un état du monde nécessitant une adaptation de la stratégie, ces données sont associées afin de former des prédicats, grâce à une grammaire regroupant les opérateurs logiques AND, OR, NOT ainsi que les opérateurs de comparaison $<$, $>$, $=$ et $\neq$. Des nombres et chaînes de caractères ne provenant pas du contexte peuvent également y être ajoutés. Nous pouvons par exemple compléter notre comportement logique *Militaire Modéré* avec deux transitions comme illustré sur la figure 5.4 :

- une transition permettant de passer du développement de l'armée (sous-comportement *Développement Militaire*) à une phase de protection (sous-comportement *Protection*) lorsque la puissance militaire est considérée suffisante (nous avons ici supposé qu'un paramètre *puissance militaire* était disponible dans le contexte et nous avons défini une valeur considérée suffisante de 10 de façon arbitraire afin d'illustrer notre exemple),
- une transition permettant de passer de la phase de protection (sous-comportement *Protection*) à une phase d'attaque (sous-comportement *Combat*) lorsque la relation diplomatique est différente d'une relation de paix et qu'un avantage est détecté (nous avons ici aussi supposé qu'un paramètre *rapport de force* était disponible dans le contexte et nous avons défini une valeur considérée avantageuse de 0.7 de façon arbitraire afin d'illustrer notre exemple).

Les opérateurs logiques permettent également de combiner des conditions sur le contexte avec des conditions sur les résultats comme nous allons le voir dans le paragraphe suivant.

L'adaptation aux résultats

Lors de son exécution, un état peut retourner des informations. Les résultats proposés par les systèmes étudiés dans le chapitre 4 se résumaient souvent à un résultat binaire de réussite ou d'échec. Pour déclencher une transition lorsqu'un résultat est reçu, celui-ci doit simplement être indiqué sur cette transition et peut être combiné avec d'autres conditions à l'aide des opérateurs logiques. Dans notre exemple complété sur le figure 5.5, nous utilisons l'issue du combat transmis par le sous-comportement *Combat* que nous associons à des conditions du contexte pour savoir si l'armée doit être renforcée.

Afin d'augmenter l'expressivité des résultats, nous avons décidé d'utiliser des messages sous la forme de chaînes de caractères. Ces résultats diffèrent également des systèmes existants car

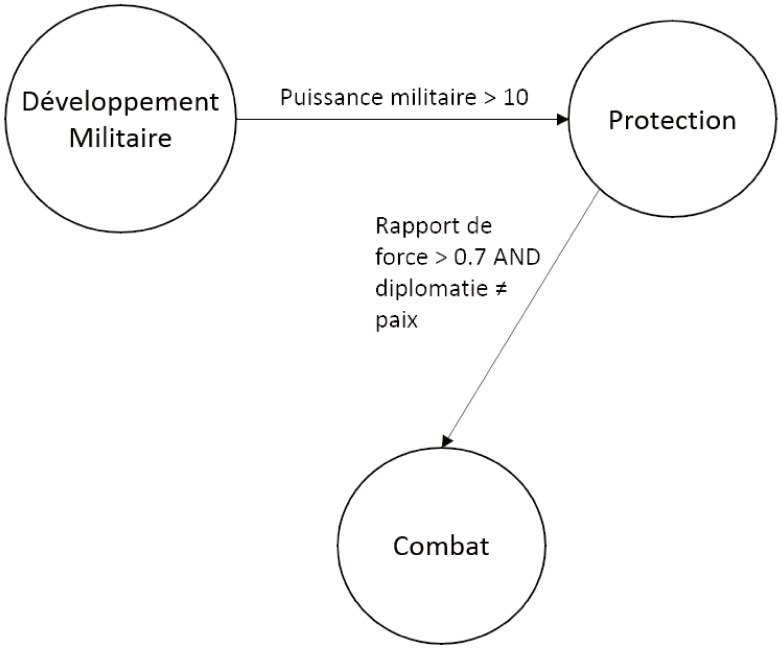


FIGURE 5.4: Les transitions sur l'état du contexte de notre exemple de comportement logique

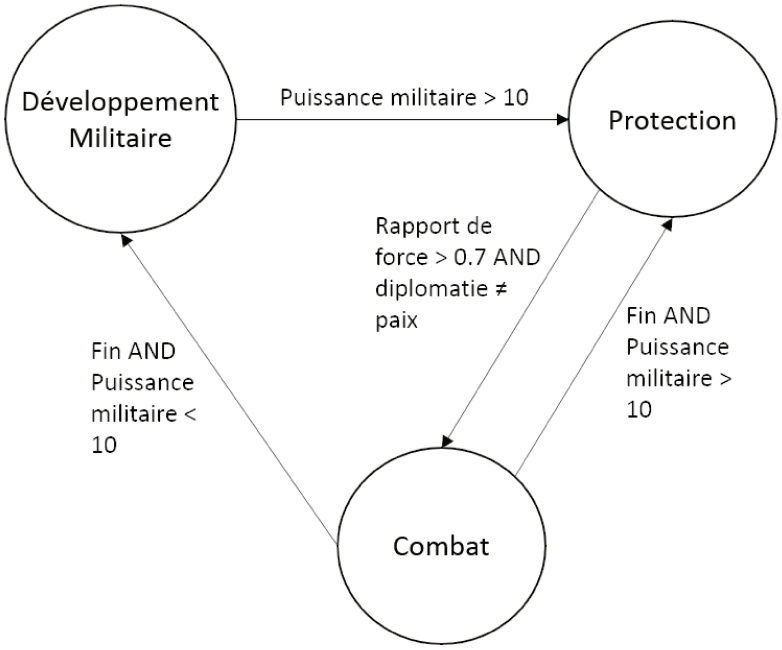


FIGURE 5.5: Les transitions sur le résultat du combat

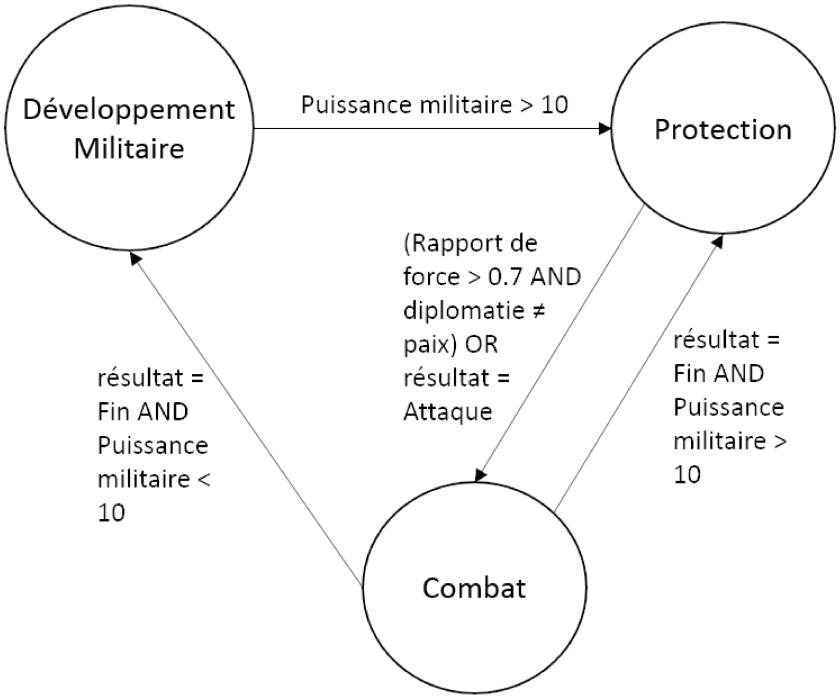


FIGURE 5.6: Les transitions sur les résultats enrichis

ils peuvent être envoyés pendant l'exécution d'un comportement et pas uniquement lorsque son exécution est finie. Cela permet de déléguer au comportement supérieur la responsabilité d'évaluer si une interruption est justifiée. Dans notre exemple représentée sur la figure 5.6 le sous-comportement *Protection* ne fait qu'alerter de la détection d'une attaque, c'est notre comportement *Militaire Modéré* qui prend la décision de modifier le comportement en évoluant dans un comportement *Combat*. Nous avons également modifié le format des prédicats pour préciser lorsqu'une chaine de caractères fait référence à un message en la comparant avec *message*.

Les messages peuvent provenir du module tactique, mais il peut également être utile pour une FSM de renseigner son comportement supérieur lorsque sa sélection change. Par exemple, il peut être utile pour notre comportement *Militaire Modéré* de transmettre l'information qu'une attaque est en cours au comportement supérieur (le comportement *Militaire*). Il sera alors de la responsabilité du comportement supérieur de déterminer si l'information est utile. Pour pouvoir envoyer un message au comportement supérieur, les états de nos FSM sont enrichis d'un message optionnel qui est envoyé au niveau supérieur lors de sa sélection. Ces messages prennent également la forme d'une chaîne de caractères et sont donc utilisés par les FSM de la même manière que ceux envoyés par le module tactique. La figure 5.7 montre donc notre comportement *Militaire Modéré* auquel nous avons ajouté un message *Attaque* sur l'état comportant le sous-comportement *Combat* de façon à ce que le comportement supérieur soit prévenu lorsqu'un combat est engagé.

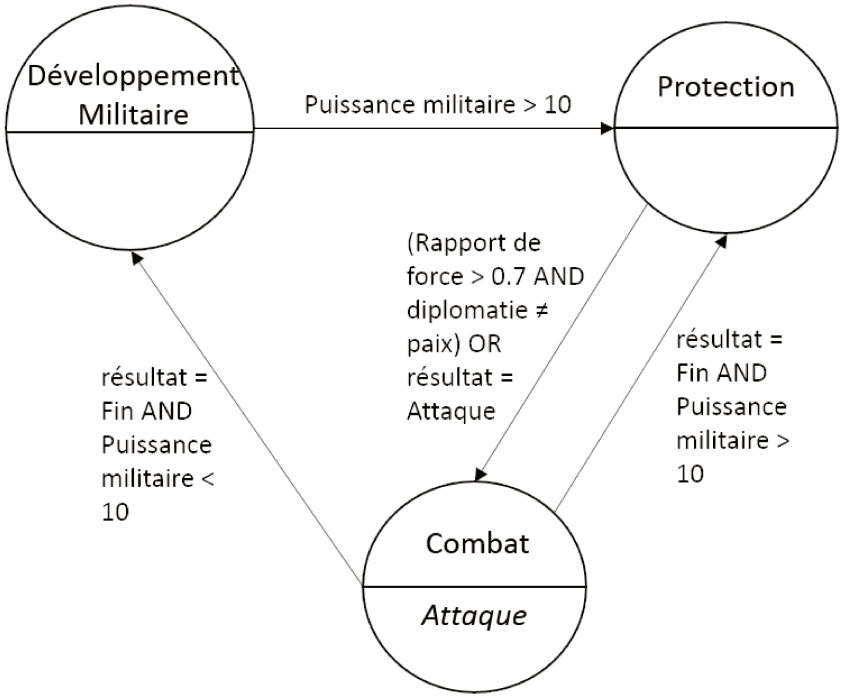


FIGURE 5.7: Les états enrichis permettent d'envoyer des messages au comportement supérieur

Conclusion

Pour résumer nous pouvons formaliser un comportement logique comme un tuple $\langle SB, Mo, Mi, \Sigma, \Psi, S, s_0, \Gamma \rangle$ où :

- SB est un ensemble de comportements,
- Mo est un ensemble de messages sortants,
- Mi est un ensemble de messages entrants,
- Σ est un ensemble de conditions externes,
- Ψ est un ensemble de prédicats composés de conditions externes $\in \Sigma$ et de messages $\in Mi$,
- S est un ensemble d'*états* décrits par des tuples $\langle sb, m \rangle$ tel que $sb \in SB$ et $m \in Mo$ ou peut être null,
- s_0 est l'état *initial*,
- $\Gamma : S \times \Psi \rightarrow S$ est la fonction de transition.

5.2.3 La répartition des ressources

Le besoin de répartir les ressources apparaît lorsque plusieurs comportements s'effectuent en parallèle : dans notre modèle cela correspond aux comportements parallèles. Afin de répartir les ressources il est nécessaire d'avoir des informations sur les ressources disponibles d'une part, et sur les besoins de chaque comportement d'autre part. Les besoins en ressources peuvent

être propagés en partant du bas de la hiérarchie et en agrégeant à chaque niveau les besoins du niveau inférieur. Si l'on considère qu'un comportement n'a besoin que de la somme des ressources demandées par ses sous-comportements, les seules informations nécessaires pour connaître les besoins d'un système sont alors les besoins de ses comportements élémentaires. La répartition des ressources revient alors à faire coïncider les besoins des comportements élémentaires avec les ressources disponibles. Pour guider cette répartition, il est nécessaire d'ajouter des informations supplémentaires sur les comportements parallèles que nous allons présenter et illustrer en nous servant du comportement *Développement Militaire* de notre exemple introduit par la figure 5.2.

Combien de ressources allouer ?

La difficulté de la répartition des ressources intervient lorsque plusieurs comportements fonctionnant en parallèle ont besoin des mêmes ressources. Deux situations sont alors possibles : ou bien le système possède suffisamment de ressources pour en affecter aux deux comportements, ou bien les ressources disponibles ne sont pas suffisantes et des informations supplémentaires sont alors nécessaires pour choisir à quel comportement attribuer chaque ressource.

Les algorithmes gloutons sont utilisés dans plusieurs techniques présentées dans la partie 4.4, ils consistent à traiter les demandes par ordre de priorité et nécessite pour cela de définir ces priorités. Appliqués à la répartition des ressources entre plusieurs comportements, ils consistent à assigner des priorités aux comportements et à assigner les ressources dont ils ont besoin les uns après les autres dans l'ordre de priorité. Notre exemple de comportement *Développement Militaire* possède trois sous-comportements : *Création Armée*, *Défense* et *Espionnage*. Nous pouvons par exemple définir que *Création Armée* est de priorité 1 et que *Défense* et *Espionnage* sont de priorité 2. *Création Armée* se verra donc allouer les ressources dont il a besoin avant que *Défense* et *Espionnage* ne puissent en demander.

Un algorithme glouton basique suppose soit un ordre strict, c'est-à-dire qu'il n'est pas possible d'avoir deux comportements de même priorité, soit une sélection aléatoire lorsqu'il y a une égalité. Pour notre exemple, ces solutions ne sont pas suffisantes car nous risquerions d'affecter toutes les ressources militaires au plus prioritaire et ne rien laisser pour l'autre. Une solution raisonnable serait alors de répartir les ressources en deux groupes de tailles égales pour en allouer un à chaque comportement. Cette méthode manque cependant de souplesse pour donner une expressivité suffisante à notre modèle et nous souhaitons l'enrichir pour offrir davantage d'options. Pour cela nous nous inspirons des travaux de Souza et al. (2014) qui proposent de définir des politiques d'investissement consistant à affecter un pourcentage à chaque domaine pour répartir les ressources. Dans notre cas cela revient à affecter des pourcentages aux comportements *Défense* et *Espionnage*. L'expression sous forme de pourcentage peut amener à des erreurs puisque la somme doit être égale à 100, il nous a semblé plus simple pour l'utilisateur de laisser libre les nombres affectés aux comportements, que nous appelons les *importances*, et d'utiliser la méthode suivante : l'ensemble des ressources disputées est réparti de façon égal en plusieurs parts, et l'importance d'un comportement représente le nombre de parts auxquelles il

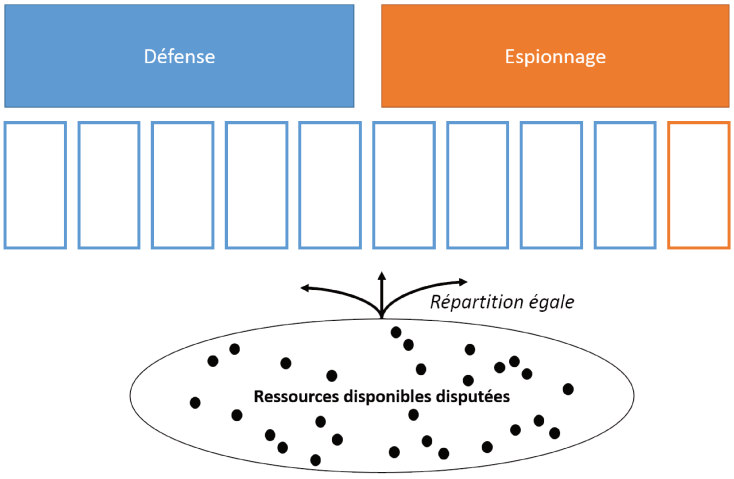


FIGURE 5.8: L'utilisation de l'importance dans la répartition des ressources

a droit. Le nombre de parts total est donc égal à la somme des importances. De façon formelle nous pouvons définir que pour chaque priorité il existe un ensemble $P = ((sb_1, i_1), ..., (sb_n, i_n))$ pour lequel le nombre de parts identiques doit être égal à $\sum_{x=1}^{n} i_x$. Dans notre exemple, si nous assignons une importance de 9 à *Défense* et 1 à *Espionnage*, les ressources militaires seront divisées en 10 groupes, soit 9 + 1, dont 9 seront affectés à *Défense*, et 1 à *Espionnage*. La figure 5.8 illustre cette répartition.

Dans un comportement parallèle, chaque sous-comportement a donc besoin d'avoir une priorité, et si plusieurs sous-comportements possèdent la même priorité, il est nécessaire de leur assigner des importances. Ces deux paramètres sont importants lorsqu'il existe un conflit lors de la demande des ressources. Lorsque deux sous-comportements demandent les mêmes ressources mais ont une priorité différente, le comportement le plus prioritaire les obtiendra toutes. Lorsque deux sous-comportements demandent les mêmes ressources et ont une priorité identique, les ressources seront partagées en plusieurs groupes qui seront distribués entre les deux sous-comportements. Ces deux paramètres présentent des contrôles différents et complémentaires.

Quelles ressources allouer

Lors de la répartition des ressources, il faut également choisir, en plus du nombre de ressources, quelles ressources attribuer à chaque comportement. Nous avons alors augmenté l'expressivité en nous inspirant des travaux de Gordon (2006) qui propose de rendre l'allocation de ressources plus souple et personnalisée en définissant deux types d'informations lors d'une requête de ressource : les caractéristiques indispensables, et celles optionnelles mais préférées. Les caractéristiques indispensables sont définies dans notre modèle par les comportements élémentaires soit par les développeurs, mais le game designer peut vouloir préciser des caractéristiques optionnelles pour créer des personnalités d'IA différentes en introduisant des différences subtiles entre les comportements. Pour notre comportement d'espionnage, nous pouvons par exemple préférer utiliser un négociateur car nous allons vouloir, s'il rencontre une unité adverse, que l'interaction

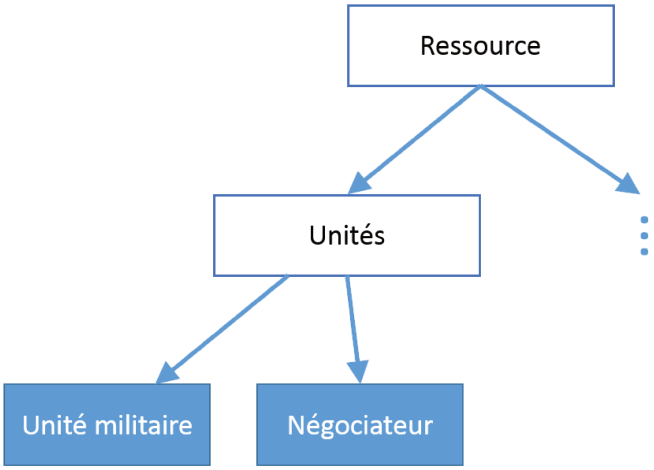


FIGURE 5.9: La hiérarchie de ressources

soit pacifique et positive pour les relations diplomatiques. Cela permet de ne pas bloquer l'action si aucun négociateur n'est disponible, mais de proposer une expérience particulière choisie par le game designer s'il y en a un.

Pour cela nous avons besoin de construire une hiérarchie de types de ressource qui formalise le fait qu'une unité peut être soit une unité militaire soit un négociateur comme sur la figure 5.9. Dans cette hiérarchie nous nommons les feuilles des *ressources élémentaires* et les autres des *ressources complexes*. Les demandes en ressources peuvent alors permettre des choix de personnalisation de la stratégie en faisant la requête d'une ressource complexe. Un négociateur (ou une unité militaire) sera alors dite un *sous-type* des unités. Nous définissons dans ce cas une *préférence de type* qui se formalise sous la forme d'un tuple $\langle t_i, t_{ij} \rangle$ où t_{ij} est un sous-type de t_i. Elle exprime la préférence pour une ressource de type t_{ij} lorsqu'une ressource de type t_i est demandée par le système. Dans notre exemple, il faudra alors créer une préférence $\langle$*Unité*, *Négociateur*$\rangle$. Lorsque le comportement *Espionnage* demandera une ressource de type *Unité*, il lui sera alors alloué en priorité un *Négociateur* s'il y en a un de disponible, sinon une unité d'un autre sous-type lui sera allouée.

Certains types de ressource sont homogènes (e.g. l'or), c'est-à-dire que les différentes ressources de ce type sont exactement les mêmes ; d'autres sont hétérogènes (e.g. les unités militaires), c'est-à-dire que deux ressources de ce type pourront présenter des différences. Ces dernières proviennent de paramètres supplémentaires associés à la ressource et qui peuvent prendre différentes valeurs. Prenons l'exemple des négociateurs : dans un jeu de stratégie un niveau de vie est souvent associé aux unités. Des négociateurs pourront donc avoir un niveau de vie différent, et pour notre comportement *Espionnage* nous pouvons préférer envoyer les négociateurs avec le moins de vie pour garder les autres pour des négociations importantes. Si plusieurs spécialisations de négociateurs existent, par exemple en *commerce* ou en *religion* le premier cherchant à créer des liens commerciaux et le deuxième à évangéliser, nous pouvons ici aussi souhaiter exprimer une préférence pour des négociateurs spécialisés en religion afin de garder les négociateurs en commerce pour plus tard. Afin de pouvoir exprimer ces préférences, nous ne pouvons

Priorité	Sous-comportement	Importance	Préférences
1	Création Armée		
2	Espionnage	1	[Unité, Négotiateur] [Négotiateur, Spécialité, [Commerce]] [Négociateur, Vie, min]
	Défense	9	

TABLE 5.1: Exemple de comportement parallèle

pas utiliser les préférences de type. Nous définissons alors des *préférences de caractéristique*. Une préférence de caractéristique est définie sous la forme d'un tuple $\langle t, par, p\rangle$ où :

- t est un type de ressource sur lequel s'applique la préférence, dans notre cas *Négociateur*,
- *par* est le nom d'un paramètre sur lequel s'applique la préférence, dans notre cas *Vie*,
- p est la préférence elle-même.

La préférence peut prendre différentes formes selon le type de valeur que prend le paramètre. Dans le cas de la vie, la valeur est numérique et nous souhaitons la minimiser, mais nous pourrions également vouloir la maximiser. Dans le cas de la spécialité des négociateurs la valeur n'est pas numérique et la notion d'ordre ne s'y applique pas, il est donc nécessaire d'énumérer les valeurs préférées. La préférence peut donc prendre deux formes :

- *min* pour minimiser la valeur du paramètre ou *max* pour le maximiser,
- ou une liste de chaînes de caractères correspondant aux valeurs préférées.

Le tableau 5.1 résume l'ensemble des caractéristiques de notre comportement *Développement Militaire* : les priorités, les importances ainsi que les préférences de chaque sous-comportement.

Conclusion

Pour résumer nous pouvons formaliser un comportement parallèle comme un tuple $\langle SB, LP\rangle$ où :

- SB est un ensemble de comportements,
- LP est un ensemble de *niveaux de priorité*.

Un *niveau de priorité* a la forme $(p, (sb_0, i_0, LPt_0, LPc_0), ..., (sb_n, i_n, LPt_n, LPc_n))$ où

- $p \in \mathbb{N}$ est la *priorité*,
- $sb_j \in SB$ est un sous-comportement ayant la priorité p,
- $i_j \in \mathbb{N}$ est l'*importance* du sous-comportement sb_j,
- LPt_j est l'ensemble des *préférences de type* du sous-comportement sb_j,
- LPc_j est l'ensemble des *préférences de caractéristique* du sous-comportement sb_j.

5.2.4 Conclusion

Notre modèle propose de représenter une stratégie sous la forme d'une hiérarchie de comportements. Chaque comportement est donc composé d'un ensemble de sous-comportements pour lesquels il possède des informations afin d'appliquer la stratégie. Ces comportements peuvent être de deux types : **logiques** ou **parallèles**. Un comportement logique permet de respecter le besoin d'adaptabilité de la stratégie en permettant de la modifier en fonction du contexte. Il possède des informations pour effectuer une sélection parmi ses sous-comportements. Un système de **messages** est mis en place pour adapter la sélection aux retours que peuvent donner les comportements aux niveaux supérieurs. Ils s'ajoutent à des conditions permettant l'adaptation au contexte dynamique. Un comportement parallèle répond au besoin de décomposition d'un objectif en plusieurs objectifs plus élémentaires pour faciliter le raisonnement. Il partage un comportement en sous-comportements indépendants qui vont être effectués en parallèle. Plusieurs comportements en parallèle signifie qu'un partage de ressources va avoir lieu, il possède donc des informations supplémentaires pour guider l'allocation des ressources. Chacun de ses sous-comportements a une **priorité** qui détermine l'ordre dans lequel les demandes en ressources sont étudiées. Il est possible de donner une priorité identique à plusieurs sous-comportements, auquel cas il est nécessaire de définir des **importances** pour déterminer comment les ressources vont être réparties lorsque les mêmes sont demandées par plusieurs de ces sous-comportements. Chacun peut également posséder des **préférences** qui permettent au game designer de personnaliser la sélection des ressources allouées. Elles peuvent se porter sur le type de la ressource ou sur un de ses paramètres.

5.3 Le moteur

Grâce au modèle défini dans la partie précédente, un game designer peut créer des stratégies pour guider les adversaires dans un jeu 4X. Nous souhaitons maintenant réunir la stratégie définie par la game designer et l'IA tactique définie par le développeur en créant un moteur qui va envoyer au module tactique des directives provenant d'une stratégie. Nous avons défini dans le chapitre 3 qu'une stratégie fixe des objectifs haut-niveau et guident l'allocation des ressources tout en s'adaptant au contexte dynamique. Ces directives vont donc consister à communiquer les objectifs sélectionnés en fonction du contexte puis à guider l'allocation des ressources.

5.3.1 La sélection des objectifs

La sélection des objectifs à poursuivre se fait via la sélection des comportements par les FSM. Afin de faire cette sélection il est donc nécessaire de vérifier les conditions sur les transitions. Afin de ne pas avoir à parcourir toute la hiérarchie à chaque fois, une liste L contenant les conditions susceptibles de faire changer la sélection est créée. Les conditions sont vérifiées de façon régulière et la liste est mise à jour à chaque changement d'état sélectionné. Celle-ci contient donc les conditions qui apparaissent sur les transitions qui ont pour origine un des états

actuellement sélectionnés. Chaque condition est également associée au comportement logique correspondant afin de faciliter l'accès à la partie de la hiérarchie concernée par la mise à jour lorsqu'une condition est satisfaite. Par exemple, si l'état possédant le comportement *Protection* est sélectionné dans notre comportement logique *Militaire Modéré*, alors la condition *Rapport de force > 0.7* sera ajoutée à notre liste L et associée au comportement *Militaire Modéré*. De cette manière si la condition est satisfaite, nous aurons directement accès au comportement où le changement a lieu et nous n'aurons pas besoin de parcourir la stratégie entière.

Les algorithmes 1 et 2 décrivent le fonctionnement de cette liste. L'algorithme 1 décrit la vérifications des conditions qui se fait régulièrement. Une liste des conditions satisfaites est créée (ligne 5), puis pour chaque condition vraie, le comportement logique correspondant est mis à jour (ligne 9) comme décrit dans l'algorithme 2 : on vérifie de nouveau que la transition peut effectivement être activée. Cela est possible lorsque le prédicat de la transition est vrai. Si la transition est effectivement activée, on supprime alors les conditions de la liste L correspondant à notre comportement logique. L'état destination de notre transition est alors sélectionné. L'étape suivante consiste à regarder si un message y est associé et si c'est le cas de le propager. Cette propagation peut entrainer un changement de sélection qui annulera le comportement à l'origine du message. S'il n'a pas été annulé, les conditions sur les différentes transitions qui ont pour origine le nouvel état sélectionné sont ajoutées à la liste L. La dernière étape est d'initialiser le comportement du nouvel état sélectionné avec la méthode INITIALIZE qui se comporte comme suit :

- si c'est un comportement logique, sélectionner l'état initial puis poursuivre comme si une transition était activée dans l'algorithme 2 : envoyer le message associé à l'état courant s'il y en a un, puis si le comportement est toujours actif ajouter les conditions associées à l'état courant à la liste L, et enfin initialiser le comportement sélectionné avec la méthode INITIALIZE,

- si c'est un comportement parallèle, appeler la méthode INITIALIZE de l'ensemble des sous-comportements,

- si c'est un comportement élémentaire, l'indiquer comme sélectionné.

En plus de cette mise à jour régulière, les messages envoyés par les comportements élémentaires peuvent venir mettre à jour notre hiérarchie à tout moment. Ces messages sont donc envoyés au niveau d'une feuille et sont propagés en remontant la branche de la hiérarchie correspondante de la même manière que lors de la sélection d'un nouvel état dans une FSM. L'ensemble de ces mécanismes de mise à jour sont représentés dans la figure 5.10.

A la fin d'une mise à jour de notre hiérarchie, nous avons une liste de comportements élémentaires sélectionnés qui correspondent à des objectifs devant être résolus par le module tactique auquel nous envoyons nos requêtes. Nous avons ensuite besoin de déterminer les ressources qui seront allouées à chaque objectif.

Algorithm 1 Algorithme de vérification des conditions

```
1: function CheckConditions(𝓛)
2:     LC ← []                          ▷ Initialiser une liste pour les conditions satisfaites
3:     for all c ∈ 𝓛 do
4:         if c then
5:             LC ← c                   ▷ Ajouter chaque condition satisfaite à la liste
6:         end if
7:     end for
8:     for all c ∈ 𝓛𝓒 do
9:         Update(c.behavior)           ▷ Mettre à jour les comportements associés
10:    end for
11: end function
```

Algorithm 2 Algorithme de mise à jour d'un comportement logique

```
1: function Update
2:     t ← CheckTransitions(selectedState) ▷ Récupérer la transition activée s'il y en a une
3:     if t then
4:         RemoveConditions(selectedState)      ▷ Supprimer les conditions observées
5:         selectedState ← t.destinationState   ▷ Modifier l'état sélectionné
6:         if selectedState.Message then
7:             SendMessage(selectedState.Message)   ▷ Propager le message
8:         end if
9:         if active then          ▷ Vérifier que le comportement est toujours sélectionné
10:            AddConditions(selectedState)      ▷ Surveiller les nouvelles conditions
11:            Initialize(selectedState)   ▷ Initialiser le nouveau comportement sélectionné
12:        end if
13:    end if
14: end function
```

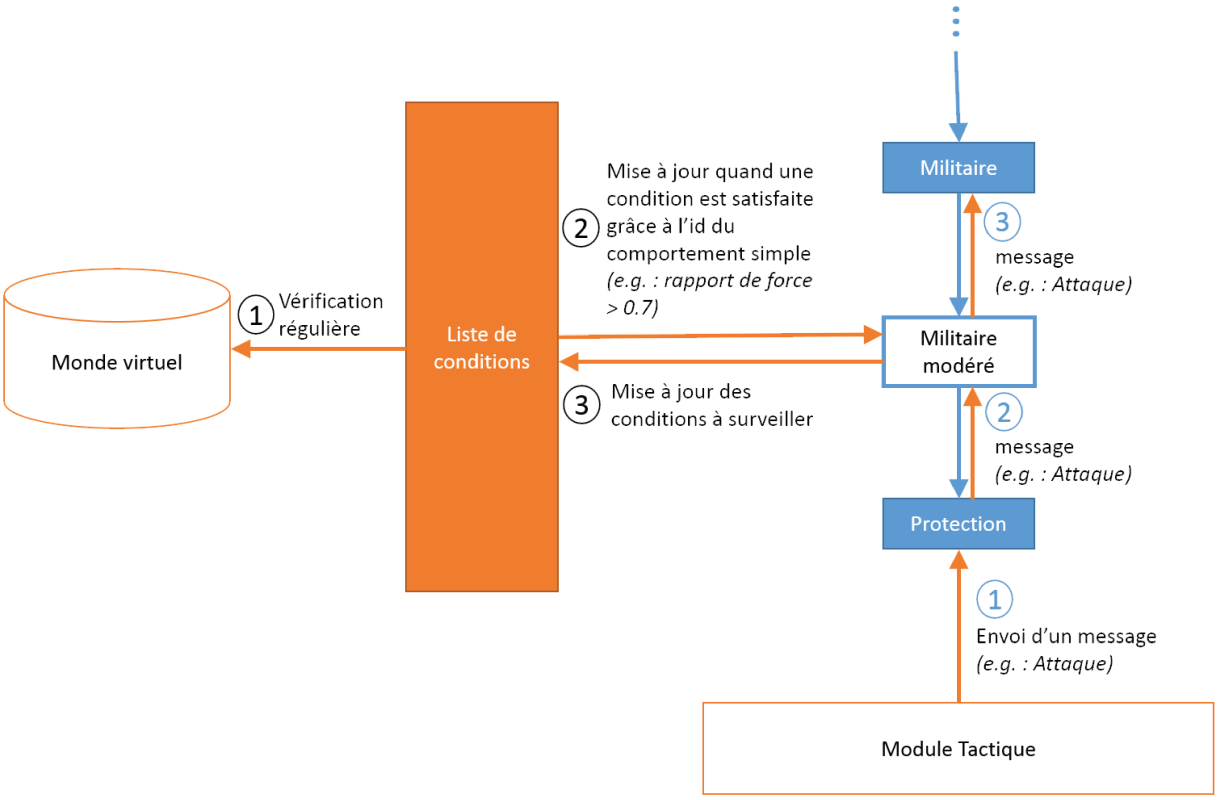


FIGURE 5.10: La mise à jour de la stratégie

5.3.2 La répartition des ressources

Nous utilisons un algorithme de répartition des ressources qui va prendre en compte les données renseignées par l'utilisateur dans les comportements parallèles telles que définies dans la partie 5.2. Ces données vont orienter la répartition au niveau du nombre (importance et priorité) et pour le choix des ressources (préférences). Ces données vont être confrontées aux ressources disponibles et aux demandes de ressources pour chaque objectif fournies par le module tactique.

Pour débuter la répartition, l'algorithme va parcourir la hiérarchie de comportements plusieurs fois en commençant par le comportement racine comme l'indique l'algorithme 3. L'algorithme sera appelé plusieurs fois jusqu'à ce qu'il ne reste plus de ressources à allouer ou bien que les comportements élémentaires ne réclament plus de ressources. Pour chaque comportement logique, l'algorithme explorera la branche correspondant au comportement de l'état sélectionné. Pour chaque comportement parallèle, les sous-comportements seront explorés des plus prioritaires au moins prioritaires. Lors d'une évaluation de la stratégie, seule une priorité se verra attribuer des ressources. Lorsque plusieurs sous-comportements possèdent cette priorité ils seront tous parcourus. Nous expliquons plus en détail l'algorithme de parcours de la hiérarchie dans les parties qui suivent.

Algorithm 3 Algorithme d'allocation des ressources

Require: Une liste de ressources à allouer R

1: $success \leftarrow true$
2: **while** $success$ & $R \neq []$ **do**
3: $success \leftarrow$ ALLOCATE($rootBehavior, R$)
4: **end while**

La répartition des ressources en nombre

Pour permettre l'allocation des ressources aux comportements, il est nécessaire de connaître les besoins en ressources des comportements élémentaires. Pour cela nous avons décidé de formaliser la demande du module tactique pour les besoins d'un objectif. Nous avons fait le choix de ne pas définir de besoins fixes pour chaque objectif pour répondre au besoin d'adaptabilité dans l'utilisation de la stratégie, et les besoins sont donc demandés à chaque allocation leur permettant ainsi d'évoluer au cours du jeu. Afin de formaliser la communication entre le module tactique et notre moteur pour les besoins en ressources nous avons mis en place un format d'expression des demandes.

Nous avons souhaité que les besoins puissent être exprimés de façon assez flexible pour que la répartition puisse s'adapter à toute configuration de ressources disponibles. Nous nous sommes ici aussi inspiré des travaux de Gordon (2006) en séparant les besoins B d'un objectif en deux : les ressources indispensables à son fonctionnement Ri, et les ressources permettant de le renforcer Rr. La différence avec les travaux de Gordon (2006) est que l'on ne définit plus les besoins sur une ressource en fonction de ses paramètres, mais sur un ensemble de ressources en fonction de ses paramètres. En effet en fonctionnant à un niveau stratégique, les demandes en ressources ne concernent plus les ressources individuelles, ni un nombre précis de ressources, mais un ensemble de ressources qui doit répondre à certains critères.

Les ressources Ri sont exprimées sous la forme d'une liste de types de ressources, chaque type étant associé à une quantité. Pour qu'un objectif puisse être réalisé il est donc nécessaire que chaque type de ressource renseigné puisse être alloué dans la quantité associée. Les ressources Rr servent une fois les ressources Ri allouées, lorsque des ressources sont encore disponibles et peuvent venir aider à la poursuite de l'objectif. Elles expriment à la fois la nature des ressources dont l'objectif peut avoir l'utilité, mais également des contraintes sur l'allocation. Par exemple l'objectif de défense peut demander à obtenir des ressources de type *unités militaires* mais estimer qu'il n'est pas nécessaire d'en obtenir plus que cinq par ville : nous pouvons alors définir un nombre maximum d'unités militaires dont l'objectif a besoin. Il peut également être utile d'exprimer des combinaisons : par exemple un objectif d'espionnage peut demander à obtenir autant d'unités que de chevaux si l'espionnage demande de faire de longs déplacements. Nous avons alors défini une grammaire permettant d'exprimer ces besoins.

Les ressources de renfort Rr s'expriment sous la forme d'une liste d'expressions. Chaque expression est associée à un type de ressource et prend une des formes suivantes :

- [i, r], soit i un entier et r un type de ressource. Cette expression signifie que seules i unités de ressource de type r peuvent être alloués à l'objectif. S'il n'y a pas suffisamment de ressources alors aucune n'est allouée.
- [max(i), r] ou [min(i), r], soit i un entier et r un type de ressource. Cette expression signifie qu'au plus (resp. au moins) i unités de ressource de type r peuvent être allouées à l'objectif.
- [iα, r], soit i un entier, α une variable et r un type de ressource. Cela signifie que seul un multiple de i unités de ressource de type r peut être alloué à l'objectif.
- [r], soit r un type de ressource. Cette expression signifie qu'aucune contrainte ne s'applique à ce type de ressource.

Si le même nom de variable (α) apparait dans plusieurs expressions pour un même objectif, cela signifie que les conditions sont liées. Par exemple si une condition [1x, ouvrier] et une condition [2x, soldat] définissent les ressources de renfort d'un objectif, cela signifie que l'objectif aura besoin de deux fois plus de soldats que d'ouvriers.

Quelques règles viennent encadrer ces expressions dans leur utilisation pour un objectif :

- si une expression [i, r] est définie, il ne peut pas y avoir une autre expression utilisant le même type de ressource r.
- si une expression [r] est définie, il n'est également pas possible d'avoir une autre expression utilisant le même type de ressource r.
- [max(i), r], [min(i), r] et [iα, r] peuvent apparaître avec le même type de ressource r.
- plusieurs expressions [iα, r] peuvent utiliser le même α.

Nous allouons les ressources aux comportements en plusieurs fois afin de nous permettre d'effectuer une répartition entre plusieurs comportements de même priorité. Pour cela nous faisons appel à une méthode Allocate qui a pour objectif d'allouer un minimum de ressources au comportement pour son fonctionnement. Cette méthode diffuse l'appel vers le bas de la hiérarchie en appelant la méthode similaire des sous-comportements jusqu'à atteindre les comportements élémentaires. Les ressources dont ont besoin les comportements élémentaires sont alors celles déterminées par le module tactique pour leur objectif. Notre moteur contient également un allocateur de ressources auquel chaque comportement élémentaire transmet ses besoins lorsque sa méthode Allocate est appelée. Le résultat des méthodes Allocate correspond à un résultat de succès si au moins une ressource à été allouée, ou à un échec dans le cas inverse. Un résultat

d'échec signifie donc qu'il n'y a plus de ressources disponibles intéressant le comportement, il n'est donc pas utile de rappeler la méthode ALLOCATE. En revanche si c'est un résultat de succès qui est renvoyé, cela signifie que des ressources ont pu être allouées et qu'il existe peut-être encore des ressources disponibles permettant d'apporter leur aide au comportement. Le fonctionnement de la méthode ALLOCATE est différent suivant le type de comportement :

- Pour un comportement élémentaire, au premier appel la méthode ALLOCATE récupère les besoins B de l'objectif qui lui est assigné en les demandant au module tactique. Il demande ensuite à obtenir les ressources indispensables Ri à l'allocateur de ressources. Lors des appels suivants la méthode ALLOCATE demande à l'allocateur un minimum de ressources tout en respectant les conditions définies dans les besoins optionnels Rr.

- Pour un comportement logique, la méthode ALLOCATE appelle la méthode similaire du comportement sélectionné et renvoie le résultat reçu par cet appel.

- L'algorithme 4 correspond à la méthode ALLOCATE d'un comportement parallèle, celle-ci a un fonctionnement plus complexe que pour les autres types de comportement. Afin d'en simplifier la compréhension les mécanismes sont schématisés sur la figure 5.11. Comme précisé précédemment, les sous-comportements possèdent chacun une priorité et les ressources sont allouées pour une priorité à la fois, de la plus haute priorité à la plus basse. Les méthodes des sous-comportements de la priorité courante sont alors appelées séquentiellement (boucle commençant à la ligne 7). Lors des appels suivants les méthodes seront de nouveaux appelées, mais les appels à la méthode d'un comportement s'arrêteront temporairement lorsqu'elle aura été appelée autant de fois que son importance (ligne 10). Une fois que tous les comportements d'une priorité auront été appelés autant de fois que leur importance, le compteur sera remis à 0 et les méthodes de tous les comportements de la priorité sont de nouveau appelées (ligne 24). Lorsqu'une méthode retourne un résultat d'échec le sous-comportement est désactivé pour cette allocation de ressources (ligne 13). Les appels s'effectuent tant que des sous-comportements de la priorité en cours sont actifs. Lorsque le dernier élément de la priorité retourne un résultat d'échec, on passe à la priorité suivante (ligne 22). La méthode du comportement parallèle ne retourne un résultat d'échec que lorsque tous les sous-comportements ont retourné un résultat d'échec (ligne 20).

La répartition des ressources en préférences

Afin de respecter les préférences définies dans la stratégie, celles-ci sont intégrées dans les appels à la fonction ALLOCATE. Elles sont donc transmises de leur emplacement dans la hiérarchie jusqu'en bas de chaque branche afin d'être intégrées aux demandes passées à l'allocateur de ressources. L'allocateur suivra alors le raisonnement décrit par l'algorithme 5. Il cherchera à satisfaire toutes les préférences en cherchant en premier dans les ressources allouées au comportement élémentaire à l'allocation précédente (ligne 3) puis en cherchant dans l'ensemble des ressources (ligne 6). Si la recherche échoue elle sera effectuée en cherchant à satisfaire une

Algorithm 4 Algorithme d'allocation des ressources pour les comportements parallèles

```
1: function ALLOCATE
2:     if p not defined then
3:         p ← priorité la plus haute                    ▷ p sauvegarde la priorité courante
4:         nbAppel ← 0
5:     end if
6:     nbAppel ← nbAppel + 1
7:     for all rb ∈ runningBehaviors do
8:         if ALLOCATE(rb) then
9:             if rb.importance = nbAppel then
10:                ADDTOTEMPDEACTIVATED(rb)
11:            end if
12:        else
13:            ADDTODEACTIVATED(rb)
14:        end if
15:    end for
16:    if runningBehaviors =[] then
17:        nbAppel ← 0
18:        if tempDeactivated =[] then
19:            if p est la priorité la plus basse then
20:                return false
21:            end if
22:            p ← priorité suivante ALLOCATE(self)    ▷ Il n'y a donc pas eu de ressources
   d'allouées, on recommence avec la priorité suivante
23:        else
24:            ACTIVATEALLTEMPDEACTIVATED
25:        end if
26:    end if
27:    return true
28: end function
```

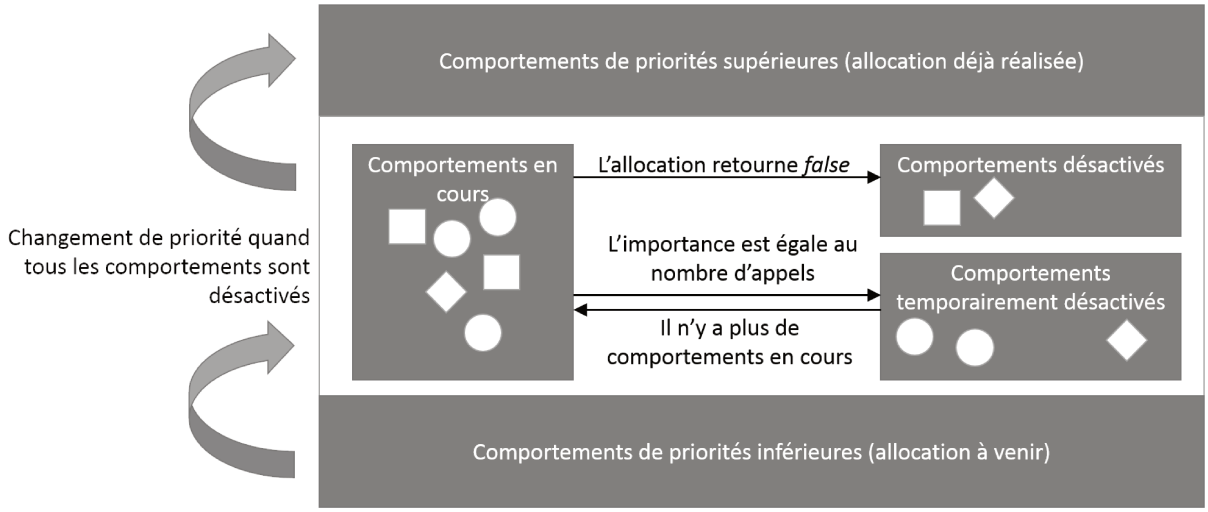


FIGURE 5.11: Schématisation de l'allocation des ressources dans un comportement parallèle

préférence de moins et ainsi de suite jusqu'à ce qu'une ressource soit trouvée ou jusqu'à ce le nombre de préférences à satisfaire descende sous zéro (ligne 9).

Algorithm 5 Algorithme de recherche des ressources satifaisant des préférences

Require: un type de ressource T et une liste de préférences associée $preferences$

1: $n \leftarrow preferences.length$
2: **while** $n \geq 0$ **do**
3: **if** $r \leftarrow$ FINDALLOCATEDRESOURCE$(T, preferences, n)$ **then** ALLOCATE(r)
4: **return** true
5: **end if**
6: **if** $r \leftarrow$ FINDRESOURCE$(T, preferences, n)$ **then** ALLOCATE(r)
7: **return** true
8: **end if**
9: $n \leftarrow n - 1$
10: **end while**
11: **return** false

Exemple d'allocation des ressources

Afin d'illustrer l'allocation des ressources, nous reprenons notre exemple et détaillons les étapes de l'allocation des ressources en supposant que le comportement *Militaire Modéré* est sélectionné et a lui même sélectionné le sous-comportement *Développement Militaire*. Nous ne détaillerons l'allocation que pour cette partie de la hiérarchie dont nous avons précédemment détaillé le contenu.

1. Nous avons tout d'abord le comportement logique *Militaire Modéré*. Comme nous l'avons vu les comportements logiques se contentent d'appeler la méthode ALLOCATE du com-

portement associé à l'état sélectionné. Dans notre exemple c'est la méthode ALLOCATE du sous-comportement *Développement Militaire* qui est appelée.

2. Nous avons donc ensuite le comportement parallèle *Développement Militaire* qui est appelé pour la première fois. La priorité courante est alors mise à 1, correspondant à la plus haute priorité. Un seul sous-comportement possède cette priorité, c'est le sous-comportement *Création Armée*. Sa méthode ALLOCATE est alors appelée.

3. Le comportement élémentaire *CréationArmée* voit sa méthode ALLOCATE appelée pour la première fois. Il va donc envoyer une requête au module tactique pour obtenir les besoins l'objectif. Le module tactique peut alors renvoyer les besoins suivants : Ri = [["ouvrier", 1]], Rr = [["ouvrier"],[2x,"nourriture"]]. Ces besoins signifient que pour créer une armée, l'objectif a besoin d'au moins un ouvrier (ressources indispensables Ri) et peut profiter d'ouvriers supplémentaires sans aucune contrainte sur l'allocation ainsi que d'unités de nourriture par groupes de 2 (ressources de renfort). Le comportement *CréationArmée* fournit alors les ressources indispensables Ri à l'allocateur de ressources qui lui renvoie si l'allocation d'un ouvrier a pu être effectuée. Si la réponse est positive le comportement renvoie alors un booléen à *true*, sinon un booléen à *false*.

4. Notre comportement *Développement Militaire* reçoit alors la réponse de l'allocation de ressources à *CréationArmée* sous la forme d'un booléen et agit en conséquence. Supposons dans un premier temps que le résultat est vrai. Le sous-comportement ne possède pas d'importance car il est le seul à posséder la priorité 1, il ne sera donc jamais temporairement désactivé. Il est donc maintenu dans la liste des sous-comportements en cours et renvoie un booléen à *true*.

5. Le comportement *Militaire Modéré* se contente de transférer le résultat qui est alors *true*.

6. Puisque le comportement *Militaire Modéré* a retourné précédemment *true*, il sera de nouveau rappeler, sauf s'il n'y a plus aucune ressource disponible. Il appellera de nouveau la méthode ALLOCATE du comportement *Développement Militaire*.

7. Le sous-comportement *Développement Militaire* se retrouvera alors dans la même situation et appellera la méthode ALLOCATE de *Création Armée*.

8. *Création Armée* est un comportement élémentaire qui s'est déjà vu alloué ses ressources indispensables. Le comportement fait de nouveau appel à l'allocateur de ressources mais transmets cette fois-ci la description des ressources de renfort, soit Rr. L'allocateur cherche alors à affecter une quantité minimale de ressources, soit un ouvrier ou 2 unités de nourriture. C'est encore une fois la réponse de l'allocateur que le comportement retournera au comportement supérieur.

9. Le comportement *Développement Militaire* reçoit encore une fois le résultat de l'allocation aux sous-comportement *Création Armée*. Si la réponse est positive nous repartons de

l'étape 4, nous allons donc cette fois détailler une réponse négative. Le sous-comportement *Création Armée* est alors désactivé. Cela entraine un changement de priorité puisque tous les sous-comportements de cette priorité sont désactivés. La priorité courant devient alors 2. La méthode ALLOCATE recommence de nouveau puisqu'aucune ressource n'a été affectée. Les méthodes ALLOCATE des sous-comportements *Défense* et *Espionnage* vont alors être appelées séquentiellement. Commençons par le sous-comportement *Espionnage*.

10. Le comportement élémentaire *Espionnage* fonctionne de la même manière que le comportement *Création Armée*. Il est appelé pour la première fois : il récupère les besoins par le module tactique, fait appel à l'allocateur de ressources en précisant les préférences transmises par le comportement supérieur, et retourne le résultat.

11. Le comportement *Développement Militaire* reçoit le résultat de *Espionnage*. Supposons qu'il soit positif, le nombre d'appels (1) correspond à son importance, le sous-comportement est donc temporairement désactivé. La méthode ALLOCATE du sous-comportement *Défense* est appelée. Son fonctionnement étant le même que les autres comportements élémentaires nous ne le détaillerons pas. Ici trois scénarios sont possibles :

    - le résultat renvoyé est négatif ; le sous-comportement est désactivé ; il n'y a plus de sous-comportements actifs et les sous-comportements temporairement désactivés, soit le sous-comportement *Espionnage*, sont donc réactivés.
    - le résultat renvoyé est positif ; la méthode ALLOCATE est appelée par la suite plusieurs fois et seul le sous-comportement *Défense* est exploré tant que le nombre d'appels n'est pas égal à 9. Si l'allocation de ressources échoue avant, le sous-comportement est désactivé et nous nous retrouvons de nouveau avec uniquement le sous-comportement *Espionnage* à évaluer les appels suivants.
    - le début est identique au cas précédent mais le nombre d'appels à *Défense* atteint 9 sans échouer, nous nous retrouvons alors dans la situation de l'étape 9.

12. S'il ne reste plus que le sous-comportement *Espionnage* d'actif, les prochains appels commenceront par appeler sa méthode ALLOCATE et si elle réussit le sous-comportement sera temporairement désactivé et immédiatement activé de nouveau. Le comportement *Développement Militaire* continuera de retourner *true* tant que le résultat de l'allocation pour *Espionnage* réussira ; lorsqu'il échouera il ne sera pas possible de descendre la priorité et le comportement retournera *false*.

13. Le comportement *Militaire Modéré* retournera alors *false* ce qui entrainera la fin des allocations à cette branche.

5.3.3 Conclusion

Suite à notre proposition de modèle de stratégie permettant au game designer de définir des stratégies, un moteur de stratégie a été créé pour permettre à une stratégie d'être utilisée en combinaison avec un module tactique afin de guider un adversaire dans un jeu 4X. En effet l'objectif de cette thèse est de fournir un contrôle direct du game designer sur la création de l'IA, pour cela il est indispensable que les stratégies créées soient interprétables pour les modules IA implémentés par les développeurs. Ce moteur sert donc de lien entre notre modèle et les modules des développeurs, pour cela il prend en compte les différents éléments renseignés dans la stratégie pour sélectionner des comportements élémentaires définissant les objectifs haut-niveau mais également les ressources qui y sont allouées. Pour réaliser l'allocation, le moteur confronte les ressources disponibles aux besoins de chaque objectif en communiquant avec le module tactique afin de respecter le besoin d'adaptabilité qui consiste à déléguer aux niveaux inférieurs le choix de la méthode de résolution des objectifs et donc la définition des ressources nécessaires pour la mettre en place. Le moteur utilise également les informations que comportent la stratégie pour réaliser une allocation qui répond aux besoins tout en permettant une personnalisation par la stratégie.

5.4 Bilan du chapitre

L'objectif de cette thèse est d'améliorer la création des IA dans les jeux 4X. Celles-ci ont un rôle primordial dans la qualité de l'expérience de jeu qui guide la conception d'un jeu. Le game designer est responsable de la conception des IA qui doit être pensée dans la globalité du jeu, mais actuellement les IA sont ensuite créées par les développeurs spécialisés qui n'ont pas toujours une vision complète du jeu et doivent donc compter sur les informations transmises par le game designer pour créer des IA participant à l'expérience de jeu. Afin d'améliorer cette création, nous proposons de donner un contrôle direct sur la création des IA au game designer.

Notre proposition se compose d'un modèle de stratégie permettant au game designer de créer un raisonnement haut-niveau : une stratégie. Ce modèle a été pensé pour être accessible à des personnes ne possédant pas de compétences en programmation. Pour cela notre modèle s'inspire de travaux existants parmi lesquels nous avons choisi les plus accessibles pour les game designers en privilégiant les représentations visuelles et les paramètres simples dans la formulation et dans la compréhension de l'impact sur la décision. Nous avons notamment choisi d'utiliser des machines à états finis, format déjà utilisé et connu de l'industrie du jeu vidéo puisque facile à la prise en main, pour définir nos *comportements logiques*. Nous les avons enrichies d'un système de messages afin que la décision s'adapte non seulement à l'environnement mais également aux résultats de l'application de la stratégie. Le modèle s'inspire des besoins particuliers dans les jeux de stratégie au tour par tour qui nécessitent de contrôler de nombreuses ressources pour effectuer plusieurs opérations en parallèle mais en gardant une cohérence globale. Pour cela nous avons défini des *comportements parallèles* qui permettent au game designer de diviser comme

il le souhaite la prise de décision en plusieurs domaines. Le game designer peut également y ajouter des informations qui permettent de guider l'allocation des ressources.

Afin que ce modèle s'intègre pleinement dans la création de stratégie, nous fournissons un moteur de stratégie permettant de faire le lien entre la stratégie du game designer et le moteur tactique des développeurs. Le modèle défini se concentrant sur la prise de décision haut-niveau, le moteur va fournir au module tactique des directives adaptées au contexte en utilisant l'adaptabilité du modèle de stratégie. Le moteur va ensuite participer à l'allocation des ressources en combinant les informations renseignées dans la stratégie avec les besoins exprimés par le module IA et en les confrontant aux ressources disponibles.

Chapitre 6

Evaluation

Nous proposons dans cette thèse une méthode permettant aux game designers de participer directement à la création des IA qu'ils ont pensées. La méthode est détaillée dans le chapitre 5. Elle est composée d'un modèle de stratégie permettant aux game designers de décrire un raisonnement haut-niveau, et d'un moteur permettant à une stratégie de guider un module tactique dans la prise de décision pour une IA dans un jeu 4X. Afin de répondre à notre problématique, il est important que le modèle soit à la fois transparent et expressif. En effet le modèle est destiné à fournir un accès direct aux game designers sur l'IA, il est donc important de s'assurer de son utilisabilité par les utilisateurs cibles qui se traduit dans notre cas par une transparence ; le modèle est également destiné à limiter les recours nécessaires aux développeurs et nécessite pour cela de fournir un modèle expressif donnant un maximum de possibilités aux game designers.

Dans ce chapitre, nous présentons le test utilisateur et l'évaluation qui ont été effectués afin de tester notre modèle sur ces deux caractéristiques. Dans un premier temps nous nous penchons plus en détail sur l'implémentation que nous avons réalisée afin de préparer à l'utilisation du modèle, à la fois celle de notre moteur, mais également celle d'un jeu de stratégie pour permettre aux utilisateurs de mettre en action les stratégies ainsi que celle d'un éditeur de stratégie. Nous présentons ensuite le déroulement du test utilisateur et de l'évaluation et enfin nous décrivons et discutons les résultats obtenus.

6.1 L'implémentation

Afin de pouvoir évaluer notre modèle, il est nécessaire d'avoir un avis extérieur en mettant notre modèle entre les mains de personnes ne le connaissant pas encore. De plus, afin de proposer un contexte proche de celui pour lequel le modèle a été pensé, il est nécessaire de permettre aux participants de manipuler le modèle dans le but de créer une IA pour un jeu 4X. Deux parties manquent alors pour répondre à ce besoin : au niveau de la création d'une stratégie, il est nécessaire de rendre la manipulation des stratégies accessible ; au niveau de l'utilisation d'une stratégie, il est nécessaire de la coupler avec un module tactique et un jeu.

Dans un premier temps nous présentons l'implémentation du jeu ainsi que des moteurs qui

y sont attachés et nous détaillons les relations entre chaque partie. Nous abordons dans un deuxième temps l'édition des stratégies.

6.1.1 Le moteur de stratégie dans le jeu

Le travail de mise en place du test utilisateur et de l'évaluation a été réalisé avec l'aide d'un étudiant ingénieur de l'UTC dont la supervision a été accomplie dans le cadre de cette thèse. Les premières discussions se sont intéressées au jeu qui allait être utilisé. Une fois le jeu défini, nous avons pu mettre en place le moteur de jeu composé de l'environnement de jeu, du module tactique, ainsi que de notre moteur. L'UML des différentes classes présentées dans cette partie est présent en annexe B.

Le jeu

Afin d'avoir un nombre suffisant de participants, nous souhaitions limiter le temps nécessaire pour réaliser une évaluation. Cette évaluation devait comprendre l'explication du modèle, l'explication de l'outil et l'explication du jeu, ainsi que les réponses au questionnaire final en plus bien entendu de l'utilisation du modèle et du jeu. Nous avons donc décidé de proposer une version très simplifiée d'un jeu 4X reprenant les points importants pour notre question de recherche :

- une diversité des domaines de développement pour le système via plusieurs conditions de victoire (complexité du problème nécessitant une décomposition),
- un système à développer, notamment par les ressources (contexte interne nécessitant une adaptation),
- un environnement dynamique (contexte externe nécessitant une adaptation).

La complexité de nos besoins provient du souhait de reprendre des codes provenant de l'univers des jeux 4X tels que la diversité des domaines et des conditions de victoires, mais le besoin d'un jeu proposant des parties courtes afin de permettre au participant de tester ses stratégies. Les jeux couramment utilisés comme environnements de test sont des RTS qui mettent en jeu une contrainte temporelle que nous ne nous sommes pas imposée, ainsi que des univers se concentrant sur les tactiques militaires qui ne possèdent en général qu'un seul objectif, celui de destruction des adversaires. Ces environnements ne correspondent donc pas à notre besoin et nous avons alors décidé de créer un jeu répondant à nos contraintes. L'étudiant a ainsi eu pour mission de définir un jeu répondant à nos besoins et a abouti aux caractéristiques suivantes :

- trois domaines de développement pour un joueur : démographique, technologique ou militaire ; trois conditions de victoires associées,
- des unités ouvrières et militaires à créer ainsi qu'un niveau technologique à faire évoluer grâce à des ressources récoltées dans l'environnement,

- une organisation ennemie à combattre.

Nous avons décidé de passer par un logiciel de conception de jeu afin de simplifier sa création : Unity. Ce logiciel a défini le language utilisé pour l'environnement de jeu, pour notre moteur ainsi que pour le module tactique : le C♯.

Les règles du jeu Deux joueurs sont opposés dans un même environnement consistant en un monde fini quadrillé où chacun possède un emplacement appelé *QG* et symbolisant sa ville. Les joueurs peuvent manipuler deux types d'unités : des ouvriers et des soldats. Les premiers servent à récupérer les ressources présentes dans l'environnement et à les ramener au QG afin qu'elles puissent être utilisées. Les secondes servent à combattre les unités adverses. Les ressources disponibles dans l'environnement sont au nombre de trois : la nourriture, le bois et l'or. La nourriture permet de créer des ouvriers, elle peut également être combinée à l'or pour créer des soldats. Le bois permet de faire évoluer la technologie symbolisée par le niveau technologique du QG compris entre 1 et 10. Ce niveau technologique détermine la force des soldats créés. Cette force influence les dégâts provoqués par les soldats. Ceux-ci peuvent attaquer soit des unités ennemies soit le QG ennemi. Lorsqu'un soldat attaque une unité ennemie, il la blesse ou la tue : une unité possède un niveau de vie initialisé à 10 qui baisse à chaque attaque et tue l'unité s'il atteint 0. Lorsqu'un soldat attaque le QG ennemi, il fait descendre le niveau technologique. Un QG ne peut pas être détruit : si son niveau technologique est à 1, les attaques n'ont aucun impact.

Pour remporter une partie, trois possibilités existent :

- détruire toutes les unités ennemies (victoire militaire),
- atteindre un niveau technologique de dix (victoire technologique),
- atteindre un nombre d'ouvriers égal à trente (victoire démographique).

L'interface du jeu Nous nous sommes rendus compte qu'en raison du manque de temps ainsi que de notre manque de compétence dans le domaine, la partie graphique n'allait pas pouvoir être réalisée dans les temps. Nous avons alors fait le choix pour notre jeu d'utiliser une carte simulée virtuellement et d'afficher uniquement les informations intéressantes pour les participants pour comprendre la mise en place de sa stratégie. L'interface textuelle que nous avons alors créée pour afficher ces informations se divise en quatre parties telles que le montre la figure 6.1.

- Les parties 1 et 3 correspondent à la description de chaque joueur : la partie supérieure décrit les ressources et les quantités possédées par chacun, la partie inférieure décrit les objectifs sélectionnés et les ressources qui y sont allouées.
- La partie 2 indique les actions qui sont faites dans l'environnement précédées du nom du joueur qui les effectue.

FIGURE 6.1: L'interface du jeu développé pour les évaluations

- La partie 4 comprend deux boutons permettant de contrôler la vitesse de la simulation de jeu.

L'environnement de jeu A partir des règles de jeu qui ont été définies, nous avons pu développer un environnement virtuel stockant l'état courant et permettant sa modification par le module tactique. Nous avons pour cela développé une classe *WorldManager* sous la forme d'un singleton car nous n'avons pas pris en compte des perceptions différentes, et l'environnement est donc le même pour nos deux joueurs. Cet environnement possède une carte virtuelle sous forme de tableau dans lequel sont placées aléatoirement au début de chaque parties des sources de bois, or et nourriture. Le *WorldManager* possède également les ressources de chaque joueur :

- les ouvriers, leurs positions et leurs niveaux de vie, ainsi qu'éventuellement la ressource qu'ils portent,
- les soldats, leurs positions, leurs niveaux de vie et leurs forces,
- les QG, leurs positions, leurs niveaux technologiques et les ressources qu'ils contiennent.

Le *WorldManager* propose également un ensemble d'actions qui permettent au module tactique d'agir sur le monde virtuel :

- la création d'un ouvrier ou d'un soldat,
- l'augmentation du niveau technologique,
- le déplacement d'une unité sur la carte,
- l'attaque d'une unité ou d'un QG,

- la récupération d'une ressource sur une source,
- la dépose d'une ressource à un QG,
- l'affectation d'un soldat à un ouvrier pour le protéger.

Lors de la réalisation de ces actions, le *WorldManager* détermine également si une condition de victoire est atteinte et met fin à la partie lorsque c'est le cas. Le *WorldManager* est responsable de la mise à jour de l'affichage des informations, il possède donc des références sur les différents labels de l'interface et effectue les modifications nécessaires après la réalisation d'une action.

Le *WorldManager* propose également de récupérer certaines informations sur le contexte. Pour notre stratégie elles sont utiles pour l'adaptation grâce aux FSM, mais ces informations peuvent également servir au module tactique pour ses décisions. Ces informations sont les suivantes :

- la distance entre deux positions,
- la position de la source la plus proche (pour un type de ressource),
- la position des QG,
- les quantités pour chaque type de ressource (unités comprises),
- le niveau technologique,
- la puissance militaire (addition des forces des soldats)
- les quantités d'unités, le niveau technologique et la puissance de l'adversaire.

Il est également possible de lui fournir un prédicat qu'il évalue et dont il renvoie le résultat (booléen).

Le module tactique

Nous avons également dû prendre le rôle des développeurs d'IA et nous avons donc implémenté un module tactique. Celui-ci se présente sous la forme d'une classe *ObjectiveManager* utilisée conjointement avec plusieurs classes héritant d'une classe *Objective* représentant les différents objectifs manipulables par la stratégie. Chaque joueur possède une instance d'*ObjectiveManager*, il y en a donc deux dans l'implémentation de notre jeu.

Chaque instance de la classe *Objective* possède un *id* permettant d'identifier le comportement élémentaire auquel il est rattaché et donc d'envoyer des messages à la stratégie. Elle possède également la liste des ressources qui lui ont été allouées ainsi que des méthodes *GetResources*, *Allocate* et *Deallocate* qui permettent respectivement de récupérer les besoins en ressources tels que décrits par notre proposition, d'allouer des ressources et de les désallouer. Une méthode *Update* est également disponible, elle permet à l'objectif d'effectuer les actions nécessaires pour sa réalisation avec les ressources allouées en appelant les méthodes associées dans le *WorldManager*.

C'est également lui qui est responsable de la mise à jour des informations sur les objectifs affichés sur l'interface (parties basses des zones 1 et 3 sur la figure 6.1).

Un *ObjectiveManager* est l'interface vers les objectifs à la fois pour la stratégie et pour le *WorldManager*. Il possède la liste des objectifs en cours et propose des méthodes permettant d'en ajouter (*AddObjective*) et d'en retirer (*RemoveObjective*), ainsi que l'accès aux méthodes des objectifs précédemment présentées par le biais de l'id du comportement élémentaire. Il possède également une méthode *Update* qui est appelée une fois que la stratégie a été mise à jour et que les ressources ont été allouées et qui appelle la méthode *Update* de chaque objectif. Il récupère alors les messages envoyés par les objectifs et les transmet à la stratégie.

Cinq objectifs ont été créés pour alimenter les stratégies :

- *PreparerArmee* : objectif de création de soldats,
- *AmeliorerTechno* : objectif d'augmentation du niveau de technologie du QG,
- *Attaquer* : objectif de préparation et de déclenchement d'une attaque lorsque la puissance de l'armée est supérieure à la puissance de l'armée ennemie,
- *DeveloppperDemographie* : objectif de création d'ouvriers,
- *Proteger* : objectif de protection du QG et des ouvriers par des soldats.

Le moteur de stratégie

Le moteur de stratégie doit faire le lien entre une stratégie décrite par un participant (par le biais de l'éditeur décrit dans la section suivante) et l'*ObjectiveManager* précédemment décrit. Le moteur de stratégie est encapsulé dans une classe *StrategyEngine*. De la même manière que pour le module tactique, nous allons avoir deux instances de *StrategyEngine*, une pour chaque joueur.

La première étape lorsque le jeu est lancé est l'initialisation de la stratégie. Pour que le participant fournisse un stratégie, il a été choisi de la stocker dans un fichier au format JSON. Chaque instance de *StrategyEngine* possède un lien vers le fichier JSON correspondant à la stratégie qu'il va devoir utiliser. Nous avons utilisé une classe *StrategyBuilder* encapsulant un algorithme de parsing JSON pour initialiser les stratégies.

Les stratégies sont représentées grâce à des instances des classes *LogicalBehavior* (comportement logique), *ParallelBehavior* (comportement parallèle) et *SimpleBehavior* (comportement élémentaire) toutes filles d'une même classe mère *Behavior* (comportement). Le comportement *Behavior* ne possède qu'un élément commun aux comportements : le nom.

LogicalBehavior possède une représentation d'une FSM sous la forme d'un dictionnaire d'états associés aux transitions sortantes. Les transitions sortantes sont représentées par une liste de prédicats associés à l'index de l'état destination. Des classes *Transition*, *Condition* et *Message* ont été créées pour encapsuler les différents formats de prédicat et leurs méthodes d'évaluation (par le biais d'une méthode commune *CheckCondition*).

ParallelBehavior possède les informations d'un comportement parallèle grâce à une classe *BehaviorLevel* regroupant un *Behavior*, une importance et une liste de préférences (qui sont également représentées à l'aide d'une classe *Preference*). Une liste de liste de *BehaviorLevel* permet de représenter l'ensemble des informations nécessaires. Deux classes héritent de *Preference* : la classe *TypePreference* propose une méthdoe *IsRespected* permettant de savoir si une ressource respecte la préférence de type, et la classe *CharacteristicPreference* qui fournit une méthode *GetComparedMethod* permettant de trier une liste de ressources de manière à ce que la première ressource soit celle répondant à la préférence.

SimpleBehavior ne possède aucune information supplémentaire car son nom doit correspondre à celui de son objectif et c'est de cette manière que nous allons communiquer avec l'*ObjectiveManager*. Une stratégie est accessible par une instance de la classe *StrategyGraph* qui possède une référence vers le comportement racine.

La mise à jour des objectifs sélectionnés De la stratégie sous forme de graphe contenue dans la classe *StrategyGraph*, nous extrayons l'arbre des comportements sélectionnés. Les comportements sont dupliqués car notre modèle de stratégie permet à un comportement de posséder plusieurs comportements supérieurs, or cela signifie qu'un comportement pourra avoir plusieurs instances dans une stratégie, une pour chaque comportement supérieur. Notre arbre est composé de *RunningBehavior* qui possèdent chacun une référence *parent* vers le *RunningBehavior* supérieur permettant ainsi de faire remonter les messages, mais également une référence *behavior* sur le *Behavior* qu'il représente ainsi qu'un id permettant de l'identifier. Les *RuntimeLogical* sont les instanciations des *LogicalBehavior*, ils possèdent également l'état sélectionné.

Le *StrategyGraph* est statique : il contient toutes les informations de la stratégie. Le *StrategyTree* est dynamique, il va évoluer et représenter la stratégie telle qu'elle est appliquée à un moment *t*. Pour cela le *StrategyEngine* possède une méthode *Update* qui va dans un premier temps mettre à jour la stratégie en deux étapes : la première c'est l'évaluation des messages transmis par l'*ObjectiveManager* au tour précédent en les envoyant aux objectifs concernés, la deuxième étape c'est la mise à jour de la sélection en fonction du contexte. Pour cela, nous utilisons comme nous l'avons décrit dans la section 5.3.1 une liste des conditions qui peuvent modifier la sélection que possède le *StrategyEngine*, ce dernier vérifie alors avec le *WorldManager* si ces conditions sont vérifiées dans le contexte courant. En ce qui concerne les informations mises à disposition des participants pour déclencher des transitions, les participants avaient accès à plusieurs variables : la quantité possédée de chaque ressource (y compris unités), le niveau technologique, la puissance militaire ainsi que les quantités d'unités, le niveau technologique et la puissance militaire de l'ennemi. Afin d'évaluer la sélection, les *RunningBehavior* implémentent une méthode *Update* dont le fonctionnement a été décrit dans la section 5.3.1. Les *RunningLogical* utilisent leur référence *behavior* pour évaluer si la sélection doit être modifiée. Les *RunningSimple* lors de leur création signalent leur création à l'*ObjectiveManager* pour que l'objectif associé soit créé. Les *RunningBehavior* implémentent également une méthode *Cancel* permettant de supprimer les conditions qui étaient surveillées pour un comportement logique

annulé.

La mise à jour des ressources allouées Une fois la stratégie mise à jour, le *StrategyEngine* met à jour les ressources allouées en faisant appel à une troisième méthode des *RunningBehavior* : la méthode *Allocate* dont le fonctionnement a été décrit dans la partie 5.3.2. Les *RunningSimple* font appel à un allocateur de ressources représenté par une classe *ResourceAllocator* qui est responsable des ressources du joueur et qui répond aux demandes d'allocation. En sauvegardant les allocations précédentes, il choisit les ressources à allouer en essayant de garder une continuité dans les allocations.

Pour les préférences, les participants peuvent utiliser la vie et la force pour les préférences de caractéristique, en indiquant *min* ou *max* suivi de *Vie* ou *Puissance*. Les préférences de type sont accessibles pour les *Unités* qui peuvent avoir pour préférence *Soldat* ou *Ouvrier*.

6.1.2 L'outil

L'édition de stratégie est un sujet complexe pour notre évaluation. En effet, nous souhaitons permettre aux participants de manipuler et d'évaluer notre modèle, cependant nous avons besoin pour cela de fournir un outil permettant cette manipulation. Cet outil indispensable introduit un biais dans notre évaluation puisque l'expérience vécue par les participants sera alors impactée par l'outil fourni. Nous n'avons cependant pas trouvé de solution permettant d'empêcher ce biais.

Afin de permettre aux utilisateurs d'avoir une vue globale de la hiérarchie et de leur permettre de manipuler la hiérarchie, nous avons dans un premier temps cherché à proposer un éditeur de graphe. Des éditeurs de graphe existants déjà, nous en avons choisi un[1] que nous avons personnalisé afin d'ajouter les contrôles et informations manquants. Cet éditeur sépare d'un côté la représentation visuelle et de l'autre les données qui vont venir nourrir cette représentation. Pour stocker les données nous avons utilisé une architecture similaire à celle de notre moteur de stratégie : une classe mère *Behavior* possède trois classes filles *LogicalBehavior*, *ParallelBehavior* et *SimpleBehavior*. La classe mère possède des méthodes communes pour changer le nom, et ajouter ou enlever des sous-comportements (qui n'exécutent rien lorsque le comportement est un objectif) qui sont appelées par l'interface lorsque l'utilisateur effectue des modifications. Chaque classe fille permet de stocker les informations qui lui sont spécifiques de la même manière que dans notre moteur de stratégie.

La vue de la stratégie sous forme de graphe permet d'indiquer les relations comportement supérieur/sous-comportement entre deux comportements et permet de modifier ces relations facilement en ajoutant ou supprimant les arcs. L'éditeur utilisé permet également d'assigner un nom à chaque nœud que nous avons rendu éditable afin de permettre à l'utilisateur de le personnaliser. L'édition du nom d'un comportement élémentaire se présente sous la forme d'une liste déroulante car les possibilités sont contraintes par les objectifs du module tactique. Nous

[1] https://www.codeproject.com/Articles/182683/NetworkView-A-WPF-custom-control-for-visualizing-a (accédé le 6 février 2017)

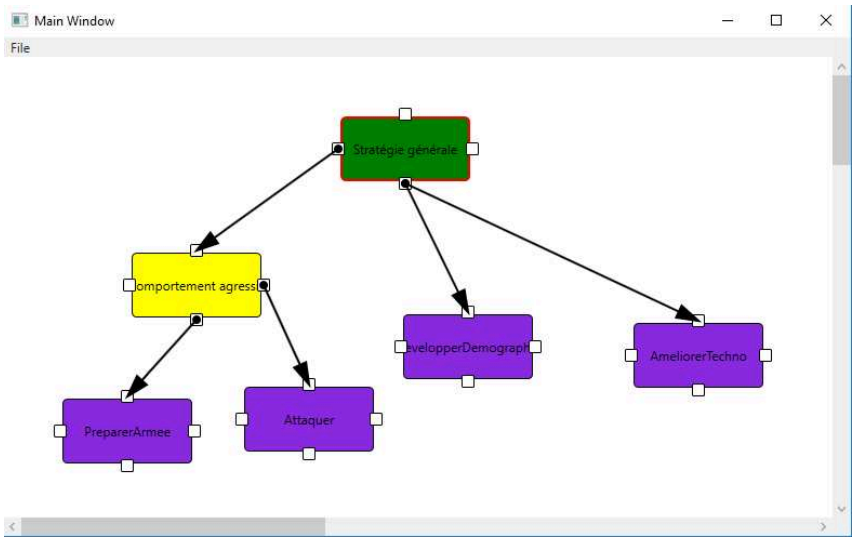


FIGURE 6.2: La stratégie sous forme de graphe affichée dans l'outil développé

n'avons pas voulu surcharger le graphe d'informations mais nous avons trouvé utile de distinguer les différents types de comportements en modifiant légèrement l'aspect de nœuds par leur couleur comme le montre la figure 6.2. Le graphe ainsi obtenu possède donc trois types de nœuds :

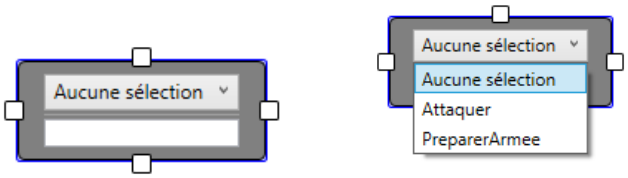
- les verts correspondent à un comportement parallèle,
- les jaunes correspondent à un comportement logique,
- les violets correspondent à un comportement élémentaire.

Nous avons modifié la création des nœuds de façon à pouvoir choisir le type de nœud désiré. Il est également nécessaire de choisir un nœud comme nœud racine de notre hiérarchie, nous l'avons rendu repérable par une fine bordure rouge et avons ajouté un menu contextuel sur les nœuds pour le désigner.

Notre modèle a pour objectif de permettre de créer les IA en gardant une vision complète de l'expérience désirée pour le joueur. De la même manière il nous a semblé nécessaire de permettre de garder une vision complète de la stratégie lors de l'édition des détails d'un comportement. Nous avons donc créé une deuxième fenêtre destinée à l'édition des détails des comportements.

L'édition d'un comportement logique

Dans le cas d'un comportement logique, nous avons besoin de représenter une FSM, nous avons donc utilisé le même éditeur de graphes auquel nous avons apporté des modifications différentes de la stratégie. Les labels sur les nœuds contiennent le nom du comportement qui y est associé et optionnellement un message tel que la figure 6.4a le montre. Cela a nécessité une édition personnalisée que la figure 6.3 représente : le message est laissé libre mais la liste des sous-comportements est construite à partir de la stratégie créée dans l'autre fenêtre afin de ne pas surcharger ici aussi l'utilisateur avec des données.



(a) (b)

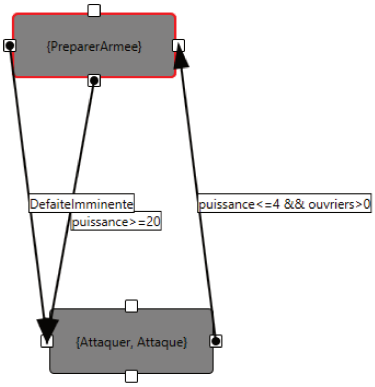
FIGURE 6.3: Édition d'un état dans un comportement logique

Un nœud doit être désigné comme l'état initial, il est alors reconnaissable par la même bordure rouge que le nœud racine de notre stratégie. Un arc entre deux nœuds représente une transition pour passer d'une structure à une autre. Un ensemble de conditions lui est associé. Nous avons alors dû enrichir la représentation du graphe de façon à pouvoir ajouter ces conditions. Pour effectuer une liaison *OU* il est nécessaire de créer deux arcs distincts possédant chacun leurs conditions comme sur les figures 6.4. Cela permet d'éviter de nécessiter d'un système complexe de parenthèses pour définir l'ordre d'évaluation des opérateurs. Afin ici aussi d'éviter de rendre l'éditeur trop complexe, celui-ci ne permet l'ajout que d'une comparaison (figure 6.4b) ou d'un message (figure 6.4c) à la fois. Un seul éditeur a ainsi eu besoin d'être ajouté, ainsi qu'un menu contextuel dynamique accessible sur le label de la transition permettant de supprimer l'une des conditions (figure 6.4d).

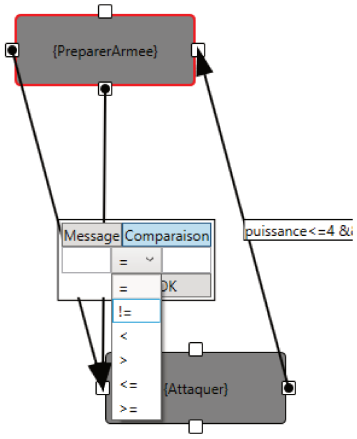
L'édition d'un comportement parallèle

Afin de représenter les comportements parallèles nous avons utilisé une vue sous la forme d'un tableau tel que représenté sur la figure 6.5 qui permet ainsi de représenter les relations d'ordre entre les différents niveaux de priorité. Pour cela chaque ligne correspond à une priorité et regroupe les sous-comportements qui possèdent cette priorité. La priorité n'est pas directement éditable car sa valeur n'est pas importante, seul l'ordre compte et peut donc être modifié. Chaque sous-comportement est ensuite représenté par son nom, son importance éditable, ainsi qu'une liste de préférences composée chacune de deux champs texte : le premier permet de renseigner le type de ressource sur lequel porte la préférence, et le deuxième possède soit le sous-type qui sera préféré dans le cas d'une préférence de type, soit *min* ou *max* suivi du nom du paramètre sur lequel porte la préférence dans le cas d'une préférence de caractéristique. Ces champs auraient bénéficié de menus déroulants mais nous avons été limité par nos connaissances techniques en Windows Presentation Foundation (WPF), le modèle graphique utilisé par l'éditeur de graphe choisi.

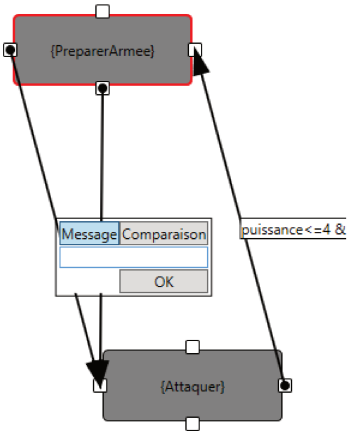
Ces différentes vues permettent de construire la totalité d'une stratégie. Il est ensuite possible par le menu présent dans la fenêtre principale d'enregistrer la stratégie sous la forme d'un fichier JSON de façon à pouvoir l'utiliser avec notre moteur de stratégie. Un exemple est présenté en annexe C.1.



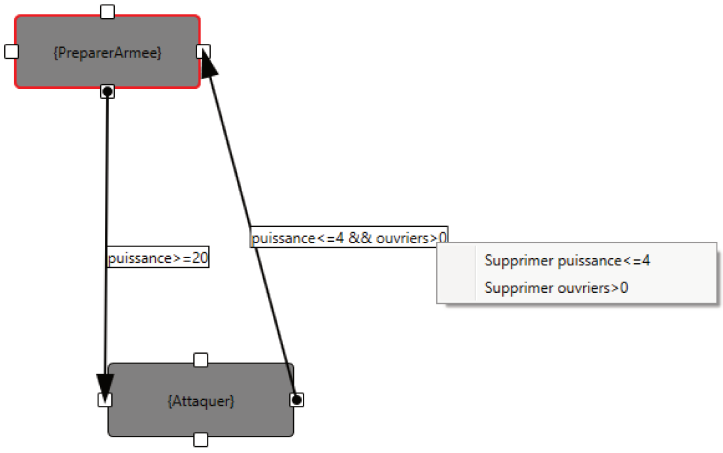
(a)



(b) Création d'une condition de comparaison



(c) Création d'une condition de message



(d) Suppression d'une condition

FIGURE 6.4: Une FSM affichée dans l'outil développé

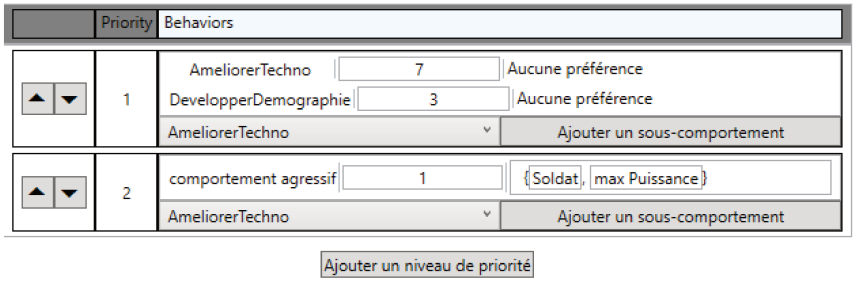


FIGURE 6.5: Un comportement parallèle affiché dans l'outil développé

6.2 Le test utilisateur

Afin de préparer notre évaluation, nous avons fait appel à des étudiants ingénieurs de l'UTC pour effectuer un test utilisateur de notre modèle en nous intéressant à sa **transparence** et son **expressivité**. Nous avons ainsi souhaité observer si notre modèle peut être compris rapidement par des personnes sans connaissances préalables et s'il permet d'exprimer un raisonnement stratégique. Nous allons détailler le déroulement de l'évaluation, les questions qui ont été posées et enfin les résultats que nous avons obtenus.

6.2.1 Déroulement

Afin de réaliser le test, nous avons suivi les étapes suivantes :

1. *Explication des différents éléments.* Le modèle de stratégie, l'éditeur de stratégie ainsi que le jeu ont été présentés et expliqués oralement aux participants.

2. *Manipulation des stratégies.* L'éditeur de stratégie est laissé entre les mains du participant sans limite de temps afin qu'il le découvre et l'utilise pour créer une stratégie. Le jeu est également mis à sa disposition afin qu'il puisse mettre en action les stratégies créées.

3. *Réponse au questionnaire.* Un questionnaire est présenté au participant pour qu'il exprime un retour sur son expérience.

Nous avons laissé la possibilité au participant de poser des questions que ce soit au moment des explications, pendant l'utilisation de l'outil ou bien durant le questionnaire.

6.2.2 Présentation du questionnaire

Afin de mesurer la transparence et l'expressivité du modèle, l'étudiant a établi un questionnaire comprenant des questions se rapportant à l'une ou l'autre des caractéristiques. Nous présentons dans cette partie les deux ensembles de questions et les méthodes utilisées pour analyser les résultats. Les questions sont soit à réponse binaire (oui/non) soit à réponse graduée (échelle de 1 (*Pas du tout*) à 6 (*Tout à fait*)).

La première caractéristique que nous avons souhaité mesurer est la transparence du modèle, c'est-à-dire sa facilité de compréhension. Nous avons alors posé une question générale graduée *(a)Trouves-tu le modèle facile à comprendre ?* ainsi que plusieurs questions à réponse binaire se concentrant sur les différents points caractéristiques de notre modèle :

- *(b)As-tu compris lorsqu'une transition était déclenchée ?*
- *(c)As-tu compris à quoi servaient les messages qui pouvaient être mis sur les états ?*
- *(d)As-tu compris à quoi sert l'importance ?*
- *(e)As-tu compris quand mettre deux comportements dans un même niveau de priorité ?*

- *(f)As-tu compris quand créer un nouveau niveau de priorité ?*

Les résultats obtenus ont été assemblés en calculant la moyenne A des résultats des questions binaires symbolisant la compréhension des éléments caractéristiques de notre modèle. Afin de réaliser cette moyenne A, une réponse oui est transformé en 1 et une réponse non en 0. La moyenne A ainsi obtenue est comprise entre 0 et 1. Afin de combiner cette moyenne A avec la réponse restante, celle de notre question graduée plus générale, cette dernière est également ramenée à un nombre B compris entre 0 et 1. Pour cela la réponse donnée est diminuée de 1, puis une division par 5 est réalisée. Une moyenne C du nombre B ainsi obtenu et de la moyenne A est ensuite calculée et les analyses sont effectuées sur ce résultat final. La formule 6.1 résume le calcul total, les lettres minuscules symbolisant le résultat des questions associées.

$$C = ((b + c + d + e + f)/5 + (a - 1)/5)/2 \tag{6.1}$$

La deuxième caractéristique que nous avons souhaité mesurer est l'expressivité perçue du modèle. Pour cela nous nous interrogeons sur la pertinence des différents éléments et des contrôles qu'ils fournissent. Quatre questions sont ainsi posées, la première fait référence à l'interface principale sur l'assemblage des comportements en stratégie, la seconde fait référence à la définition plus précise des comportements logiques, la troisième fait référence à la définition des comportements parallèles, et la dernière est une question portant sur l'ensemble de l'outil.

- *(g)Trouves-tu la représentation de la stratégie sous forme de graphe adaptée ?*
- *(h)Trouves-tu les informations données par la fenêtre détail des comportements logiques suffisantes ?*
- *(i)Trouves-tu les informations données par la fenêtre détail des comportements parallèles suffisantes ?*
- *(j)Es-tu satisfait des possibilités offertes par l'outil ?*

De la même manière que pour le premier ensemble de questions, nous établissons une méthode de calcul afin d'en extraire un résultat D cumulé. Nous souhaitons de manière similaire donner un poids équivalent à la réponse de la question générale (j), et à la moyenne des autres questions portant sur des points plus précis précis du modèle, nous obtenons alors la formule 6.2.

$$D = ((g - 1)/5 + (h - 1)/5 + (i - 1)/5 + 3(j - 1)/5)/6 \tag{6.2}$$

Le questionnaire et les réponses complets sont disponibles en annexe D.

6.2.3 Résultats

Le questionnaire était précédé d'une série de quatre questions destinées à établir un profil de le participant : le statut étudiant, la capacité à programmer, la pratique du game design et l'utilisation de jeux de stratégie. L'expérience a été menée au sein de l'UTC qui propose une

a	b	c	d	e	f	A	B	C
5	Oui	Oui	Oui	Oui	Non	0.8	0.8	0.8
5	Oui	Oui	Oui	Oui	Oui	1	0.8	0.9
4	Oui	Oui	Non	Oui	Oui	0.8	0.6	0.7
5	Oui	Oui	Oui	Oui	Oui	1	0.8	0.9
4	non	Oui	Oui	Non	Non	0.4	0.6	0.5
5	Oui	Oui	Oui	Oui	Oui	1	0.8	0.9
5	Oui	Oui	Oui	Oui	Oui	1	0.8	0.9
6	Oui	Oui	Oui	Oui	Oui	1	1	1
5	Oui	Oui	Oui	Oui	Oui	1	0.8	0.9
4	Oui	Oui	Oui	Oui	Oui	1	0.6	0.8
4	Oui	Oui	Oui	Oui	Oui	1	0.6	0.8
6	Oui	Non	Oui	Oui	Oui	0.8	1	0.9
5	Oui	Non	Oui	Oui	Oui	0.8	0.8	0.8
							moyenne	0.83077
							écart-type	0.12506

TABLE 6.1: Résultats des questions associées à la transparence

spécialisation en informatique mais ne propose pas de formation spécifique aux jeux vidéo, ce qui s'est reflété dans les résultats puisqu'une majorité (69,2%) savait programmer mais une majorité (76,9%) n'avait pas pratiqué le game design. Treize personnes ont participé à cette première évaluation dont douze étaient étudiants. Tous avaient déjà joué à un jeu de stratégie mais à des fréquences très variables et seuls dix participants avaient joué à un jeu de stratégie au tour par tour.

Les réponses aux questions précédemment présentées sont résumées dans les tableaux 6.1 et 6.2. On voit que globalement les participants ont trouvé le modèle transparent et ont compris les différents éléments (score moyen de 83%). L'expressivité obtient également un bon résultat (score moyen de 70%) mais il est moins marqué que pour la transparence : la moyenne est plus basse et l'écart-type plus grand (0.18 contre 0.13 pour la transparence). En regardant le détail des réponses il semble que les informations affichées dans la fenêtre des détails ne soient pas toujours suffisantes (les notes moyennes sur la suffisance des informations sont 3.6/6 pour les comportements logiques et 4/6 pour les comportements parallèles), le modèle pourrait donc être enrichi pour donner encore davantage de contrôle. De nombreuses questions à réponse ouverte ont été également posées aux participants ce qui nous a permis de récolter des informations supplémentaires et des suggestions sur les évolutions possibles de notre modèle. L'ensemble des réponses des participants peut être trouvé en annexe D.

	g	h	i	j	D
	4	4	3	4	0.56667
	6	2	5	5	0.73333
	4	2	4	5	0.63333
	4	2	4	4	0.53333
	6	3	2	3	0.46667
	5	4	5	6	0.86667
	5	5	3	5	0.73333
	6	5	6	6	0.96667
	5	3	3	5	0.66667
	5	3	2	3	0.43333
	5	4	5	4	0.66667
	6	6	6	6	1
	5	4	4	6	0.83333
moyenne	5.07692	3.61538	4	4.76923	0.7
écart-type	0.75955	1.26085	1.35401	1.09193	0.17951

TABLE 6.2: Résultats des questions associées à l'expressivité

6.2.4 Discussion

Les résultats obtenus sont encourageants mais doivent cependant être nuancés pour plusieurs raisons.

Le temps. Nous avions estimé le temps nécessaire à une heure lors de l'appel aux participations. Nous avons ensuite laissé les participants utiliser l'outil le temps qu'ils le souhaitaient mais en raison de cette estimation les participants n'avaient généralement pas plus de temps à nous accorder même si la durée a finalement semblé trop courte pour une utilisation de l'outil en profondeur. Les explications prenaient en moyenne entre 15 et 20 minutes, et le questionnaire étant assez long nous prévoyions environ 15 minutes pour y répondre, laissant donc moins d'une demi-heure pour découvrir et utiliser l'outil. Les participants avaient généralement le temps de créer une première stratégie mais pas de la tester. Les réponses données par les participants correspondent donc à une utilisation superficielle de l'outil.

Le profil des participants. Les participants avaient un profil qui différait de celui des utilisateurs cibles pour notre modèle, notamment sur les connaissances en programmation et en game design. Le modèle a été créé de façon à donner un contrôle partiel, la sélection des contrôles doit donc se faire en fonction des utilisateurs finaux. Il est donc important de prendre en compte que les participants avaient un profil d'ingénieur, parfois informatique, et que leurs besoins et leurs envies peuvent donc différer de ceux de game designers. Il faut également relativiser la facilité avec laquelle les participants ont pris en main l'outil puisque de façon similaire, leur profil étant

davantage informaticien il est nécessaire de confronter les résultats avec ceux de personnes non informaticiennes.

Le questionnaire et les résultats. Les questions et les calculs utilisés ont été créés pour ce test utilisateur et auraient besoin d'un processus de validation scientifique qui n'a pu être effectué en raison d'un manque de temps. En effet une validation du questionnaire aurait pu assurer l'absence de biais dans les questions ainsi que leur pertinence et la pertinence des calculs effectués.

La qualité de l'outil. L'outil s'est avéré être relativement instable, provoquant des crashs qui ont interrompu l'expérience des participants et qui leur ont surtout fait parfois perdre les stratégies déjà créées. Cela a pu influencer négativement les participants. Malheureusement nous avons manqué de ressources pour améliorer l'outil avant la fin de la thèse. De plus comme nous l'avons remarqué plus tôt, l'utilisation d'un outil d'un point de vue général implique obligatoirement un biais sur l'utilisation du modèle. Cependant, malgré un outil instable les réponses ont été globalement positives.

Les réponses obtenues aux questions posées sur notre proposition lors de ces tests utilisateurs sont donc globalement encourageantes : notre outil semble compris et apprécié des participants à ces tests. Ces tests utilisateurs sont un atout pour mettre en place une telle évaluation : en effet, nous pouvons par exemple estimer grâce à ces tests le temps nécessaire à une évaluation. Nous présentons dans la section suivante l'évaluation que nous avons mise en place.

6.3 L'évaluation

Ayant obtenu des résultats assez encourageants nous avons souhaité effectuer une deuxième évaluation en y apportant des améliorations :

- *Le temps.* Le temps fixé pour une évaluation a été augmenté à deux heures afin de permettre aux participants de découvrir et comprendre plus en détail le modèle, et les participants ont pour la plupart pu rester plus longtemps, parfois jusqu'à trois heures car les séances ont été organisées en fin de journée.

- *Le profil des participants.* Nous avons contacté une école de game design, l'ISART Digital, afin de mener cette évaluation auprès de personnes ayant un profil plus proche des utilisateurs cibles. Cependant il s'est avéré que plusieurs étudiants avaient également une formation partielle en développement informatique.

- *Le questionnaire et les résultats.* Nous avons voulu nous appuyer sur un questionnaire connu de la recherche et validé. Cependant cela a été difficile de trouver des questionnaires répondant à nos besoins.

6.3.1 La méthodologie

L'évaluation de notre modèle nécessite une comparaison avec une autre méthode offrant un accès partiel à la création d'IA haut-niveau. Nous avons décidé de le comparer à une méthode utilisant des paramètres dans un fichier XML. Nous avons choisi cette méthode car elle correspond à la définition d'un raisonnement haut-niveau non structuré. Autrement dit, nous pouvons la définir comme une façon brute de définir des directives à fournir au module tactique quand notre modèle permet de fournir des informations plus structurées. Nous définissons ainsi notre modèle comme une version qui se veut améliorée de cette deuxième méthode.

Cette méthode est notamment utilisée en modding. Le modding consiste à permettre aux joueurs de venir apporter des modifications sur un jeu. Le modding sur l'IA est une méthode que l'on retrouve fréquemment dans les jeux de stratégie. Par exemple Civilization V propose huit fichiers XML sur lesquels les joueurs peuvent effectuer des modifications pour que les IA agissent différemment[2]. Chaque fichier contient plusieurs balises XML qui définissent chacune un paramètre, et chaque balise contient la valeur du paramètre. Ce sont ces valeurs qui peuvent être modifiées.

De plus c'est une méthode qui a été simple à ajouter à notre processus de test puisqu'un simple éditeur de texte suffit. En ce qui concerne le jeu nous avons gardé le même mais nous en avons créé une seconde version prenant cette fois deux fichiers XML pour guider les IA et fonctionnant avec un moteur légèrement différent. Nous avons essayé de limiter au maximum les modifications afin d'avoir une vraie comparaison des modèles et non des moteurs. Nous avons essayé de proposer des contrôles assez similaires et avons proposé au total neuf paramètres à définir. Cinq d'entre eux font référence aux cinq systèmes élémentaires utilisés dans la stratégie, les quatre restants permettent de définir des préférences pour l'attaque et la protection. Un exemple d'un fichier XML correspondant est présenté en annexe C.2.

6.3.2 Les critères d'évaluation

Les *variables dépendantes* que nous souhaitons évaluer sont la **transparence** et l'**expressivité** de notre modèle en le comparant avec la méthode utilisant le XML décrite dans la section précédente. Notre première *variable indépendante* est donc la méthode utilisée (modèle de stratégie ou XML).

Concernant l'expressivité, en laissant libre l'utilisation du modèle sans fixer de but, le risque est que le participant se fixe des objectifs en fonction de ce que permet le modèle et passe à côté des limites du modèle. Nous avons donc décidé d'introduire dans notre évaluation une deuxième *variable indépendante* sous la forme d'une directive sur le comportement à créer, distinct du modèle, et d'insérer dans notre questionnaire une évaluation de l'atteinte de l'objectif. Cette contrainte permet non seulement d'évaluer l'expressivité mais également la transparence. En effet, pour suivre la directive le participant doit réussir à faire le lien entre ce qu'il crée et ce qu'il obtient dans le jeu puisque la directive fait référence au comportement observé dans le jeu alors

[2] http://modiki.civfanatics.com/index.php?title=Civ5_XML_Reference (accédé le 6 février 2017)

que le participant possède uniquement des contrôles sur l'édition de la stratégie. La directive va guider le participant vers une utilisation précise de la stratégie qui peut là aussi laisser de côté certaines limites, nous avons donc défini plusieurs directives prenant des formats différents qui peuvent correspondre à des manières de penser différentes. Cela permet également d'évaluer si le modèle est plus adapté à certaines utilisations qu'à d'autres. Nous avons donc défini trois directives de design différentes :

- une directive centrée sur le but final de l'IA qui est de *gagner avec le but technologique* (atteindre un niveau 10 de technologie),
- une directive qui est de réaliser un *comportement autonome*, cette directive est volontairement vague afin de laisser davantage de liberté au participant et de donner une directive davantage dirigée vers la personnalité de l'IA,
- une directive qui est de réaliser un comportement *s'adaptant aux ressources de l'environnement*. Cette directive se concentre davantage sur l'adaptabilité de l'IA.

Nous avons alors établi les deux hypothèses suivantes :

- **H1** : Le modèle de stratégie sera perçu comme rendant la création d'IA haut-niveau plus accessible que la méthode utilisant le XML.
- **H2** : Le modèle de stratégie permet davantage d'exprimer un comportement voulu que la méthode utilisant le XML.

L'hypothèse **H2** fait référence à la capacité à suivre les directives données et donc à l'expressivité du modèle, alors que l'hypothèse **H1** porte davantage sur l'intelligibilité et la transparence du modèle.

6.3.3 Déroulement

Chaque participant utilise les deux méthodes de contrôle haut-niveau sur l'IA, soit notre modèle de stratégie et la méthode utilisant un fichier XML. La méthode est alors une *variable indépendante intra-sujet*. Afin de limiter son impact sur les résultats, la moitié des participants a évalué le XML en premier, et l'autre moitié a utilisé notre éditeur de stratégie en premier. Les directives de design ont été utilisées comme *variable indépendante inter-sujet*, les participants ont ainsi été répartis en six groupes. Nous avons estimé qu'un minimum de dix-huit personnes, soit trois par groupe, permettrait d'avoir des résultats exploitables. Les participants ont donc été répartis tel que la figure 6.6 le montre.

Nous avons réalisé une version textuelle des explications de l'évaluation, du modèle, des outils et du jeu, ainsi, chaque participant avait des explications similaires. Les PDF fournis aux participants peuvent être trouvés en annexe E.1. L'évaluation a suivi les étapes suivantes :

1. *Explication des éléments communs.* Le jeu et le déroulement de l'évaluation sont présentés au participant sous la forme d'un PDF.

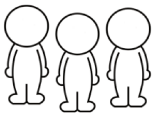
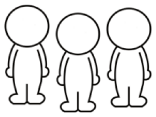
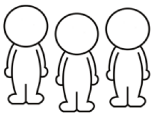
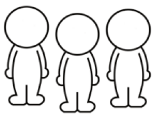
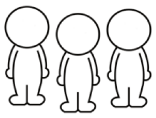
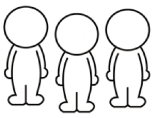
Variable indépendante inter-sujet / Variable indépendante intra-sujet	Directive 1 : Gagner avec la technologie	Directive 2 : Créer un comportement économe	Directive 3 : S'adapter aux ressources découvertes
XML puis Editeur de Stratégie	 1ère moitié groupe 1	 1ère moitié groupe 2	 1ère moitié groupe 3
Editeur de stratégie puis XML	 2ème moitié groupe 1	 2ème moitié groupe 2	 2ème moitié groupe 3

FIGURE 6.6: Répartition des participants dans les groupes de test

2. *Réponse au questionnaire et affectation à un groupe.* Le participant remplit la première partie du questionnaire portant sur son profil et se voit assigner un groupe avec l'objectif correspondant.

3. *Explication de la première méthode testée.* Soit le modèle de stratégie, l'éditeur et les éléments du jeu pouvant être utilisés dans la stratégie, soit le principe de l'XML et ses paramètres, sont présentés au participant sous la forme d'un PDF.

4. *Utilisation de la première méthode.* Le joueur peut soit manipuler l'éditeur et créer une stratégie, soit modifier les paramètres du XML, puis utiliser le jeu pour tester et essayer d'atteindre son objectif.

5. *Réponse au questionnaire pour la première méthode.* Un questionnaire présenté ci-dessous est fourni au participant pour qu'il exprime un retour sur son expérience.

6. *Explication de la seconde méthode testée.* Identique au 3.

7. *Utilisation de la seconde méthode.* Identique au 4.

8. *Réponse au questionnaire pour la seconde méthode.* 5.

Les participants n'étaient autorisés à poser des questions qu'à l'étape 1, et chaque méthode a été testée uniquement avec les explications fournies dans les documents PDF.

6.3.4 Présentation du questionnaire

Le même questionnaire est proposé pour les deux méthodes de façon à pouvoir effectuer des analyses de comparaison sur les résultats.

L'hypothèse **H2** est évaluée par une simple affirmation *La stratégie créée correspond aux instructions qui vous ont été données* notée sur une échelle de Likert à cinq niveaux (de *pas*

1. *Je pense que j'aimerais utiliser ce modèle fréquemment si j'étais game designer.*
2. *Je trouve ce modèle inutilement complexe.*
3. *Je trouve que ce modèle est facile à utiliser.*
4. *Je pense que j'aurais besoin de l'aide d'un spécialiste pour être capable d'utiliser ce modèle.*
5. *J'ai trouvé que les différentes fonctionnalités de ce modèle ont été bien intégrées.*
6. *Je pense qu'il y a trop d'incohérence dans ce modèle.*
7. *J'imagine que la plupart des gens serait capable d'apprendre à utiliser ce modèle très rapidement.*
8. *Je trouve ce modèle très lourd à utiliser.*
9. *Je me sentais très confiant en utilisant ce modèle.*
10. *J'ai besoin d'apprendre beaucoup de choses avant de pouvoir utiliser ce modèle.*

FIGURE 6.7: Affirmations du formulaire SUS

du tout d'accord à *tout à fait d'accord*) permettant d'estimer la réalisation de l'objectif avec le modèle utilisé.

En ce qui concerne l'hypothèse **H1** nous avons recherché des questionnaires adaptés. Etant donné la spécificité du problème, il a été difficile de trouver un questionnaire validé qui réponde exactement à nos besoins. Nous avons décidé d'utiliser le questionnaire SUS (System Usability Scale) défini par Brooke (1996) permettant d'évaluer l'utilisabilité d'un système qui se rapproche des notions d'intelligibilité et de transparence. Le questionnaire est composé de dix affirmations notées par le participant grâce à une échelle de Likert à 5 niveaux. Il a cependant été démontré que la version originale en anglais n'était pas toujours bien comprise par les personnes dont ce n'est pas la langue maternelle (Brooke, 2013), nous avons donc souhaité utiliser une traduction française mais aucune n'ayant été validée nous avons effectué notre propre traduction qui nécessiterait une validation. La version du questionnaire utilisée comprend les affirmations en figure 6.7. Nous avons également fait quelques modifications pour adapter le questionnaire à notre contexte : cette évaluation n'ayant pas été faite auprès de game designers qui sont les utilisateurs cibles de notre modèle, la première affirmation (à l'origine *I think that I would like to use this system frequently*) s'est vu ajouter *si j'étais game designer* afin de préciser cette cible. De plus le questionnaire utilise à l'origine le terme *system* pour désigner le système évalué, or il a été démontré que ce terme peut être modifié sans que cela n'ait d'impact sur les résultats

(Brooke, 2013), nous l'avons donc remplacé par *modèle* afin de préciser que ce n'est pas l'outil qui est évalué mais bien le modèle de stratégie.

6.3.5 Résultats

Cette évaluation a été réalisée auprès d'étudiants de l'école ISART Digital de Paris[3] spécialisée dans le jeu vidéo, nous avons cependant dû compléter par des personnes extérieures pour atteindre les 18 participants minimum requis. Afin de limiter les biais nous avons réparti autant que possible entre les six groupes les participants suivant leurs connaissances en programmation, en game design et en modding, ainsi que les étudiants de l'ISART et les autres.

Résultats SUS Pour analyser les résultats du questionnaire SUS, la procédure est la suivante :

- Obtenir un score entre 0 et 4 pour chaque affirmation.
  - Pour les affirmations 1, 3, 5, 7 et 9, l'échelle de Likert est numérotée de 0 (pas du tout d'accord) à 4 (tout à fait d'accord).
  - Pour les affirmations 2, 4, 6, 8 et 10, l'échelle de Likert est numérotée de 4 (pas du tout d'accord) à 0 (tout à fait d'accord).
- Les scores des affirmations sont additionnés pour obtenir un score total entre 0 et 40.
- Le score est multiplié par 2,5 pour obtenir un score entre 0 et 100.

A partir de ces résultats nous avons effectué des tests de student sur chacun des groupes ainsi que sur l'ensemble des résultats. Les moyennes, écarts-types et degrés de significativité sont indiqués sur la figure 6.8. Nous pouvons voir que les résultats ne sont pas significatifs et donc que notre hypothèse **H1** n'est pas validée mais n'est pas non plus réfutée. Les réponses tendent cependant à aller à l'inverse de cette hypothèse. Le détail des résultats peut être trouvé en annexe E.2.2.

Résultats sur l'expressivité Afin d'analyser les résultats obtenus avec l'affirmation *La stratégie créée correspond aux instructions qui vous ont été données*, l'échelle de Likert a été numérotée de 1 à 5. Nous avons également effectué des tests de student dont les résultats sont présentés sur la figure 6.9. Les résultats sont plus partagés que pour l'utilisabilité et sont très différents selon la directive donnée. Les résultats ne sont pas significatifs sauf dans le cas de la directive 3. L'hypothèse **H2** n'est donc ni validée ni réfutée.

L'ensemble du questionnaire et le détail des réponses des participants peuvent être trouvés en annexe E.2.

[3] `https://www.isartdigital.com/fr/` (accédé le 6 février 2017)

Directive 1 : Gagner avec la technologie	**Directive 2 : Créer un comportement économe**	**Directive 3 : S'adapter aux ressources découvertes**
Moyenne Editeur : 45,83 Ecart-type Editeur : 19,2787 Moyenne XML : 58,75 Ecart-type XML : 13,5785 P-value (degré de significativité) : **0,21**	Moyenne Editeur : 61,25 Ecart-type Editeur : 18,8911 Moyenne XML : 81,67 Ecart-type XML : 11,2546 P-value (degré de significativité) : **0.052**	Moyenne Editeur : 57,5 Ecart-type Editeur : 22,8035 Moyenne XML : 61,25 Ecart-type XML : 28,4495 P-value (degré de significativité) : **0.81**
Toutes directives confondues		
Moyenne Editeur : 54,86 Ecart-type Editeur : 20,3186 Moyenne XML : 67,22 Ecart-type XML : 21,0023 P-value (degré de significativité) : **0.08**		

FIGURE 6.8: Résultats des tests de student sur le questionnaire SUS

Directive 1 : Gagner avec la technologie	**Directive 2 : Créer un comportement économe**	**Directive 3 : S'adapter aux ressources découvertes**
Moyenne Editeur : 3,67 Ecart-type Editeur : 1,5055 Moyenne XML : 3,67 Ecart-type XML : 1,5055 p-value (degré de significativité) : **1**	Moyenne Editeur : 3 Ecart-type Editeur : 1,2649 Moyenne XML : 3,67 Ecart-type XML : 1,2111 p-value (degré de significativité) : **0.37**	Moyenne Editeur : 3,83 Ecart-type Editeur : 0,7528 Moyenne XML : 1,83 Ecart-type XML : 1,169 p-value (degré de significativité) : **0.007**
Toutes directives confondues		
Moyenne Editeur : 3,5 Ecart-type Editeur : 1,2005 Moyenne XML : 3,0556 Ecart-type XML : 1,5136 p-value (degré de significativité) : **0.34**		

FIGURE 6.9: Résultats des tests de student sur le respect des directives

6.3.6 Discussion

L'évaluation avait pour but d'évaluer l'expressivité et la transparence de notre modèle. Les hypothèses posées n'ont pas été validées ni réfutées. Nous revenons sur ces résultats et discutons à la fois la méthode d'évaluation ainsi que la signification des résultats obtenus.

Pour expliquer les résultats du questionnaire SUS pour l'hypothèse **H1** qui ne permettent pas de conclure sur l'utilisabilité de notre modèle, plusieurs explications nous semblent possibles mais nécessiteraient des évaluations supplémentaires.

- Une première explication provient d'une erreur que nous avons fait lors de la mise en place de ce test : pour évaluer l'utilisabilité des deux méthodes il aurait fallu proposer le même degré d'expressivité. Pour obtenir la même expressivité, le XML proposé aurait nécessité beaucoup plus de paramètres, or si l'utilisabilité du XML avec si peu de paramètres ne semble pas poser de problèmes, nous supposons néanmoins que celle-ci se détériore avec une complexité plus grande. Il serait donc intéressant de réaliser une évaluation sur un jeu de plus grande complexité ou simplement sur le jeu utilisé mais en augmentant son expressivité, de façon à obtenir un nombre de paramètres plus grand. Cependant dans le cas d'un jeu plus complexe, une telle évaluation nécessiterait une grande disponibilité des participants puisque l'on peut supposer d'après cette première évaluation qu'une journée voire plusieurs seraient nécessaires. Cette évaluation aurait cependant un intérêt double puisqu'elle permettrait également d'évaluer la capacité à passer à l'échelle pour notre modèle.

- Une deuxième piste pour expliquer les résultats obtenus est l'influence de la qualité du prototype sur l'expérience des participants. En effet de nombreux retours ont mentionné des crashs du logiciel et d'autres ont mentionné des difficultés techniques sur son utilisation. De plus le questionnaire SUS s'adresse à un système dans son ensemble et il ne semble pas certain que la modification du terme *système* en *modèle* soit suffisant. L'aspect instable et rudimentaire de l'outil peut donc avoir desservi notre expérimentation et a probablement influencé les réponses des participants. L'amélioration de l'interface est une piste qui pourrait diminuer l'impact négatif de l'outil sur la perception du modèle. A cela s'ajoute la qualité du jeu qui ne présentait malheureusement pas de représentation graphique, cependant cela s'appliquant aux deux méthodes (la nôtre ainsi que le XML) les résultats obtenus n'ont pas dû être influencés.

- Enfin une dernière explication qui rejoint en partie la première est le temps nécessaire pour la prise en main de notre modèle. Nous retirons de l'évaluation un résultat auquel nous nous attendions : notre modèle nécessite malgré tout un temps de formation, et les explications fournies n'étaient pas suffisantes pour maîtriser entièrement le modèle. Il serait alors intéressant d'évaluer la durée du temps de formation nécessaire mais cela suppose une évaluation plus longue ainsi qu'une personne maîtrisant le modèle et disponible pour répondre aux interrogations.

En ce qui concerne les résultats pour l'hypothèse **H2**, ceux-ci sont très différents selon la directive fournie. La directive ayant fournie un résultat positif pour notre modèle est celle centrée sur l'adaptation de la stratégie au cours du jeu. En effet une divergence fondamentale entre le XML et notre modèle est la possibilité d'avoir des informations différentes qui sont fournies au module tactique au cours de la partie. Il semble difficile d'imaginer un ensemble de paramètres fixes permettant cette liberté d'expression. Nous nous sommes également rendu compte après coup du manque de pertinence de notre méthodologie. En effet, l'expressivité n'est pas quelque chose qui se mesure dans l'utilisation mais qui dépend de notre définition des paramètres XML. L'hypothèse devrait davantage porter sur le temps nécessaire pour atteindre un certain objectif de design. Nous avions voulu prendre en compte ce temps dans notre évaluation et nous avons récupéré le temps passé par chaque participant sur chaque méthode, cependant les résultats étaient trop faussés pour être exploitables : le temps d'édition de la stratégie a été fortement ralenti par les crashs de l'éditeur, et les expressivités étant différentes, les comportements obtenus ne sont pas comparables. De plus, si une nouvelle évaluation devait être mise en place, il serait nécessaire de travailler en collaboration avec des game designers pour définir des directives représentatives de leur travail.

6.4 Bilan du chapitre

Nous avons mené un test utilisateur qui a eu pour objectif d'obtenir des avis extérieurs sur le modèle présenté dans cette thèse et pour mesurer son expressivité et sa transparence. Suite à des résultats encourageants, nous avons mis en place une évaluation qui a eu pour objectif d'évaluer notre modèle en le comparant à une méthode proposant des paramètres dans un fichier XML. L'évaluation s'est faite sur deux caractéristiques : la transparence et l'expressivité. Les résultats obtenus ne permettent pas de valider le modèle mais encouragent à effectuer une évaluation de plus grande envergure utilisant un jeu plus complexe. Celle-ci nécessiterait un temps de formation plus long sur les deux méthodes mais permettrait également d'évaluer leur passage à l'échelle. Le modèle peut entre temps être retravaillé pour améliorer son expressivité, et l'outil peut être amélioré afin de le rendre plus utilisable et de limiter son influence sur la perception du modèle. Il serait également intéressant d'effectuer des évaluations de notre modèle face à d'autres méthodes utilisées par l'industrie.

Chapitre 7

Conclusion Générale

Dans le chapitre 2 nous avons pu voir que l'IA a une place importante dans l'expérience de jeu proposée et que celle-ci est contrôlée par le game designer. Or, lors de la création d'un jeu le game designer délègue souvent l'implémentation de l'IA à des développeurs spécialisés qui ont comme objectif de créer des IA produisant l'expérience voulue. Le game designer doit alors communiquer sa vision de l'expérience de jeu, ainsi que les attentes plus particulières pour les IA, aux développeurs. Cette communication manque de formalismes adaptés ce qui entraine une communication incomplète et parfois erronée à laquelle s'ajoutent souvent des évolutions de l'expérience désirée. Cela induit de nombreuses boucles de développement-tests.

Les travaux présentés dans ce manuscrit ont pour objectif de fournir au game designer un contrôle partiel mais direct sur l'IA afin de réduire ces échanges coûteux en temps. Nous avons défini qu'un accès sur haut-niveau qui vient personnaliser l'IA était le plus adapté. Nous nous sommes alors intéressés à la création d'IA guidant le comportement des opposants dans un jeu de stratégie au tour par tour car le niveau stratégique de la prise de décision et l'impact de la personnalité y sont particulièrement développés. Nous avons défini que ce contrôle devrait être à la fois expressif, pour limiter les interventions des développeurs, et transparent, afin que le game designer puisse l'utiliser efficacement.

Dans le chapitre 3 nous avons étudié les stratégies d'entreprise et militaires afin de mieux définir ce qu'est une stratégie et quelles en sont les composantes. Dans le chapitre 4 nous nous sommes intéressés aux méthodes informatiques utilisées actuellement dans l'industrie et dans la recherche académique pour les jeux de stratégie. Nous avons particulièrement mis en avant leur potentiel à être personnalisées par des contrôles intelligibles et qui pourraient donc, en réponse à notre problématique, être mis à disposition des game designers.

Pour répondre à notre problématique, nous proposons donc un modèle de stratégie accompagné d'un moteur de stratégie présentés dans le chapitre 5. Nous allons dans un premier temps revenir sur cette proposition et sur les résultats obtenus lors de l'évaluation décrite dans le chapitre 6, puis évaluer les limites et perspectives de ces travaux.

7.1 Résumé de la proposition

Afin de fournir un contrôle haut-niveau au game designer, notre proposition se décompose en deux parties : un modèle de stratégie qui lui permet de définir un raisonnement stratégique, et un moteur qui permet d'interfacer la stratégie créée avec le module IA créé par les développeurs que nous appelons module tactique. Pour mieux définir la composition d'une stratégie dans notre modèle, nous nous sommes inspirés des travaux étudiés dans les chapitres 3 et 4. Nous avons notamment mis en avant les points communs à la fois du fonctionnement des stratégies mais également des différents contextes dans lesquels elles sont mises en place.

7.1.1 Le contexte

Dans un jeu 4X, un joueur doit contrôler une organisation afin de la faire prospérer dans un environnement fournissant quelques ressources de base et dans lequel évoluent d'autres organisations similaires. L'évolution des autres organisations peut être contrôlée par d'autres joueurs ou par des IA. C'est à ces IA que nous nous intéressons. Elles contrôlent donc une organisation qui doit prospérer dans un environnement où d'autres organisations évoluent également et dans un contexte de compétition.

Les organisations vont alors évoluer en taille et en compétence. En atteignant un certain stade, leur gestion peut devenir compliquée et nécessite une méthode définissant comment les ressources vont être utilisées pour continuer à faire prospérer l'organisation : c'est l'apparition d'un besoin de stratégie. Celle-ci permet en effet d'apporter une cohérence aux actions d'une organisation de grande taille qui doit prendre des décisions car plusieurs évolutions sont possibles. Dans les jeux de stratégie au tour par tour, on retrouve bien souvent de nombreux domaines sur lesquels les joueurs peuvent agir : le militaire presque toujours, mais également la diplomatie, la religion, le commerce, les sciences et techniques... Des décisions qui relèvent davantage des préférences que de l'optimisation doivent donc être prises. C'est ici que le game designer peut venir ajouter de la personnalité à l'IA.

De plus l'organisation se place dans un environnement dynamique et imprévisible. En effet l'environnement en lui-même n'est pas entièrement connu car les jeux de stratégie font fréquemment appel au brouillard de guerre qui conditionne les connaissances de l'environnement à la position des unités dans celui-ci. Par ailleurs les autres organisations dans l'environnement vont évoluer sous la direction d'une autre IA ou bien d'un joueur, or ces évolutions ne peuvent pas être prévues avec certitude. Les indications données par le game designer doivent donc pouvoir s'adapter à un environnement dynamique.

7.1.2 La construction de la stratégie

Les décisions multiples appellent à une division de la prise de décision sous forme hiérarchique, nous avons pu le voir à la fois dans la définition commune d'une stratégie ainsi que dans les travaux précédents sur l'IA dans les jeux de stratégie. Cette décomposition est double : elle

peut servir à diviser en plusieurs solutions ou plusieurs étapes d'une solution qui seront choisies en fonction du contexte, ou bien elle peut servir à décomposer la prise de décision en plusieurs domaines donnant lieu à plusieurs prises de décision plus simples qui sont effectuées en parallèles. A partir du premier type de décomposition nous avons créé des comportements logiques utilisant les FSM à la fois pour leur expressivité et pour leur facilité d'utilisation. Pour le deuxième type de décomposition nous avons créé les comportements parallèles que nous avons définis en nous inspirant de plusieurs travaux. Notre modèle laisse libre l'agencement de ces deux types de comportements mais garde une forme hiérarchique. En laissant libre cet agencement, nous donnons au game designer une plus grande expressivité et nous permettons également au modèle de pouvoir être personnalisé pour différents jeux.

L'adaptation avec les comportements logiques

Les comportements logiques ont été définis afin de s'adapter à l'environnement dynamique mais également aux retours sur la mise en place de la stratégie actuelle. C'est pour cela que nous avons choisi d'utiliser des FSM. Celles-ci proposent en effet une représentation visuelle et accessible d'un choix entre plusieurs options, qui permet de prendre en compte toute sorte d'information par le biais de conditions binaires placées sur des transitions entre deux options. Elles permettent également de prendre en compte les retours observés sur la mise en place d'une stratégie puisque les transitions sont dépendantes de l'état sélectionné. Nous avons également ajouté un système de messages permettant d'obtenir des informations spécifiques au comportement sélectionné et également d'en envoyer. Les messages se présentent sous la forme de chaines de caractères personnalisables par le game designer de façon à ce qu'ils soient intelligibles. Les messages reçus sont ensuite utilisés de la même manière que les conditions, en étant placés sur des transitions qu'ils peuvent alors déclencher.

La répartition des ressources avec les comportements parallèles

Les comportements parallèles permettent de définir différents domaines à étudier en parallèle. Ces domaines peuvent posséder leur propre stratégie sous la forme d'un comportement logique ou parallèle, ou peuvent se présenter sous la forme d'un comportement élémentaire afin de donner directement un objectif au module tactique. L'agencement des comportements étant laissé libre, la définition des différents domaines l'est également. De la même manière cela permet à la fois une plus grande expressivité et une réutilisabilité du modèle. Nous avons donné plusieurs contrôles supplémentaires au game designer sur cette décomposition afin d'améliorer l'expressivité. Pour cela nous nous sommes inspirés de travaux précédents et avons sélectionné ceux présentant des outils facilement compréhensibles, afin de garder notre objectif de transparence. Les éléments que nous avons alors intégrés à notre modèle sont les suivants :

- des priorités permettant de définir un ordre dans la répartition des ressources,
- des importances permettant d'influencer les quantités de ressources allouées,

- des préférences permettant d'influencer la nature des ressources allouées.

Nous avons également eu besoin, pour la répartition des ressources, de créer un protocole de demande de ressources pour communiquer avec le module tactique. Nous avons ainsi divisé les ressources demandées en deux parties : les ressources indispensables pour entamer le comportement, et les ressources qui vont venir le renforcer.

7.1.3 L'évaluation du modèle

Afin d'évaluer notre modèle, nous avons mis en place un test utilisateur puis une évaluation. Les résultats du premier ont été encourageants à la fois sur l'expressivité et la transparence du modèle, cependant pour plusieurs raisons nous avons nuancé ces résultats (voir la section 6.2.4). A partir des observations qui ont été faites sur ce test, nous avons mis en place une évaluation dont les résultats ont mis en avant les limites de notre modèle et ont ouvert plusieurs pistes de réflexion sur les améliorations à y apporter.

7.2 Limites et Perspectives

Suite à l'évaluation, nous proposons dans cette partie plusieurs pistes d'amélioration pour dépasser les limites actuelles du modèle. Nous proposons également des perspectives plus larges sur les réflexions ouvertes par les travaux menés au cours de cette thèse.

7.2.1 Les évolutions du modèle et du moteur

A court terme nous pouvons envisager de venir apporter des modifications sur le modèle afin d'améliorer son expressivité tout en cherchant à maintenir sa transparence. Plusieurs pistes ont été envisagées mais laissées de côté dans un premier temps pour garder un modèle intelligible, et il serait intéressant d'étudier la pertinence de chacune et son influence sur la transparence.

Comportements logiques

Les comportements logiques proposent actuellement de sélectionner un comportement en fonction des messages reçus et de conditions dans le contexte observé. Nous proposons plusieurs perspectives d'amélioration.

Contexte Actuellement les conditions proviennent d'un contexte indéfini qui possède pour seule contrainte d'être une association de noms et de valeurs. Les conditions sont définies avec une grammaire logique composée de comparaisons agrégées par les opérateurs logiques AND, OR et NOT. Plusieurs types d'opérateurs pourraient venir augmenter l'expressivité de ces conditions. Par exemple les opérateurs de calcul +, -, * et / permettraient d'effectuer des calculs sur les valeurs numériques récupérées. Les quantificateurs $\forall$ et $\exists$ permettraient également de diversifier les conditions possibles. De plus si l'on suppose un contexte hiérarchique, les variables peuvent

avoir des ramifications et il serait alors possibles de faire plusieurs conditions sur plusieurs ramifications d'un même objet (7.1).

Listing 7.1: Exemple de l'utilisation de ∃ dans les conditions

```
∃ x && x.type == voiture && x.couleur == rouge
```

Gestion avancée des messages Nous avons également pensé à deux mécanismes supplémentaires qui permettraient d'enrichir les transitions : une gestion du temps et une gestion de décompte des messages. Pour le décompte des messages on peut imaginer un tableau de messages pour lequel chacun aurait un compteur et leur utilisation sur les transitions serait alors similaire aux autres conditions puisqu'un message serait alors une autre variable numérique du contexte observé. La gestion du temps peut également être vue comme un message global qui serait diffusé à tous les comportements en cours et qui viendrait incrémenter son compteur. Pour les jeux de stratégie au tour par tour il serait logique d'utiliser plutôt un compteur de tour. Il serait alors possible de faire évoluer la stratégie au cours du jeu même si l'environnement ne présente pas de conditions particulières. Cela permettrait notamment de ne pas avoir un jeu trop monotone et de déclencher des attaques par exemple si le jeu n'en déclenche pas autrement.

Nous aurions également souhaité explorer la possibilité de passer des valeurs numériques dans les messages de façon à pouvoir les utiliser dans des comparaisons sur des transitions. Il faudrait alors voir comment un compteur de messages pourrait être conjugué à la possibilité de remonter une valeur numérique. On peut envisager la possibilité d'additionner les valeurs remontées, ou bien de ne garder que la dernière. D'autres possibilités peuvent probablement être imaginées.

Paramétrage Le modèle bénéficierait grandement d'un système de paramétrage des comportements ; en effet actuellement pour modifier légèrement une répartition des ressources il est nécessaire de créer deux comportements parallèles différents et d'effectuer la modification par le biais d'une transition dans un comportement logique. De plus si l'on veut par exemple déclencher un comportement correspondant à une attaque sur un ennemi en particulier, il n'existe aucune manière de prendre la décision à un haut-niveau et de la transférer le long d'une branche. Les paramètres pourraient ainsi être définis soit sur l'état soit provenir d'une transition contenant le quantificateur ∃. Pour cela il serait nécessaire d'autoriser aux objectifs de définir des paramètres. Il pourrait être nécessaire pour cela de définir des types agrégés, comme dans notre exemple une armée, qui serviraient à définir le type de paramètre demandé. Il pourrait être intéressant de voir le parallèle qui existe entre le contexte agrégé que nous avons évoqué précédemment avec cette notion de type agrégé pour les paramètres. On peut voir un premier lien entre les deux puisque le paramètre de type agrégé pourrait être utilisé sur une transition. Reprenons notre exemple d'un comportement qui effectue une attaque, ce comportement pourrait être un comportement logique qui veut modifier le sous-comportement choisi en fonction du rapport de force : si l'opposant prend un avantage trop important il va par exemple sélectionner

un comportement de fuite. L'exemple de l'armée ennemie serait alors à la fois un type agrégé correspondant à un ensemble de soldats, et serait également nécessaire dans le contexte agrégé notamment pour pouvoir effectuer des conditions sur la force de l'armée.

Stack-Based Finite State Machines Le système de stack-based FSM, présenté par Tozour (2004), propose de garder une trace des états sélectionnés. De cette manière lorsqu'un état est sélectionné pour réagir à un évènement en particulier et interrompt son comportement en cours, il peut lorsque l'évènement a été géré retourner à son comportement d'origine. Ce fonctionnement peut se faire plusieurs fois de façon à avoir plusieurs comportements interrompus en attente. Il peut être intéressant d'utiliser cette méthode de la manière dont Tozour (2004) la présente, mais nous voyons également un intérêt à son utilisation dans les conditions sur les transitions. En effet, cette pile des états sélectionnés peut servir à garder une trace des actions faites précédemment. Actuellement les décisions prises ne sont affectées que par le comportement courant, mais on peut imaginer enrichir le raisonnement avec une mémoire plus longue de la stratégie.

Comportements parallèles

Les comportements parallèles actuels proposent plusieurs paramètres influençant la répartition des ressources entre les différents sous-comportements. Les paramètres *importance* et *priorité* influencent les quantités allouées à chaque sous-comportement alors que les préférences viennent modifier la nature des ressources allouées.

Adaptabilité Les comportements parallèles sont actuellement fixes et limitent l'adaptabilité du modèle. Une demande qui a été faite régulièrement de la part des participants à notre test et notre évaluation était la possibilité de régler les importances en fonction d'une variable numérique dans le contexte. On peut alors imaginer que l'importance soit exprimée sous la forme d'une variable du contexte éventuellement à l'intérieur d'une opération dont le résultat déterminerait l'importance. Nous avons évoqué la possibilité d'utiliser des paramètres pour personnaliser les comportements et leur utilisation dans les comportements logiques. Les variables utilisées dans le calcul des importances pourraient alors également provenir d'un paramètre passé au comportement. On peut évidemment également imaginer qu'il serait possible de faire évoluer la priorité de la même manière.

Répartition des ressources Plusieurs modifications pourraient venir améliorer notre répartition des ressources pour la rendre plus efficace. La première serait de prendre en compte la position des ressources dans la répartition des ressources, cela suppose cependant une localisation du point d'action de l'objectif qui serait alors une information supplémentaire que ce dernier devrait transmettre. On peut également se demander comment cela serait couplé avec des préférences autres. Dans le cas de l'attaque d'une ville à laquelle on veut affecter des ressources pour la défense, doit-on préférer le soldat avec le plus haut niveau de vie mais qui

est situé loin de l'attaque ou un soldat plus proche mais dont le niveau de vie est plus faible et qu'on risque alors de tuer s'il va défendre ?

Enfin nous avons observé un problème au niveau du fonctionnement de notre moteur : les poids des comportements parallèles influencent la répartition des ressources à un instant t, or dans le jeu utilisé comme dans beaucoup de jeux de stratégie au tour par tour plusieurs actions sont ponctuelles et ne nécessitent pas plusieurs tours de jeux pour être effectuées. Cela résulte en un sous-comportement se voyant affecter à chaque tour l'unique ressource *bois* disponible par exemple car il possédait une priorité et un poids plus importants que les autres. Il serait plus intéressant et plus juste d'utiliser les poids sur la durée à la manière du système mis en place par Souza et al. (2014). Si un sous-comportement possède la même priorité et un poids deux fois plus important qu'un autre, tout le long de la réalisation de l'objectif le premier se verrait remettre deux fois plus de ressources que le deuxième.

7.2.2 Les modules annexes

Les messages peuvent facilement être étendus pour prendre en compte des messages venant de modules annexes tel qu'un module de modélisation du joueur. On peut également imaginer passer par le contexte qui peut contenir toute sorte d'analyse de l'état du jeu.

La modélisation du joueur

La modélisation du joueur sera particulièrement intéressante étant donné l'objectif d'un jeu de produire une expérience enrichissante et puisque comme nous avons pu le voir les joueurs peuvent être très différents et donc vivre différemment une même situation. La modélisation du joueur permettrait alors d'avoir un retour sur l'expérience provoquée. Le modèle ayant été conçu afin de pouvoir s'adapter, il permettrait d'utiliser ce retour pour modifier le comportement des IA.

Certains travaux se sont déjà intéressés à l'adaptation du comportement des PNJ au profil du joueur. C'est le cas par exemple de Thue et al. (2007) qui propose une sélection des PNJ en fonction de leur correspondance au profil obtenu par la modélisation du joueur. Chaque PNJ possède son comportement propre et est annoté par le game designer pour sa correspondance à cinq profils distincts prédéfinis. L'adaptation est assez contraignante car elle nécessite de définir des comportements différents et change complètement le comportement sélectionné, néanmoins avec notre modèle il serait possible de garder le comportement mais de venir y apporter des modifications en fonction du profil du joueur.

L'analyse du jeu

Afin de rendre plus pertinentes les informations du contexte, on peut imaginer coupler le modèle avec un contexte abstrait tel que dans les travaux de Galvao Madeira (2007). Ce dernier propose en effet un système de raisonnement hiérarchique qu'il annexe d'un système d'abstraction des données. Ces travaux s'inspirent fortement du fonctionnement militaire, c'est pourquoi les données abstraites concernent les armées : un groupes d'unités militaire est ainsi représenté

par son centre de masse et ses niveaux de concentration, de santé, de qualité, de mobilité et de munitions. Cependant il serait intéressant d'étendre cette méthode à des notions plus abstraites. Par exemple pour la diplomatie, plus que d'étudier l'ennemi dans son individualité, c'est le système de forces reliant tous les acteurs de l'environnement qu'il est intéressant d'étudier : Fiévet (1992) parle du *centre de gravité des forces* pour définir l'étude des différents acteurs afin de déterminer ses propres points forts et points faibles. En étudiant les capacités de chacun (ressources, développement possible...), on peut repérer les grandes lignes de force et les possibilités d'alliances et de conflits.

7.2.3 La place du game designer

Notre proposition participe à un débat plus large qui est le rôle du game designer dans la création d'un jeu. Dans cette thèse nous avons pris la décision de donner un accès direct au game designer sur les IA car nous nous sommes intéressés à cet aspect du jeu en particulier. Cependant le game designer tel que nous l'avons décrit supervise l'ensemble de la création du jeu et il serait intéressant de mettre en parallèle le contrôle que nous donnons sur les IA avec les contrôles qui pourraient être donnés sur les autres domaines. Une réflexion sur une plateforme unique pour les contrôles donnés au game designer pourrait par exemple être mise en place.

Le chapitre 2 aborde la question de l'expérience qui se crée lors de l'utilisation d'un jeu par un joueur. Il est alors intéressant de voir que les IA y contribuent car elles dirigent des participants à l'histoire qui se crée à ce moment-là. Notre modèle offre un contrôle sur une IA guidant l'un des opposants d'un jeu 4X, mais pour offrir une vision globale au game designer, il serait nécessaire d'étudier et de formaliser les relations entre les différents éléments d'un jeu qui créent l'expérience complète. Une première piste serait d'étudier les possibilités de liens entre notre modèle et les différents descriptifs du fun.

7.3 Bilan

Les travaux présentés dans ce manuscrit proposent un début de réflexion sur un contrôle haut-niveau des IA de jeux de stratégie accessible pour des non-développeurs. La proposition est composée d'un modèle de stratégie et d'un moteur offrant un contrôle partiel inspiré de l'utilisation commune d'une stratégie. Les stratégies créées avec ce modèle permettent de définir des objectifs haut-niveau ainsi que des paramètres influençant l'allocation des ressources. De plus les stratégies s'adaptent au contexte observé dans le jeu. L'évaluation qui a été menée montre que nos travaux ne sont pas aboutis et nécessitent d'être retravaillés et améliorés. Plusieurs pistes d'amélioration ont été proposées ainsi que des pistes sur des modes d'évaluation plus pertinents que ceux utilisés. Parmi les perspectives décrites, nous proposons plusieurs améliorations du modèle qui avaient été mises de côté afin de commencer avec un modèle simple dans le but de le complexifier progressivement car l'équilibre entre l'intelligibilité et l'expressivité est difficile à maintenir. Ces perspectives supposent donc des évaluations permettant d'évaluer l'apport en expressivité et la perte en intelligibilité. De plus notre modèle offre plusieurs possibilités de

combinaison avec des modules annexes permettant ainsi d'explorer plusieurs problématiques de la recherche en IA comme par exemple la modélisation du joueur. Enfin notre problématique s'inscrit dans une autre plus large s'intéressant à l'expérience du joueur et à sa formalisation.

Annexes

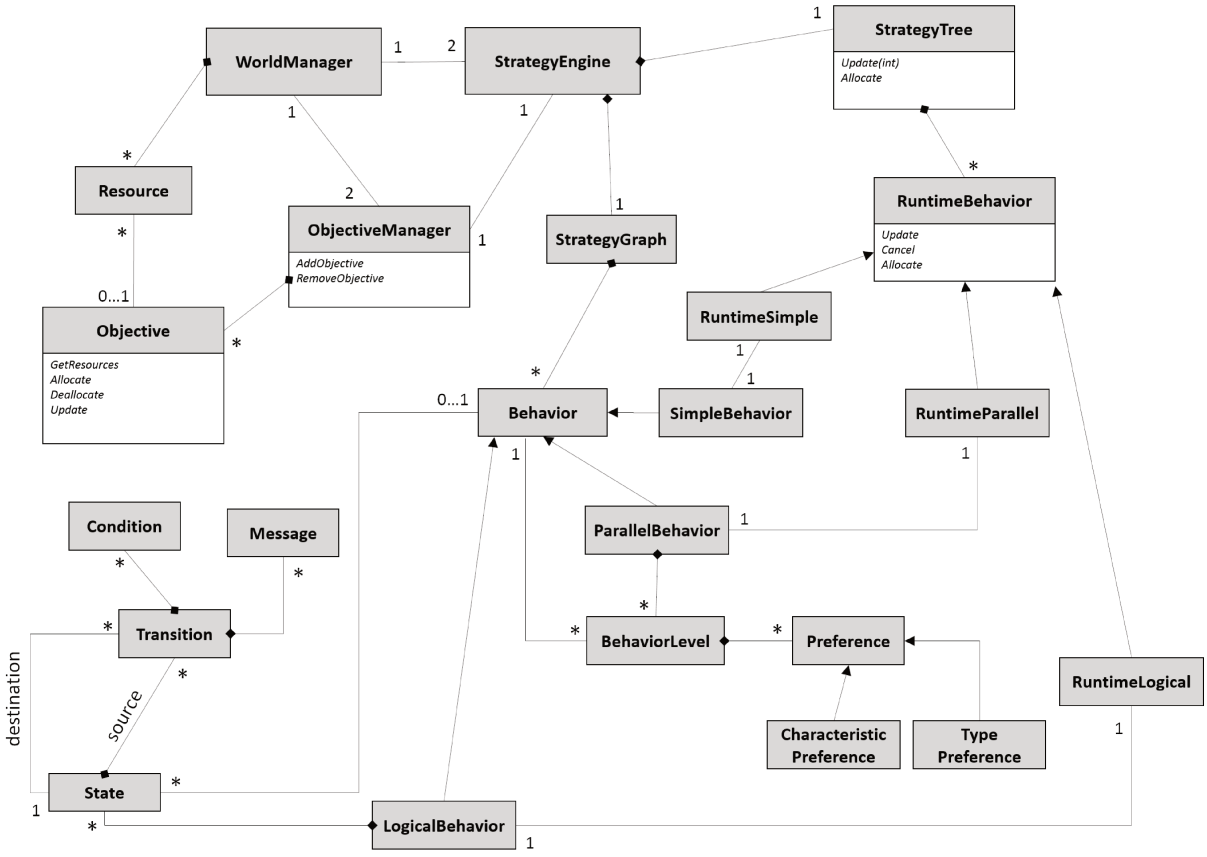
Annexe A

Publications

- *Towards a resource-based model of strategy to help designing opponent AI in RTS games* - Juliette LEMAITRE, Domitile LOURDEAUX, Caroline CHOPINAUD, 7th International Conference on Agents and Artificial Intelligence (ICAART), 2015, p.210-215.
- *Vers un modèle de stratégie basé sur la gestion des ressources pour la création des IA dans les RTS* - Juliette LEMAITRE, Domitile LOURDEAUX, Caroline CHOPINAUD, Treizièmes Rencontres des Jeunes Chercheurs en Intelligence Artificielle (RJCIA), 2015.

Annexe B

UML


WorldManager
StrategyEngine
StrategyTree
Update(int)
Allocate
Resource
ObjectiveManager
AddObjective
RemoveObjective
StrategyGraph
RuntimeBehavior
Update
Cancel
Allocate
Objective
GetResources
Allocate
Deallocate
Update
RuntimeSimple
Behavior
SimpleBehavior
RuntimeParallel
Condition
Message
ParallelBehavior
Transition
BehaviorLevel
Preference
RuntimeLogical
destination
source
Characteristic Preference
Type Preference
State
LogicalBehavior

Annexe C

Exemples de fichiers

C.1 Exemple de stratégie en JSON

```
{
  "NBehavior":6,
  "Root":4,
  "Behaviors":
  {
    "Objectives":[
          {"idx":0,"name":"AmeliorerTechno","x":560.0,"y":198.04},
          {"idx":1,"name":"DeveloppperDemographie","x":347.0,"y
             ↪ ":190.04},
          {"idx":2,"name":"Attaquer","x":200.0,"y":257.04},
          {"idx":3,"name":"PreparerArmee","x":32.0,"y":268.04}
        ],
        "Parallels":[
          {"idx":4,"name":"Strategie generale","x":289.0,"y
             ↪ ":7.0399999999999991,
          "behaviors":[
                {"behavior":0,"source":2,"dest":0},
                {"behavior":1,"source":2,"dest":0},
                {"behavior":5,"source":3,"dest":0}
          ],
          "NLevels":2,
          "levels":[
            {"behaviors":[
                  {"behavior":0,"importance":7,"preferences":[]},
                  {"behavior":1,"importance":3,"preferences":[]}]
                },
```

```
                {"behaviors":[
                  {"behavior":2,"importance":1,
                  "preferences":[
                    {"globaltype":"Soldat","specifictype":"max
                       ↪ Puissance"}
                  ]}
      ]}
    ]}
  ],
  "Logicals":[
        {"idx":5,"name":"comportement agressif","x":96.0,"y
           ↪ ":133.04,
        "behaviors":[
              {"behavior":3,"source":2,"dest":0},
              {"behavior":2,"source":1,"dest":0}
        ],
        "initial":0,
        "states":[
      {"behavior":0,"message":"","x":132.0,"y":41.039999999999992},
              {"behavior":1,"message":"","x":207.0,"y":321.04}],
        "transitions":[
          {"source":{"state":0,"connector":2},"destination":{"state
             ↪ ":1,"connector":3},
            "conditions":[
                  {"m1":"puissance","signe":">=","m2":"20"}
              ],
              "messages":[]
            },
            {"source":{"state":1,"connector":1},"destination":{"
               ↪ state":0,"connector":1},
              "conditions":[
                {"m1":"puissance","signe":"<=","m2":"4"},
                {"m1":"ouvriers","signe":">","m2":"0"}
              ],
              "messages":[]
            },
            {"source":{"state":0,"connector":3},"destination":{"
               ↪ state":1,"connector":3},
              "conditions":[],
              "messages":[{"text":""}]
```

```
                }
    ]}
  ]}
}
```

C.2 Exemple de fichier XML

```
<strategy>
        <importanceArmy>1</importanceArmy>
        <importanceTechno>3</importanceTechno>
        <importanceDemo>2</importanceDemo>
        <attackTendency>0.1</attackTendency>
        <protectTendency>0.3</protectTendency>
        <preferenceAttackTarget>Ouvrier</preferenceAttackTarget>
        <preferenceAttackSoldier></preferenceAttackSoldier>
        <preferenceProtectSoldier>max Vie</preferenceProtectSoldier>
        <preferenceProtectTarget>Ouvrier</preferenceProtectTarget>
</strategy>
```

Annexe D

Test utilisateur

D.1 Questionnaire

1. Es-tu étudiant ? (oui/non)
2. Si oui, dans quel domaine ?
3. Si non, quel est ton domaine d'activité ?
4. Sais-tu programmer ? (oui/non)
5. Si oui, as-tu déjà fait de l'intelligence artificielle ? Précisez l'expérience.
6. As-tu déjà fait du game design ? (oui/non)
7. Quel âge as-tu ?
8. Joues-tu ou as-tu déjà joué à des jeux-vidéo de stratégie ? (1 : Jamais à 6 : Très souvent)
9. As-tu déjà joué à des jeux de stratégie au tour par tour ? (oui/non)
10. Si oui, lesquels ?
11. Qu'attends-tu de l'IA contre laquelle tu joues ?
12. Trouves-tu la représentation de la stratégie sous forme de graphe adaptée ? (1 : Non, pas du tout à 6 : Oui, tout à fait)
13. As-tu compris lorsqu'une transition était déclenchée ? (oui/non)
14. As-tu compris à quoi servaient les messages qui pouvaient être mis sur les états ? (oui/non)
15. Connaissais-tu déjà le concept de machine à états ? (oui/non)
16. Trouves-tu les informations données par la fenêtre détail des comportements logiques suffisantes ? (1 : Non, pas du tout à 6 : Oui, tout à fait)

17. Qu'enlèverais-tu ou rajouterais-tu ?
18. As-tu des remarques à faire sur la fenêtre de création des comportements logiques ?
19. As-tu compris à quoi sert l'importance ? (oui/non)
20. As-tu compris quand mettre deux comportements dans un même niveau de priorité ? (oui/non)
21. As-tu compris quand créer un nouveau niveau de priorité ? (oui/non)
22. Trouves-tu les informations données par la fenêtre détail des comportements parallèles suffisantes ? (1 : Non, pas du tout à 6 : Oui, tout à fait)
23. Qu'enlèverais-tu ou rajouterais-tu ?
24. Trouves-tu l'interface facile d'utilisation ? (1 : Non, pas du tout à 6 : Oui, tout à fait)
25. Quels sont les points positifs de l'interface ?
26. Qu'enlèverais-tu ou rajouterais-tu à propos de l'interface ?
27. Es-tu satisfait des possibilités offertes par l'outil ? (1 : Non, pas du tout à 6 : Oui, tout à fait)
28. Que changerais-tu ?
29. Trouves-tu le modèle facile à comprendre ? (1 : Non, pas du tout à 6 : Oui, tout à fait)
30. Quelle a été ton impression générale de l'outil et du modèle sous-jacent ?
31. Quels sont pour toi les points forts de l'outil et du modèle présenté ?
32. Quels sont pour toi les points faibles de l'outil et du modèle présenté ?
33. Quelles seraient tes principales pistes d'amélioration ?
34. Le travail présenté offre-t-il pour toi de bonnes perspectives dans l'intelligence artificielle et le domaine des jeux-vidéo de stratégie ?
35. Pourquoi ?

D.2 Réponses

1. 92,3% oui
2. Génie des Procédés / Informatique(x5) / Tronc commun UTC (x4) / Mécanique / HuTech 01
3. IR Informatique embarquée

4. 69,2% oui

5. IA01 (systèmes experts, algorithmes génétiques, réseaux de neurones) / IA01, IA02 / IA01, IA02, et IA de jeux pour projets perso / Prolog et python (EdX Berkeley Artificial Intelligence)

6. 23,1% oui

7. 17(x2)/18/19(x2)/20(x5)/21/22(x2)

8. 2/3(x3)/4(x6)/5(x3)

9. 76,9% oui

10.
   - Heroes 3, Civ 4, Civ 5
   - civilization V
   - Civ 5 + STR (age of empire, starcraft)
   - Seigneur des anneaux la bataille pour la terre du milieu 1 et 2, endless legend, shetlers
   - Advanced Wars
   - warcraft/ageof (pour le tour/tour je ne sais plus !)
   - CIV IV, Europa Universalis, Total War
   - Heroes if might and magic
   - Civilisation
   - Civilization, Thea : The awakaning...

11.
   - Qu'elle soit assez forte et réactive pour ne pas me laisser gagner facilement
   - Qu'elle soit meilleure que les IA actuelles et avec des comportements plus variés (dans Civ certains adversaires ont toujours les mêmes stratégies)
   - qu'elle soit forte, et puisse me battre sans avoir des avantages de départ (plus de ressources, soldats plus puissant,...)
   - Proposé un challenge sans avoir un avantage injuste sur la vitesse de production ou de recolte. de la surprise, de l'adaptation à mes actions
   - Qu'elle sache réagir à mes actions et qu'elle n'attende pas sans rien faire.
   - Du challenge, pour pouvoir progresser en tant que joueur
   - Adaptation au style de jeu engagé, impliquer une notion de "faute strategique" pour rendre humain le comportement
   - Je sais pas
   - Innovante, surprenante
   - Qu'est ne soit pas trop idiote sinon c'est pas drôle

- qu'elle résiste, qu'elle pousse à réfléchir sur notre façon de jouer et à développer de nouvelles stratégies
- Imprévisibilité et originalité,

12. 4(x3)/5(x6)/6(x4)

13. 92,3% oui

14. 84,6% oui

15. 46,2% oui

16. 2(x3)/3(x3)/4(x4)/5(x2)/6

17.
- Il faut connaître le principe de machine à état (rapide à comprendre)
- Un ratio des objectifs à réaliser suivant tel parallèle me semble manquer d'informations car on ne réalisera pas forcément le même ratio si on a beaucoup de soldats/ouvrier que si on en a peu
- Possibilité de passer d'un état a un autre en fonction de choses comme le nombre de ressources disponibles (ex: plus de ressources -> attaque totale) ou la durée de la partie.
- Il aurait été bien d'avoir le nom des domaines sur les cases (logique, parallèle, objectif) ou une légende
- Un système plus claire sur les conditions, et messages
- Des jolies couleurs, et en vrai des indications sur les actions possibles ...
- Des aides sur ce qu'on peut mettre ou non, des exemples
- Guider un peu plus sur l'ajout de préférences
- – ce serait bien de pouvoir écrire les conditions de passage d'un état à l'autre (dans les comportements logiques) sur une seule ligne sinon quand il y a beaucoup de conditions (type plusieurs 'ou' logique), ça devient rapidement illisible et en plus il n'y a plus assez de 'ports' pour tirer ou accrocher les flèches
  – éventuellement enlever les comportements logiques et permettre directement des liens conditionnels d'un comportement parallèle à l'autre
- Rajouter le nombre de nœud pour tirer des flèches

18. As-tu des remarques à faire sur la fenêtre de création des comportements logiques ?

- Une liste déroulante serait plus adaptées car on ne sait pas vraiment les action a notre disposition.
- Pas très intuitive, il faut du temps pour comprendre comment l'utiliser (bulles d'aide ?)

- De base, le fait que la fenêtre soit vide alors même qu'on a déjà créé des relations père - fils entre les noeuds est perturbant, le fait de devoir faire clic-droit -> ajouter etat n'est pas très intuitif, peut être proposer de base à l'utilisateur de définir les comportements de transition pour les états fils? A voir.
- " As-tu compris à quoi servaient les messages qui pouvaient être mis sur les états ? " » je n'ai pas eu les explications sur ce sujet
- Il pourrait être intéressant de rajouter différents paramètres comme la vitesse de développement des paramètres parallèles.

19. 92,3% oui

20. 92,3% oui

21. 84,6% oui

22. 2(x2)/3(x3)/4(x3)/5(x3)/6(x2)

23.
- on réalise les options des niveaux de priorité suivant l'état du jeu actuel (ennemi proche de l'attaque, bâtiment en train de se faire détruire ...)
- Liste déroulante pour savoir exactement les sous comportements disponibles.
- Après explication j'ai compris quand/comment créer un nouveau niveau de priorité
- Ce n'est pas tant enlever ou rajouter, mais réorganiser ou expliciter, de sorte que les rôles respectifs de l'importance et de la priorité soient plus distincts. Déjà sémantiquement, importance et priorité sont peut être trop proches par rapport à leurs rôles dans l'interface.
- Plus d'explications sur ce qu'on peut mettre ou non et des exemples
- Rajouter quelques notes d'explications de l'effet de chaque flèche sur le résultat.

24. 2/3(x2)/4(x4)/5(x5)/6

25.
- l'interface est simple
- Pour l'instant 3 couleurs donc facile à comprendre
- graph, simple à comprendre, simple à réaliser
- Interface avec des boîtes et des nœuds bien adapté car permet d'avoir une vision d'ensemble de l'IA.
- Intuitif pour les mécanismes de base (créer un objectif et le relier à un comportement parallèle)
- Simple et intuitif, on se rend facilement compte de qu'est ce qui fait quoi.
- la visualisation sous forme de graphe, avec des couleurs, on s'y fait vite et le schéma de pensée de la

- construction d'une IA est plutôt clair
- C'est joli, nan vraiment ! Et c'est assez clair (sauf quelques exceptions)
- Très visuelle sur la logique de déroulement des tâches
- Graphique
- Pas compliqué
- le système de graphe qui visuellement rend bien les liens entre les différents noeuds, les couleurs, la fenêtre de détail
- Sa facilité d'utilisation, sa logique type algorithmique

26.
- j'ajouterais une palettes à outils pour éviter de trop cliquer
- Des systèmes de modules et de sous-graphes
- un modèle plus construit de base pour cadrer les choses avec par exemple l'état initial au milieu et un enchainement d'étapes en forme de cercle autour, avec la spécifications en l'action en fonction du recul par rapport à l'état initial
- Possibilité de mettre des commentaires, par exemple pouvoir donner un nom a une partie du graphe. pouvoir cacher certains objectifs aiderait aussi a pouvoir mieux se reperer lorsque le graphe deviendrait plus grand.
- Une légende avec les catégories d'actions, ce qu'on peut en faire (créer/supprimer/ajouter conditions) (avec des boutons/menus déroulants)
- Un système plus claire pour les messages, comment en créer de nouveaux etc...
- l'affichage des transitions dans la vue générale, on ne se rend pas assez bien compte lorsqu'on a beaucoup d'états de la logique générale.
- Des indications , info-bulles, boutons add/delete plutot que de passer par le clic droit. Et virer les plantages intempestifs <3
- Une barre sur le haut ou le côté avec les éléments que l'on peut utiliser de façon à pouvoir les drag and drop. Box selection.
- Une doc, des info bulles etc...
- le double-clic un peut pénible, il faudrait que cela ouvre directement la fenêtre contextuelle au clic simple, plus de 'ports' pour tirer les flèches et les ancrer
- Rajouter quelques notes notamment au niveau du développement des contraintes parallèles pourrait éclaircir l'utilisation pour un

27. 3(x2)/4(x3)/5(x4)/6(x4)

28.
- pour créer une bonne IA l'interface devient chargée, faire des développement caché sur les noeud père
- cadré plus au départ le modèle, mais avec possibilité de tout modifier si l'envie est présente

- Peut être faire un chemin d'accès plus simple pour créer/modifier des conditions sur les actions
- Je tenterais d'homogénéiser l'ensemble de l'interface, il y a trop d'action de sélection/-validation différentes, il faudrait que l'utilisateur puisse savoir comment fonctionne l'interface de façon plus intuitive (par exemple appuyer sur entrer pour ajouter une préférence, mais appuyer sur ok pour valider une transition.)
- L'ajout de données circonstancielles comme la valeur d'une unité, le cout du prochain paler technologie etc
- Une simulation
- J'ajouterai une case help, notamment pour aider les gens n'ayant jamais utilisé ce genre d'outils.

29. 4(x4)/5(x7)/6(x2)

30.
- un peu désarmée pour assimiler les informations
- SImple à comprendre une fois qu'on comprend les règles, donc facile pour discuter du modèle et le modifier
- très simple, on peut vraiment faire ce que l'on veut quasiment
- L'outil est adapté a des IA simple comme celle proposé ici, mais je pense qu'avec un plus grand projet, le graphe deviendra vite illisible.
- Amusant mais ça aurait été mieux avec le jeu réel pour pouvoir voir le lien entre ce que l'on fait et les conséquences
- L'outil est simple d'utilisation
- Plutôt ludique, un peu complexe à prendre en main les paramètres au début
- Euh, c'était marrant ?
- Bonne impression
- Pas assez complet
- Le modèle est bien adapté, l'outil une fois plus travaillé et documenté est bien adapté aussi
- très flexible, permettant une variété de réactions et plusieurs niveaux de précision (très macro pour une stratégie "à la louche" ou plus détaillé, point par point avec beaucoup de conditions)
- L'outils est très agréable à utiliser, et assez facile à comprendre.

31.
- sa simplicité
- simple à la compréhension, grande possibilité de personnalisation
- Representation graphique beaucoup plus lisible que le code

- Graphe pratique pour comprendre où on en est
- Intuitif, on clique sur la case et on à le détail
- la visualisation sous forme de graphe, et la gestion des transitions sous forme de machine à état
- Souplesse et rapidité de mise en place d'une strategie
- La compréhension facile par n'importe qui
- L'idée d'utiliser les machines à état
- Simple à comprendre avec l'arbre
- simplicité de prise en main, clarté du rendu (on a un aperçu global en un coup d'oeil
- Sa clarté malgré le fait qu'il un peu graphiquement vétuste.

32.
- rapidement fouillé quand il y a trop de lien et d'idée
- personnalisation trop libre au début, esquisser un petit modèle de base, ou cadrer au moins le concept pour permettre de bien comprendre les choses que l'on va pouvoir réaliser
- Impossibilité de dupliquer un objectif qui posera un gros problème de lisibilité pour des IA avec une dizaines d'états logiques.
- Ne pas pouvoir voir le lien entre ce que l'on fait et les conséquences. Peut être aussi que la "programmation" parait un peu compliquée pour quelqu'un qui n'a pas l'habitude de faire ça
- A grande échelle (avec beaucoup de cases) on risque de se perdre un peu avec toutes les flèches.
- points faibles: ergonomie.
- Un peu long à réaliser.
- Le manque d'explications (besoin de poser des questions à l'oral)
- L'outil est buggué, le modèle est bon
- Stabilité de l'appli, modéliser une stratégie plus complexe risque de devenir un peu plus galère
- dès qu'on entre un peu dans la précision, il y a des flèches partout, les labels et les traits s'entrecroisent et on perd en lisibilité
- Peut-être que sa logique pourrait être diffile à comprendre pour les personnes non habituées à la pensée type algorithme (conditions, si, et/ou).

33.
- les menus déroulants pour simplifier l'affichage
- Faire un système avec CTRL pour selectionner tous les nœuds en dessous du nœud selectionné et tout bouger en meme temps (ou pouvoir réduire le sous-graphe à un nœud)

- cadrer la personnalisation en la laissant tout de même libre comme elle est actuellement
- Possibilté de bien separer chaque branche, ce qui est ici impossible puisqu'il est impossible de dupliquer les objectifs
- Un bouton retour en arrière, la possibilité de sélection la flèche et la déplacer à la sourie, la supprimer avec suppr.
- l'ergonomie, homogénéiser, proposer un peu plus d'explications et ... les couleurs. C'est un peu moche quand même, même pour un produit test.
- Affiner et rajouter les objectifs, améliorer l'ergonomie de l'interface
- Ergonomie, documentation,
- la stabilité (:)), la lisibilité, la possibilité de déplacer les labels le long des flèches pour organiser les textes lorsqu'il y a des croisements
- Développer des critères à choisir pour la création des boîtes parallèles pourrait faciliter la compréhension des possibilités.

34.
- oui (x9)
- le graph est sans doute un moyen très simple pour créer une IA, après je ne sais pas si il représente de meilleures perspectives que d'autres principes puisque je ne connais que celui-là
- Bon point pour les game designeurs qui ne programment pas => résultats conformes aux attentes
- Oui, même si j'ai du mal à imaginer comment cela pourra être applicable à tous les types de jeux vidéos
- Il le paraît intéressant de permettre à des non développeurs de développer des IA, afin d'améliorer la compréhension de cette partie du jeu vidéo par tous.

35.
- bien définir la logique de l'IA
- Je ne pense pas qu'il existe déjà un modele pour modeliser l'IA
- Beaucoup plus simple pour le game designer mais aussi pour une personne inexperimenté qui souhaite créer un jeu video sans avoir de connaissances en programmations.
- Les games designer pourront décrire les divers comportement des IA sans passer par un programmer. Cela leurs permettrons de directement tester l'efficacité du comportement.
- calibrer une IA est loin d'être évident, et pousse à de nombreux aller-retours entre dev et designer, cet outil permet de donner aux designer plus d'autonomie, et surtout permet de faciliter la création de comportements.
- J'imagine qu'il peut permettre l'amélioration de l'efficacité dans la communication entre les equipes de dev et de game design et donc d'arriver à un resultat cohérent avec le CdC en utilisant moins de ressources projet.

- Plus accessible et permet un dialogue sûrement plus facile entre game designers et développeurs
- Les API pour développer des IA sont historiquement mauvaise
- Modélise quelque chose de relativement abstrait et complexe difficile à saisir
- le modèle est très ouvert, très flexible et personnalisable (noms des nœuds et des liens, conditions), il peut s'adapter à de nombreux types de jeux (moyennant quelques adaptations par exemple dans les objectifs) et à toutes sortes de stratégie

Annexe E

Evaluation

E.1 Explications fournies

E.1.1 Evaluation et jeu

Le Déroulement

Pendant cette évaluation vous allez devoir créer une intelligence artificielle qui contrôle un joueur de jeu de stratégie. Une consigne sur le type d'intelligence artificielle que vous devez créer vous sera donnée au début de la séance. Ces consignes sont plus ou moins précises, elles peuvent laisser libre l'interprétation, c'est volontaire.
Pour créer les intelligences artificielles nous allons vous proposer deux méthodes permettant d'avoir un contrôle partiel et haut-niveau. Il vous est demandé pour chaque méthode de lire dans un premier temps la feuille de description qui vous sera fournie, et de réaliser les instructions qu'elles contiennent. Dans un deuxième temps il vous est demandé de créer l'intelligence artificielle en essayant de respecter la consigne fournie. Cette deuxième étape doit être chronométrée, vous pouvez utiliser un chronomètre en ligne (http://www.chronometre-en-ligne.com/) ou simplement noter l'heure de début et noter à la fin le nombre de minutes qui vous a été nécessaire. Arrêtez-vous lorsque vous ne pensez pas pouvoir respecter davantage la consigne.
Lorsque vous avez fini l'intelligence artificielle, il vous est demandé de répondre à un court questionnaire sur votre expérience. Une fois que vous avez fini la première méthode et répondu au questionnaire correspondant, il vous sera fourni la fiche descriptive de la deuxième méthode.

Le Jeu

Le jeu créé pour cette évaluation oppose 2 civilisations composées d'un QG, d'ouvriers et de soldats. Chaque civilization possède son QG dès le début de la partie et commence avec un ouvrier.

- Les ouvriers ont la capacité de se déplacer pour aller chercher des ressources dispersés dans l'environnement (de façon aléatoire à chaque début de jeu) et venir les déposer au QG.
- Les soldats ont la capacité de se déplacer et d'attaquer les ouvriers et soldats ennemis de même que le QG ennemi. Ils possèdent une force définie à leur création qui détermine les dégâts provoqués par une attaque.
- Les ouvriers possèdent un niveau de vie initial de 10 qui diminue à chaque attaque sur eux.
- Les QG possède un niveau de technologie initial de 1 qui peut être augmenté en consommant des ressources ou diminué lors d'une attaque. Ce niveau de technologie définit la force des soldats créés. Un QG ne peut pas être détruit et son niveau de technologie ne peut pas descendre en dessous de 1.

Il existe 3 types de ressources dans l'environnement : la nourriture, l'or et le bois. Elles peuvent être utilisées pour créer un ouvrier, un soldat, ou pour augmenter le niveau de technologie. Pour pouvoir effectuer ces actions les ressources doivent être disponibles au QG. Les quantités nécessaires sont les suivantes :

- Création d'un ouvrier : 3 nourriture.
- Création d'un soldat : 2 nourriture et 2 or.
- Augmenter le niveau de technologie (+1) : 3 bois.

Il existe trois manières de gagner :

- Atteindre 30 ouvriers
- Éliminer toutes les unités (soldats et ouvriers) ennemis.
- Atteindre un niveau de technologie de 10.

E.1.2 XML

Ouvrez le fichier “xml”.
Ce dossier comporte 3 documents qui nous intéressent : Game.exe, strategy1.xml, strategy2.xml.

Game.exe vous permettra de lancer le jeu présenté dans “Présentation évaluation.pdf”. Lorsque vous lancez l’exe celui-ci fait tourner le jeu avec 2 intelligences artificielles, ce n’est pas vous qui les contrôlez. Ces intelligences artificielles utilisent le fichiers strategy1.xml pour le joueur 1 et strategy2.xml pour le joueur 2. Vous pouvez lancer le jeu si c’est votre première méthode pour voir comment il marche.

Ouvrez maintenant le fichier “strategy1.xml”. C’est ce fichier que vous allez pouvoir modifier pour créer votre propre intelligence artificielle (merci de ne pas toucher au fichier strategy2.xml). Ce fichier présente 8 paramètres que vous allez pouvoir définir :

- importanceArmy
- importanceTechno
- importanceDemo
- attackTendency
- protectTendency
- preferenceAttackTarget
- preferenceAttackSoldier
- preferenceProtectSoldier
- preferenceProtectTarget

Les 3 premiers paramètres fonctionnent ensemble et sont définis par des réels (pas plus de 2 chiffres après la virgule). Ils représentent les importances relatives des 3 ressources que vous pouvez créer ou modifier : les soldats, les ouvriers et le niveau de technologie. Ces importances influent sur les ressources allouées à chaque actions. ImportanceArmy correspond à la création de soldat, importanceTechno à l’augmentation du niveau de technologie et importanceDemo à la création d’ouvrier. Pour simplifier si la valeur que vous attribuez à importanceArmy est le double de la valeur que vous attribuez à importanceTechno, alors deux fois plus de ressources seront attribuées à la création de soldats qu’à l’augmentation du niveau de technologie.

Le paramètre “attackTendency” est utilisé pour définir lorsque vous attaquez. C’est un float compris entre 0 et 1 (inclus). Il symbolise le rapport de force maximum que l’ennemi peut avoir pour qu’une attaque soit lancée. Par exemple si vous indiquez 0.1 soit 10%, cela signifie que vous n'attaquez que si vous possédez au moins 90% de la puissance militaire totale, soit si votre ennemi possède au plus 10% de la puissance militaire totale. 0 correspond alors à n’attaquez que si l’ennemi ne possède aucune puissance militaire (aucun soldat) et 1 correspond à attaquer dans 100% des cas.
Le paramètre “protectTendency” est utilisé en cas d’attaque pour définir la proportion de soldats qui reste protéger le QG et les ouvriers. C’est également un réel compris entre 0 et 1 (pas plus de 2 chiffres après la virgule).

E.1.3 Modèle et éditeur de stratégie

Ouvrez le dossier “Editeur de stratégie”.
Ce dossier comporte 4 fichiers qui nous intéressent : SimpleSample.exe, game.exe, strategy1.json et strategy2.json.

Game.exe vous permettra de lancer le jeu présenté dans “Présentation évaluation.pdf”. Lorsque vous lancez l’exe celui-ci fait tourner le jeu avec 2 intelligences artificielles, ce n’est pas vous qui les contrôlez. Ces intelligences artificielles utilisent le fichiers strategy1.json pour le joueur 1 et strategy2.json pour le joueur 2. Vous pouvez lancer le jeu si c’est votre première méthode pour voir comment il marche.

Lancer SimpleSample.exe. Ce programme va vous permettre de créer des stratégies sous la forme de fichiers json comme ceux fournis (strategy1.json et strategy2.json). Pour comprendre à quoi ressemble une stratégie cliquez sur “File” puis “Open”. Naviguez ensuite pour pouvoir sélectionner “strategy2.json”. Cliquer sur “Ouvrir”.

Vous devez désormais voir un graphe composé de noeuds jaunes, verts et violets. Les noeuds violets représentent les objectifs évoqués rapidement dans “Présentation évaluation.pdf” (dans la description de l’interface). Il en existe 5 :

- PreparerArmee, qui consiste à développer la création de soldats,
- AmeliorerTechno, qui consiste à augmenter le niveau de technologie du QG,
- Attaquer, qui déclenche une attaque lorsque la puissance de l’armée est supérieure à la puissance de l’armée ennemie,
- DeveloppperDemographie, qui consiste à développer la création d’ouvriers,
- Proteger, qui consiste à développer la protection du QG et des ouvriers par des soldats.

Le rôle de cette stratégie est de définir lesquels de ces objectifs doivent être poursuivis et avec quelles ressources. Pour cela il est mis à votre disposition des “comportements logiques” (les noeuds jaunes) et des comportements parallèles (les noeuds verts). Ceux-ci sont composés de plusieurs sous-comportements définis par les flèches du graphe. Par exemple sur la stratégie proposée, le comportement “comportement agressif” a 2 sous-comportements : l’objectif “PreparerArmee” et l’objectif “Attaquer”. Le comportement parallèle a lui trois sous-comportements : l’objectif “DeveloppperDemographie”, l’objectif “AmeliorerTechno” et le sous-comportement logique “comportement agressif”. Le nombre de sous-comportements n’est pas limité. Un comportement “racine” est défini (entouré en rouge), il ne doit être le sous-comportement d’aucun autre comportement. Ici c’est le comportement “Stratégie générale”.

Double-cliquez sur le noeud jaune que vous voyez. De nouvelles informations apparaissent alors dans la fenêtre Détails. Ici, vous voyez le détail du comportement logique “comportement agressif”. Un comportement logique sert à définir lequel des sous-comportements effectuer en fonction du contexte. Il est basé sur le principe de “machine à états finis” (FSM) qui se présente également sous la forme d’un graphe. Les noeuds du graphe sont appelés “états”, et un seul de ces états est sélectionné à la fois. Ici on a créé 2 états correspondants aux 2 sous-comportements. et on navigue entre les deux états en fonction de paramètres de l’environnement. Ici aussi un noeud est entouré de rouge, c’est l’état initial, le premier qui sera sélectionné. Les conditions pour changer d’état

E.2 Détails des résultats

E.2.1 Questions générales

Questions

1. Quel âge as-tu ?
2. Sais-tu programmer ? (de 0 : Pas du tout à 4 : Très bien)
3. Quelles sont tes compétences en game design ? (de 0 : Je ne connais rien à 4 : Je suis spécialiste)
4. As-tu des compétences en intelligence artificielle ? (oui/non)
5. Si oui, préciser l'expérience (outils, modèles utilisés...)
6. Joues-tu ou as-tu déjà joué à des jeux-vidéo de stratégie ? (de 0 : Jamais à 4 : Très souvent)
7. Si oui, lesquels ?
8. As-tu déjà fait du modding ? (0 : Je ne sais pas ce que c'est, 1 : Je sais ce que c'est mais je n'en ai jamais fait, 2 : J'ai un peu regardé comment ça marchait j'ai vite fait essayé, 3 : Oui je connais bien j'en ai déjà fait plusieurs fois)
9. Qu'attends-tu de l'IA contre laquelle tu joues ?

Réponses

1. 17 / 18 (x3) / 20 (x2) / 21 (x2) / 23 / 25 (x4) / 26 (x2) / 27 (x2) / 40
2. 0 (x2) / 1 (x5) / 2 (x4) / 3 (x5) / 4 (x2)
3. 0 / 1 (x5) / 2 (x5) / 3 (x4) / 4 (x3)
4. 38,9% oui
5.
   - Behaviour tree, path finding, reseaux de neurones, IA jeux echec
   - Design, balancing, warcraft 3
   - Pathfinding, comportement simple.
   - Système expert, représentation de connaissances
   - master en IA
   - Programmation d'une IA sur Unity en C# pour un jeu d'infiltration. Utilisation d'une machine à état tout ce qu'il y ade plus simple, raycasts etc... L'ia patrouille, repère, poursuit, et tire.

- Petite expérience sous Unreal Engine

6. 1 (x2) / 2 (x7) / 3 (x5) / 4 (x4)

7.
   - Warcraft III, Offworld trading company, Age of empire III, Cossack Back to war, Civilization, Galactic Civilizations
   - Civilization, Medieval, XCom
   - Civilisation V, Rome II Total War, Age Of Empire et Age Of Mythology
   - AOM, AOE, Warcraft 3, Romet Total war...
   - Civilization, Banished, Total War, Planetary Annihlation
   - Age of empires, Stronghold
   - age of empire - civilisation
   - Age of Empire, Sins of a solar Empire, Civilisation, , Ogame,...
   - age of empire
   - Battle Realms/ Age of Mythologie/ Civilisation/ la série Total War/ Cartoon War..
   - Starcarft, Company of hearoes, toute la saga Total War, etc...
   - Mega lo mania, C&c: red alert, Company of Heroes, Men of War: AS2, Crusader king 2, civilization 5, RUSE,
   - Dawn of War 1&2, Starcraft 1&2, Warcraft 3
   - Age of Empires. Halo Wars. Dawn of War
   - Warcraft III, Starcraft I
   - Civilisation
   - RTS (Red Alert, Warcraft, Starcraft, supreme commander, age of empire, etc.), 4X (civ, sword of the stars, endless legend et endless space, etc.), grand strategy (europa universalis, heart of iron)
   - Civilisation, Age of Empire, Starcraft, Warcraft III

8. 0(x5) / 1 (x8) / 2 (x4) / 3

9.
   - Qu'elle puisse s'adapter aux situations et s'adapter vis à vis du comportement du joueur.
   - Addaptation
   - Une flexibilité selon mes capacités en tant que joueur.
   - Logique et adaptative (en terme de stratégie et pourquoi pas de difficulté).
   - Qu'elle agisse comme un joueur humain.
   - Qu'elle ne soit pas trop facile à battre ni trop difficile non plus.
   - Rien en particulier...découvrir

- Qu'elle soit capable de me challenger
- Qu'elle apporte de l'intérêt au jeu, et me donne envie de jouer contre elle. Qu'elle ait un comportement différent à chaque partie.
- S'adapter à mon style de jeu. Me battre.
- Qu'elle réagisse à ma stratégie et ne se focus pas que sur son objectif.
- une opposition dosée et "agréable"
- Entrainement, adapté a mon niveau
- Qu'elle semble disposer des même capacités que moi. Qu'elle me laisse gagner mais me rende la tache difficile.
- Jouant essentiellement sur des phases tutorielles tactiques, je tends à apprécier une IA réactive et cohérente mais qui puisse être contrée une fois bien analysée. Je ne souhaite pas particulièrement jouer contre des simili-humains, et donc la part de prévisibilité doit rester présente de manière intelligente.
- Qu'elle s'adapte a ma manière de jouer et me force a changer de stratégie.
- Une grande variété de stratégies, la capacité de s'adapter à ce que je fais pour tenter de me contrer, et une IA qui ne triche pas (bénéficie des mêmes ressources et informations que le joueur).

E.2.2 SUS

E.2.3 Expressivité

Directive 1 : Gagner avec la technologie	**Directive 2 : Créer un comportement économe**	**Directive 3 : S'adapter aux ressources découvertes**
Editeur : 52,5 XML : 50	Editeur : 72,5 XML : 65	Editeur : 60 XML : 80
Editeur : 40 XML : 67,5	Editeur : 75 XML : 92,5	Editeur : 30 XML : 15
Editeur : 72 ,5 XML : 70	Editeur : 65 XML : 87,5	Editeur : 67,5 XML : 80
Editeur : 55 XML : 75	Editeur : 57,5 XML : 92,5	Editeur : 85 XML : 37,5
Editeur : 40 XML : 45	Editeur : 25 XML : 72,5	Editeur : 30 XML : 70
Editeur : 15 XML : 45	Editeur : 72,5 XML : 80	Editeur : 72,5 XML : 85
Moyenne Editeur : 45,83 **Ecart-type Editeur : 19,2787** **Moyenne XML : 58,75** **Ecart-type XML : 13,5785**	**Moyenne Editeur : 61,25** **Ecart-type Editeur : 18,8911** **Moyenne XML : 81,67** **Ecart-type XML : 11,2546**	**Moyenne Editeur : 57,5** **Ecart-type Editeur : 22,8035** **Moyenne XML : 61,25** **Ecart-type XML : 28,4495**
Moyenne Editeur : 54,86 **Ecart-type Editeur : 20,3186** **Moyenne XML : 67,22** **Ecart-type XML : 21,0023**		

Directive 1 : Gagner avec la technologie	**Directive 2 : Créer un comportement économe**	**Directive 3 : S'adapter aux ressources découvertes**
Editeur : 4 XML : 4	Editeur : 4 XML : 2	Editeur : 4 XML : 4
Editeur : 5 XML : 5	Editeur : 4 XML : 3	Editeur : 4 XML : 1
Editeur : 4 XML : 5	Editeur : 4 XML : 5	Editeur : 3 XML : 1
Editeur : 3 XML : 4	Editeur : 3 XML : 5	Editeur : 4 XML : 1
Editeur : 1 XML : 1	Editeur : 1 XML : 3	Editeur : 5 XML : 2
Editeur : 5 XML : 3	Editeur : 2 XML : 4	Editeur : 3 XML : 2
Moyenne Editeur : 3,67 **Ecart-type Editeur : 1,5055** **Moyenne XML : 3,67** **Ecart-type XML : 1,5055**	**Moyenne Editeur : 3** **Ecart-type Editeur : 1,2649** **Moyenne XML : 3,67** **Ecart-type XML : 1,2111**	**Moyenne Editeur : 3,83** **Ecart-type Editeur : 0,7528** **Moyenne XML : 1,83** **Ecart-type XML : 1,169**
Moyenne Editeur : 3,5 **Ecart-type Editeur : 1,2005** **Moyenne XML : 3,0556** **Ecart-type XML : 1,5136**		

Bibliographie

Gustavo Andrade, Geber Ramalho, Hugo Santana, and Vincent Corruble. Extending reinforcement learning to provide dynamic game balancing. In *Proceedings of the Workshop on Reasoning, Representation, and Learning in Computer Games, 19th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 7–12, 2005. 10

Tom Anthony, Daniel Polani, and Chrystopher L. Nehaniv. General self-motivation and strategy identification: Case studies based on Sokoban and Pac-Man. *IEEE Transactions on Computational Intelligence and AI in Games*, 6(1):1–17, 2014. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6687219`. 10

Richard Bartle. Virtual Worlds: Why People Play. In *Massively multiplayer game development 2*, pages 3–18. 2005. xv, 11, 12, 13

Général André Beaufre. *Introduction à la stratégie*. 1964. xv, 22, 23, 24, 28, 29, 31

Maurice Bergsma. *Adaptive Spatial Reasoning for Turn-Based Strategy Games*. PhD thesis, 2008. 38, 40, 48, 50

Edward Booth, John Thangarajah, and Fabio Zambetta. Flexible story generation with Norms and Preferences in computer role playing games. In *2015 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 68–74. IEEE, 2015. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=7317953`. 14

John Brooke. SUS-A quick and dirty usability scale. *Usability evaluation in industry*, 189(194): 4–7, 1996. 100

John Brooke. SUS: a retrospective. *Journal of usability studies*, 8(2):29–40, 2013. URL `http://dl.acm.org/citation.cfm?id=2817913`. 100, 101

Michael Buro. Real-time strategy games: A new AI research challenge. In *IJCAI*, pages 1534–1535, 2003. URL `https://skatgame.net/mburo/ps/RTS-ijcai03.pdf`. 18, 35

Donald C. Hambrick and James W. Fredrickson. Are you sure you have a strategy ? *Academy of Management Executive*, 19(4), 2005. 21, 24, 29

Ellen Earle Chaffee. Three models of strategy. *Academy of management review*, 10(1):89–98, 1985. 31

David Churchill, Mike Preuss, Florian Richoux, Gabriel Synnaeve, Alberto Uriarte, Santiago Ontanón, and Michal Certicky. StarCraft Bots and Competitions. *Encyclopedia of Computer Graphics and Games*, pages 1–18, 2016. URL http://webdocs.cs.ualberta.ca/~cdavid/pdf/ecgg15_chapter-competitions.pdf. 40, 41

Daniel Cook. The Chemistry Of Game Design, 2007. URL http://www.gamasutra.com/view/feature/129948/the_chemistry_of_game_design.php. xv, 8, 9

Anders Dahlbom and Lars Niklasson. Goal-Directed Hierarchical Dynamic Scripting for RTS Games. In *AIIDE*, pages 21–28, 2006. URL http://www.aaai.org/Papers/AIIDE/2006/AIIDE06-008.pdf. 37

Kevin Dill. Prioritizing Actions in a Goal-Based RTS AI. In *AI Game Programming Wisdom 3*, pages 321–330. 2006. 49, 50

Kutluhan Erol, James Hendler, and Dana S. Nau. HTN planning: Complexity and expressivity. In *AAAI*, volume 94, pages 1123–1128, 1994. URL http://www.aaai.org/Papers/AAAI/1994/AAAI94-173.pdf. 39

Richard Evans. Varieties of Learning. In *AI Game Programming Wisdom*. 2002. 15

Général Gil Fiévet. *De la stratégie militaire à la stratégie d'entreprise*. Interéditions, 1992. xv, 23, 24, 26, 28, 29, 30, 31, 112

Charles Andrye Galvao Madeira. *Agents Adaptatifs dans les jeux de stratégie modernes : Une approche fondée sur l'apprentissage par renforcement*. PhD thesis, UPMC, 2007. 37, 38, 111

Laura Mata García. Understanding design thinking, exploration and exploitation: Implications for design strategy. *International Design Business Management*, (2), 2012. 26, 29

Elizabeth Gordon. A Goal-Based, Multitasking Agent Architecture. In *AI Game Programming Wisdom 3*, pages 265–274. 2006. 49, 50, 51, 64, 71

Sir Basil Henry Liddell Hart. *Histoire mondiale de la stratégie*. PLON, 1962. 25, 26, 28, 32

Chris Hecker. Structure vs Style, 2008. 14, 16

Thomas R. Hinrichs and Kenneth D. Forbus. Analogical Learning in a Turn-Based Strategy Game. In *IJCAI*, pages 853–858, 2007. URL http://www.aaai.org/Papers/IJCAI/2007/IJCAI07-137.pdf. 39

Robin Hunicke, Marc LeBlanc, and Robert Zubek. MDA: A formal approach to game design and game research. In *Proceedings of the AAAI Workshop on Challenges in Game AI*, pages 1–5, 2004. xv, 2, 11, 13

Damian Isla. GDC 2005 Proceeding: Handling Complexity in the Halo 2 AI, May 2005. URL http://www.gamasutra.com/view/feature/2250/gdc_2005_proceeding_handling_.php. 4, 13, 14, 15

Damian Isla. Transparent Decision-Making and AI Design, 2008. 4, 15, 55

Ulit Jaidee, Héctor Muñoz-Avila, and David W. Aha. Case-based learning in goal-driven autonomy agents for real-time strategy combat tasks. In *Proceedings of the ICCBR Workshop on Computer Games*, pages 43–52, 2011. URL http://www.nrl.navy.mil/lasr/sites/www.nrl.navy.mil.lasr/files/pdfs/Jaidee-CBL-combat-tasks.pdf. 47

Geraint Johnson. Goal Trees. In *AI Game Programming Wisdom 3*, pages 301–310. 2006. 39, 40, 46, 47, 48, 51

Tom Kent. Designing a Multi-Tiered AI Framework. In *AI Game Programming Wisdom 2*, pages 457–466. 2004. 38

Raph Koster. *A theory of fun*. 2004. 8

Jasper Laagland. A HTN planner for a real-time strategy game. *Unpublished manuscript.(hmi. ewi. utwente. nl/verslagen/capita-selecta/CS-Laagland-Jasper. pdf)*, 2008. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.406.8722&rep=rep1&type=pdf. 39

Raúl Lara-Cabrera, Carlos Cotta, and Antonio J. Fernández-Leiva. A review of computational intelligence in RTS games. In *Foundations of Computational Intelligence (FOCI), 2013 IEEE Symposium on*, pages 114–121. IEEE, 2013. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6602463. 35

Philippe Lorino and Jean-Claude Tarondeau. De la stratégie aux processus stratégiques. *Revue française de gestion*, pages 307–328, 2006. 21, 25

Simon M. Lucas, Michael Mateas, Mike Preuss, Pieter Spronck, and Julian Togelius. Artificial and computational intelligence in games (Dagstuhl Seminar 12191). *Dagstuhl Reports*, 2(5): 43–70, 2012. URL http://drops.dagstuhl.de/opus/volltexte/2012/3651/. 15, 35

Michael Mateas and Andrew Stern. A Behavior Language: Joint action and behavioral idioms. In *Life-Like Characters*, pages 135–161. Springer, 2004. URL http://link.springer.com/chapter/10.1007/978-3-662-08373-4_7. 46

Joshua McCoy and Michael Mateas. An Integrated Agent for Playing Real-Time Strategy Games. In *AAAI*, volume 8, pages 1313–1318, 2008. URL http://www.aaai.org/Papers/AAAI/2008/AAAI08-208.pdf. xv, 41, 42

Manish Mehta, Santiago Ontanon, and Ashwin Ram. Adaptative Computer Games-Easing the authorial burden. In *AI Game Programming Wisdom 4*. 2008. 39, 45, 47

Chris Miles, Juan Quiroz, Ryan Leigh, and Sushil J. Louis. Co-evolving influence map tree based strategy game players. In *Computational Intelligence and Games, 2007. CIG 2007. IEEE Symposium on*, pages 88–95. IEEE, 2007. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4219028. 48

Henry Mintzberg. The structure of unstructured decision processes. *Administrative Science Quarterly*, 21(2):246–275, 1976. 23, 29, 31

Henry Mintzberg. Patterns of strategy formation. *Management Science*, 24(9), 1978. 23, 25, 27, 31, 32

Jeanne Nakamura and Mihaly Csikszentmihalyi. The Concept of Flow. In *Handbook of positive psychology*, pages 89–105. 2002. 1

Jon Lau Nielsen, Benjamin Fedder Jensen, Tobias Mahlmann, Julian Togelius, and Georgios N. Yannakakis. AI for General Strategy Game Playing. In *Handbook of Digital Games*, pages 274–304. John Wiley & Sons, Inc., March 2014. xv, 10

Miguel A. Nieves. Personality Reinforced Search for Mobile Strategy Games. In *Game AI Pro 2*, pages 243–253. 2015. 42, 44, 45

Rodolphe Ocler. *Vers la notion de stratégie proactive : éléments de définition et de mise en œuvre*. PhD thesis, 2002. 26, 28, 30

Santiago Ontañón and Ashwin Ram. Case-Based Reasoning and User-Generated Artificial Intelligence for Real-Time Strategy Games. In *Artificial Intelligence for Computer Games*, pages 103–124. Springer New York, New York, NY, 2011. ISBN 978-1-4419-8187-5 978-1-4419-8188-2. URL http://link.springer.com/10.1007/978-1-4419-8188-2_5. 47, 48

Santiago Ontanón, Gabriel Synnaeve, Alberto Uriarte, Florian Richoux, David Churchill, and Mike Preuss. A Survey of Real-Time Strategy Game AI Research and Competition in StarCraft. *Computational Intelligence and AI in Games, IEEE Transactions on*, 5(4):293–311, 2013. xv, 18, 35, 41

Nils Pettersson. What is a game and why do we play?, 2013. URL http://www.gamasutra.com/blogs/NilsPettersson/20130116/184876/What_is_a_game_and_why_do_we_play.php. 7

Bart Presnell, Ryan Houlette, and Dan Fu. Making Behavior Modeling Accessible to Non-Programmers. 2007. 15

Armelle Prigent, Pascal Estraillier, Vincent Courboulay, and Matthieu Perreira Da Silva. Adaptive Storytelling Based On Model-Checking Approaches. *IJIGS*, 5(2), November 2009. 14

Steve Rabin, editor. *AI Game Programming Wisdom*. Charles River Media, 2002. 35

Steve Rabin, editor. *AI Game Programming Wisdom 2*. Charles River Media, 2004. 35

Steve Rabin, editor. *AI Game Programming Wisdom 3*. Charles River Media, 2006. 35

Steve Rabin, editor. *AI Game Programming Wisdom 4*. Charles River Media, 2008. 35

Steve Rabin, editor. *Game AI Pro*. A K Peters/CRC Press, 2013. 35

Steve Rabin, editor. *Game AI Pro 2*. A K Peters/CRC Press, 2015. 35

Richard Rumelt. Good Strategy/Bad Strategy: the difference and why it matters, 2011. URL https://www.youtube.com/watch?v=UZrTl16hZdk. 29

Manu Sharma, Michael P. Holmes, Juan Carlos Santamaría, Arya Irani, Charles Lee Isbell Jr, and Ashwin Ram. Transfer Learning in Real-Time Strategy Games Using Hybrid CBR/RL. In *IJCAI*, volume 7, pages 1041–1046, 2007. URL http://www.aaai.org/Papers/IJCAI/2007/IJCAI07-168.pdf. 38

Thiago A. Souza, Geber Lisboa Ramalho, and Sergio R. M. Queiroz. Resource Management in Complex Environments: Applying to Real Time Strategy Games. pages 21–30. IEEE, 2014. ISBN 978-1-4799-8065-9. doi: 10.1109/SBGAMES.2014.27. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7000029. 49, 50, 51, 63, 111

Marius Stanescu, Nicolas A. Barriga, and Michael Buro. Hierarchical adversarial search applied to real-time strategy games. In *Tenth Annual AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, 2014. URL https://skatgame.net/mburo/ps/HierarchicalSearch-AIIDE-2014.pdf. 48

David Thue, Vadim Bulitko, Marcia Spetch, and Eric Wasylishen. Interactive Storytelling: A Player Modelling Approach. In *AIIDE*, pages 43–48, 2007. URL http://www.aaai.org/Papers/AIIDE/2007/AIIDE07-008.pdf. 111

Paul Tozour. Stack-Based Finite-State Machines. In *AI Game Programming Wisdom 2*, pages 303–306. 2004. 110

Mathieu Triclot. *Philosophie des jeux vidéo*. 2011. 7, 8

Sun Tzu. *L'art de la guerre*. 30

Suhas Virmani, Yatin Kanetkar, Manish Mehta, Santiago Ontanon, and Ashwin Ram. An intelligent IDE for behavior authoring in real-time strategy games. 2008. 39, 45

Ben George Weber, Michael Mateas, and Arnav Jhala. Applying Goal-Driven Autonomy to StarCraft. In *AIIDE*, 2010. URL http://alumni.soe.ucsc.edu/~bweber/pubs/gda_aiide2010.pdf. 46

Jay Young and Nick Hawes. Evolutionary Learning of Goal Priorities in a Real-Time Strategy Game. In *AIIDE*, 2012. URL https://pdfs.semanticscholar.org/37eb/9cb895dad1cad128fd5c53f0396761a54343.pdf. 42, 44, 45