UNIVERSITÉ DU QUÉBEC À MONTRÉAL

DÉVELOPPEMENT D'UN SYSTÈME TUTORIEL INTELLIGENT POUR L'APPRENTISSAGE DU RAISONNEMENT LOGIQUE

MÉMOIRE

PRÉSENTÉ

COMME EXIGENCE PARTIELLE

DE LA MAÎTRISE EN INFORMATIQUE

PAR

ANGE ADRIENNE NYAMEN TATO

FEVRIER 2016

REMERCIEMENTS

Je tiens à exprimer mes sincères remerciements à toutes les personnes qui ont contribué, de près ou de loin au bon déroulement de mon projet de maîtrise.

Je présente une profonde gratitude au professeur Roger NKAMBOU, mon directeur de recherche, pour son implication, son soutien, sa disponibilité et surtout ses précieux conseils. Je le remercie également pour son orientation et pour la confiance qu'il m'a accordée.

Je remercie toute l'équipe du projet Muse-logique, dont le professeur Serge Robert, Dr Clauvice Kenfack, Janie, Pamela, Christophe et Carolane pour leur soutien et leur contribution énorme dans ce projet.

J'adresse mes plus sincères remerciements à mes parents Mr Tato Eulalie et Mme Mbeunke Solange, mes frères et sœurs Ariane, Joyce et Adam Tato, à ma famille qui de loin a toujours su m'encourager et me soutenir dans mes travaux.

Enfin je tiens à remercier ma cousine Doris Sadou et sa famille pour leur soutien inestimable depuis mon arrivée au Québec.

TABLE DES MATIERES

LISTE DES TABLEAUX

LISTE DES FIGURES

RÉSUMÉ

Le raisonnement est un processus cognitif nous permettant de tirer à partir de règles, des conclusions sur des faits et situations de la vie. Il est considéré dans la littérature comme étant une partie intégrante de plusieurs autres processus humains comme la perception (résultat d'une combinaison inductive entre les senseurs et la mémoire), la catégorisation, la compréhension, la prise de décision, la résolution de problèmes. L'humain a tendance à effectuer des raisonnements erronés sans s'en rendre compte. Les erreurs de raisonnement seraient des inférences créatives dans un contexte où il aurait été approprié de faire une inférence déductive. L'acquisition de la compétence en raisonnement consiste alors à apprendre à être déductif et ainsi à acquérir des structures plus organisées pour mieux systématiser notre information et pour devenir cognitivement plus performants.

L'avènement des technologies de l'information allié à l'évolution du domaine de l'Intelligence Artificielle, a permis le développement de systèmes tel que les systèmes tutoriels intelligents (STI). Ces systèmes sont caractérisés par le fait qu'ils permettent d'automatiser l'enseignement et de favoriser l'apprentissage sans l'intervention d'un tuteur humain. Ce projet vise le développement d'un STI générique appelé Muse-logique et dédié à l'apprentissage du raisonnement logique.

Dans ce document, nous présentons en premier lieu la problématique et les objectifs de notre projet. Une étude détaillée des domaines en jeu est faite par la suite. Enfin, l'architecture, les éléments conceptuels et les résultats d'implémentation du système, sont exposés ainsi qu'une validation préliminaire du système. Le contenu et l'élaboration des différents composants du STI on fait l'objet d'un travail minutieux effectué avec la participation active des membres d'une équipe multidisciplinaire. Le composant expert est soutenu par un ensemble de règles du domaine. Le modèle cognitif de l'apprenant est soutenu par un réseau bayésien et enfin, le modèle pédagogique est soutenu par des règles tutorielles étudiées et validées théoriquement. La particularité et la robustesse de Muse-logique résident dans son cadre de développement composé d'experts en sciences cognitives, en systèmes tutoriels intelligent et en raisonnement logique. Des perspectives futures sont envisagées pour les prochaines versions.

Mots clés : raisonnement logique, systèmes tutoriels intelligents, logique, système adaptatif, réseau bayésien, feedbacks, représentation des connaissances.

CHAPITRE I

INTRODUCTION

Ce travail est réalisé dans le cadre d'un projet de recherche au sein des laboratoires GDAC et LANCI de l'Université du Québec À Montréal avec une équipe pluridisciplinaire, dont des experts logiciens, cogniticiens, psychologues du raisonnement et dans le domaine des systèmes tutoriels intelligents. Ce projet vise à construire un système tutoriel intelligent pour l'apprentissage du raisonnement logique. Le but est de pouvoir aider les humains à accroître leur capacité cognitive, leur rigueur intellectuelle, leur aptitude à faire des abstractions et leur sens critique.

1.1. Problématique

Considérons une situation dans laquelle nous sommes devant une maison et le propriétaire Mr X nous affirme que si l'on lance une pierre sur l'une des fenêtres de sa maison, cette dernière se brisera. Une semaine plus tard, Mr X nous appelle pour nous dire que des enfants se sont amusés dans sa cour et ont lancé par accident une pierre sur l'une de ses fenêtres. Instinctivement, nous conclurons que cette fenêtre s'est brisée. Cette faculté de pouvoir conclure est appelée raisonnement; selon Peter A. (Angeles, 1981), le raisonnement est un processus d'inférence permettant de tirer des conclusions à partir de faits, d'observations et ou de déclarations. Considérant la même situation ci-dessus et supposons maintenant qu'une autre semaine après, Mr X nous contacte pour nous dire qu'il vient de constater que l'une de ses fenêtres est brisée. De manière

instinctive nous conclurons fort probablement que des pierres ont été lancées, ce qui est un raisonnement erroné car d'autres événements peuvent avoir causé cet incident. Cette conclusion erronée est ce que l'on appelle en logique un *sophisme*. Nous avons augmenté le poids de la première partie de l'information (si on lance une pierre sur une fenêtre) que nous avait fournie Mr X. Il existe d'autres types de raisonnement erronés que nous verrons en détail plus tard dans ce mémoire. Des erreurs de raisonnement peuvent menés à des actions dangereuses ou non prudentes. Imaginons le cas où il s'agissait plutôt d'une personne qui soit entrée par effraction dans la maison, Mr X ne s'étant pas rendu compte déduit que c'est sûrement une pierre qui a été lancée par une personne alors qu'un malfrat se trouve actuellement dans la maison avec lui. Il devient donc intéressant de se pencher sur l'amélioration de notre façon de raisonner.

Selon (Rips, 2002), le raisonnement est une partie intégrante de plusieurs autres processus humains comme la perception (résultat d'une combinaison inductive entre les senseurs et la mémoire), la catégorisation, la compréhension, la prise de décision, la résolution de problèmes. Afin d'éviter de tirer des conclusions erronées, être de meilleurs raisonneurs, il est important d'apprendre les règles de raisonnement valides et invalides qui découlent de la logique. Savoir raisonner logiquement est donc une connaissance cruciale qui s'applique à la vie de tous les jours et nécessaire à la survie et à l'estime de soi. Or ce raisonnement logique qui peut sembler facile à acquérir de prime à bord, est un processus cognitif non trivial et difficile à acquérir. En effet, de nombreuses expériences en sciences cognitives montrent que des erreurs systématiques sont courantes dans l'usage du raisonnement logique chez l'humain (Evans, Newstead, & Byrne, 1993; Noveck, Van der Henst, Rossi, & Mercier, 2007). Un certain nombre de questions se posent sur la manière d'améliorer les compétences de l'humain dans ce domaine : Quels sont les éléments impliqués dans l'apprentissage des compétences en raisonnement logique ? Quelles sont les stratégies pour favoriser le développement de ces compétences ? Est-il possible d'envisager un cadre automatique pour accompagner les humains dans cette tâche ? Quels sont les moyens que nous disposons à cet effet ?

Les systèmes tutoriels intelligents (STI) peuvent-ils aider ? Est-ce qu'un tel système peut soutenir le raisonnement logique et son apprentissage ? Quelles seraient alors les caractéristiques d'un tel STI ? Sera-t-il extensible à d'autres types de raisonnement et de logique ? Les réponses à ces questions font partie intégrante de ce travail

1.2. Hypothèse de solution

D'un point de vue scientifique, pourquoi vouloir construire un système capable de soutenir l'apprentissage du raisonnement logique ? D'une part, cette idée se fonde sur les constats suivants :

- une partie importante de la connaissance humaine relève du traitement de l'information par des raisonnements (Henry Markovits, 2014);
- le raisonnement logique occupe une place importante dans nos mécanismes de raisonnement (Evans, 2002);
- le raisonnement logique humain est plus souvent non classique que classique (Robert, 2005);
- en tant que machine cognitive en situation de lutte pour sa survie, l'humain tend à faire des erreurs systématiques dans ses raisonnements logiques (Aleven, 2010a; Evans, 2002; Evans et al., 1993);
- ces erreurs étant dues à l'emploi de stratégies de découverte (inductives, analogiques, adductives) dans un contexte de justification (Robert, 2005);
- apprendre à raisonner logiquement, c'est apprendre comment fonctionne le traitement cognitif de l'information, tout en apprenant les lois et procédures valides du raisonnement logique, de manière inséparable (Robert, 2005);
- les mécanismes cognitifs humains sont le produit de modules cérébraux relativement autonomes mais également en interaction étroite entre eux.

Par ailleurs, les STI ont fait avancer la compréhension de l'apprentissage en général (Vanlehn, 2006; Woolf, 2010), mais ont aussi fait leurs preuves quant à leur capacité de soutenir l'apprentissage du raisonnement et des habiletés de résolution de problèmes en sciences, mathématiques, programmation, et même en logique (Aleven, 2010a; Lesta & Yacef, 2002; Tchetagni, Nkambou, & Bourdeau, 2007). Ils offrent plusieurs services dont le suivi du raisonnement de l'apprenant, la détection et le diagnostic des erreurs de ce dernier et même des démarches remédiatives à ces erreurs. Cependant, même s'il existe quelques STI pour l'apprentissage de la logique, ils se réduisent en des banques d'exercices sans une base de connaissances explicite et sans un modèle de la compétence logique (Croy, Barnes, & Stamper, 2008). L'analyse des systèmes existants nous permet de formuler les deux limites suivantes :

- ils ne se fondent pas sur une élicitation des connaissances et structures logiques essentiels au raisonnement, limitant ainsi leur portée explicative du raisonnement de l'apprenant ;
- ils ne se fondent pas sur des théories fondamentales de l'erreur de raisonnement (Stanovich, 2009).

Nous nous proposons donc de développer un système tutoriel intelligent pour l'apprentissage du raisonnement logique qui, en plus d'offrir les différents services cités plus haut, lèvera les deux limites observées. Le système doit être capable de diagnostiquer, accompagner l'apprenant efficacement dans son apprentissage du raisonnement, donner des feedbacks constructifs, identifier les erreurs que font les apprenants et les corriger.

1.3. Objectifs

Ce travail de maîtrise vise à construire un premier prototype de MUSE-Logique, un système tutoriel capable de raisonner et pouvant aider au développement de la

compétence en raisonnement logique. Pour y arriver, plusieurs sous-objectifs doivent être atteints :

a) Élaborer une architecture du système basé sur l'architecture classique d'un STI (Nkambou, 2010a).

b) Éliciter et implémenter les structures du raisonnement notamment les règles d'inférences valides et invalides (représentation de la connaissance du domaine (Alonso, Aranda, & Martn–Mateos, 2007)).

c) Intégrer des services fondamentaux, notamment le mécanisme de diagnostic cognitif et les structures de connaissances sous-jacentes. A ce niveau, la nécessité d'intégrer un catalogue des erreurs et d'utiliser un modèle probabiliste (réseau Bayesien (Pardos & Heffernan, 2010)) mettant en liens les éléments de connaissances et de performances associés au raisonnement, a été établi au sein de l'équipe Muse. Nos services de diagnostic cognitif seront construits sur ces éléments.

d) Élaborer une stratégie de support métacognitif durant l'apprentissage afin d'encourager l'autocorrection et la réflexion. A ce niveau, nos premières hypothèses font entrevoir l'usage des méta-structures (ex : treillis de Boole pour la logique propositionnelle) de la logique pour le traçage et la visualisation du raisonnement. L'apprenant pourra à travers ces méta-structures observé l'évolution de son propre raisonnement. On parle alors d'ouverture du modèle apprenant, une pratique reconnue comme favorisant la métacognition (Bull, 2004; Bull & Kay, 2010).

e) Implémenter des règles tutorielles construites à partir des stratégies de tutoring qui ont prouvées leurs efficacités (Narciss, 2008) et sont largement utilisées dans des environnements d'apprentissage.

f) Implémenter et évaluer MUSE-Logique pour le raisonnement déductif et la logique propositionnelle dans un premier temps, puis par la suite intégrer les autres

formes de raisonnement logique (inductif, abductif etc…) ainsi que la logique des prédicats et la logique flou. Il est donc important de s'assurer que les choix conceptuels et d'implémentation constituant le cadre (framework) de MUSE-Logique tiennent compte de son caractère évolutif c'est-à-d que ces choix faciliteront l'extension à d'autres types de raisonnement et de logique.

L'intelligence artificielle est au cœur de ce projet car le système visé devra être avant tout, un système expert logicien capable de produire des raisonnements automatiques valides et de les expliquer, en tenant du contexte. Ce système de par sa généricité, pourra être étendu à d'autres logiques et servira également de support au développement d'autres types de raisonnement logique.

1.4. Méthodologie

Muse-Logique est un système à base de connaissances dont le développement met en jeu au moins deux ingénieries : l'ingénierie des connaissances et l'ingénierie du logiciel. Nous avons eu la chance d'avoir accès à l'expertise et la connaissance nécessaire pour l'acquisition et l'encodage des connaissances du domaine enjeu de l'apprentissage, soit la logique et le raisonnement logique. Le processus d'ingénierie des connaissances est passé par une phase d'élicitation des connaissances suivi par des choix d'outils efficaces pour l'encodage. Bien évidemment, l'ingénierie logicielle n'est pas négligée car la gestion du projet, le design, la conception et le développement du STI ont reposé sur une approche pratico pratique du génie-logiciel. Ainsi, la méthodologie adoptée pour le développement de Muse-Logique (en tant que logiciel) est la méthode agile (http://www.agilemanifesto.org/). Il a été prouvé que des projets réalisés en suivant cette méthode sont généralement des projets réalisés dans un environnement de grande qualité et aboutissent presque toujours à des résultats de grande qualité (Qumer & Henderson-Sellers, 2008). Sa particularité est que,

premièrement elle est axée principalement sur le facteur humain qui est l'agent principal du projet et deuxièmement elle offre une gestion de temps et des livrables flexibles afin de prendre en compte les nouveaux besoins au fur et à mesure. Cette dernière particularité est effectivement ce que nous cherchons dans le projet car ce projet étant réalisé dans un cadre de recherche, est caractérisé par de nouveaux besoins pouvant surgir à tout moment et qui ont tendance à être modifiés au fil du temps.

A cet effet, dans le développement de Muse, nous procédons par étapes, c'est-à-dire fixons des objectifs à court terme et commençons le développement sans perdre de temps. Chaque fois que l'objectif fixé est atteint, nous passons au prochain sachant que l'on pourra y revenir et ainsi de suite jusqu'à ce que le but ultime soit atteint. En plus de la méthode agile, nous avons adopté une approche participative, car ce projet implique toutes les parties prenantes nécessaires pour déboucher sur un système valide et efficace tant du point de vue de son contenu que de sa pédagogie. Muse-Logique est donc développé avec la participation conjointe des informaticiens spécialistes des EIAH (Environnements informatiques pour l'Apprentissage Humain) ainsi que des experts du domaine de la logique et de la psychologie du raisonnement. L'ensemble des composants développés sont à cet effet validés et argumentés par ces experts tout au long du processus.

1.5. Organisation du mémoire

Ce mémoire rend compte du développement de Muse-Logique, un système tutoriel intelligent pour l'apprentissage du raisonnement logique. Il comporte sept chapitres dont le premier est l'introduction. Le chapitre II présente la cadre théorique du mémoire à savoir, la logique, le raisonnement logique et son apprentissage. Dans ce chapitre, nous examinons de plus près le raisonnement logique en tant que concept fondamental dans notre étude. Le chapitre III présente le concept de STI, son intérêt,

le pourquoi de son choix mais également un état de l'art des systèmes tutoriels d'apprentissage du raisonnement de la logique existants en s'attardant sur leurs points faibles. Après cet état de l'art, nous présentons dans le chapitre IV les éléments d'architecture et de conception de notre nouveau système, comment les différents modèles communiquent, les techniques qui permettent au système de pouvoir raisonner, les éléments qui font de notre système un système générique. Le chapitre V expose les stratégies de développement de Muse-logique. Dans ce chapitre une attention particulière est portée sur l'implémentation à proprement dite, les techniques utilisées pour permettre au système de raisonner, l'encodage des connaissances en logique, les résultats et des scénarios de fonctionnement. Dans le chapitre VI nous présentons les résultats des tests et évaluations interne effectués sur le prototype et de Muse. Enfin, le chapitre VII conclut le mémoire par un résumé des contributions ainsi qu'un exposé des perspectives et des travaux futurs que nous visons pour le projet.

CHAPITRE II

CADRE THEORIQUE : LOGIQUE ET RAISONNEMENT LOGIQUE

L'objectif de ce travail est de développer un système générique (dans le sens où il pourra être étendu selon le besoin à toutes formes de logique) capable d'apprendre la logique, de raisonner, de faire la différence entre une inférence valide et invalide. D'autre part, ce système doit aussi être capable de faire apprendre aux humains à raisonner, de diagnostiquer et corriger leurs erreurs. Cet objectif ne peut être atteint sans qu'il ne soit nécessaire d'éliciter et comprendre les connaissances du domaine visé. Nous devons chercher à savoir quels sont les éléments qui entrent en jeu pour pouvoir instruire un tel système dans l'apprentissage de la logique. Le but de ce chapitre est donc d'exposer le cadre théorique de notre projet : La logique et le raisonnement logique. Voici les questions auxquelles nous tenterons de répondre : Qu'est-ce que le raisonnement ? Qu'est-ce que le raisonnement logique ? Comment apprendre à raisonner et quels en sont les défis? Quels sont les mécanismes pour faciliter l'apprentissage du raisonnement ?

Le présent chapitre contient des informations qui nous servirons de bagages nécessaires pour envisager la mise en place d'un système capable de soutenir l'apprentissage du raisonnement logique. Nous serons également muni à la fin de ce chapitre, de notions qui nous aiderons par la suite pour l'encodage du raisonnement et des connaissances reliées; cet encodage étant un élément-clé pour l'automatisation de l'apprentissage et l'émulation du raisonnement logique dans un système informatique.

2.1. Définitions

Jusqu'ici nous avons parlé de logique, de raisonnement logique, de son importance dans la vie de l'humain, de règles pour raisonner sans toutefois s'y attarder avec attention. Cette section a pour objet de compléter avec détails, les quelques éléments fournis depuis le début de ce document concernant ces termes.

2.1.1 Qu'est-ce que le raisonnement ?

Le raisonnement est fondamentalement un processus cognitif visant à tirer une information (conclusion) à partir des informations données (prémisses) par l'application d'une règle (Evans et al., 1993) . Comme mentionnée à l'introduction de ce mémoire, le raisonnement est une partie intégrante de plusieurs autres processus humains comme la perception (résultat d'une combinaison inductive entre les senseurs et la mémoire), la catégorisation, la compréhension, la prise de décision, la résolution de problèmes. Le cerveau humain acquiert de l'information de manière assez directe, par la perception. Il en acquiert de manière plus indirecte par le raisonnement. C'est donc un processus incontournable dans la vie de l'humain. Il existe différents raisonnements dont le raisonnement logique qui nous intéresse. Quel est le rôle de la logique dans le raisonnement ?

2.1.2. Qu'est-ce que la logique ?

La logique est à l'origine la recherche de règles générales et formelles permettant de distinguer un raisonnement concluant de celui qui ne l'est pas . Selon le dictionnaire français Larousse, la logique est la « science du raisonnement en lui-même, abstraction faite de la matière à laquelle il s'applique et de tout processus psychologique ». Nous

avons déjà des éléments de réponse à la question qui était de savoir : Quel est le rôle de la logique dans le raisonnement ? Il est naturel d'affirmer qu'elle sert de support au raisonnement logique. En effet, c'est est un langage formel pour faire des inférences déductives valides (Robert, 2013). Le mécanisme d'élaboration d'inférence est appelé raisonnement. Il existe différentes logiques dont la logique classique (logique des propositions et logique des prédicats) et la logique non classique (logique modale, logique multivalente, logique floue, etc...). Une logique classique est vérifonctionnelle, assertorique, extensionnelle, bivalente alors qu'une logique non classique n'a pas toutes les propriétés précédentes. Dans la suite de ce mémoire, une attention particulière sera portée sur la logique des propositions. Cela dit, nous n'excluons pas pour autant les autres, car le système a été conçu de manière à pouvoir être étendu à d'autres logiques. Ces affirmations pourront se confirmer dans le chapitre IV portant sur les éléments conceptuels du STI Muse-logique.

2.1.3. Qu'est-ce que le raisonnement logique ?

La définition du raisonnement selon (Evans et al., 1993) nous dit clairement qu'un raisonnement est fait à partir d'une règle. La nature de la règle utilisée dans le raisonnement définit la nature de ce raisonnement. Par exemple si la règle utilisée est logique, alors on parle de raisonnement logique. Le raisonnement logique est important dans la vie quotidienne pour notre autodéfense intellectuelle. Il est important dans notre activité cognitive parce qu'une grande partie de notre cognition est inférentielle. Il est également important en informatique car les ordinateurs sont des machines inférentielles (des si...alors règles) où la logique joue un grand rôle.

Le raisonnement logique (ou déductif) est non ampliatif (les informations contenues dans la conclusion ne découlent que des informations données), monotone (la valeur de la conclusion ne peut être modifiée même si de l'information a été ajouté

dans les prémisses), certain (il y a toujours une conclusion possible lorsque la règle est vraie, il existe toujours une conclusion valide). Ce type de raisonnement nous sert à organiser l'information en systèmes. Il est contraire au type de raisonnement créatif qui lui est ampliatif, non monotone, incertain et nous sert à générer de nouvelles informations (Robert, 2005). Les raisonnements créatifs sont des sophismes, mais ils sont nécessaires pour connaître. Il existe plusieurs formes de raisonnement qui sont (Varin, 2007):

- Le raisonnement abductif : C'est la forme la plus complexe du raisonnement. Il permet, face à un certain effet dont nous possédons la règle, de parvenir à la cause. Selon (Elayne, 2009), il est un procès réversible où les résultats sont toujours impliqués aux causes et vice versa. Le sujet cherche les causes des situations.
- Le raisonnement inductif : Encore appelé raisonnement créatif, il consiste à créer dans la conclusion de l'information nouvelle qui n'était pas présente dans les prémisses. Selon (Elayne, 2009), cette forme de raisonnement se produit à partir des observations du concret. A travers l'induction, le raisonneur peut arriver à la généralisation. Le sujet généralise une idée à partir des observations effectuées.
- Le raisonnement déductif ou logique : Ce dernier consiste à appliquer une règle générale à des cas particuliers. Selon (Elayne, 2009), elle implique de déduire la connaissance à partir des connaissances antérieures. Le sujet déduit les nouvelles connaissances à partir des connaissances déjà acquises. C'est un processus de fiabilité totale car, l'individu utilise l'abstraction, la recombinaison et la connaissance.

2.2. Les règles du raisonnement logique.

Après avoir défini le raisonnement logique, nous allons maintenant nous intéresser aux règles (valides et invalides) qui régissent ce raisonnement. La première version de notre système porte sur le raisonnement déductif. Cependant, nous envisageons dans nos futurs travaux, une extension aux autres formes de raisonnement.

Une élicitation des connaissances procédurales nécessaires au raisonnement logique a été faite au sein de l'équipe. Ce processus a abouti à la spécification de toutes les règles d'inférence logiques en logique propositionnelle dont les règles valides et non valides. Chaque règle représente une inférence. L'inférence peut être immédiate, c'est-à-dire constituée d'une seule prémisse qui mène à la conclusion, ou médiate (syllogisme), composée de deux ou plusieurs prémisses dont découle la conclusion. Un syllogisme est un raisonnement (inférence) à deux prémisses dont une prémisse majeure, qui est constitué de deux propositions (appelé antécédent et conséquent et lié par un connecteur), d'une prémisse mineure, qui affirme ou nie l'une de ces deux propositions et d'une conclusion. La conclusion est tirée, affirmant ou niant l'autre proposition impliquée dans la prémisse majeure (différente de la mineure). Rappelons qu'il est possible de tirer une conclusion à partir d'un syllogisme parce qu'il existe des relations entre les prémisses (Varin, 2007).

Exemple de syllogisme :

Majeure : Si on lance une roche dans une fenêtre, alors elle brisera.
Information : On a lancé une roche dans une fenêtre.
Conclusion : La fenêtre s'est brisée.

La prémisse majeure est identifiée par la majeure dont l'antécédent est « on lance une roche dans une fenêtre », le conséquent est « la fenêtre brisera » et le connecteur est « Si alors » qui représente 'L'implication'. La prémisse mineure est l'information. La

conclusion a été obtenue après un raisonnement sur la majeure et la mineure. On peut constater que les informations contenues dans la conclusion se trouvent en partie dans la majeure. Cette remarque nous amène à définir de façon plus concrète le raisonnement déductif comme étant une réorganisation des informations que nous avons sous les yeux. Notons que nous partons d'un raisonnement dont les prémisses sont supposées vraies pour déduire la véracité de la conclusion. Un raisonnement correct doit aboutir à une conclusion vraie puisque les prémisses sont toujours vraies pour un syllogisme correct. Chaque syllogisme présenté dans ce mémoire et contenu dans la base de données de Muse-Logique a fait l'objet d'une validation minutieuse au sein de l'équipe des logiciens avec qui nous travaillons. Un syllogisme doit respecter un certain nombre de règles que nous pouvons trouvés en détail dans (André, 2005).

Il existe des syllogismes catégoriques (syllogisme Aristoliens) (J. Barnes, 1969) et non catégoriques (syllogismes Stoïciens). Parmi les syllogismes non-catégoriques, on trouve les syllogismes hypothétiques lesquels feront l'objet du présent mémoire (voir Figure II.1). Il existe 3 formes de syllogisme hypothétique qui sont différenciés uniquement par le connecteur qui se trouve dans la majeure :

- Le syllogisme implicatif encore appelé conjonctif ou conditionnel est représenté par le connecteur implication « → ». Ce connecteur est détecté par l'expression « Si…alors… »
- Le syllogisme disjonctif est représenté par le connecteur ou « V ». Ce connecteur est détecté par l'expression « Ou » ou « Ou bien »
- Le syllogisme d'incompatibilité ou conjonctif est représenté par le connecteur « | ». Cette forme de syllogisme est très peu étudiée dans la littérature.

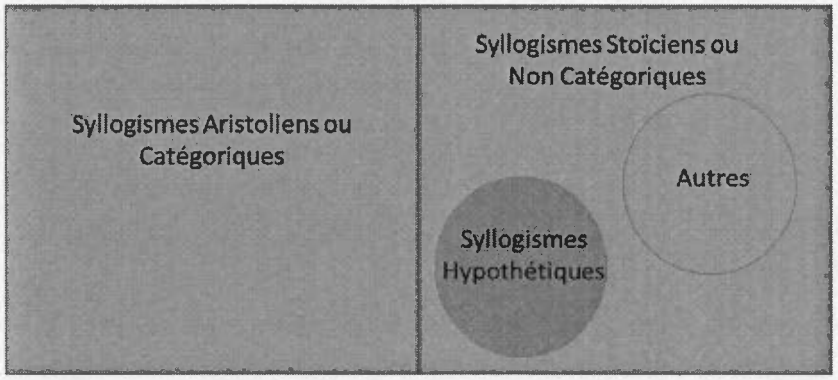


Figure II.1 : Types de syllogismes

Chaque syllogisme hypothétique peut s'écrire sous la forme d'une expression propositionnelle constituée de propositions et de connecteurs les liants. Prenons l'exemple du syllogisme pris en exemple plus haut :

antécédent = « on lance une roche dans une fenêtre » = proposition P,

conséquent = « elle brisera » = proposition Q,

connecteur = « si…alors' = implication,

information = « On a lancé une roche dans une fenêtre » = proposition P,

conclusion = « La fenêtre s'est brisée » = proposition Q.

La forme propositionnelle de ce syllogisme s'écrira $\mathbf{((P \rightarrow Q)\ \&\ P) \rightarrow Q}$ qui se lit « Si P implique Q et on a P donc on a Q » où « & » veut dire « et ». Il devient important de s'attarder sur les différents connecteurs logiques car ils nous permettrons de déterminer la valeur de vérité d'un syllogisme.

2.2.1. Connecteurs logiques

De ce que nous avons présenté ci-dessus, nous pouvons déjà dénombrer 4 connecteurs dont le Ou « V », l'implication « → », l'incompatibilité « | » et le et « & » sur lesquels nous rajoutons un opérateur unaire qui est la négation « ¬ ». Chacun de ces opérateurs est présenté avec sa table de vérité, si besoin est. Les tables de vérité sont une technique de décision pour déterminer si une expression propositionnelle bien formée est (Anellis, 2004) :

- une loi logique (une tautologie, toutes les valeurs de la table finale sont vraies),
- une contradiction (toutes les valeurs de la table finale sont fausses),
- une formule factuellement vraie ou factuellement fausse (au moins une valeur vraie et au moins une valeur fausse dans la table finale).

2.2.1.1. La négation «¬ »

La négation est un connecteur qui s'applique à une seule proposition et sert à la nier. Par exemple, considérons la proposition p = « On lance une roche dans une fenêtre »; la négation de cette proposition sera « On n'a pas lancé de roche dans une fenêtre » et sa forme propositionnelle « ¬p ». Nier une proposition c'est inversé sa valeur de vérité (vrai devient faux et vice-versa), en d'autres termes, si la valeur de vérité de p est égale à vrai, alors celle de ¬p est faux. La dernière colonne de la table de vérité de cet opérateur, représente la valeur de vérité de l'expression en fonction des valeurs de vérité de la proposition en jeu.

2.2.1.2. Le Et « & »

Le connecteur « & » est le plus simple à comprendre. Il correspond à la conjonction de coordination « Et » en français. Il s'applique à deux propositions et non une seule comme dans le cas de la négation. Par exemple, considérons les propositions p = « l'argent ne fait pas le bonheur » et q = « seul l'amour peut faire le bonheur »; la forme propositionnelle « p & q » veut dire « l'argent ne fait pas le bonheur et seul l'amour peut faire le bonheur ». Quand est ce que cette expression est vraie? En d'autres termes : Quelle est la valeur de vérité de cette forme ? La réponse à cette question se trouve dans la table de vérité de l'opérateur. La dernière colonne représente la valeur de vérité de « p & q ». Nous déduisons de cette table que, la valeur de vérité de l'expression est vraie uniquement lorsque les deux propositions qui la composent sont également vraies. Donc, dire que « p & q » est vraie c'est dire que p et q sont vrais en même temps.

2.2.1.3. Le Ou « V »

Le connecteur « V » correspond au « Ou » de la langue française. Il s'applique également à deux propositions. Reprenons l'exemple de tout à l'heure où p = « l'argent ne fait pas le bonheur » et q = « seul l'amour peut faire le bonheur »; la forme propositionnelle « p V q » veut dire « l'argent ne fait pas le bonheur ou seul l'amour peut faire le bonheur ». Quand est ce que cette expression est vraie ? Sans passé par la table de vérité, il est clair qu'elle est vraie si au moins l'une des deux propositions impliquées est vraie, et faux dans le cas contraire.

Cependant il existe un autre connecteur appelé « ou exclusif » que l'on représentera par « W » qui est semblable au « Ou inclusif » que nous venons de voir, à la seule différence que, lorsque les deux propositions sont vraies, l'expression est

fausse. Soit l'exemple suivant : « Demain, j'irais au cinéma ou au à la ronde », si l'on considère un contexte où je n'ai pas assez d'argent pour faire les deux activités en même temps, alors les deux propositions ne pourront être vraies en même temps. Dans le cas où elles se réaliseraient toutes les deux (sont vraies), alors l'expression sera fausse. Ce cas illustre un « Ou exclusif ». Si l'on considère un contexte où j'ai assez d'argent pour les deux activités, alors nous tombons dans le cas du « Ou inclusif ».

2.2.1.4. L'implication « → »

Le connecteur « → » qui est « l'implication », est un connecteur conditionnel et se détecte par la présence d'un « Si…alors ». Il est un peu plus compliqué que ceux qui ont été vu plus tôt car il n'est pas commutatif. Par exemple, si nous considérons p = « l'argent ne fait pas le bonheur » et q = « seul l'amour peut faire le bonheur », les deux expressions « l'argent ne fait pas le bonheur et seul l'amour peut faire le bonheur » et «Seul l'amour peut faire le bonheur et l'argent ne fait pas le bonheur» veulent dire la même chose. Par contre les expressions « Si l'argent ne fait pas le bonheur alors seul l'amour peut faire le bonheur » et « Si seul l'amour peut faire le bonheur alors l'argent ne fait pas le bonheur » n'ont pas le même sens. Quand est-il de la valeur de vérité de l'expression « p → q»? La valeur de vérité de l'expression est vraie si les deux propositions impliquées se produisent, ce qui est tout à fait normale. En effet, lorsque l'on a une proposition p qui implique la réalisation de l'autre (p → q) cela ne veut pas dire que q ne peut pas se réaliser indépendamment de p. Exemple : Si je lance une roche dans une fenêtre alors la fenêtre brisera. Dans cet exemple, si la fenêtre se brise peu importe si l'on a lancé une roche ou pas, l'implication est vérifiée. Par contre, si l'on lance une roche dans une fenêtre (p vraie) et que cette fenêtre ne brise pas (q faux) alors cette implication devient fausse.

2.2.1.5. L'incompatibilité « | »

L'incompatibilité qui se note « | » est un connecteur qui s'applique à deux propositions. On dit que deux propositions sont incompatibles si les deux ne peuvent pas être vraies en même temps et on note « p | q ». Il est nécessaire ici de présenté la table de vérité du connecteur :

Tableau II.1 : Table de vérité de l'incompatibilité

p	q	p | q
V	V	F
V	F	V
F	V	V
F	F	V

2.2.2. Syllogismes valides

Nous avons jusqu'ici, présentée une liste non exhaustive de connecteurs, mais suffisante pour comprendre la suite. Ces connecteurs nous permettent de déterminer quel est le type de syllogisme hypothétique en jeu. Ils servent également à construire des propositions plus complexes à partir de propositions plus simples.

Un raisonnement logique est fait à partir d'une règle logique. Cependant, ce raisonnement peut être valide (raisonner à partir d'une règle valide) ou erroné (raisonner à partir d'une règle fausse). C'est pourquoi il est important de pouvoir ainsi distinguer les syllogismes valides de ceux qui ne le sont pas. Nous présentons dans ce qui suit ces différentes règles (syllogismes). Pour chaque syllogisme hypothétique, il y'a deux inférences valides et deux erronés.

- **Syllogismes implicatifs valides**

Les deux inférences valides sur l'implication sont appelées MPP (Modus Ponendo Ponens qui signifie 'en posant de je pose') et MTT (Modus Tollendo Tollens qui signifie 'en niant je nie') (Varin, 2007). Ponere (P), c'est affirmer et tollere (T), c'est nier. Dans le MPP, la mineure (information) affirme l'antécédent et la conclusion valide est l'affirmation du conséquent. Concrètement, dans le cas où $p \rightarrow q$, l'affirmation de p m'autorise à conclure que q. Cette interprétation correspond à la première ligne de la table de vérité du connecteur « implication ». Formellement le MPP s'énonce comme suit : « Si p alors q. Or p donc q » et sa forme propositionnelle s'écrit « $((p \rightarrow q) \& p) \rightarrow q$ ». Exemple : S'il pleut, alors il y a des nuages. Or il pleut, donc il y a des nuages. Dans le MTT, la mineure nie le conséquent et la conclusion est la négation de l'antécédent. Concrètement, dans le cas où $p \rightarrow q$, la négation de q m'autorise et me contraint à conclure en niant p. Cette interprétation correspond aux deux dernières lignes de la table de vérité de l'implication. Formellement, le MTT s'énonce comme suit : « Si p alors q. Or non q donc non p » et sa forme propositionnelle s'écrit « $((p \rightarrow q) \& \neg q) \rightarrow \neg p$ ». Exemple : S'il pleut, alors il y a des nuages. Or, il n'y a pas de nuages, donc il ne pleut pas.

- **Syllogismes disjonctifs valides**

Les deux inférences valides sur le disjonctif sont tous les deux appelés MTP (Modus Tollendo Ponens). Dans le MTP, la mineure nie un des termes de la majeure (antécédent ou conséquent) et la conclusion affirme l'autre. Concrètement, dans le cas où p V q, on a deux situations valides :

- La négation de p m'autorise à conclure que q. Formellement il s'énonce comme suit: « p ou q. Or non p donc q » et sa forme propositionnelle s'écrit « $((p \text{ V } q) \& \neg p) \rightarrow q$ ».

- La négation de q m'autorise à conclure que p. Formellement il s'énonce comme suit : « p ou q. Or non q donc p » et sa forme propositionnelle s'écrit « ((p V q) & ¬q) →p ».

Le MTP est valide tant avec le « ou » qu'avec le « ou bien…ou bien ».

- **Syllogismes d'incompatibilités valides**

Les deux inférences valides sur l'incompatibilité sont tous les deux appelés MPT (Modus Ponendo Tollens). Dans le MPT, la mineure affirme l'un des termes de la majeure (antécédent ou conséquent) et la conclusion nie l'autre. Concrètement, dans le cas où p | q, on a deux situations valides:

- L'affirmation de p m'autorise à conclure que non q. Formellement il s'énonce comme suit : « p incompatible avec q. Or p donc non q » et sa forme propositionnelle s'écrit « ((p | q) & p) →¬q ».
- L'affirmation de q m'autorise à conclure que non p. Formellement il s'énonce comme suit : « p incompatible avec q. Or q donc non p » et sa forme propositionnelle s'écrit « ((p | q) & q) →¬p ».

2.2.3. Syllogismes erronées

Les syllogismes erronés sont encore appelés sophismes. Les sophismes sont des inférences non valides, des erreurs logiques. Selon le dictionnaire Larousse, un sophisme est un argument qui, partant de prémisses vraies, ou jugées telles, aboutit à une conclusion absurde et difficile à réfuter. Chaque syllogisme hypothétique possède deux inférences invalides.

- **Syllogismes implicatifs erronés**

Les deux sophismes d'implication sont le AC (affirmation du conséquent) et le DA (négation de l'antécédent). Dans le AC, la mineure affirme le conséquent et la conclusion est l'affirmation de l'antécédent. Concrètement, dans le cas où $p \rightarrow q$, l'affirmation de q ne m'autorise en aucun cas de conclure que p or c'est cette situation qui se produit dans un AC. Formellement le AC s'énonce comme suit : « Si p alors q. Or q donc p» et sa forme propositionnelle s'écrit « $((p \rightarrow q) \& q) \rightarrow p$ ». Exemple : S'il pleut, alors il y a des nuages. Or il y a des nuages, donc il pleut, ce qui nous le rappelons est un raisonnement erroné.

Dans le cas du DA, la mineure nie l'antécédent et la conclusion est la négation du conséquent. Concrètement, dans le cas où $p \rightarrow q$, la négation de p ne m'autorise en aucun cas de conclure que q n'a pas lieu or c'est cette situation qui se produit dans un DA. Formellement le DA s'énonce comme suit : « Si p alors q. Or non p donc non q» et sa forme propositionnelle s'écrit « $((p \rightarrow q) \& \neg p) \rightarrow \neg q$ ». Exemple : S'il pleut, alors il y a des nuages. Or il ne pleut pas, donc il n'y a pas de nuages.

L'erreur que nous commettons dans ces situations est que, nous considérons l'antécédent comme seul antécédent possible, donc comme cause unique. Ainsi, l'implication est traité comme une équivalence et donc le AC et le DA sont considérés comme valides. On ne tient pas compte des antécédents alternatifs, des causes multiples. Les antécédents alternatifs sont des contre-exemples aux conclusions des sophismes, ce qui inhibe ces sophismes (Henry Markovits, Brunet, Thompson, & Brisson, 2013). Nous augmentons indûment l'information de la prémisse majeure et allégeons notre mémoire de travail. La correction passe par la sensibilisation aux antécédents alternatifs (Robert, 2014).

- **Syllogismes disjonctifs erronés**

Les deux sophismes sur le disjonctif sont tous les deux appelés MPT (Modus Tollendo Ponens). Dans le MPT, la mineure affirme un des termes de la majeure

(antécédent ou conséquent), et la conclusion nie l'autre ce qui bien sûr est faux. Concrètement, dans le cas où on a comme majeure p V q, on a deux situations invalides:

- L'information est l'affirmation de p et la conclusion est non q. Formellement il s'énonce comme suit : « p ou q. Or p donc non q » et sa forme propositionnelle s'écrit « ((p V q) & p) →¬q ». Exemple : Je prends un café ou un biscuit. Or, je prends un café, donc je ne prends pas de biscuit.
- L'information est l'affirmation de q et la conclusion est non p. Formellement il s'énonce comme suit : « p ou q. Or q donc non p » et sa forme propositionnelle s'écrit « ((p V q) & q) →¬p ». Exemple : Je prends un café ou un biscuit. Or, je prends un biscuit, donc je ne prends pas de café.

Le MPT est valide seulement avec le « ou bien…ou bien ». Nous utilisons les « Ou » pour construire des catégories (d'objets, d'événements…). Nous avons tendance à catégoriser avec des « Ou exclusif », plutôt qu'avec des « Ou » ainsi le MPT est considéré comme valide. La correction passe par la sensibilisation aux possibilités de catégories hiérarchiques: des objets qui seraient à la fois des A et des B (des chiens et des mammifères) (Robert, 2014).

- **Syllogisme d'incompatibilités erronées**

Les deux sophismes sur l'incompatibilité sont tous les deux appelés MTP (Modus Tollendo Ponens). Dans le MTP, la mineure nie l'un des termes de la majeure (antécédent ou conséquent) et la conclusion affirme l'autre ce qui est bien sûr faux. Concrètement, dans le cas où p | q, on a deux situations invalides qui peuvent se produire:

- La négation de p m'autorise à conclure que q. Formellement il s'énonce comme suit : « p incompatible avec q. Or non p donc q » et sa forme propositionnelle s'écrit « ((p | q) & ¬p) →q ».

- La négation de q m'autorise à conclure que p. Formellement il s'énonce comme suit : « p incompatible avec q. Or non q donc p » et sa forme propositionnelle s'écrit « $((p \mid q) \& \neg q) \rightarrow p$ ».

Nous utilisons les incompatibilités pour construire des catégories (d'objets, d'événements…). Nous avons tendance à catégoriser avec des « Ou exclusif », plutôt qu'avec des incompatibilités et ainsi le MTP est considéré comme valide. De même que le sophisme disjonctif, la correction passe par la sensibilisation aux possibilités de catégories hiérarchiques (Robert, 2014).

Les sophismes ne sont pas les seules erreurs observables lors d'un raisonnement logique. Une deuxième catégorie d'erreurs est constituée de suppressions d'inférence valide. Ce type d'erreur survient lorsque le raisonneur ne fait pas une inférence logiquement valide, en considérant qu'elle ne l'est pas. En d'autres termes, il refuse de tirer une conclusion dans des situations où l'on devrait pourtant le faire. La prise en compte de ce type d'erreur est actuellement considérée dans le projet MUSE-Logique mais non encore implémentée dans le prototype actuel. Techniquement, il s'agira d'étendre le catalogue des erreurs pour y inclure les suppressions d'inférence, soit par un encodage explicite des règles de raisonnement sous-jacentes s'il y a lieu, soit par un simple marquage du catalogue d'erreur dans le modèle de l'apprenant pour indiquer la présence de ces erreurs (c'est une déviation de l'application d'une règle d'inférence valide). Par ailleurs, comme pour les sophismes, des actions tutorielles appropriées devront être élaborées pour prendre en charge ses erreurs en termes de stratégies rémédiatives. Dans le cadre de ce mémoire, nous nous concentrerons uniquement sur les sophismes.

2.3. Apprentissage du raisonnement logique

Le raisonnement n'est pas un processus absolu. De nombreuses études ont montré que des personnes en situation de raisonnement tirent des conclusions différentes à des inférences formellement identiques qui ne diffèrent que par le contenu des prémisses (Cummins, Lubart, Alksnis, & Rist, 1991; Thompson, 1995). Par exemple, donner une réponse logique pour un syllogisme d'implication invalide (AC ou DA) dans un contexte donné ne signifie pas que la même réponse sera donnée dans un contexte différent. Pour apprendre à raisonner correctement, il faut apprendre les règles valides et pouvoir corriger des raisonnements erronés. L'inhibition des sophismes et donc la correction de nos erreurs se fait via les contre-exemples à la conclusion.

2.3.1. Inhibition des sophismes

Dans le cas des sophismes d'implication, les stratégies d'inhibition sont (Henry Markovits et al., 2013) :

- utiliser un contenu qui contient beaucoup d'antécédents alternatifs,
- donner une tâche de production de contre-exemples (préalablement aux tâches de raisonnement) afin de stimuler la sensibilité aux contre-exemples.

Dans le cas du syllogisme disjonctif, une stratégie d'amélioration du raisonnement (par production de contre-exemples avec P&Q) n'existe pas dans la littérature de même que pour le syllogisme d'incompatibilité (Robert, 2014).

Afin de manipuler le contenu (contexte) des syllogismes, nos experts ont identifié quatre principales classes de catégorie de contenu, dont les difficultés augmentent compte tenu de la nature du contenu: Concret, Contrefactuel, Abstrait informel et Abstrait formel. Les contenus concret et contrefactuel ont chacun été affinés à deux

sous-catégories. Pour ce qui est de la classe Concret on a le Concret avec peu d'alternatives CFA (*Croncrete with Few Alternatives*) et le Concret avec beaucoup d'alternatives CMA (*Concrete with Many Alternatives*). Pour ce qui est de la classe Contrefactuel on a le Contrefactuel avec peu d'alternatives CFFA (*Contrary-to-fact with Few Alternatives*) et le Contrefactuel avec beaucoup d'alternatives CFMA (*Contrary-to-fact with Many Alternatives*). Le Tableau II.2 présente les détails de ces contextes. Par exemple, la prémisse majeure du syllogisme d'implication présenté plus haut qui était « Si on lance une roche dans une fenêtre alors la fenêtre brisera», appartient à la catégorie CMA car il y'a plusieurs objets alternatifs qui s'offrent à nous si nous désirons briser une fenêtre par exemple : un bâton, une balle de baseball etc… Cette construction de catégorie de contextes est faite sur la base de résultats empiriques en psychologie (H. Markovits, 2013; Henry Markovits, 2014).

Tableau II.2 : Classes de catégories de contenus

<table>
<tr><td colspan="3">DESCRIPTIVE</td><td>NORMATIVE</td></tr>
<tr><td colspan="2">Causal</td><td rowspan="4">Abtract informal (AI)

Abstract formal (AF)</td><td rowspan="4"></td></tr>
<tr><td>Concret</td><td>Contrary-to-fact</td></tr>
<tr><td>Few alternatives (CFA)</td><td>Few alternatives (CFFA)</td></tr>
<tr><td>Many alternatives (CMA)</td><td>Many alternatives (CFMA)</td></tr>
</table>

Ces classes présentées dans le ci-dessus nous fournissent non seulement un moyen de classifier les compétences en raisonnement mais aussi nous permettra d'organiser les syllogismes afin de mieux cerner s'il y'a lieu, les causes des erreurs de raisonnement. Elles serviront aussi comme cadre pour l'organisation des exercices de raisonnement (items).

2.3.2. Apprentissage des règles de raisonnement

Parce que nous nous sommes concentrés sur trois connecteurs (disjonction, implication et incompatibilité) ayant chacune quatre règles d'inférence, Muse-logique pour la logique propositionnelle est faite de 3x4x6 (72) compétences en raisonnement. Il y'a donc 36 règles d'inférences valides et 36 règles d'inférences invalides (ces dernières constituant le Catalogue des erreurs). Toutes ces règles devront être apprises par (encoder dans) le système pour lui permettre de raisonner logiquement tout en sachant faire la différence entre des raisonnements valides de ceux qui ne le sont pas.

La connaissance des règles d'inférences, leur compréhension et leur utilisation constituent des mécanismes permettant de faciliter l'apprentissage du raisonnement logique.

Le présent chapitre a fait état du cadre théorique portant sur le raisonnement logique. Nous avons vu que le raisonnement, bien qu'il soit une tâche effectuée de façon innée chez l'homme, est régi de formules. Ces formules étant des règles qui peuvent être valides (inférences valides) ou erronés (sophisme et suppression d'inférence). La logique joue un très grand rôle dans le processus de raisonnement logique. En effet, nous avons vu que les syllogismes qui se présentent sous forme de propositions reliées entre elles par des connecteurs, peuvent être traduites en expressions logiques, ce qui rend leur évaluation plus facile. La valeur de vérité d'une expression logique se détermine facilement grâce à la table de vérité qui en découle. Chaque syllogisme hypothétique (associé à chacun des 3 connecteurs : Et, Ou, implicatif) possède 2 règles d'inférence valides et 2 autres invalides. Raisonner logiquement c'est apprendre ces règles d'inférence et savoir les mettre en pratique; c'est aussi éviter de faire des sophismes en les inhibant. Une façon d'inhiber des sophismes dans le cas de l'implication, c'est premièrement pouvoir distinguer la catégorie de contenu en jeu (être sensibilisé aux alternatives) et deuxièmement pouvoir produire des contre-exemples. Avec la distinction des catégories de contenu dans

Muse-logique, les règles d'inférence valides dans le cas de la logique des propositions sont de 36 de même que les règles d'inférence invalides.

Nous avons présenté deux approches pour notre système : une approche computationnelle qui fait des prédictions sur des sophismes d'incompatibilité, contribue à générer des stratégies d'inhibition; une approche algorithmique qui intègre les contextes de génération des sophismes, intègre la psychologie cognitive des raisonneurs. Ces deux approches contribuent à générer des stratégies d'inhibition et peuvent être appliquées aux logiques non classiques. La combinaison de ces deux approches permet de construire un système à la fois performant dans l'apprentissage du raisonnement logique et dans l'enseignement de ce raisonnement.

Le soutien à l'apprentissage du raisonnement logique par un système passe par une assimilation des informations présentées dans ce chapitre. Une machine qui se veut capable de raisonner doit être à mesure de connaître, de comprendre et d'utiliser les règles. L'automatisation de l'apprentissage et l'émulation du raisonnement logique passe par un encodage de ce raisonnement. L'encodage consistera en la mise en place d'un moteur d'inférences et un mécanisme de détection d'erreurs de raisonnement. Un STI possède-t-il des caractéristiques lui permettant de supporter le raisonnement logique ?

CHAPITRE III

ETAT DE L'ART SUR LES SYSTEMES TUTORIELS INTELLIGENTS

Nous avons vu ce qu'est le raisonnement logique, et son importance dans la vie humaine. Nous avons également présenté les différentes règles qui définissent un bon ou un mauvais raisonnement. Dans le présent chapitre, nous traitons du support technologique à l'apprentissage de ce raisonnement logique. Ainsi, ce chapitre introduit la notion de système tutoriel intelligent (STI). Il est important de comprendre le fonctionnement et connaitre les capacités d'un tel système. Nous présenterons dans un premier temps un aperçu du domaine des STI; par la suite, nous présenterons les différents STI en logique qui existent en nous attardant sur les problèmes auxquels ils font face, et sur des solutions potentielles pour pallier à ces problèmes. Il sera aussi question de mettre en évidence les points non couverts de ces systèmes et comment une solution pourrait être envisagée. Les STI peuvent être vus comme une amélioration des environnements d'apprentissage assistés par ordinateur; nous énoncerons un bref historique de ces technologies.

3.1. Les systèmes tutoriels intelligents

Dans cette section, l'accent est mis sur les systèmes tutoriels intelligents en général. D'où et comment est venue l'idée d'un tel système ? De quoi se compose-t-il et quel en est l'utilité ?

3.1.1. Historique

Les années 1940 et 1950 voient l'apparition des premiers véritables ordinateurs. Mais ces machines ne servaient qu'à faire des calculs massifs. Par ailleurs, il est évident que la référence par excellence pour accomplir des tâches telles que l'enseignement ou l'apprentissage reste la personne humaine, à travers notamment ses capacités de jugement en situation et de réaction face à l'imprévu (Rialle, 1996). Cependant, la ressource humaine est épuisable et pas toujours disponible. Quel était alors le meilleur moyen de rendre cette ressource accessible et inépuisable ? Reproduire voire dépasser, l'intelligence humaine dans toutes ses dimensions en était une hypothèse. D'où la nécessité de doter les machines d'une intelligence quasiment comparable à celle de l'humain. Alors que les scientifiques s'interrogent sur la possibilité de construire des machines pensantes, Turing propose, un test pour déterminer l'intelligence d'une machine. On ne peut qu'admirer l'ingéniosité du dispositif qui, en s'éloignant des préjugés et en évitant l'écueil d'avoir à définir dans l'absolu ce qu'est la pensée ou l'intelligence, propose pour ce faire "le jeu de l'imitation" : un interrogateur humain dialoguant avec deux entités (sans les voir) doit déterminer laquelle est l'humain et laquelle est la machine. S'il se trompe plus souvent que quand il a à distinguer, dans les mêmes circonstances, une femme d'un homme, alors la machine franchit le test (Marquis, Papini, & Prade, 2009).

Il n'est pas question ici du développement des programmes capables de tromper l'interrogateur, le test de Turing veut d'abord que la machine rivalise par ses performances avec l'humain. Mais il évoque aussi le fait qu'une machine à qui l'on veut conférer une intelligence dispose d'une capacité importante qui est l'apprentissage. Puisque l'enseignement et l'apprentissage relèvent de l'intelligence, il fallait de ce fait mettre au point des systèmes capables de remplir ces tâches. Pour mettre au point de tels systèmes une question s'imposait : Comment faisons-nous ? C'est à cette question que tentent de répondre les sciences de la cognition.

Les sciences cognitives ont pour objet de décrire, d'expliquer et le cas échéant de simuler voire d'amplifier les principales dispositions et capacités de l'esprit humain à savoir le langage, raisonnement, perception, coordination motrice, planification, décision, émotion, conscience, culture (Andler, 2005). Dans ce sens, l'approche cognitive a influencé le développement des logiciels en particulier ceux qui sont destinés à l'apprentissage (tant machine que humaine), en offrant des modèles pour l'organisation et l'utilisation des connaissances humaines et en soulignant les facteurs qui favorisent ou diminuent l'efficacité de celles-ci.

Il y a quelques années donc, les sciences cognitives étaient jusque-là méconnues et les logiciels d'apprentissage humain se comportaient comme une succession d'informations, questions, réponses, commentaires prévus et implémentés par avance par les concepteurs du système, et n'ayant pas la capacité d'apprendre soi-même, de détecter et comprendre les erreurs et les incompréhensions d'un apprenant. Ces derniers n'étaient pas capables de s'adapter à un apprenant ou une situation spécifique (Ochs, 2004) implémentant ainsi une approche dite behavioriste dans laquelle tout ce qui importe est la réponse finale de l'apprenant, c'est-à-dire la performance observée. Par conséquent, tout apprenant commettant une erreur dans une activité d'apprentissage recevait la même rétroaction préétablie du système (Fournier, 2010). La venue des systèmes tutoriels intelligents (STI) au début des années 70 avait pour but de pallier à ces limites.

3.1.2. La notion de STI

Selon Wenger (Wenger, 2014) et Ohlsson (Ohlsson, 1987) un système tutoriel intelligent est un système d'enseignement basé sur ordinateur, avec des modèles de pédagogies qui précisent ce qu'il faut enseigner et des stratégies qui spécifient, comment les enseigner. Les STI sont des logiciels de formation (qui peuvent être en

ligne ou pas) dotés d'une intelligence artificielle leur permettant de s'adapter aux apprenants et de gérer une interaction complexe avec ces derniers dans des situations d'apprentissage. Ces systèmes ont vu le jour au début des années 70 et ayant pour principal objectif de simuler l'enseignant dans ses capacités d'expert pédagogue et d'expert du domaine. Ils sont une amélioration des environnements d'apprentissage par ordinateur (années 60-70) dans ce sens que, il intègre la dimension accompagnement (expert pédagogue) et la dimension adaptation qui jusque-là n'étaient pas des objectifs à atteindre en soi. Plusieurs STI ont vu le jour et ont fait leurs preuves, notamment GUIDON (Clancey, 1992) (pour le diagnostic des maladies infectieuses), RomanTutor (Belghith, Nkambou, Kabanza, & Hartman, 2012) (pour l'apprentissage de la manipulation d'un bras robot), Andes (Vanlehn, Lynch, Schulze, Shapiro, & Shelby, 2005) (pour l'apprentissage de la physique), etc.

3.1.3. Les composants d'un STI

Traditionnellement, les STI sont dotés de connaissances du domaine, de connaissances pédagogiques et d'un modèle de l'apprenant mis à jour dynamiquement (Andler, 2005; Clancey, 1992). Bien qu'ayant pris différentes formes selon les visées de chaque développeur, on retrouve de façon minimale dans l'architecture d'un STI, les quatre composantes suivantes (voir également Figure III.1):un modèle apprenant, un modèle expert, un e modèle tuteur et une interface.

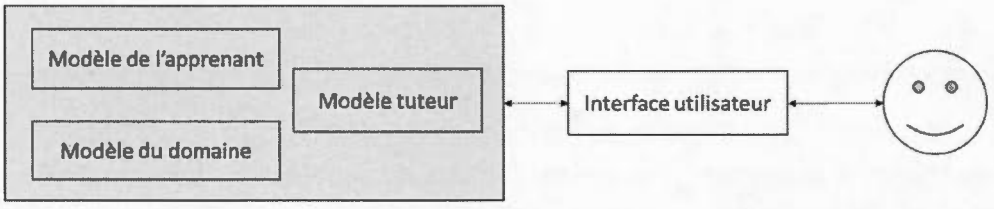


Figure III.1 : Architecture d'un système tutoriel intelligent

3.1.3.1. Modèle Tuteur

Cette composante contient les stratégies pédagogiques (nous les appellerons également règles tutorielles) du système. Comme un tuteur humain, il sélectionne les tâches à exécuter par l'apprenant en fonction des besoins de ce dernier. Il planifie les activités à présenter à l'apprenant, lui fournit les explications appropriées, détermine quand et comment intervenir. Il est capable d'adapter son intervention en fonction du sujet. Il aide, accompagne, guide l'apprenant dans sa tâche d'apprentissage. Le tuteur s'appuie sur des approches éducatives appropriées pour prendre ses décisions.

Lorsque l'apprenant effectue une action (par exemple répondre à une question), l'interface transmet l'action au modèle tuteur. Le tuteur fait appel au modèle du domaine pour avoir la réponse (connaissances en jeux dans la question). En fonction de ce que l'apprenant a donné comme réponse et la réponse de l'expert, le tuteur met à jour le modèle de l'apprenant et donne un feedback approprié (explication, réponse juste, etc.). Les tactiques d'interventions et stratégies du tuteur dépendent de l'état du modèle de l'apprenant.

Il existe différentes stratégies pédagogiques parmi lesquelles l'entraînement (Coaching) qui permet d'offrir à l'apprenant des conseils et le guider (pas à pas) lorsqu'il s'éloigne de la solution; ou encore l'approche socratique à l'apprentissage (ou à l'enseignement) qui permet une articulation sur les éléments de connaissances du domaine pour amener à identifier les règles de niveau supérieur et les concepts et l'apprentissage par la pratique (Nkambou, 2010b). Les stratégies d'intervention (feedbacks ou rétroactions) du tuteur sont contenues dans les stratégies pédagogiques. Elles ne dépendent pas entièrement du domaine étudié, raison pour laquelle il existe des stratégies d'intervention de base (dites indépendantes du domaine), que tout tuteur devrait respecter afin d'améliorer le gain d'apprentissage chez les apprenants (Narciss, 2008). Comme exemple, le tuteur ne devrait pas donner de feedbacks négatifs comme « Faux », « Non, ce n'est pas correct », car il est généralement admis que ce type de

feedback peut entraîner de la frustration chez l'apprenant. Il est donc important de se pencher sur ce point lors de la conception des règles tutorielles.

3.1.3.2. Modèle expert ou Modèle du domaine

C'est dans cette composante que sont codées les connaissances du domaine ainsi que les mécanismes liés à leur exploitation. Il consiste en une base de connaissances que le système tente d'enseigner à l'apprenant. La constitution (élicitation, acquisition et encodage) du module expert représente souvent 50% de l'effort de développement d'un STI (Nkambou, 2010b). Les connaissances du domaine ne sont pas stockées de manière brute dans ce composant sinon on ne l'appellerait pas « expert ». L'expert doit être capable de les manipuler, les expliquer voir de les démontrer. Ces connaissances étant souvent généralisées, le système doit être capable d'y raisonner et d'en tirer des conclusions valides qui n'y sont pas explicitement encodées. Le domaine de connaissance devra être encodé et exploitable par la machine, ceci afin d'assurer une manipulation flexible des données durant le processus d'enseignement et d'apprentissage d'où une nécessité de bonne représentation. Plusieurs formalismes sont proposés dans le domaine de l'intelligence artificielle pour organiser la base de connaissances dans le module d'expertise. Les réseaux sémantiques, les systèmes de production de règles, les réseaux connexionnistes neuronaux, les représentations procédurales et la construction de scripts ou de frames en sont des exemples (Paquette, 2011). Ces méthodes de représentation amènent à déduire 3 possibilités de modélisation de l'expert du domaine (Nkambou, 2010b) :

- approche «boîte noire» : appliquer une quelconque méthode de raisonnement sur le domaine. On dispose des entrées (données, problèmes), des sorties (réponse correcte, réponse de l'apprenant) mais pas d'explications sur le processus qui a mené aux sorties. Il n'y a aucun moyen d'accéder à la structure

de raisonnement de l'expert. Par exemple, un système boîte noire ne serait pas capable de justifier pourquoi il attribue une réponse positive ou négative à une question qui lui est posée.

- système expert ou de type boîte de verre: capable d'expliciter ses raisonnements.
- modèle cognitif: simuler la façon dont l'humain utilise les connaissances et résout les problèmes. Cette modélisation s'appuie sur les théories de la cognition. Les connaissances sont décomposées et organiser de manière à être communicable par le tuteur.

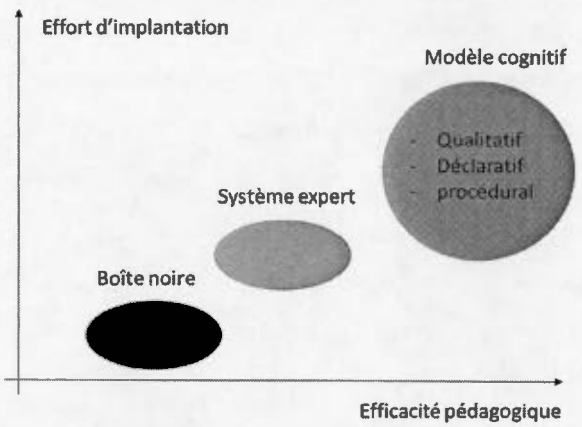


Figure III.2 : Comparatif des méthodes de modélisation du composant expert

La figure 3.2 présente un comparatif de différentes modélisations exposées. Le modèle cognitif bien que demandant en effort d'implémentation, assure une efficacité pédagogique plus importante. C'est également le modèle qui se rapproche le plus de l'intelligence humaine. Bien que plusieurs STI se situent dans la deuxième catégorie (à l'instar de GUIDON), une tendance récente fonde le développement du modèle expert sur des architecture cognitive (c-à-d l'implémentation d'une théorie de la cognition humaine) comme ACT-R. De telle architecture sont alors utilisées comme support à l'encodage du domaine d'expertise. Les tuteurs issus de cette tendance sont dits 'cognitifs' (Conati, Gertner, & Vanlehn, 2002).

3.1.3.3. Modèle Apprenant

Nous aimerions pouvoir dire que le modèle de l'apprenant, c'est l'apprenant lui-même. Cependant, ce n'est pas le cas, mais il est construit par le système pour représenter à un instant donné, les états courants (cognitifs, affectifs, psychologiques, émotionnels, etc...) de l'apprenant. Ce modèle est consulté périodiquement par le tuteur et l'expert pour déterminer le focus de la formation (Nkambou, 2010b). Il est généralement constitué de plusieurs sous-modèles : affectif, inférentiel et cognitif. Le modèle inférentiel permet au système de faire une inférence sur la compréhension (état des connaissances) de l'apprenant. Cette inférence ou encore diagnostic, a pour but de déterminer les causes des erreurs de l'apprenant. Il existe deux principales approches pour le diagnostic de l'état de l'apprenant. La première approche est le « *Model tracing* » de Anderson (Anderson, 1983) qui propose de créer et d'analyser la trace des activités de l'apprenant. Cette approche nécessite une bonne modélisation du processus de résolution de problèmes. La deuxième approche dite «*Knowledge Tracing* » propose d'analyser un fragment (épisode) d'apprentissage afin d'identifier les connaissances qui ont été utilisées. Le modèle cognitif est souvent de type expertise partielle (overlay) c'est-à-dire, les connaissances de l'apprenant sont considérées comme formant un sous-ensemble de celles de l'expert.

La plupart des systèmes tutoriels intelligents représentent l'état des connaissances de l'apprenant comme un sous-ensemble des connaissances de l'expert. Ainsi, le modèle de l'apprenant est construit en comparant la performance de l'apprenant à celle de l'expert, ceci pour une même tâche ou un même problème (Paquette, 2011). C'est cette technique qui a été nommée : modèle par recouvrements (overlay model) par Carr et Goldstein. Il existe deux variantes de ce modèle. Une variante simple (Figure III.3) dans laquelle, les connaissances de l'apprenant sont entièrement incluses dans celles de l'expert, et une variante avec erreurs (Figure III.4)

dans laquelle, sont stockées les informations de conceptions erronées de l'apprenant. Ce stockage des informations mal maîtrisées par l'apprenant permet une meilleure planification des interventions.

Figure III.3 : Modèle overlay simple

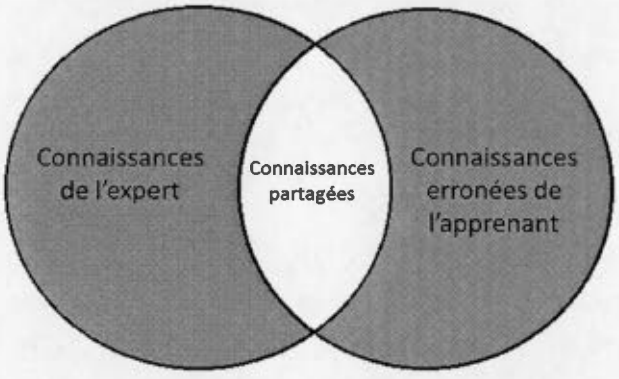


Figure III.4 : Modèle overlay avec erronées (buggy)

La modélisation de la connaissance de l'apprenant et de son comportement pendant l'apprentissage utilise fondamentalement deux procédures (Paquette, 2011) : 1) inventorier les domaines que l'apprenant a maîtrisés ou qu'il a tenté d'apprendre; 2) appliquer une méthode de reconnaissance de plans à ces patterns de réponse dans le but de comprendre le processus de raisonnement utilisé par l'étudiant pour produire une réponse. Des techniques ont été développées pour de modélisation de l'apprenant. Parmi ces techniques nous retrouvons (Nkambou, 2010b):

- le réseau sémantique : les nœuds et liens sont ajoutés au fur et à mesure que les concepts sont appris par l'apprenant,
- le réseau bayésien (Aleven, 2010b): chaque nœud du réseau a une valeur qui indique la probabilité que l'apprenant connaisse l'élément de connaissance concerné. Il permet un raisonnement probabiliste sur l'état des connaissances de l'apprenant en tenant compte des observations notées lors de ses interactions avec le tuteur.

La mise à jour du modèle de l'apprenant se fait généralement soit par une observation du comportement lors de la résolution de problèmes de l'apprenant (données psychométriques) tel qu'observer par le système via le composant interface, soit par des questions directes posées à l'étudiant.

3.1.3.4. L'interface utilisateur

Cette composante est l'entrée du STI. C'est en quelque sorte l'élément qui joue le rôle de support à travers lequel, les apprenants communiquent avec le système. Elle fournit à l'étudiant un environnement avec lequel il interagit, soit pour recevoir des messages du système, soit pour lui en donner. On l'appelle aussi parfois « module de communication » (Paquette, 2011). Les apprenants effectuent des actions sur l'interface et cette dernière, se charge d'interpréter ces actions et de les envoyer au système. De même, lorsque le système rend une réponse, l'interface se charge de communiquer la réponse à l'apprenant. Elle est autant indispensable que les autres composants.

Elle peut se présenter sous différentes formes selon le type d'environnement d'apprentissage et l'objectif visé. Elle peut prendre la forme d'une interface graphique (boutons, icônes, menus), d'un environnement à réalité virtuelle ou augmentée, d'une inerface tangible, d'un langage de commandes ou même du langage naturel (GUIDON

(Clancey, 1992)) etc. L'interface d'un STI doit être conçue en tenant compte du fait qu'elle ne devrait en aucun cas être ou devenir un obstacle à l'apprentissage.

3.1.4. Utilité des STI

D'un point de vue fondamental, les STI permettent de faciliter l'apprentissage. Ils permettent de comprendre et d'étudier les différents processus qui peuvent intervenir durant l'apprentissage. Ils peuvent fournir une assistance adaptée au niveau et aux besoins de l'apprenant. Les échanges entre le système et l'apprenant servent à affiner non seulement le modèle de l'apprenant, mais permettent également au système de savoir quel processus il déclenchera dans la suite des échanges.

D'un point de vue socio-économique, ils permettent de pallier au manque de ressources humaines. Ces systèmes sont capables de reproduire certains aspects de l'intelligence humaine voire de les dépasser. Ils font déjà partie intégrante des systèmes éducatifs des universités, des écoles.

3.2. STI en raisonnement logique

Jusqu'ici, nous ne nous sommes intéressés qu'aux STI en général, qu'en est-il des STI dans le domaine que nous visons dans ce mémoire (le raisonnement logique) ? En existe-t-il ? Si oui, pourquoi vouloir en développer un nouveau ? Nous répondrons à toutes ces questions dans cette section.

3.2.1. STI pour l'apprentissage de formules logiques

Un outil a été mis sur pied dans le but de faire apprendre la dérivation de formules logiques, et ce dans le cadre des cours de mathématiques enseignés aux étudiants de l'Université de Caroline du Nord depuis 2002 et à l'UNC Charlotte depuis 2006 (T. Barnes, 2006). Les apprenants doivent construire des preuves de dérivation d'une proposition à partir d'autres propositions logiques. Ainsi, le STI assigne à chaque apprenant, un ensemble de 10 problèmes qui vont de simples demandes d'équivalence logiques à des preuves d'inférence plus complexes. L'interface de l'outil permet aux utilisateurs d'entrer sous formes de lignes consécutives, leurs preuves. Après que l'étudiant ait construit des preuves à son raisonnement sur un exercice, il valide et le système vérifie la réponse. Si l'apprenant a commis une erreur, le STI affiche un message d'avertissements et marque en rouge les lignes erronées (Figure III.5). Il sauvegarde également cette erreur pour une analyse ultérieure. Concrètement, le STI donne un ensemble de formules (par exemple, les 3 premières formules de la Figure III.5), l'apprenant doit prouver que de ces formules, peut dériver une autre formule (qui est b^¬c dans ce cas) en faisant un ensemble de dérivations. Ces dérivations sont donc faite sur la base des propositions données. L'apprenant donne au fur et à mesure les formules qu'il arrive à dériver tout en mentionnant la ou les lignes concernées, ainsi que la raison (règle logique) d'une telle dérivation.

Statement	Line	Reason
1. a → b		Given
2. c → d		Given
3. ¬ (a → d)		Given
¬ a v d	3	rule IM (error)
4. a ^ ¬ d	3	rule IM implication
5. a	4	rule S simplification
b	4	rule MP (error)
b	1	rule MP (error)
6. b	1,5	rule MP modus ponens
7. ¬ d	4	rule S simplification
8. ¬c	2,7	rule MT modus tollens
9. b ^ ¬c	6,8	rule CJ conjunction

Figure III.5 : Exemple de preuves de formules logique

En considérant l'exemple de la figure ci-dessus, on constate que l'apprenant a mal utilisé la règle du (MPP) Modus Ponendo Ponens qui nous le rappelons est une règle de raisonnement valide. La règle valide du MPP stipule que : si on a p qui implique q et que p se réalise, alors on peut conclure que q aussi se réalisera. Or dans l'exemple, l'apprenant conclut b deux fois en utilisant respectivement et uniquement les formules propositionnelles 4 et 1, ce qui est une mauvaise réponse car pour pouvoir conclure b (un MPP), on a besoin des formules 1 et 5.

Bien que utile à l'apprentissage de dérivations logiques et à l'application de règles logiques, nous pouvons constater que cet outil ne se focalise pas sur l'apprentissage du raisonnement logique. D'un côté, il est centré uniquement sur la manipulation de symboles et propositions logiques. Cette méthode d'apprentissage est très abstraite, si l'on considère que l'objectif est d'apprendre à être de meilleurs raisonneurs. De plus, il ne peut être facilement utilisable par tout le monde car les apprenants non aptes à ces expressions et symboles logiques ne pourront pas facilement voir l'utilité d'un tel STI. D'un autre côté, il y a un manque de support métacognitif et une non explicitation des connaissances en jeu dans le raisonnement. Les erreurs commises par l'apprenant ne font pas objet d'une analyse cognitive pour déceler quel est le problème exact que l'apprenant rencontre. Le système se contente d'appliquer la règle fournit par l'apprenant. Si cette règle ne peut être appliquée, le système signale une erreur et oriente l'apprenant en l'aidant sur les prochaines étapes sans toutefois comprendre le pourquoi de l'erreur et n'est ainsi pas capable de lui donner un feedback plus approprié. Le système n'est pas capable de déterminer les compétences maitrisées ou non par l'apprenant.

3.2.2. Logic-ITA pour l'apprentissage de la logique propositionnelle

Logic-ITA (Lesta & Yacef, 2002) est un système dont le but est d'enseigner la logique propositionnelle. La Figure III.6 présente l'architecture du système. Il est composé du STI Logic-Tutor destiné aux apprenants et de deux autres composantes (LT-Analyser et LT-Configurator) destinés aux tuteurs. Il permet aux apprenants d'apprendre à prouver la validité d'un argument logique tout comme le précédant STI. Un argument est dit valide s'il dérive logiquement (utilisation de lois logiques comme le MPP, MTT, etc.) de certaines informations considérées vraies (les prémisses). A chaque étape, le tuteur du STI évalue la validité à la volée.

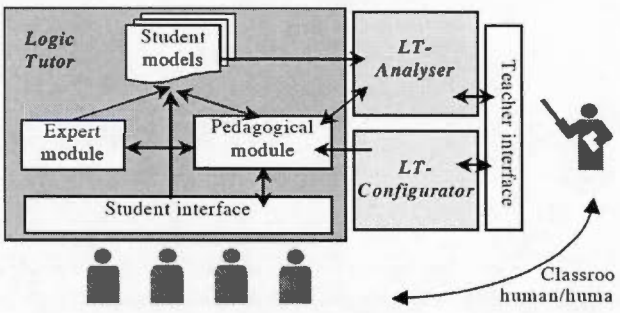


Figure III.6 : Architecture de Logic-ITA (T. Barnes, 2006)

Le système crée et maintient un modèle de l'apprenant. Le modèle de l'apprenant est un recouvrement partiel du modèle de l'expert, avec une partie définissant les incompréhensions de l'apprenant (buggy model). Toutes les actions de l'apprenant sont enregistrées dans ce modèle. Le module Expert contient l'expertise en logique propositionnelle. Il est en mesure de vérifier dynamiquement l'exactitude des réponses des élèves ainsi que la production de feedbacks appropriés lorsque des erreurs surviennent. Chaque ligne saisie par l'étudiant est évaluée, selon un principe d'erreurs en cascade. D'abord, Logic-ITA vérifie que tous les champs de la réponse sont remplis avec le bon type de données. Ensuite, la syntaxe de la formule est vérifiée. Pour ce qui est de l'évaluation des preuves de dérivations en elles-mêmes, si l'erreur est d'ordre

générale (mauvaise référence à une règle, mauvaise application d'une règle), le module expert propose une règle plus appropriée ou alors corrige l'erreur en proposant par exemples les numéros de lignes qui peuvent être appliqués à la règle en cours (si erreur d'application de règle). Le système propose également une correction se fondant sur une base de données des erreurs récurrentes et leurs solutions. Le module pédagogique contient les règles pour le séquençage des exercices en utilisant le modèle de l'apprenant.

Dans ce deuxième système, nous constatons aussi un manque d'analyse cognitive des différentes tâches, des compétences mises en jeu. D'un côté, le raisonnement humain n'est pas simulé en tant que tel. Le système apporte des solutions aux erreurs de l'apprenant sans toutefois comprendre pourquoi l'apprenant a commis cette erreur et comment le lui faire comprendre. D'un autre côté, le système fonctionne avec des niveaux. Chaque apprenant se voit attribuer le niveau 1 lors de sa première connexion. Or, il peut avoir des apprenants ayant un niveau élevé dès la base. De plus, le modèle de l'apprenant n'est pas assez élaboré, car il ne se base que sur les données brutes pour diagnostiquer les compétences de l'apprenant. Le STI n'est pas capable à un instant t, de dire quelles compétences sont maîtrisées ou non par un apprenant.

3.2.3. Prolog Tutor

Prolog Tutor (Tchetagni et al., 2007) est un STI destiné à soutenir l'apprentissage des notions de base en programmation logique. Prolog est un langage destiné à la programmation logique et très utilisé en IA. Ce langage fonde son calcul sur un mécanisme de raisonnement connu: la résolution, qui permet de formaliser les connaissances déclaratives et procédurales. En effet, il permet d'encoder les connaissances procédurales sous forme de règles, d'encoder toute situation concrète pouvant être logiquement décrite et ce en utilisant la logique des prédicats comme

formalisme. Parce qu'il est un langage purement déclaratif, prolog utilise un moteur d'inférence dont le calcul est basé sur un mécanisme de raisonnement appelé 'résolution'. Il simule le raisonnement logico-déductif pour résoudre les problèmes (Legrand, 1992). Acquérir les notions de base en Prolog porte généralement sur: (1) l'apprentissage du vocabulaire des structures de données utilisées pour représenter les informations (par exemple, comment sont exprimées les connaissances déclaratives en Prolog?) Et (2) l'apprentissage des principes régissant les algorithmes de raisonnement sur ces structures de données (Tchetagni et al., 2007).

Dans Prolog Tutor, les apprenants apprennent à prouver des théorèmes ou à résoudre des buts en utilisant les techniques de raisonnement de Prolog (résolution et d'unification). Ils apprennent ainsi à raisonner comme prolog, à faire de la programmation logique, à faire de la résolution par réfutation pour prouver des buts. Cependant, bien qu'il s'agisse d'un raisonnement logique, la résolution, contrairement au MPP et autres, n'est pas un mécanisme de raisonnement commun chez l'humain. Elle est plutôt efficace pour le raisonnement machine. Or MUSE-Logique vise l'apprentissage du raisonnement humain. La résolution par réfutation n'est donc pas simple à mettre en œuvre par les humains et n'est directement pas applicable à la vie de tous les jours. En plus de cela, le modèle du domaine de ce STI (modèle expert) ne se fonde que sur les connaissances du langage Prolog et non sur des connaissances globales en raisonnement logique.

Il existe également des environnements de e-learning pour l'apprentissage de la logique à l'instar du projet SELL (Huertas, Humet, López, & Mor, 2011) dont le but est de faire apprendre la logique en général, ou encore *power of logic web tutor* . Malheureusement, ils ne constituent qu'une banque d'exercices avec solutions. Les systèmes en eux-mêmes ne sont pas capables de résoudre les problèmes de manière autonome. Il n'y a pas de modèle apprenant et l'enseignement n'est pas adaptatif. Une même erreur à un problème spécifique produira le même feedback. Les exercices sont présentés dans le même ordre à chaque apprenant peu importe ses besoins.

Nous avons présenté un certain nombre d'outils existant pour l'apprentissage de la logique et ou du raisonnement logique et nous avons pu constater de façon générale un certain nombre de limites. Tout d'abord, il y a peu de systèmes qui traitent du raisonnement en tant qu'objectif d'apprentissage, et ceux qui s'y intéressent ont des limites quant à une explicitation profonde des connaissances et des compétences en jeu. Ces systèmes ne visent pas nécessairement une approche globale de l'apprentissage du raisonnement par son incarnation par exemple dans différents contextes (situation de raisonnements). Ils ne sont pas capables d'expliquer le choix d'une règle quelconque. Ils ne reproduisent pas le raisonnement humain. De plus aucune extension à d'autres logiques n'est prévue dans ces systèmes. Il y a donc une nécessité d'aller plus loin en matière de support technologique étant donné ces limites des STI actuels dans le domaine.

Parce que l'apprentissage du raisonnement logique est important et parce que le support technologique actuel est insuffisant, il est donc nécessaire de proposer une meilleure solution: Muse-Logique.

CHAPITRE IV

LE STI MUSE-LOGIQUE : ARCHITECTURE ET ELEMENTS CONCEPTUELS

Après une présentation détaillée des domaines clés dans ce rapport (Système Tutoriel Intelligent et Raisonnement Logique), et une revue des manques à combler dans la littérature concernant les systèmes en raisonnement logique, nous entrons dans le vif du sujet : la conception du STI Muse-logique. Muse-logique est un système tutoriel intelligent capable de modéliser et simuler le raisonnement humain tout en expliquant son cheminement, et de faire apprendre aux humains à être de meilleurs raisonneurs. Muse-logique ce veut être un outil extensible à d'autres types de raisonnements et d'autres logiques. Dans ce chapitre, nous présenterons l'architecture du système ainsi que les éléments conceptuels qui ont servi au développement. Nous démontrerons que l'approche d'ingénierie adoptée dans cette conception, ainsi que nos modèles développés, sont conçus de sorte à faciliter l'extension à d'autres logiques et d'autres types ou modes de raisonnement.

4.1. Composants de Muse-logique

Tout système tutoriel intelligent doit posséder les 4 composantes présentées plus haut à savoir le modèle apprenant, tuteur, expert et l'interface. Dans cette section, l'attention sera portée particulièrement sur les 3 premiers. Concernant l'environnement d'apprentissage (l'interface) de Muse-Logique, nous avons choisi de faire dialoguer le

système avec ses utilisateurs par une interface web dont les détails techniques seront présentés au chapitre suivant (0).

4.1.1. Muse-Apprenant

Le modèle apprenant est une représentation de l'apprenant dans Muse-Logique. Il se compose de 6 modèles à savoir le modèle psychométrique, le modèle statistique, le modèle affectif, le modèle psychologique, le modèle de connaissances et le modèle de compétences (voir Figure IV.1). Le modèle psychométrique contient les informations sur l'apprenant qui permettra à Muse-logique d'initialiser le modèle cognitif de l'apprenant. Le modèle statistique ou mémoire épisodique correspond aux traces que le système récupère des activités (réponses, temps de résolution, nombre d'erreurs, etc.) de l'apprenant. Le modèle affectif, qui utilisera des technologies telle que des *eyes-trakers* et ou des *face-readears*, permettra d'interpréter les expressions du visage ou positions du regard afin de modéliser l'état affectif de l'apprenant. Le modèle psychologique correspond aux comportements et processus mentaux de l'apprenant. Nous tenons à mentionner que ces deux dernières sous-composantes du modèle apprenant ne sont actuellement pas encore développés, mais la construction de l'architecture permettra d'y tenir compte très facilement le moment opportun. Le modèle cognitif (modèle de connaissances et de compétences), modélise l'ensemble des connaissances et des compétences en raisonnement logique acquises correctement ou pas par l'apprenant. Dans Muse-logique, les connaissances de l'apprenant ont été modélisées comme un recouvrement (overlay) sur les connaissances de l'expert, mais à l'aide d'un réseau bayésien (voir détails dans la section 4.3 de ce chapitre).

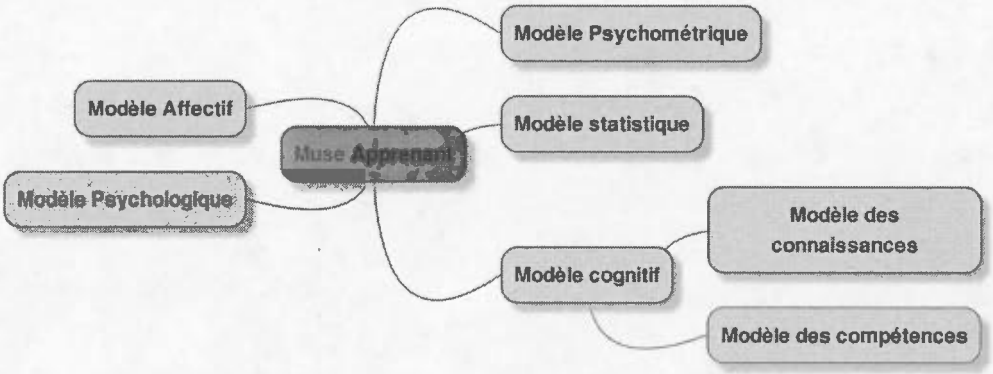


Figure IV.1 : Modèle de Muse-Logique Apprenant

4.1.2. Muse-Tuteur

Le modèle tuteur de Muse-Logique joue plusieurs rôles (voir Figure IV.2) :

- détermine les compétences visées par l'apprentissage (en interaction avec l'apprenant),
- conduit l'interaction avec l'apprenant,
- fait un suivi du raisonnement de l'apprenant,
- détecte les erreurs de l'apprenant (en collaboration avec le module expert et le modèle de l'apprenant),
- effectue un diagnostic cognitif des erreurs détectées,
- planifie des activités remédiatives pour défaire le mauvais et reconstruire le bon modèle de raisonnement;
- évalue les apprentissages donnant lieu à une mise à jour du modèle de l'apprenant (en collaboration avec l'expert du domaine).

Son modèle pédagogique est sous forme d'une ontologie des théories d'apprentissage et d'instruction; soutenant un ensemble de scénarios pédagogiques reliés à différentes situations authentiques et associé à des objectifs (compétences)

précis (acquisition d'une compétence logique particulière). Les interactions avec les apprenants dans Muse-Logique sont gérées par un ensemble de stratégies métacognitives, afin de 1) susciter la réflexion chez l'apprenant, 2) contrôler l'acquisition des compétences métacognitives de l'apprenant en matière de raisonnement. Ces interactions s'appuient également sur les bonnes règles que doivent respecter les feedbacks dans les technologies éducatives (Narciss, 2008).

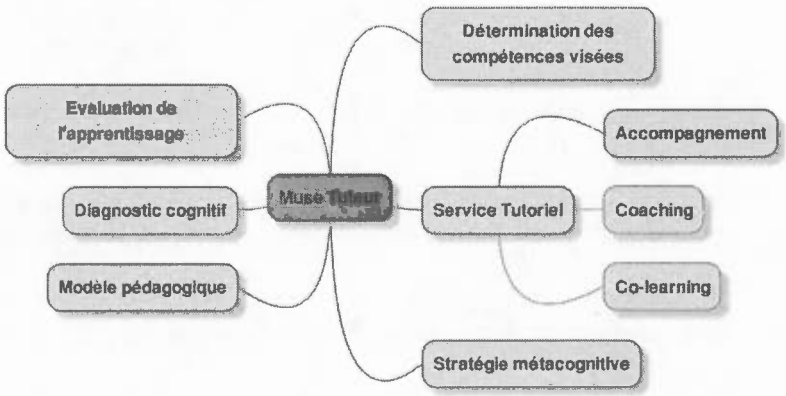


Figure IV.2 : Modèle de Muse-Logique Tuteur

4.1.3. Muse-Expert

Muse-Logique expert (Figure IV.3) a pour but de produire des raisonnements valides dans toutes les situations authentiques qui lui sont soumises et d'expliquer son processus de raisonnement. Il est également capable de détecter des erreurs de raisonnement et vérifier les actions légales dans le contexte suivant les règles qui le régissent, suivre des schémas de raisonnement dans le but de les valider et d'en proposer des correctifs. Il collabore avec le tuteur pour l'aider à évaluer et valider les productions de l'apprenant, et à déterminer ses compétences logiques. Il comporte un ensemble d'éléments illustrés dans la Figure IV.3 que nous décrivons ci-dessous.

Le modèle des connaissances logiques est composé de la mémoire sémantique logique (MSL) codée sous forme d'une ontologie du domaine et associée à chaque Logique, du modèle de règle d'inférence (MRI) représentant les règles du domaine (les connaissances procédurales du domaine sont codées avec une référence sur les concepts de l'ontologie du domaine) et d'un modèle de structures métalogiques pour la gestion des méta-structures (supports) des procédures de raisonnement.

Le catalogue des erreurs sert à l'encodage des erreurs d'inférence logique (sophismes) connues dans chaque logique mais aussi des erreurs relatives à des suppressions d'inférence ou encore des erreurs de logique non nécessairement liées à un raisonnement (par exemple, certaines confusions reliées aux concepts de la logique). C'est pour cette raison que ce catalogue indexe à la fois le modèle sémantique (MSL) et le modèle des règles d'inférence (MRI) permettant ainsi d'apparier les sophismes et autres erreurs aux connaissances et structures logiques concernées se trouvant tant dans la MSL que dans la MRI.

Le modèle de compétences décrit les compétences de l'expert dépendamment de la situation, du contexte et du domaine. Ceci se définit comme étant une mobilisation des connaissances et habilités pour résoudre un problème donné.

Un ensemble de situations authentiques modélisées sous forme d'espaces problèmes (par une analyse cognitive de tâches de raisonnement) est associé aux compétences.

Un moteur de raisonnement permet d'exploiter les différents sous-composants du modèle expert ci-dessus, dans toutes les situations authentiques identifiées.

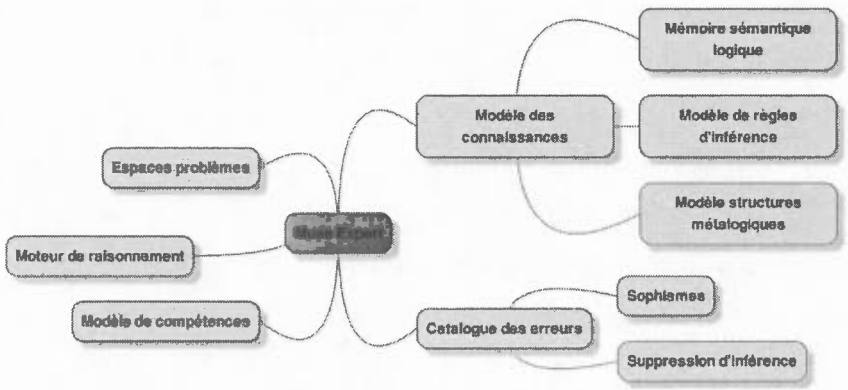


Figure IV.3 : Modèle de Muse-logique Expert

Pour que Muse-logique puisse tenir compte d'autres types de raisonnement, et d'autres logiques, nous avons doté le système d'un contrôleur dont le but est de déterminé la logique appropriée selon le contexte du raisonnement. L'architecture interne de Muse-Expert qui peut être visualisée à la Figure IV.4, a été conçue de manière que le STI puisse remplir les attentes fixées. Le contrôleur général est défini pour déterminer la logique et les compétences en jeu. Il s'appuie sur le contexte pour définir la (les) logiques admissibles. Il existe pour chaque logique (classique, floue, etc...), un contrôleur local. Chaque contrôleur local a pour but de gérer les composants (MRI, MSL) de la logique cible. Dans chacun de ces contrôleurs locaux, est encodé l'algorithme qui permet de sélectionner le bon raisonnement suivant le problème posé. Nous tenons à mentionner que la logique implémentée pour le moment est la logique des propositions appartenant à la logique classique. Toutefois, l'architecture de notre module expert tel que présentée à la Figure IV.4 nous permet d'étendre les fonctionnalités du STI Muse-Logique.

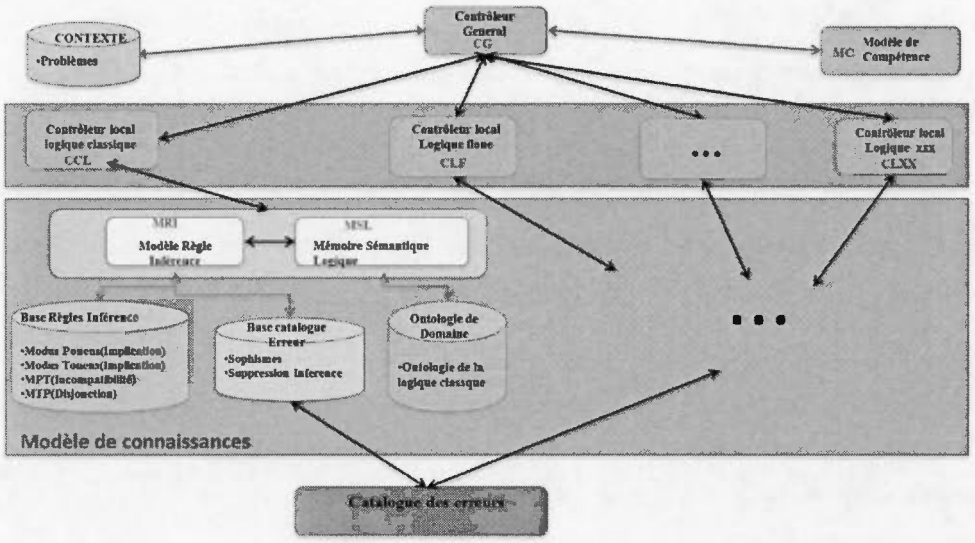


Figure IV.4 : Architecture interne de Muse-Logique Expert

4.2. Architecture fonctionnelle et modélisation UML de Muse-Logique

Afin de comprendre le fonctionnement de Muse-logique, il faut connaître les différentes interactions entre ses composantes, classes et comprendre le pourquoi de ces interactions. A cet effet, cette section présente l'architecture fonctionnelle du système. Dans un premier temps nous présenterons le cycle de fonctionnement, ensuite le diagramme de classes du projet et nous terminerons par l'architecture de la base de données et l'architecture globale de Muse-logique.

4.2.1. Cycle de fonctionnement

Le diagramme d'activité est un diagramme UML (outil de communication entre les membres du projet et les clients du projet) servant comme moyen graphique pour donner une vision d'ensemble sur les différentes actions qui s'effectuent dans un

système. La Figure IV.5 est le diagramme d'activité (version simplifiée) de Muse-logique qui représente le cycle de fonctionnement du système. Ce diagramme servira par la suite à l'implémentation des interactions du tuteur avec l'apprenant.

Lors de la connexion d'un apprenant, s'il s'agit d'une première connexion, il est prompté vers le test de niveau. Ce test de niveau sert d'initialisation du modèle de l'apprenant à travers le modèle psychométrique. Il permet ainsi à Muse-Logique de se faire une idée sur le modèle cognitif de l'apprenant et préparer ses interventions en conséquence. Lorsque l'apprenant passe le test ou si ce n'est sa première connexion, il devra choisir entre l'option « *Tutoring* » et l'option « *Training* » afin de continuer. La première option lui permet de profiter des capacités du tuteur, c'est-à-dire que, le tuteur lui présentera les exercices et réagira avec lui en fonction de son niveau de compétences. Toutes les activités de l'apprenant (temps mis pour répondre, nombre de clics sur l'aide, nombre d'échecs et de succès, etc.) seront enregistrées et prisent en compte par le système. Le modèle de l'apprenant sera mis à jour au fur et à mesure des interactions. Par contre, en mode « *Training* », la liberté est laissée à l'apprenant. Il peut s'exercer sur le type d'exercice qu'il souhaite faire. Cependant, les interactions enregistrées dans ce mode ne modifieront pas le modèle de l'apprenant. Ces informations seront gardées pour des fins de datamining par la suite.

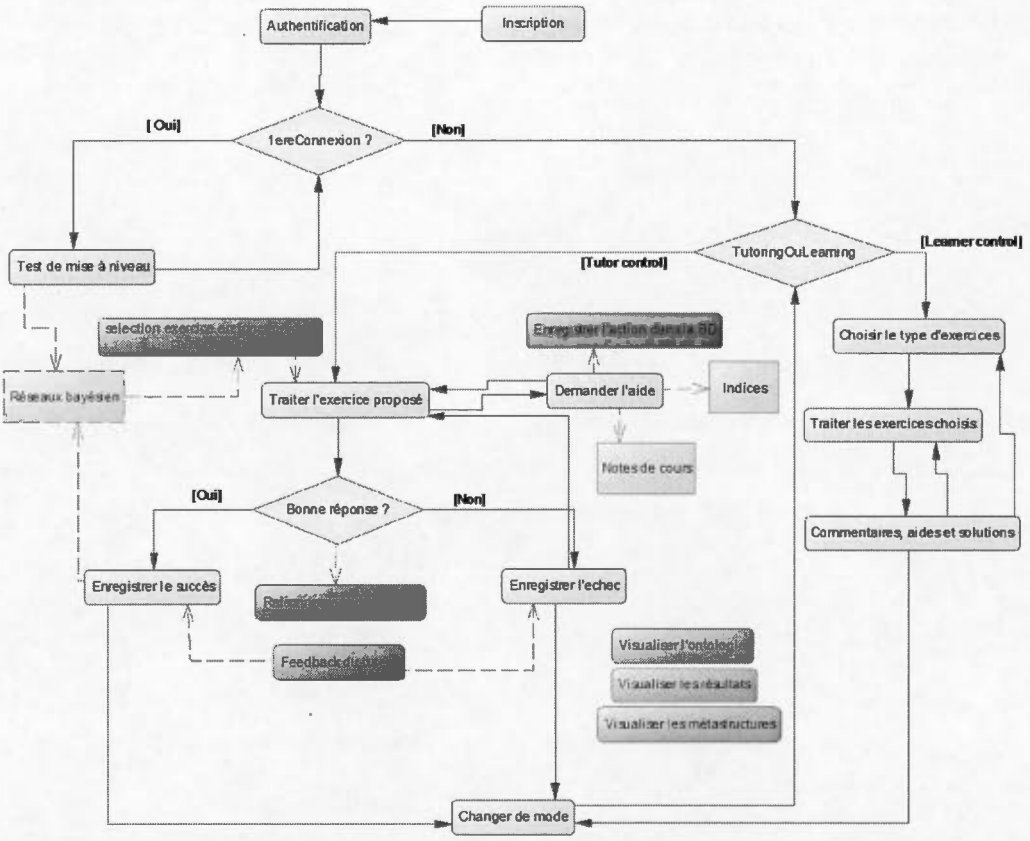


Figure IV.5 : Diagramme d'activité simplifiée de Muse-Logique

Lorsqu'un exercice est présenté à un apprenant, si cet exercice n'a jamais été traité, le tuteur sollicite l'aide de l'expert pour raisonner sur l'exercice, et lui fournir la bonne réponse. Cette réponse est enregistrée dans la base de données pour que, les prochaines fois l'expert ne soit plus sollicité pour un travail déjà effectué. Le tuteur compare alors la réponse de l'expert et celle de l'apprenant. A cet instant, si la réponse de l'apprenant ne concorde pas avec celle du tuteur, le tuteur sollicite une seconde fois l'expert pour savoir quel type d'erreur l'apprenant a commis et pourquoi (les détails sur les interactions entre le tuteur et l'expert sont présentés dans la section 4.3.2 de ce chapitre). Bien entendu, chaque réponse de l'apprenant est stockée dans la mémoire

épisodique du modèle statistique de l'apprenant et sert à mettre à jour son niveau de compétence (voir les détails de cette approche dans la section 4.3.1 de ce chapitre).

4.2.2. Diagramme de package

Nous présentons à la Figure IV.7 le diagramme de package du projet Muse-Logique. Ce diagramme définit l'architecture des packages du projet. Il représente aussi l'organisation de notre code. Nous avons opté pour l'un des plus célèbres *design patterns* qui est l'architecture MVC (Modèle Vue Contrôleur). Le pattern MVC permet de bien organiser le code source. Son but est de séparer la logique du code en trois parties qui sont le modèle, la vue et le contrôleur . La Figure IV.6 résume cette architecture.

- Modèle : cette partie gère les données du projet. Son rôle est d'aller récupérer les informations « brutes » dans la base de données, de les organiser et de les assembler pour qu'elles puissent ensuite être traitées par le contrôleur. On y trouve donc les requêtes SQL. Parfois, les données ne sont pas stockées dans une base de données. On peut être amené à aller chercher des données dans des fichiers (dans notre cas, chargement du fichier contenant le modèle de compétences ou réseau bayésien). Dans cette situation, le rôle du modèle est de faire les opérations d'ouverture, de lecture et d'écriture de fichiers.
- Vue : cette partie se concentre sur l'affichage. Elle se contente de récupérer des variables que le contrôleur lui fournit pour savoir ce qu'elle doit afficher. On y trouvera dans notre cas, essentiellement du code HTML, JSP et JavaScript.
- Contrôleur : cette partie gère la logique du code qui prend des décisions. C'est en quelque sorte l'intermédiaire entre le modèle et la vue : le contrôleur va demander au modèle les données, les analyser, prendre des décisions et

renvoyer des informations à la vue. Dans le cas de Muse-Logique, le contrôleur contient exclusivement du Java et du JESS (pour l'écriture des règles).

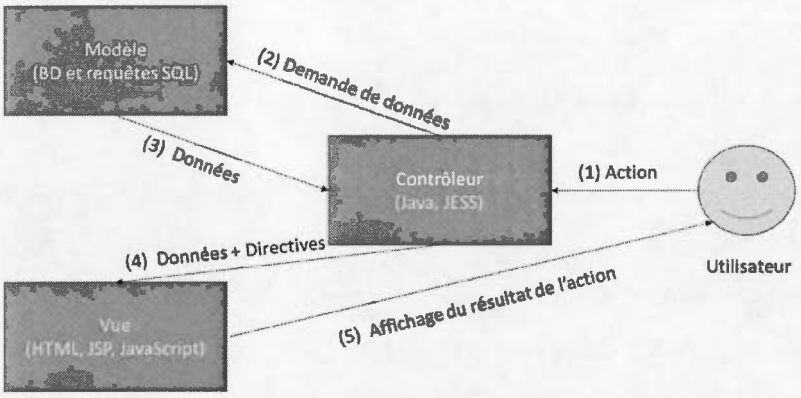


Figure IV.6 : Architecture MVC

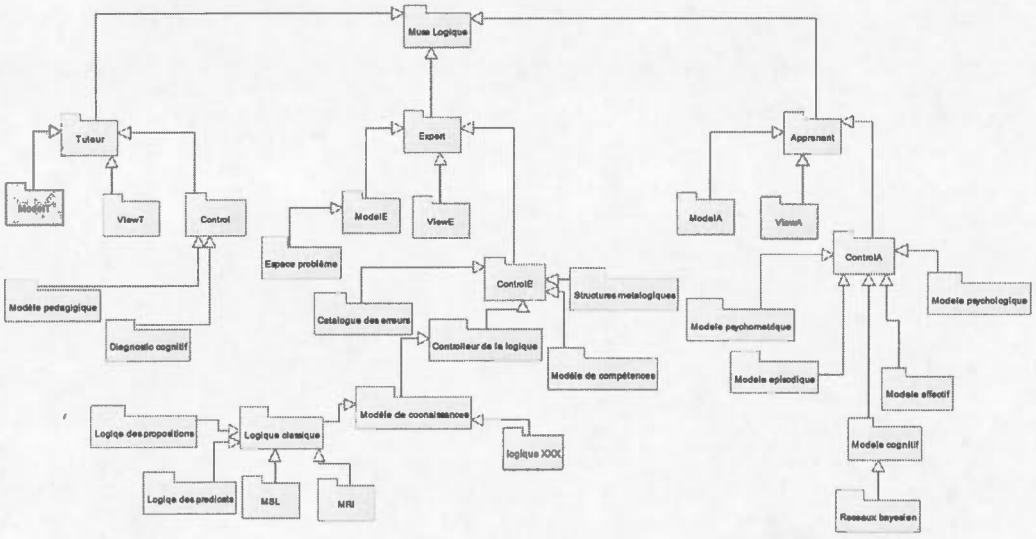


Figure IV.7 : Diagramme de package du projet Muse-Logique

Chaque composante de Muse-Logique (à l'exception de l'interface) a une architecture MVC (voir Figure IV.7). Par exemple, dans le composant Muse-Tuteur, nous retrouvons le modèle dans lequel on a les données sur les tuteurs humains qui font la création d'exercice et la gestion des apprenants. Nous y trouverons également la vue qui contient les interfaces (interface de création d'exercice, etc.) du tuteur et le

contrôleur qui contient les règles tutorielles et les fonctions de diagnostic du modèle cognitif de l'apprenant.

4.2.3. Architecture de la Base de données Muse-Logique

La banque d'exercices, les données sur les apprenants et les tuteurs, les traces des différentes interactions sur le système, sont enregistrés dans une base de données accessible à tout instant par les modules du système. L'architecture de cette base de données est présentée à la Figure IV.8. Nous avons dans cette première version un ensemble de 13 tables. Les tables contenant les exercices sont « énonces raisonnement » pour les exercices de raisonnement et « énonces logiques » pour des exercices de manipulation d'expressions logiques. Les tables « épisodiques » contiennent les résultats aux exercices, le nombre de temps passé à les résoudre, le nombre d'appels à l'aide etc.

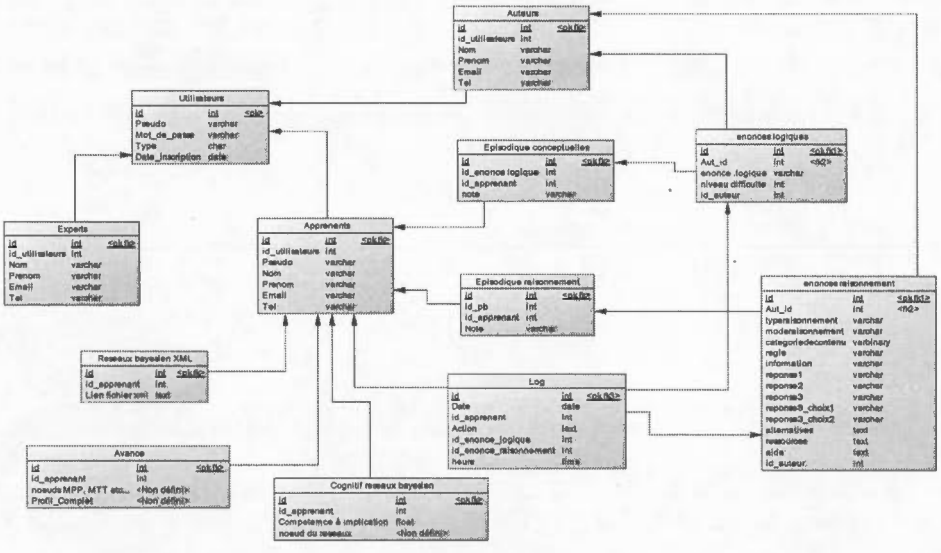


Figure IV.8 : Architecture de la BD de Muse-Logique

Les tables « Avancé» et « cognitif réseau bayésien » contiennent les informations sur l'état du réseau bayésien de chaque apprenant. Ils permettent également au tuteur de savoir à quel niveau se trouvait l'apprenant lors de la dernière connexion afin de lui permettre de continuer son apprentissage sans toutefois réinitialiser son modèle. La table « auteurs » contient les informations sur les différents tuteurs humains actifs sur le système, les tables « utilisateurs » et « apprenants » font de même respectivement pour tous les utilisateurs du système et les apprenants.

4.3. Éléments conceptuelles de Muse-Logique

Comment est supporté techniquement parlant, le fonctionnement de Muse Logique? Jusque-là, nous avons énuméré et venté les différentes fonctions que peut remplir le système. Maintenant, nous présentons les éléments de conception de Muse-Logique nous aidant à réaliser les tâches qui lui incombent.

4.3.1. Réseaux bayésien

Un réseau bayésien est un modèle probabiliste qui se présente sous forme de graphe acyclique représentant un ensemble d'éléments (nœuds) liés entre eux par des liens de causalités, avec chaque élément ayant sa table de probabilité. Les réseaux bayésiens sont à la fois des modèles de représentation des connaissances, des machines à calculer les probabilités conditionnelles, etc (Tsamardinos, Brown, & Aliferis, 2006). Nous allons étudier un exemple pour comprendre cette extraordinaire structure.

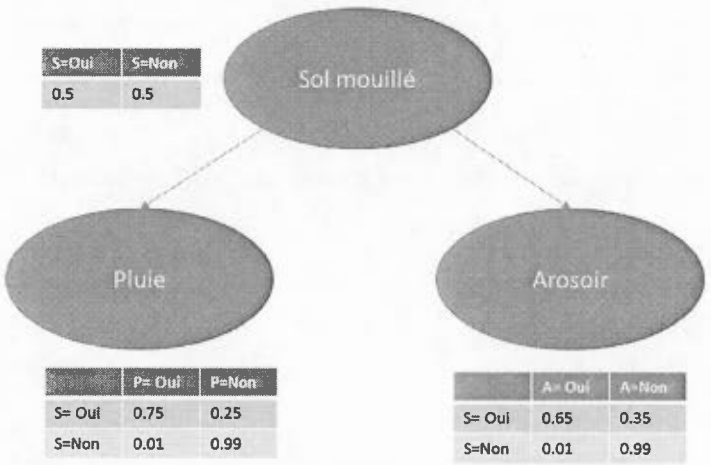


Figure IV.9 : Exemple d'un réseau bayésien

La figure 4.9 présente un réseau bayésien créé dans le cadre de ce mémoire juste à titre d'exemple. Il représente une situation où, on se trouve devant un sol mouillé en sortant de la maison et on ne sait pas s'il a plu ou si l'arrosoir a été mis en marche. Dans cette figure, le nœud parent est le « sol mouillé » qui lui, n'a aucun effet sur un autre nœud (ne cause rien), raison pour laquelle il n'a pas de parents. Par contre, les nœuds « pluie » et « arrosoir » causent tous les deux un sol mouillé et c'est la raison pour laquelle leur parent commun est le « sol mouillé ». Les tables de probabilités représentent les probabilités de réalisation de différents événements dans ce contexte. Par exemple, la table de probabilité du nœud parent signifie que, il y a 50 % de chance de trouver un sol mouillé en sortant de la maison. La probabilité qu'il ait plu sachant que le sol est mouillé est 75% et la probabilité qu'il n'ait pas plu sachant que le sol n'est pas mouillé est 90 % sont des résultats lisibles dans la table de probabilité de la « pluie ». Le même raisonnement peut être fait sur le nœud « arrosoir ». Il y a des probabilités qui ne sont pas directement lisibles sur le réseau. Par exemple, si l'on veut savoir la probabilité que le sol soit mouillé, qu'il ait plu et que l'arrosoir a été mis en marche, on ne saurait avoir le résultat sans passer par des algorithmes d'inférences. A cet effet, il existe des algorithmes qui peuvent faire des inférences exactes (Arbre de jonction…) et des algorithmes avec inférences approchées. La bonne nouvelle est que,

nous n'avons pas besoin d'entrer dans des détails algorithmiques, car il existe des outils de manipulation de réseaux bayésiens qui fournissent le nécessaire. Nous verrons dans le chapitre suivant les détails d'implémentation.

La mise en place d'un réseau bayésien passe par 2 principales étapes .

- Le développement du modèle qualitatif : il s'agit de la construction de la structure (nœuds et arcs) du réseau. Quels sont les éléments de connaissances mis en jeu et quels liens existent-ils entre ces éléments.
- Le développement du modèle quantitatif : il s'agit de la spécification des paramètres (probabilités à priori, totales et conditionnelles) du réseau.

L'obtention de ces modèles peut se faire de 2 différentes manières (Brusilovsky & Millán, 2007). La première façon est d'inférer les informations à partir des données en utilisant des algorithmes d'apprentissage (Heckerman, 2008) et la seconde façon est de faire appel aux opinions des experts du domaine. La combinaison de ces deux méthodes est également possible. Dans notre cas, nous avons opté pour la deuxième méthode d'autant plus que nous avions accès en tout temps à l'expertise des experts.

Comme mentionné plus tôt dans ce chapitre, les connaissances de l'apprenant seront modélisées comme un recouvrement sur les connaissances de l'expert mais, à l'aide d'un réseau bayésien mettant en perspective les interdépendances probabilistes entre les connaissances logiques et les différents modes de raisonnement. Cette approche nous offre ainsi, un ensemble de fonction pour l'élaboration du diagnostic cognitif, l'initialisation et la mise à jour du modèle, l'accès aux informations du modèle et même la visualisation du modèle par l'étudiant lui-même ou par une tierce personne. Le réseau bayésien de Muse-Logique pour la compétence en raisonnement implicatif se trouve à l'annexe A du présent mémoire.

Le réseau bayésien développé pour MUSE Logique prend en compte deux types de nœuds. Les noeuds mesurant la connaissance de l'apprenant ou nœuds de compétences (nœuds visibles dans le réseau à l'annexe A) et les nœuds contenant les

évidences, c'est-à-dire les résultats aux exercices (ne sont pas présentés dans le réseau pour des raisons de clarté). Les nœuds de compétences représentent des compétences acquises ou non par l'apprenant qui seront observés par le système. Ces nœuds ont une probabilité continue comprise entre 0 et 1. Les nœuds des évidences sont utilisés pour récupérer de l'information sur l'apprenant. Ils représentent les réponses de l'apprenant aux questions de la banque d'items. Les nœuds des évidences sont représentés par une variable aléatoire Q avec une distribution de Bernoulli. Q = 1 signifie que l'étudiant a répondu correctement à l'exercice Q = 0 signifie que sa réponse est incorrecte. Dans ce réseau, le nœud racine « Implication » représente la compétence globale d'un apprenant au raisonnement conditionnel. Sa compétence à faire des inférences dans les 4 contextes (Familier, Contrefactuel, Abstrait et symbolique) décrits dans le chapitre 1, est représentée avec les noeuds « Factuel», « Contre-Factuel », « Abstrait », « Symbolique ». On représente également ses compétences pour les 3 étapes nécessaires au raisonnement conditionnel « Inhiber P&nonQ », « Avoir une bonne gestion des 3 modèles », « Générer nonP&Q ». Enfin, on modélise ses compétences dans les différentes inférences avec les noeuds « MPP », « MTT », « AC » « DA » et les cas particulier en contexte causal des inférences avec beaucoup d'alternatives (MD : *Many Disablers*) « MPPMD », « MTTMD », « ACMD », « DAMD » et peu d'alternative (FD : *Few Disablers*) « MPPFD », « MTTFD », « ACFD », « DAFD ». Le raisonnement implicatif est fait dans 4 contextes différents c'est pourquoi nous avons un lien entre le nœud compétence à l'implication et chacun des nœuds compétence à l'implication dans un contexte particulier. De plus, pour faire un raisonnement implicatif, l'apprenant doit Inhiber P&nonQ, Générer nonP&Q et Avoir une bonne gestion des 3 modèles mentaux dans le raisonnement. Chaque nœud de compétence dans un contexte particulier est lié à la compétence à Inhiber P&nonQ dans ce contexte, à Générer nonP&Q dans ce contexte et Avoir une bonne gestion des 3 modèles mentaux dans ce contexte. Cependant, Générer nonP&Q est une sous compétence du fait d'Avoir une bonne gestion des 3 modèles. Il n y a donc pas de lien direct entre la compétence à l'implication dans un contexte et la compétence à Générer nonP&Q mais

un lien indirect en passant par la compétence à Avoir une bonne gestion des 3 modèles mentaux. Enfin, on sait que la capacité à Inhiber P&nonQ permet de réussir des exercices de type MPP. Il y a donc un lien entre le nœud Inhiber p&nonQ et le nœud de compétence aux exercices de type MPP. Les compétences à Inhiber P&nonQ et à Avoir une bonne gestion des 3 modèles permettent de réussir des exercices de type MTT il y a donc un lien entre ces deux compétences et la compétence aux exercices de type MTT. Pour finir, la capacité à Générer nonP&Q permet de réussir des exercices de type AC et DA. Il y a donc un lien entre les nœuds AC et Générer nonP&Q et DA et Générer nonP&Q. L'étude approfondit sur la construction de la structure du réseau bayésien dans Muse-logique est disponible dans le rapport de Mascle C. (Carolane, 2015).

Il est possible d'automatiser le diagnostic initial de l'état de connaissance de l'apprenant en se basant uniquement sur son pattern de réponses à des questions ou items (Tatsuoka, 1983). Cette automatisation se fait par le biais de l'entraînement d'un modèle de prédiction permettant de faire des hypothèses probabilistes sur l'état des connaissances de l'apprenant étant donné ses réponses à un ensemble d'items. Cet ensemble d'items constituent en fait, le pré test que l'apprenant passera à chaque première connexion sur le système. Le développement de ce modèle de prédiction passe par la préparation par les experts d'une Q-matrix. La Q-Matrix met en relation les concepts ou compétences du domaine (en colonne dans la matrice), et les items ou questions (en ligne). La matrice contient 1 si la compétence est nécessaire pour réussir l'item, 0 sinon. Dans la cadre du projet Muse-Logique, la Q-Matrix (voir annexe B) a été fournie par les experts en logique. Il est à noter que les items contenus dans le pré test sont du même type que ceux prévus dans la Q-Matrix. Ceci permet de recueillir le patron de réponse de chaque apprenant sous forme d'un vecteur de 1 et 0 (1 pour un exercice réussi, 0 sinon). La dimension psychométrique vient nourrir l'initialisation du modèle cognitif de l'apprenant. Cette dimension psychométrique est supportée par la librairie CDM (*Cognitive Diagnostic Models*) (Templin & Henson, 2006). CDM utilise

les informations telles que le pattern de réponses de l'apprenant au test de niveau (le vecteur mentionné ci-dessus), une matrice appelée DINA (qui contient en ligne des étudiants et en colonnes des exercices) et la Q-Matrix. Cette librairie est très puissante et permet d'obtenir d'autres résultats comme le GUESS (probabilité qu'une personne réussisse à un exercice sans avoir la compétence requise), le SLIP (probabilité qu'une personne rate un exercice pourtant il a la compétence requise), le *Marginal skill probabilities* (probabilité de disposer de telle ou telle compétence), le Posterior (permet de connaitre la probabilité de maitriser tel ou tel pattern de compétence à partir d'un pattern de réponse) etc. Grâce notamment au Postérior (voir Figure V.5), CDM est capable de prédire le niveau de connaissances d'un apprenant.

L'estimation ainsi faite sur les compétences à priori de l'apprenant est guidée par les données. Cette estimation contribue à nourrir le Réseau Bayésien initial de tout apprenant. La démarche entreprise avec le CDM nous permettra par la même occasion de valider la structure (les liens entre les noeuds, etc.) du réseau bayésien. En effet, parmi les résultats obtenus avec le CDM, nous avons une matrice de corrélation dans laquelle les lignes et les colonnes représentent les compétences en jeu. Cette matrice nous permet de savoir si une compétence est liée fortement ou pas à une autre, et à quel degré. En observant un exemple de cette matrice dans le Tableau IV.1 : Exemple d'une matrice de corrélation entre les compétences A et B, on voit que les compétences A et B sont faiblement corrélées, ce qui implique qu'il ne devrait pas avoir de lien direct entre ces deux compétences dans le réseau. Nous présentons ces quelques détails dans le but de mettre en perspective la multidisciplinarité évoquée dans le modèle de l'apprenant. L'intégralité de cet initialisation du modèle de l'apprenant a fait l'objet d'un sujet de stage dont le rapport rédigé par Adam C.(Christophe, 2015).

Tableau IV.1 : Exemple d'une matrice de corrélation entre les compétences A et B

	A	B
A	1	0.03
B	0,03	1

4.3.2. Règles tutorielles

L'un des buts de tuteur est d'encadrer l'étudiant dans l'application de règles de raisonnement valide. Un autre objectif est, avec l'aide de l'expert, de corriger les idées fausses qui se manifestent chez l'apprenant. Le tuteur doit fournir un feedback à l'apprenant (feed-back positif, neutre et négatif), poser des questions à l'apprenant pour comprendre les raisons de ses actions ("Pourquoi ne peut-on pas conclure dans ce cas ?"). Les règles pédagogies sont très importantes dans le développement des systèmes tutoriels intelligents. Le comportement du tuteur doit régir de ces règles. Ces règles doivent remplir un certain nombre de critères pour ne pas être plutôt un obstacle à l'apprentissage. Les interactions du tuteur doivent être fondées sur les connaissances élicitées et sur les techniques d'élaboration de feedbacks dans les technologies éducatives.

Grâce aux experts du domaine, nous avons pu étudier les types d'erreurs qui peuvent survenir chez les apprenants, comment les reconnaître, les causes de ces erreurs et des techniques remédiatives à cela. Nous avons donc adapté le comportement du tuteur en conséquence. Par exemple, soit l'une des règles tutorielles de Muse-logique: 'Si l'exercice est de type AC ou DA et que l'apprenant à donner une mauvaise réponse au prompt 2 alors il faut lui rappeler qu'il est important de toujours faire comme si la règle était vraie'; la raison pour laquelle le tuteur réagit de la sorte est que, nous avons découvert que certains apprenants peinent à raisonner logiquement sur une

règle qui semble à leur yeux ne pas être logique. Concrètement considérons le syllogisme :

Règle : Si on lance du ketchup sur une chemise, alors elle deviendra propre.

Information : Une chemise est propre.

Un apprenant pourrait ne pas conclure dans cette situation, car il n'est pas certain que si on y lance du ketchup, alors la chemise deviendra propre. Or c'est une erreur de raisonnement, car il doute de la règle qui lui a été donnée. A la suite du feedback du tuteur lui expliquant son erreur, puisque l'apprenant n'a pas réussi ce type d'exercice, le tuteur en choisit un autre du même type. Dans le cas où l'apprenant ferait une deuxième erreur, le tuteur se verra dans l'obligation de lui expliquer la règle logique derrière ce type d'exercice.

Lorsqu'un apprenant échoue à un exercice, il n'est pas préférable de le lui annoncé de façon farouche comme « Non, ce n'est pas correct ». Il a été démontré pédagogiquement que les STI qui ont un comportement de la sorte, entraînent chez l'apprenant un découragement, une perte d'estime de soi (Narciss, 2008). Cela pourrait même engendrer l'abandon chez le sujet. Le système ne doit pas donner le résultat à un exercice immédiatement après une réponse de l'apprenant. Le système doit se réserver de le faire uniquement quand c'est nécessaire. Il faut absolument éviter cela et à cet effet nos règles tutorielles sont construites dans le respect des normes en matière de feedback dans les technologies éducatives. Nous avons construit deux banques de message; une pour lorsque l'apprenant réussit un exercice et l'autre dans le cas contraire. Chaque banque contient une vingtaine de messages, chaque message est tiré au hasard afin d'éviter une redondance dans les interactions du tuteur. Nous avons pu constater que cela évite l'effet rétroaction machine qu'on a l'habitude de rencontrer. Des exemples de messages qu'on peut rencontrer sont :

- Si exercice trouvé : « Bravo ! tu mérites tout mon respect », « Génial ! Continue comme ça », « Tu me surprends de plus en plus. Magnifique, encore un exercice de réussit »
- Si exercice échoué : « Pas grave, la prochaine fois sera la bonne », « Tu as commis une petite erreur, mais on va arranger çà », « Tu y es presque continue d'apprendre »

Ces messages lorsqu'ils sont affichés à l'apprenant, ils sont suivis d'un feedback particulier à un exercice. Si nous reprenons l'exemple du syllogisme ci-dessus, si l'apprenant considère qu'il ne peut pas conclure parce qu'il doute de la véracité de la règle, le tuteur lui affichera un message tiré de la banque des messages lorsque échec et ajoutera à cela un commentaire spécifique à ce type d'erreur, qui est dans ce cas « Tu ne devrais jamais douter de la véracité de la règle ». L'implémentation de ces règles sera vue en détail dans le prochain chapitre. La construction des interactions du tuteur s'est faite sur la base d'un Framework (Narciss, 2008; Narciss & Huth, 2004) dédié à la mise en place de feedbacks plus élaborés.

4.3.3. Modélisation du domaine

Nous avons présenté au 0, un ensemble de règles valides qu'il fallait maîtriser pour aspirer à devenir un expert raisonneur. Ces règles sont exprimées sous forme d'expressions logique. Nous avions donc besoin de développer un algorithme capable à partir d'une situation de raisonnement, de transformer un syllogisme en proposition logique. Les marqueurs (si…alors, non, ou, etc.) sont identifiés dans le syllogisme grâce au contrôleur général qui déduit alors de quel type de logique et de quel type de raisonnement il s'agit. Lorsque l'apprenant répond à un exercice, le tuteur envoie le même exercice à l'expert qui le transforme en expression logique sur lequel il applique

la règle de raisonnement valide correspondant. L'expert revoit alors la conclusion valide de l'exercice au tuteur qui la compare avec celle de l'apprenant.

Hormis le fait pour le système de pouvoir raisonner en se basant sur des règles de raisonnement valides, il doit aussi être capable de reconnaître les erreurs de raisonnement. Cette fonction est remplie grâce au catalogue des erreurs. Le catalogue des erreurs est une banque contenant toutes les règles de raisonnement invalides (sophismes) ainsi que les suppressions d'inférences. Ces règles erronées tout comme celles valides, sont également codées dans Muse-Logique et accessibles en tout temps lorsque le besoin se présente. Lorsqu'un apprenant rate un exercice, le tuteur interroge l'expert pour savoir de quel type d'erreur il s'agit et comment y remédier. L'expert consulte donc son catalogue d'erreurs pour faire matcher la réponse de l'apprenant à l'une des règles erronées. C'est la réponse de l'apprenant transformé en expression logique par notre algorithme, qui est alors comparé avec l'ensemble du catalogue des erreurs de la logique sélectionnée par le contrôleur général.

Chacune des règles du domaine est documentée afin de pouvoir fournir au tuteur les explications dont il aura besoin pour corriger les erreurs de l'apprenant. Cependant, ces règles du domaine ne représentent que le côté syntaxique de la chose. Si nous les considérons uniquement, alors nous concevons un outil génial pour l'application de règles sans toutefois comprendre et exploiter les liens sémantiques existant entre les concepts en jeu. C'est pour cette raison, que notre système est doté d'une ontologie (Vanlehn, 2006) du domaine. Une ontologie est comme un dictionnaire doté de liens sémantiques entre les éléments le composant. Scientifiquement parlant, c'est un ensemble de termes et de concepts organisés dans un graphe dont les relations entre ces différents concepts peuvent être des relations sémantiques ou de subsomption. L'ontologie de Muse-Logique a été conçue grâce notamment à nos experts en logique et en systèmes experts. Cette ontologie peut être visualisée à la Figure V.4 du présent mémoire.

4.4. Déroulement du projet Muse-Logue

4.4.1. Méthodologie du projet Muse

La gestion de projet est une démarche assurant le bon déroulement du projet avec une planification (présentée à l'annexe C) et une gestion des différentes tâches. Pour la gestion de notre projet, nous avons opté pour une méthodologie agile appelée SCRUM (Rialle, 1996). SCRUM est un processus de développement d'application informatique, qui permet une certaine flexibilité dans l'exécution du projet. Muse-Logique est un projet de recherche, le changement de besoins fonctionnels est fréquent, il y a de nouveaux artefacts qui entrent, qui doivent être modifiés voire même supprimés, au fur et à mesure que la recherche avance. Il est donc crucial de pouvoir tenir compte et gérer ces instabilités, d'où l'approche agile. Les méthodologies agiles favorisent l'adaptation au changement plutôt que la conformité aux plans.

Scrum est une méthode Agile qui permet de produire la plus grande valeur métier dans la durée la plus courte (Eduscol). Le cycle de vie de SCRUM est présenté à la Figure IV.10. Les étapes de SCRUM dans un ordre chronologique sont présentées ci-dessous.

- Construction de la backlog : La backlog est considérée comme le squelette du projet. Il spécifie toutes les tâches à exécuter. Par exemple, dans notre backlog nous pouvons trouver les tâches telles que la « construction de la structure du réseau bayésien », la « spécification des règles tutorielles » etc.
- Définition des priorités pour chaque tâche de la backlog.
- Spécification des sprints : un sprint est un ensemble de sous tâches de la backlog. Ces tâches sont choisies de manière à fournir une partie fonctionnelle

du projet à la fin du sprint. Ces sprints sont généralement d'une durée de 3 à 4 semaines.

- Clôture du sprint en cours : à chaque fin de sprint, tous les membres de l'équipe peuvent voir le fonctionnement du produit courant et décider soit de le valider dans cet état, soit de continuer à l'améliorer pendant un sprint supplémentaire.

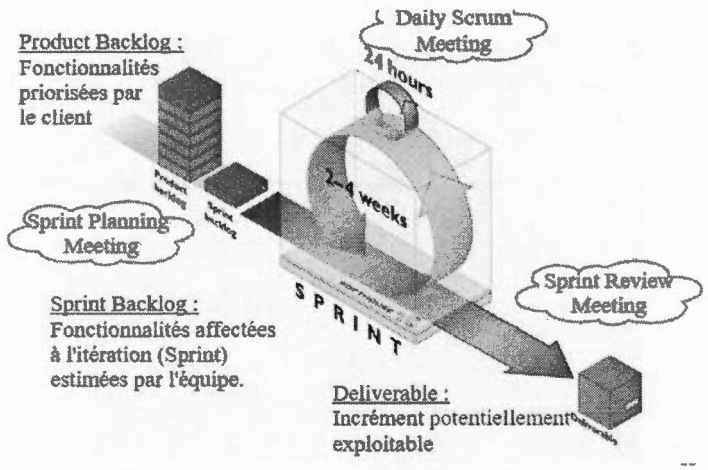


Figure IV.10 : cycle de vie du SCRUM (Schwaber, 1997, 2004)

L'équipe étant multidisciplinaire, nous avons fait appel à nos ressources en ingénierie et aux bonnes techniques d'application de SCRUM (Schwaber) pour pouvoir mettre en pratique la méthode. Après construction et validation de la backlog, nous avons produit la liste des sprints et des différents jalons pour le développement du premier prototype complet de Muse-Logique. L'annexe D présente les détails des différents sprints.

4.4.2. Approche participative

La richesse et l'expertise du STI Muse-Logique sont dues principalement au fait qu'il a été développé suivant une approche participative. On entend par approche participative, une implication active des acteurs du projet aux processus décisionnels

(planification, mise en œuvre et évaluation). L'équipe projet de Muse-Logique est une équipe multidisciplinaire dans laquelle nous retrouvons un peu de tout comme, des experts du domaine de la logique, des cogniticiens, des experts dans le domaine des STI, des ingénieurs, des développeurs, etc. Cette pluridisciplinarité a un effet très positif sur le système. Toutes les connaissances du domaine ont été élicitées, documentées de façon minutieuse et validé au sein de l'équipe. Toutes les techniques utilisées pour la construction du STI ont été également étudiées et validées. La méthodologie agile choisit, a favorisé cette approche participative puisque, elle met l'accent sur la collaboration et privilégie les individus et leurs interactions par rapport aux processus et aux outils (Eduscol).

Rappelons que l'objectif de ce chapitre était de présenter l'architecture et les éléments conceptuels du STI Muse-logique tout en prouvant que cette conception permet de produire un système générique. Nous avons vu que le modèle de l'apprenant est conçu avec un réseau bayésien. Quant au modèle du tuteur, il est principalement conçu grâce aux règles tutorielles qui ont été dument étudiées, validées par les experts pédagogues et peaufinées grâce aux techniques générales de construction de feedbacks dans les technologies éducatives. Le modèle de l'expert qui est considéré comme le cœur de ce système, est conçu grâce aux règles de raisonnement valides, au catalogue d'erreurs et à une ontologie du domaine. Tous ces composants sont mis ensemble et communiquent afin d'obtenir le système escompté, capable de raisonner, expliquer son raisonnement, faire apprendre à raisonner, diagnostiquer les erreurs des apprenants et appliquer des techniques remédiatives à ces erreurs. Nous n'oublions pas de mentionner qu'il y a eu un réel challenge lors de l'établissement de l'architecture du système, celui de pouvoir construire un système générique. Cependant, les techniques, les outils choisis et bien sûr l'expertise de l'équipe a permis de construire un modèle expert non seulement intelligent mais aussi extensible et générique pour tout autre forme de raisonnement ou type de logique.

Nous avons passé en revue les détails conceptuels. Dans le prochain chapitre nous présenterons les détails d'implémentation de Muse-Logique ainsi que les résultats de cette implémentation, en s'attardant sur les moyens utilisés pour l'implémentation des différents éléments qui entre dans le fonctionnement du système.

CHAPITRE V

IMPLEMENTATION DE MUSE-LOGIQUE

Après une présentation détaillée de l'architecture et des éléments de conception de Muse-Logique pour l'apprentissage du raisonnement logique, une question qui se pose est de savoir comment techniquement tout est pris en charge. Tout ce qui est exposé dans le présent chapitre est purement technique et vise à fournir les détails d'implémentation du système. À cet effet, nous nous attellerons à présenter dans un premier temps les choix techniques pour l'implémentation. Ensuite, nous exposerons les principaux algorithmes que nous avons développés pour le fonctionnement du système et enfin nous présenterons les résultats d'implémentation.

5.1. Choix techniques d'implémentation de Muse-Logique

5.1.1. Java EE pour l'interface

Java EE (*Java Enterprise Edition*) est une plate-forme qui permet de faciliter le développement d'applications d'entreprise en fournissant un ensemble de 22 composants sous forme d'APIs . Puisque nous avons opté pour le développement d'une interface web, nous n'utiliserons que la partie application web de JEE. Une interface web pour Muse-logique car, elle offre une meilleure accessibilité, une gestion centrale plus facile et de plus, les utilisateurs n'auront pas à installer quoi que ce soit. Les applications web JEE ont une architecture en couches comme le montre la Figure V.1.

Le client web représente l'interface web de l'application. Le serveur web englobe les services, l'accès aux données et la persistance. Le dernier élément constitue la base donnée de l'application. L'IDE choisit pour le développement est Netbeans et le serveur d'applications Apache Tomcat.

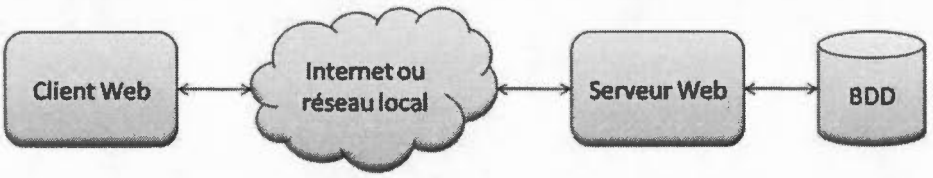


Figure V.1 : Architecture des applications Web JEE

Avec JEE, les pages web sont des JSP. Des JSP sont des pages spécifiques au JEE, qui sont à la base des pages HTML, mais dans lesquelles on peut insérer du code JAVA. Pour une expérience utilisateur de haut niveau et une convivialité meilleure, nos interfaces ont été développées en utilisant JavaScript et la bibliothèque Bootstrap. Twitter Bootstrap est une collection d'outils utile à la création de sites et d'applications web. C'est un ensemble qui contient des codes HTML et CSS, des formulaires, boutons, outils de navigation et autres éléments interactifs. Actuellement, c'est l'un des outils les plus populaires pour le développement d'interface web. La Figure V.2 présente l'interface d'accueil de Muse-Logique. Les services (connexion, interactions apprenant-tuteur, etc.) quant à eux, sont codés en java, et la base de données est gérée avec MySQL.

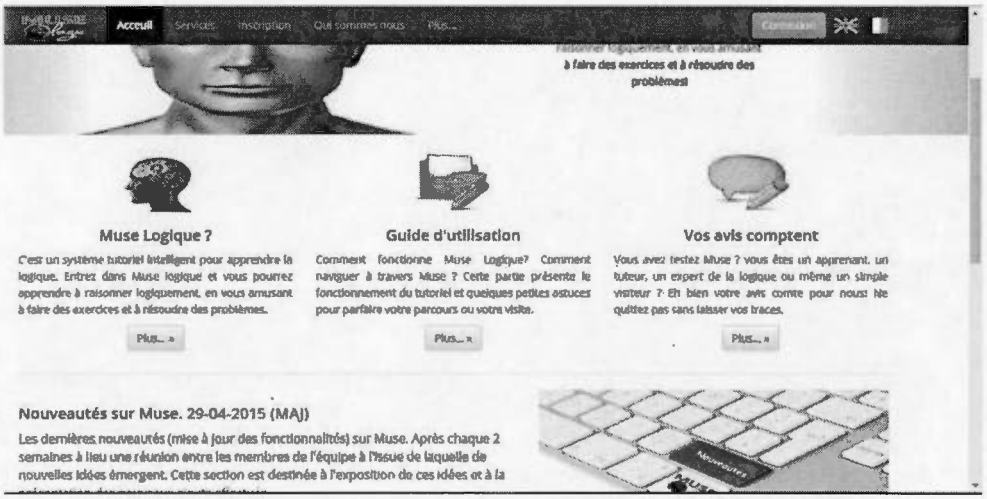


Figure V.2 : Une partie de l'interface d'accueil dans Muse-Logique

5.1.2. JESS pour les règles

Dans cette section, nous accorderons de l'importance à comment nos règles sont implémentées en JESS et comment la communication est établie entre elles et l'application web. Un moteur de règles contient un moteur d'inférence et un système de représentation des connaissances. Il existe plusieurs moteurs de règles dont les plus connus sont Prolog, Clips et JESS. JESS (JESS) pour *Java Expert System Shell,* est un moteur de règle intégré à la plate-forme Java. Il permet de construire des systèmes d'aide à la décision, des systèmes de capitalisation des connaissances ou encore des outils d'exploration des mécanismes du raisonnement (Amiguet). Nous l'utilisons pour construire un système capable de raisonner à partir de règles que nous avons défini plus tôt dans le CHAPITRE II. Nous l'avons choisi car il est facilement intégrable à toute application codée en Java.

Toujours en rappel de ce que nous avons présenté au CHAPITRE II, il existe les règles valides et invalides dans le raisonnement logique. Cependant l'accent n'a pas été mis sur le fait qu'il existe des erreurs typiques et atypiques de raisonnement. Dans notre

banque d'exercices, les syllogismes se présentent sous formes de questions à choix multiples.

Exemples : Extrait de la banque d'exercices de Muse-Logique

1-(MPP) **Règle** : Si on lance une roche dans une fenêtre, alors la fenêtre brisera.

Information : On a lancé une roche dans la fenêtre.

On peut conclure que :

☐1 La fenêtre brisera. (*Bonne réponse typique*)

☐2 La fenêtre ne brisera pas. (*Mauvaise réponse atypique*)

☐3 On ne peut conclure si la fenêtre brisera ou non. (*Mauvaise réponse typique*)

2-(AC) **Règle** : Si on lance une roche dans une fenêtre, alors la fenêtre brisera.
Information : La fenêtre est brisée.

On peut conclure que :

☐1 On a lancé une roche dans la fenêtre. (*Mauvaise réponse typique*)

☐2 On n'a pas lancé de roche dans la fenêtre. (*Mauvaise réponse atypique*)

☐3 On ne peut conclure si on a lancé une roche dans la fenêtre ou non. (*Bonne réponse typique*)

- Si réponse # 3 : Pourquoi avez-vous répondu qu'on ne peut conclure si on a lancé une roche dans la fenêtre ou non?

  ☐1 Ce n'est pas certain que si on y lance une roche, alors la fenêtre brisera.

  ☐2 La fenêtre peut être brisée pour d'autres raisons que d'y avoir lancé une roche. (*Bonne réponse*)

  ☐3 Autre.

Tableau V.1 : Implémentation sous JESS de quelques syllogismes implicatifs

MPP Valide	((p → q) & p) →q	(defrule MPPvalide ?problem <- (Problemes (operateur ?op) (consequent ?q) (antecedent ?p) (information ?p)) ?reponse <- (Reponse (reponse ?x)) (test (eq ?op "Implication")) => (modify ?problem (conclusion ?q)) (modify ?reponse (reponse "valide") (typeraisonnement "Implicatif") (moderaisonnement "MPP") (conclusion ?q)))
AC valide - Catalogue des erreurs	((p → q) & q) →On ne peut conclure	(defrule ACon_ne_peut_conclure ?problem <- (Problemes (operateur ?op) (consequent ?q) (antecedent ?p) (information ?q)) ?reponse <- (Reponse (reponse ?x)) (test (eq ?op "Implication")) => (modify ?problem (conclusion "On ne peut pas conclure")) (modify ?reponse (reponse "valide") (typeraisonnement "Implicatif") (moderaisonnement "AC") (conclusion "On ne peut pas conclure")))
MPP erroné Atypique - Catalogue des erreurs	((p → q) & p) →¬q	(defrule MPP_erreur_atypique ?problem <- (Problemes (operateur ?op) (consequent ?q) (antecedent ?p) (information ?p) (conclusion "not" ?q)) ?reponse <- (Reponse (reponse ?x)) (test (eq ?op "Implication")) => (modify ?reponse (reponse "invalide") (typeraisonnement "Implicatif") (moderaisonnement "MPP") (conclusion "atypique")))
MPP erroné Typique - Catalogue des erreurs	((p → q) & p) →on ne peut conclure	(defrule MPP_erreur_typique ?problem <- (Problemes (operateur ?op) (consequent ?q) (antecedent ?p) (information ?p) (conclusion "On ne peut pas conclure")) ?reponse <- (Reponse (reponse ?x)) (test (eq ?op "Implication")) => (modify ?reponse (reponse "invalide") (typeraisonnement "Implicatif") (moderaisonnement "MPP") (conclusion "typique")))

Il y a deux types de mauvaises réponses (mauvais raisonnement) qui devront être pris en compte dans le codage des règles non valides. Les mauvaises réponses typiques correspondent aux règles invalides présenter plus haut alors que les mauvaises réponses atypiques sont des erreurs qui n'ont pas encore d'interprétation scientifique (généralement commis par erreur ou par gaming). Le Tableau V.1 illustre comment les règles sont codées sous JESS.

Le fait que JESS soit conçu pour fonctionner principalement avec les applications JAVA nous facilite la connexion entre Muse-logique application web et nos règles du domaine. Les étapes techniques effectuées pour donc faire communiquer notre application et JESS sont les suivantes :

- écriture des règles (valides et catalogues d'erreurs) dans des fichiers d'extensions « .clp »,
- création d'une variable contenant les chemins d'accès aux fichiers dans le code source de l'expert,
- importation de la librairie JESS,
- initialisation du « RETE », moteur d'inférence de JESS (Figure V.3 ligne 1),
- indexer les fichiers contenant les règles pour que JESS les trouvé (Figure V.3 ligne 3),
- injecter les faits dans la base de connaissances (Figure V.3 lignes 4 et 5),
- exécuter le moteur d'inférence à partir du code java « rete.run » (Figure V.3 lignes 6).

Les détails de ces manipulations sont disponibles dans la documentation officielle de l'outil JESS à l'adresse (JESS).

```
Rete engine = new Rete();
engine.reset();
engine.batch(nom);
engine.add(p);
engine.add(r);
engine.run();
```

Figure V.3 : Appel de JESS dans un programme JAVA

Le système est ainsi capable à partir d'une formule logique de déduire si elle est valide ou pas, de tirer une conclusion valide, de donner le mode et le type de raisonnement dont il est question et de dire si elle est erronée ou pas et pourquoi. Une

question fondamentale qui se pose ici, est de savoir comment le système passe d'un syllogisme écrit en langage naturelle pour obtenir une forme propositionnelle qui sera évalué par l'expert ? La section 5.2.2 apporte une réponse.

5.1.3. Protege pour l'ontologie

L'ontologie définit formellement les termes et concepts employés pour décrire et représenter un domaine de connaissance (Tchouanto, 2014). En rappel au 0, ces concepts sont organisés dans un graphe dont les relations peuvent être des relations sémantiques ou des relations de subsomption. Nous avons besoin d'un tel outil dans Muse-logique pour comprendre et exploiter les liens sémantiques existant entre les concepts en jeu dans le raisonnement logique et la logique en général.

Il existe une panoplie d'outils de création d'ontologie dont Protégé . Protégé est un système auteur pour la création d'ontologies. Il a été créé à l'université Stanford et est très populaire dans le domaine du Web sémantique et au niveau de la recherche en informatique . L'ontologie dans Muse-logique a été développée avec cet outil et grâce à l'aide des experts. La Figure V.4 présente cette ontologie qui est accessible aux apprenants et aux tuteurs via l'interface web de l'application. On peut constater que tout en haut de la chaîne, nous avons la logique qui est lié à plusieurs sous-logiques dont la logique classique qui est l'objet de ce premier prototype de Muse. Cette ontologie a été générée en format .owl puis transformer en fichier JSON (*JavaScript Object Notation*) afin d'être exploitée dans l'interface graphique grâce à la librairie d3.js .

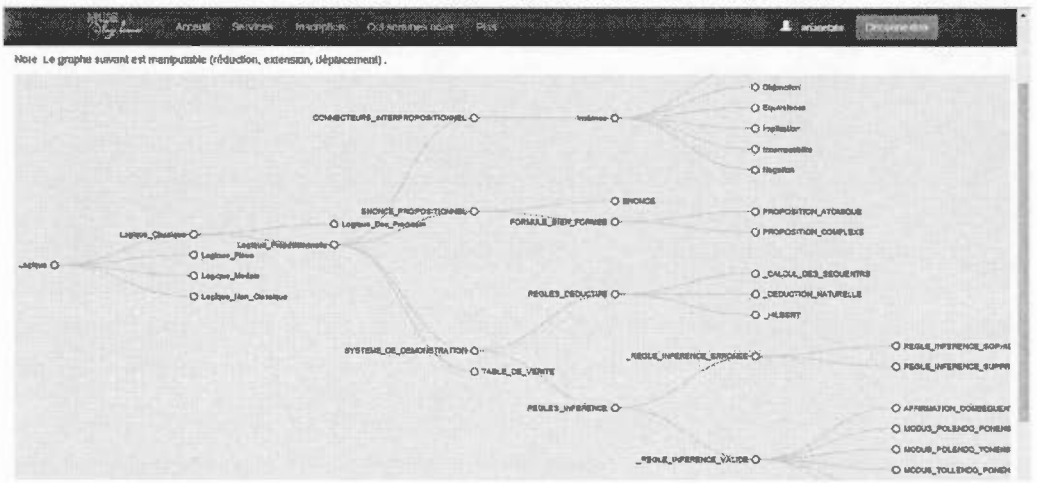


Figure V.4 : Visualisation de l'ontologie dans Muse-logique

5.1.4. CDM pour l'initialisation du modèle de l'apprenant

Pour que le système puisse se faire une idée initiale sur les connaissances d'un apprenant qui se connecte pour la première fois, il doit lui faire passer un test de niveau. Nous avons dit au 0 que la dimension psychométrique devrait nourrir cette initialisation du modèle cognitif de l'apprenant. Le modèle psychométrique intégré au modèle de l'apprenant est entrainé à partir des données mais il évoluera au fur et à mesure qu'on disposera de données plus importantes. Ces données étant essentiellement les réponses des étudiants aux 'items' (48) du prétest. Cette dimension psychométrique est supportée par la librairie CDM (*Cognitive Diagnostic Models*). Comment cela se passe-t-il concrètement ? Cette question a fait l'objet d'un stage se déroulant au sein de l'équipe dont le rapport est disponible pour plus de détails (Carolane, 2015).

L'utilisation du réseau bayésien limite l'individualisation lors de l'initialisation du réseau. En effet, on module seulement la probabilité a priori. Les probabilités a postériori étant dépendantes des parents il serait compliqué de les modifier pour chaque apprenant. De ce fait, les données du modèle psychométrique doivent être retravaillées

pour passer de probabilités pour chaque compétence à une probabilité à la compétence en raisonnement à l'implication. Le choix est fait sur le nœud racine car nous ne pouvons pas initialiser n'importe quel nœud avec une valeur précise. Nous pouvons uniquement mettre un nœud à vrai ou à faux, mais il n'est pas possible de dire qu'un nœud est vrai à 50% par exemple car les tables de probabilités conditionnelles des nœuds ont plus que 2 entrées (ils sont dépendants de leurs parents). Or, le nœud racine lui, n'a que deux entrées dans sa table de probabilités à priori (probabilité qu'il soit vrai et probabilité qu'il soit faux), car il n'a pas de parents. Donc le seul nœud sur lequel nous pouvons agir est ce dernier car il ne dépend pour l'instant d'aucuns autres. Nous pouvons dire qu'il est vrai à 50% en modifiant sa table de probabilité à priori.

Ainsi, pour l'initialisation du modèle de l'apprenant, nous prédisons sa probabilité de maîtrise de la compétence globale à l'implication (nœud racine) à travers ses résultats au pré-test. Nous recherchons à partir d'un pattern de réponse d'un étudiant la ligne de la matrice posterior contenant ce pattern ou bien un pattern proche. La Figure V.5 (Christophe, 2015) illustre en détail un exemple du processus d'extraction de la probabilité jointe de maîtrise de la compétence à l'implication à l'aide du Posterior. Cette probabilité jointe que nous fournit le CDM et qui est calculée à partir des probabilités associées à chaque élément de connaissance du domaine, est utilisée comme probabilité à priori de maîtrise du raisonnement conditionnel (la compétence ultime visée par cette première version de MUSE-Logique). En clair, cette probabilité sera affectée comme valeur de la probabilité a priori du nœud racine du réseau bayésien comme expliqué ci-haut. Par exemple, si la probabilité jointe (probabilité à l'implication) est 0.6 pour la maîtrise de la compétence, cela veut dire que l'apprenant maîtrise à 60 % le raisonnement implicatif et ne la maîtrise pas à 40%. Cette initialisation est donc distribuée sur le réseau grâce aux liens de causalité qui existent. Ainsi, Muse-Logique travaillera à faire évoluer l'apprenant en essayant d'améliorer cette probabilité de maîtrise initiale.

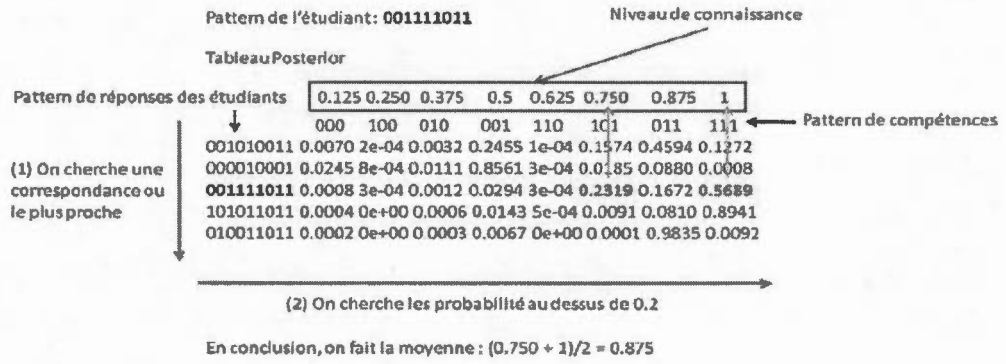


Figure V.5 : Exemple d'extraction de la probabilité d'initialisation à l'aide du prosterior

En résumé, l'initialisation du modèle de l'apprenant a nécessité l'utilisation de RStudio et de la librairie CDM. Ce dont nous avons besoin pour l'initialisation du réseau bayésien (pour le raisonnement implicatif) est la probabilité que l'apprenant lors de sa première authentification, maîtrise ce type raisonnement. Cette information est obtenue grâce au CDM, à partir d'une matrice DINA (qui contient en ligne des étudiants et en colonnes des exercices) et de la Q-Matrix (qui nous le rappelons contient en ligne exercices et en colonne des compétences). Le logiciel R a servi de support technique à toutes les manipulations nécessitant le CDM.

5.1.5. Hibernate et MySQL

L'architecture de la base de données qui sera utilisée dans Muse-logique a été présentée à la section 4.2 du chapitre IV. Techniquement, cette base sera gérée avec MySQL à travers le logiciel WampServer. WampServer met à disposition une base de données MySQL ainsi qu'une interface Web permettant de naviguer dans le contenu des bases. Le pont entre la BD et l'application, et la représentation de cette base dans notre application JEE s'est fait par le biais de l'outil hybernate. Hibernate est un open

source de type ORM (*Object Relational Mapping*) qui permet de représenter une base de données en objets java et vice versa. Grâce à cet outil, nul besoin d'être un expert en requête SQL pour pouvoir extraire des données d'une BD. Hibernate facilite la persistance et la recherche de données dans une base de données en réalisant lui-même la création des objets et les traitements de remplissage de ceux-ci en accédant à la base de données. La quantité de code ainsi épargnée est très importante d'autant que ce code est généralement fastidieux et redondant . Ce choix technique nous a donc permis de diminuer le temps de développement de Muse-Logique.

5.2. Algorithmes utilisés

Dans cette section, nous présentons les algorithmes utilisés dans le développement de Muse-Logique. Certains algorithmes ont été développés par nous-mêmes et d'autres, empruntés à la littérature puis adaptés.

5.2.1. Algorithme d'extraction de marqueurs pour les connecteurs

Muse-Logique se veut être un système générique pour tous types de raisonnement et de logique. À cet effet, l'expert doit être capable de reconnaître le type de raisonnement et ou de la logique en jeu avant tout raisonnement. Il doit également être capable de reconnaître les connecteurs présents dans le syllogisme. Pour une première version de Muse-logique, il nous fallait doter le système d'une capacité à reconnaître les types de syllogismes ou de raisonnement (implicatif, disjonctif, conjonctif) en jeu et les connecteurs présents dans les syllogismes. Heureusement, avec l'aide des experts du domaine, nous avons pu collecter une grande partie des marqueurs identifiant les types de syllogismes et les opérateurs. En rappel, nous en avons vu au premier chapitre à l'instar de « Si…alors » « et » qui définissent les marqueurs des

syllogismes d'implication et de conjonction respectivement et le « ne pas » qui définit le connecteur négation.

Le système doit être capable de dire que le syllogisme numéro 1 de l'exemple à la section 5.1.2 est un syllogisme de type implicatif, car il possède comme marqueurs « Si…alors ». Également, si l'on considère l'information « On n'a pas lancé de roche dans la fenêtre », Muse-logique doit être capable de déceler la négation en identifiant le marqueur « n'…pas ». Le Tableau V.2 présente l'ensemble des marqueurs que nous utiliserons et les opérateurs auxquels ils correspondent. Les lignes 1, 3 et 5 représentent les opérateurs et les autres représentent les marqueurs. L'algorithme développé pour ce besoin utilise la librairie Regex de java pour le parcours de chaînes de caractères. Le pseudo code est le suivant :

Fonction *obtenir_operateur (phrase)*
{
Si *phrase contient marqueurs implication* ***alors retourner*** *Implication;*
Sinon *si phrase contient marqueurs conjonction* ***alors retourner*** *conjonction;*
Etc...
}

Tableau V.2 : Quelques marqueurs et opérateurs de la logique classique

NÉGATION ($\neg$P)	CONJONCTION (P&Q)	DISJONCTION INCLUSIVE (P $\vee$ Q)	
- non … - ne … pas - n'… pas - Ce n'est pas le cas que…	- …et… -…et également… -… et aussi… -…, mais … -…, par contre … -…, par ailleurs … -…, cependant …	-… et/ou … -… ou …	
INCOMPATIBILITÉ (P | Q)	TAUTOLOGIE (P$\top$Q)	DISJONCTION EXCLUSIVE (P $\vee\vee$ Q)	
-… est incompatible avec … -…n'est pas compatible avec … -Pas … ou pas …	-Il y a une tautologie entre … et … -Il y a une loi logique entre … et …	Ou bien …, ou bien …	
IMPLICATION	AFFIRMATION	EQUIVALENCE	CONTRADICTION (P$\perp$Q)
-Si…, alors … -…si … -…suffit pour … -…est la condition suffisante pour … -… est la cause de … -… seulement si … -… donc…	P (Q) : P peu importe ce qui arrive à Q -…quelle que soit la valeur de … $\neg$P (Q) : non-P peu importe ce qui arrive à Q -Non… quelle que soit la valeur de …	-…est équivalent à … -…si et seulement si … -…est nécessaire et suffisant pour … -…est la condition nécessaire et suffisante pour …	-il y a une contradiction entre … et … -… et … se contredisent

5.2.2. Algorithme de transformation de syllogisme en expression logique

Après avoir décelé le ou les connecteurs présents dans un syllogisme et déterminer le type de syllogisme en jeu, Muse-Logique doit maintenant transformer ce syllogisme en proposition logique. Cette transformation est nécessaire pour entamer le processus de raisonnement, car les règles sont exprimées en formules logiques. Pour ce faire, il faudra extraire les parties du syllogisme à savoir l'antécédent, le conséquent et ce à quoi réfère l'information (affirmation ou non de l'antécédent ou du conséquent).

Pour plus de clarté, reprenons l'exemple 1 de la section 5.1.2 et essayons de le transformer en expression logique.

Règle : Si on lance une roche dans une fenêtre, alors la fenêtre brisera.

Information : On a lancé une roche dans la fenêtre.

Après passage dans l'algorithme d'extraction de marqueurs, le système déterminera que ce syllogisme est de type implicatif. Après cette étape, il passe à l'algorithme de transformation qui consiste à extraire les propositions P et Q. Dans l'exemple, entre le « Si » et le « alors » on a P = on lance une roche dans une fenêtre. Après le « alors » on a Q= la fenêtre brisera. Il reste l'identification de l'information. Cette dernière étape est une opération très complexe parce que, l'information n'est pas toujours forcément égale mot pour mot à P ou Q car il y a les temps des verbes qu'il faut prendre en compte. Nous humains, voyons que l'information de l'exemple est l'affirmation de la première partie de la majeure, l'antécédent (P). Si l'information était « on n'a pas lancé de roche dans une fenêtre » alors il serait une négation de l'antécédent. Comment doter le système d'une telle capacité?

Nous avons développé et testé différentes approches et nous avons finalement choisi de faire la racinisation (racine d'un mot) qui a fourni un résultat concluant. La racinisation consiste en la suppression du préfixe et du suffixe d'un mot afin d'en obtenir sa racine. Prenons l'exemple des mots lancé et lance, après racinisation, on obtiendra lanc et lanc qui sont identiques. Avec cette méthode, le système en comparant la racinisation de tous les mots de l'antécédent et du conséquent avec ceux de l'information, sera à mesure d'identifier l'appartenance de l'information. Lorsque les comparaisons donnent des résultats égaux, c'est le dernier mot des parties du syllogisme qui feront la différence (Voir ci-dessous le pseudo code de l'algorithme). Le pionner de la racinisation est Porter, qui dans les années 80 avait développé un algorithme permettant de supprimer les suffixes des mots dans la langue anglaise (tartarus). Les années ont évolué et les versions améliorées de cet algorithme ont vu le

jour, tant en français qu'en d'autres langues. Celui que nous utilisons est *Snowball stemmer Français* (tartarus). L'appel de l'algorithme sur le mot 'continuerait' a comme résultat 'continu '. La librairie regex est aussi utilisée pour la comparaison des mots dans des chaînes de caractères.

Ainsi, la transformation d'un syllogisme en formule logique suit les étapes suivantes :

- extraction des parties du syllogisme,
- racinisation de ces parties si il ne s'agit pas d'un contenu abstrait ou symbolique,
- comparaison mots à mots de l'information avec l'antécédent et le conséquent,
- identification de l'information,
- déduction de l'expression logique.

Ces étapes se traduisent par le pseudo code de l'algorithme de transformation de syllogisme en formule logique suivant :

```
FONCTION chaîne Transformer_Syllogisme (règle, information)
{
 Si règle.Marqueurs() est l'implication alors
  Si règle.Marqueurs() égale à "si...alors" alors
    p= extraire partie de la règle entre "si" et "alors"
    q= extraire partie de la règle après "alors"
  Sinon si règle.Marqueurs() égale à "...seulement si..." alors
    p= extraire partie de la règle avant "seulement si"
    q= extraire partie de la règle après "seulement si"
        etc.
        ...
    n1= Comparer (p, information)
    n2= Comparer (q, information)
    Si n1> n2 ou p.dernierMot()= information.dernierMot() alors
```

```
        Si information.Marqueurs() égale à la négation alors
            retourner ((p → q) & ¬p)
        Sinon
            retourner ((p → q) & p)
    Sinon si n2 < n1 ou q.dernierMot()= information.dernierMot() alors
        Si information.Marqueurs() égale à la négation alors
            retourner ((p → q) & ¬q)
        Sinon
            retourner ((p → q) & q)

  Sinon si règle.Marqueurs est le disjonctif alors
    etc.
}
FONCTION entier Comparer (chaine1, chaine2)
{
  rac_chaine1 = Racinisation(chaine1)
  rac_chaine2 = Racinisation(chaine2)
  compter le nombre de mots qui se ressemblent dans les deux racinisations
  retourner le nombre
}
```

5.2.3. Algorithme de vérification de formes logique

Pour mieux comprendre pourquoi ou comment une conclusion déduite d'une règle et d'une information est vraie ou fausse, il faut parfois passer par l'établissement de la table de vérité. La construction de la table de vérité d'un syllogisme passe par la transformation de ce syllogisme en une formule logique. Puisque le système veut tracer le raisonnement de l'apprenant pour déceler les points à problèmes tout en lui faisant voir ses erreurs, Muse-logique impose la construction d'une table de vérité lorsque nécessaire. C'est donc l'apprenant qui est amené à transformer un syllogisme en

formule logique. Pour que Muse-Logique soit capable de vérifier que cette formule respecte les normes d'énoncés bien formés (respect des parenthèses, de l'ordre des connecteurs, etc.), il faut le doter d'un algorithme capable de vérifier ces formules. À cet effet, nous avons développé un algorithme qui permet de vérifier si un énoncé est un EBF (Énoncé Bien Formé). Les étapes de l'algorithme sont :

- si la formule est de taille 1 et que c'est un caractère alphabétique alors c'est un EBF,
- si la formule est de taille 2 et que le premier caractère est la négation et le deuxième un caractère alphabétique, alors c'est un EBF,
- si la formule est de taille supérieure à 3, respecte l'ordre des parenthèses si il y'en a, toute négation est placée devant une proposition, tout autres connecteurs est entouré de deux propositions et toutes les propositions sont alphabétiques alors, c'est un EBF.

5.2.4. Algorithme de génération de table de vérité

Dans la logique dite "classique", chaque proposition peut être vraie ou fausse. L'attribution d'une valeur de vérité à toutes les propositions d'une formule s'appelle une interprétation de cette formule. On peut en déduire une valeur de vérité pour la formule complète sur la base de tables de vérité.

Comme précisé dans la section précédente, lorsqu'il le jugera nécessaire en particulier en cas d'erreurs de raisonnement, Muse-Logique demandera aux apprenants d'effectuer des tables de vérité de syllogismes. Cette action permettra au système non seulement de comprendre et de tracer la raison des erreurs de l'apprenant, mais permettra également à l'apprenant de comprendre son erreur. L'apprenant est amené dans un premier temps à transformer le syllogisme en formule logique puis à construire la table de vérité associée à ce syllogisme. Après cette tâche, le système corrige la table

et fait comprendre à l'apprenant ses erreurs. Pour pouvoir corriger cette table, encore faut-il que le système puisse la construire lui-même. C'est la raison pour laquelle nous avons développé un algorithme qui à partir d'un EBF, construit sa table de vérité. Les étapes de cet algorithme sont présentées ci-dessous.

- Transformer l'EBF en notation Post Fixé. Exemple : pVq devient p q V.
- Stocker cette notation dans une pile dont le premier caractère est en tête.
- Dépiler jusqu'à l'obtention d'une pile vide.
- Pendant cette dépilation, si le caractère dépilé n'est pas un connecteur, stocker ce caractère dans une autre pile. Si le caractère dépilé est un connecteur, extraire le caractère ou les 2 caractères de la nouvelle pile suivant que le connecteur est la négation ou pas.
- A l'aide de la table de vérité de l'opérateur et les valeurs de vérités possibles des opérateurs concernés, déduire la colonne de la table de vérité correspondant au connecteur. Exemple : au chapitre 1, nous avons vu les tables de vérité de certains opérateurs comme le ou (V), alors la table de vérité de la formule pVq sera celle de l'opérateur (V).
- Le numéro de la colonne résultante est placé en tête de la deuxième pile et est considéré comme une proposition avec ses valeurs de vérité égales à celles calculées précédemment,
- Boucler à partir de la 4ème étape jusqu'à obtenir la première pile vide et la deuxième contenant le résultat final (le numéro de la colonne résultat de la table de vérité globale).

5.2.5. Algorithme de manipulation du réseau bayésien

Le réseau bayésien (voir annexe A) dans Muse-Logique a été créé avec l'outil JavaBayes et est géré par la librairie Jayes (Codetrails). Lorsqu'un apprenant se

connecte pour la première fois, le réseau bayésien est initialisé grâce à ses résultats obtenus lors du test de niveau. Une fois cette étape passée, l'apprentissage peut commencer. Le tuteur sélectionne les exercices en fonction du niveau de compétence de l'apprenant. Par exemple, si l'apprenant possède une faible probabilité de maîtrise de MPP, alors il lui présente les exercices de ce type. Après réponse de l'apprenant, qu'elle soit juste ou fausse, ce résultat est marqué dans le réseau et sera pris en compte lors du choix des prochains exercices et lors des interactions du tuteur avec l'apprenant. Les étapes de l'algorithme permettant la manipulation du réseau bayésien sont :

- si c'est la première connexion de l'apprenant, lui faire passer le test de niveau et initialiser son réseau avec ses réponses,
- si ce n'est pas sa première connexion ou s'il a déjà passé le test de niveau, sélectionner les 3 nœuds « Mode de raisonnement » (MPP en mode concret avec beaucoup d'alternatives, MPP en mode abstrait, etc.) ayant les plus faibles probabilités de maîtrise et en choisir un aléatoirement parmi ces 3,
- présenter à l'apprenant un exercice ayant comme type de raisonnement et mode de raisonnement le nœud choisit précédemment,
- après la réponse de l'apprenant, injecter le résultat dans le réseau et remonter à l'étape 2.

5.3. Résultat de l'implémentation du système

La Figure V.6 présente les services offerts par Muse-Logique. Cette partie est consacrée à la présentation des résultats d'implémentation de ces services.

Figure V.6: Les services de Muse-Logique

5.3.1. Interface Tuteur humain

Dans la version 1 de Muse-Logique, le tuteur humain peut créer deux types d'exercices à savoir les exercices de logique (construction table de vérité, vérification d'EBF, etc. voir Figure V.8) et exercices de raisonnement (voir Figure V.7). Il peut visualiser les exercices créés (voir Figure V.9). Nous mentionnons que derrière cette création d'exercices, il y a un processus de validation. L'expert du domaine valide les exercices saisis par le tuteur avant que le système ne le sauvegarde s'il n'existe pas encore. Par exemple, si le tuteur saisit « pVVq » comme énoncé logique, l'expert lui fera comprendre que ce n'est pas un EBF et ne peut donc être enregistré dans la BD. Cependant, le système laisse la main au tuteur qui a le choix de garder l'exercice ou pas. Le tuteur peut choisir de garder l'exercice pour tester les apprenants sur leur compréhension d'EBF (Énoncé Bien Formé).

Le tuteur peut également gérer les apprenants. Il peut voir les statistiques de sa classe, ajouter des apprenants ou en supprimer etc. Cette partie est prévue pour les

prochaines versions de l'application. La visualisation du domaine est la même que celui de l'apprenant, sauf que le tuteur peut rajouter ou supprimer des explications et commentaires associés.

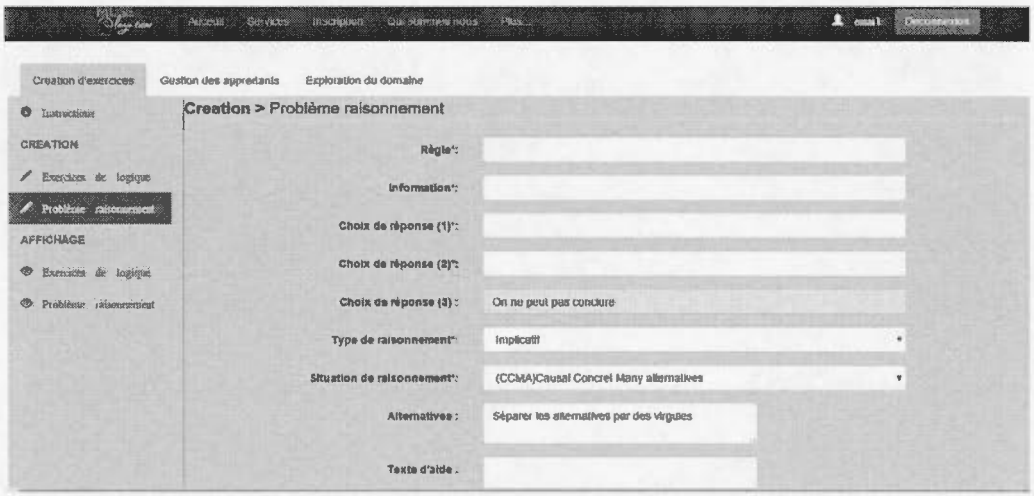


Figure V.7 : Interface du tuteur humain - création de problèmes de raisonnement

Figure V.8 : Interface du tuteur humain - création d'exercices de logique

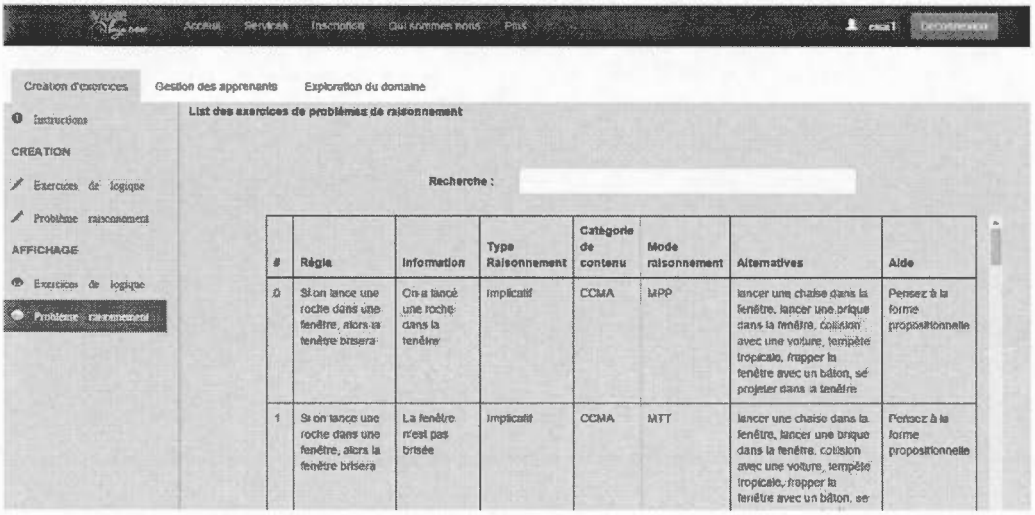


Figure V.9 : Interface du tuteur humain - affichage des problèmes de raisonnement

5.3.2. Raisonnement de l'expert

Grâce à nos architectures, conception et choix techniques, nous avons pu mettre au point un système capable de raisonner et d'expliciter son raisonnement. Le système raisonne non seulement sur les exercices entrés par le tuteur, mais sur n'importe quel exercice qu'on lui propose. Nous avons donné à l'expert le syllogisme suivant :

Règle : Si une personne morp, alors elle deviendra plede

Information : Pierre est devenu plede

La réponse de l'expert à ce syllogisme est présentée à la Figure V.10. Bien sûr, notre expert ne sert pas qu'à raisonner. Il est capable de dire si un raisonnement est erroné ou pas, déduire la forme propositionnelle d'un syllogisme, construire les tables de vérités, etc. Nous n'entrerons pas dans les détails, car une section destinée à l'évaluation de l'expertise de l'expert sera faite dans le prochain chapitre axé sur l'évaluation de Muse-Logique.

```
mode de raisonnement = AC
type de raisonnement = Implicatif
conclusion = On ne peut pas conclure
```

Figure V.10 : Réponse de l'expert à un syllogisme

5.3.3. Interfaces Apprenant

L'apprenant qui est l'utilisateur principal du système, interagit avec Muse-logique par l'intermédiaire d'une interface à travers laquelle il accède à plusieurs services offerts. Lorsque l'apprenant n'est pas à sa première connexion, le système lui propose 2 choix : apprendre avec l'aide du tuteur (mode Tutorat) ou en mains libres sur le type d'exercice que ce dernier veut faire (Figure V.11). Le développement de la 1ere version de notre STI s'est concentré dans un premier temps sur le mode « Tutorat ». Toujours sous forme d'onglet comme celui du tuteur, l'interface en mode tutorat offre 3 services dont la visualisation métacognitive (Treillis de Boole à la Figure V.12) et cognitive (Réseau Bayesien à la Figure V.13), l'exploration du domaine (ontologie) et les exercices (zone d'apprentissage à proprement dite). Il est à noter que le réseau bayésien présenté à l'apprenant est rétroactif dans le sens que l'on peut accéder aux probabilités de maîtrise de chaque nœud en cliquant simplement dessus. La zone exercices (voir Figure V.14, Figure V.15, Figure V.16 et Figure V.17) est la partie principale où l'apprentissage se déroule. La Figure V.14 illustre une situation où l'apprenant a mal répondu à un syllogisme et le tuteur, pour faire comprendre à l'apprenant son erreur, l'amène à construire la forme propositionnelle de l'énoncé et sa table de vérité (Figure V.15). La Figure V.16 illustre une situation où l'apprenant a fait une erreur de raisonnement et le tuteur tente de la rectifier. L'apprenant peut à tout moment consulter ses résultats sous forme statistique (voir Figure V.17). Les exercices de vérifications d'EBF, de construction de table de vérité sont illustrés à l'annexe E, ainsi que d'autres captures d'écran pertinentes de Muse-apprenant.

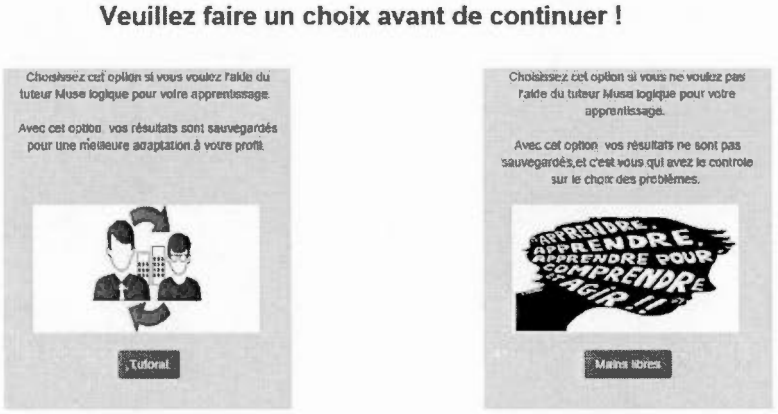


Figure V.11 : Interface de l'apprenant - choix d'apprentissage

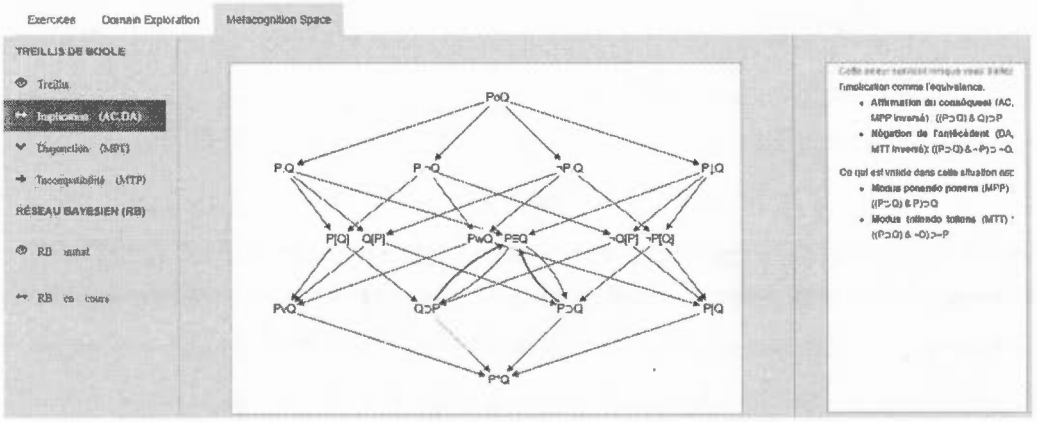


Figure V.12 : Interface apprenant - Visualisation du treillis de Boole

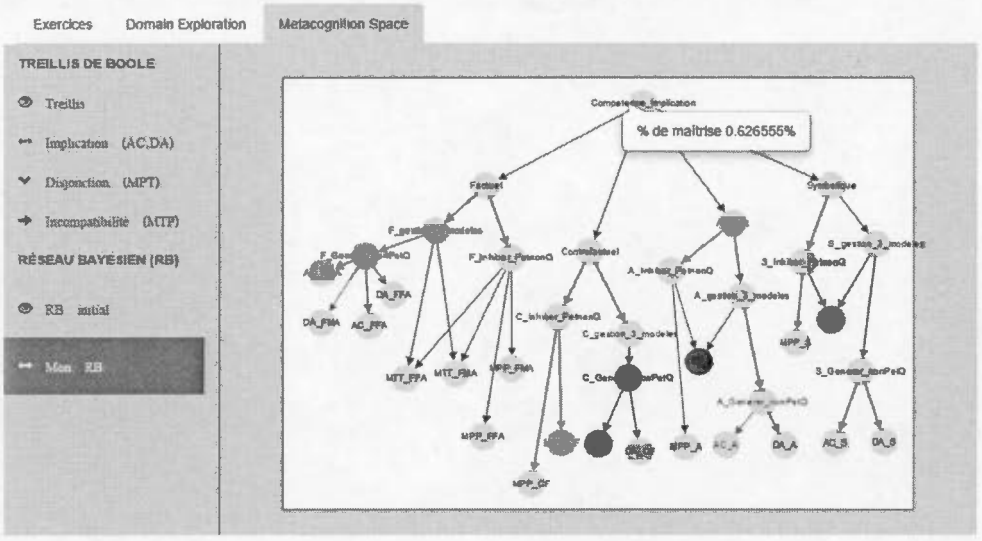


Figure V.13 : Interface apprenant – Visualisation du Réseau bayésien

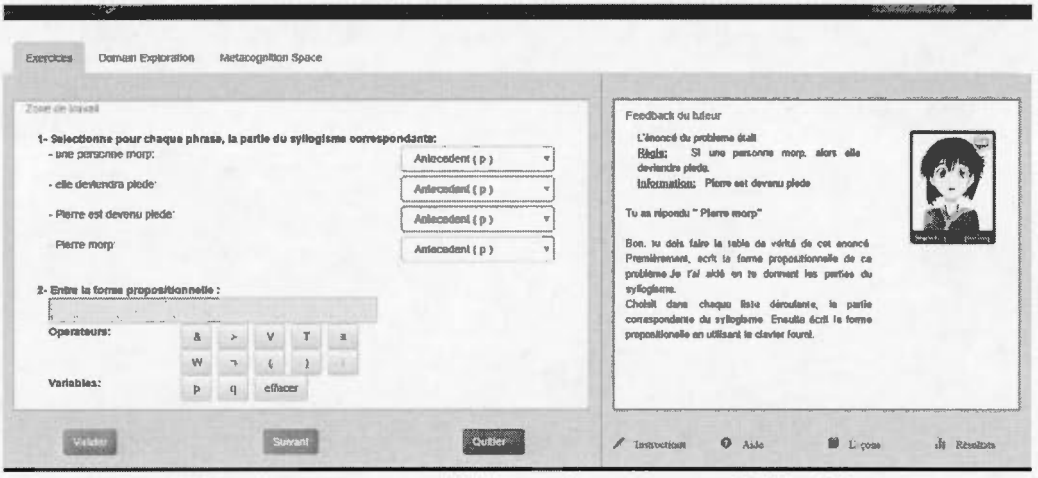


Figure V.14 : Interface apprenant – construction forme propositionnelle

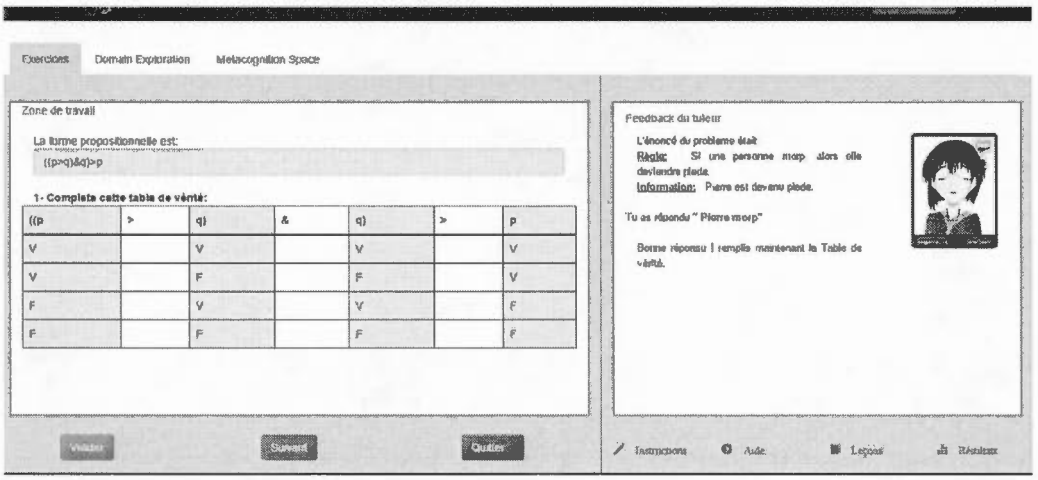


Figure V.15 : Interface apprenant – Construction Table de vérité après mauvaise réponse

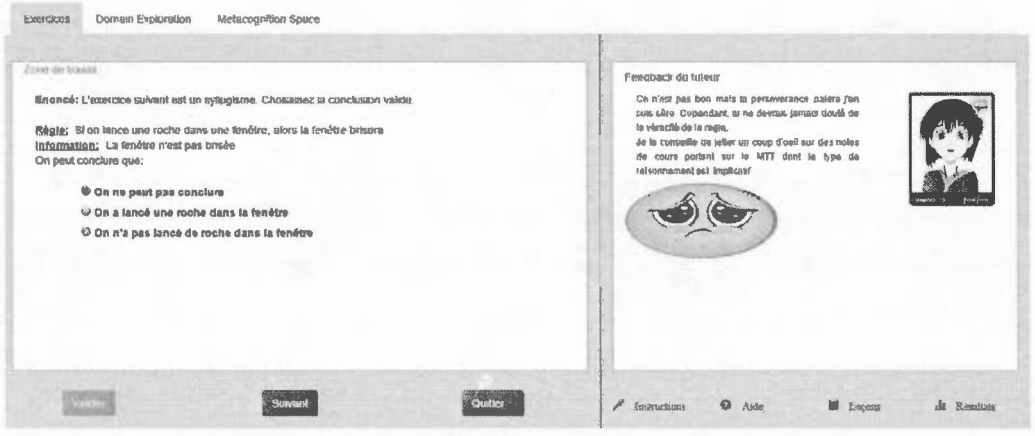


Figure V.16 : Interface apprenant – Rétroaction négatif

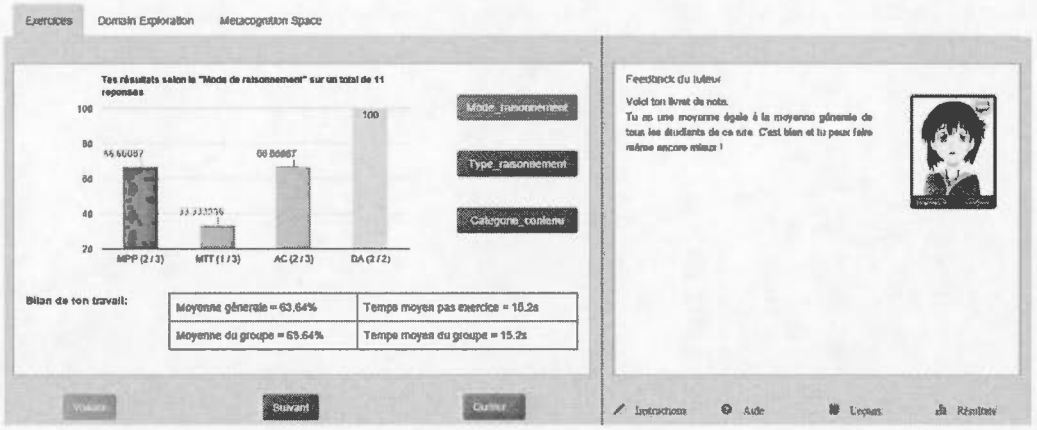


Figure V.17 : Interface apprenant – Visualisation des résultats

5.3.4. Interactions Apprenant-Tuteur

Les interactions du tuteur régissent des actions de l'apprenant. Toute action de l'apprenant est accompagnée d'un feedback du tuteur. Lorsqu'un apprenant fait une mauvaise réponse (voir Figure V.16), le tuteur essaye de lui faire comprendre son erreur et lorsqu'il répond correctement, le tuteur le félicite et l'encourage comme il se doit (voir Figure V.19). Muse-Logique peut également poser des questions pour s'assurer de la compréhension de l'apprenant sur un exercice particulier (voir Figure V.18).

Il est important de rappeler que, les interventions du tuteur ne sont pas toujours les mêmes et les exercices sont présentés à l'apprenant selon son niveau de maîtrise des compétences visées dans le système. De même que dans le cas des exercices, la visualisation des résultats de l'apprenant, de son réseau bayésien et du treillis de Boole sont accompagnés des commentaires du tuteur. Par exemple, la Figure V.20 illustre les commentaires du tuteur après affichage du réseau bayésien de l'apprenant se trouvant

à la Figure V.13 et la Figure V.17 présente l'évaluation des résultats de l'apprenant par le tuteur.

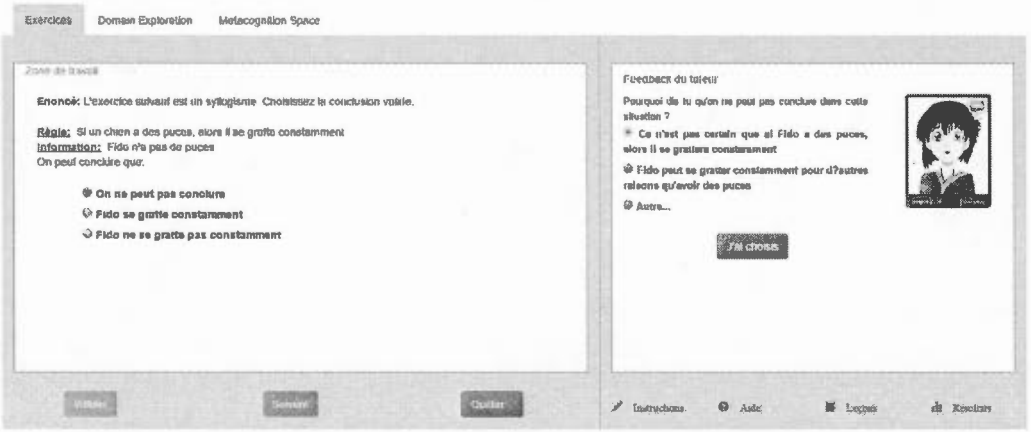


Figure V.18 : Interface apprenant – Rétroaction double du tuteur

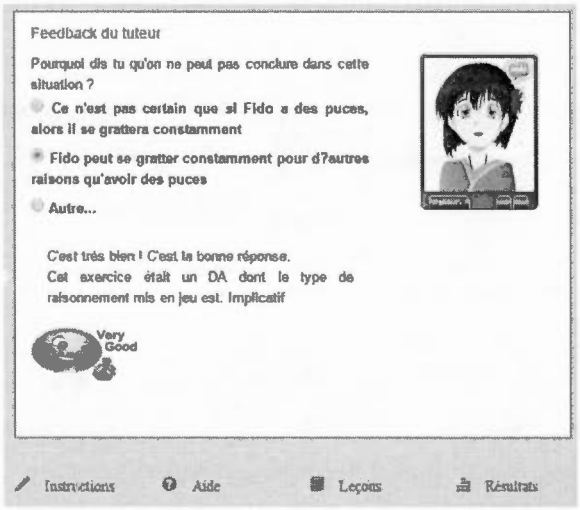


Figure V.19 : Interface apprenant – Rétroaction positif du tuteur

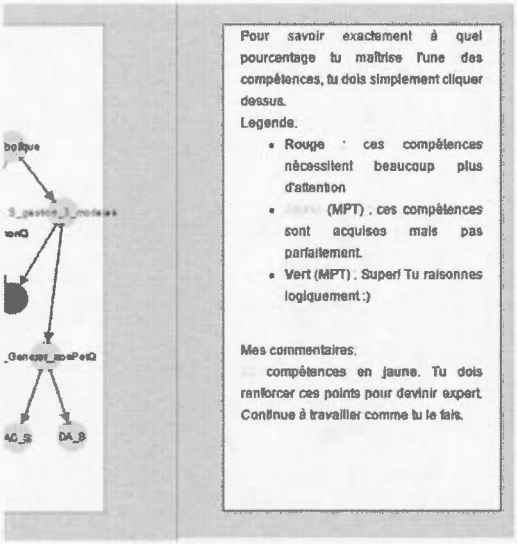


Figure V.20 : Interface apprenant – Commentaires du tuteur sur le réseau bayésien de l'apprenant

De la présentation des outils aux résultats d'implémentation en passant par les algorithmes développés dans le cadre de ce projet, nous avons à travers ce chapitre, exposer les détails techniques qui ont servi à mettre au point le STI Muse-Logique. Muse-logique est non seulement capable de raisonner logiquement comme il a été démontré, d'expliquer sa démarche et même de jouer un rôle d'accompagnateur dans l'apprentissage du raisonnement logique.

Cependant, dans l'initialisation du modèle de l'apprenant, la matrice « Posterior » dont nous avons parlé contient en ligne les différents patterns de réponse des étudiants qui se trouvent également dans la matrice DINA. Mais si nous nous retrouvons face à un pattern différent de ceux prévus, il y aura un problème car aucune ligne de Posteriror ne correspondra à ce pattern. Il faudra alors trouver par la suite une solution pour effectuer une approximation de pattern de réponse d'apprenant.

Muse-logique est à présent fonctionnel, mais est-il en mesure de raisonner devant toute situation logique ? De supporter un apprentissage réel ? Est-ce que le réseau

bayésien conçu est capable de refléter l'état de connaissances réel de l'apprenant ? Le système peut-il prédire le comportement des apprenants ? Bien évidemment la réponse à ces questions nécessite une évaluation profonde du système suivant un protocole préétabli et connu.

CHAPITRE VI

VERS UNE EVALUATION DE MUSE-LOGIQUE

Nous avons fait un certain nombre d'affirmations sur les capacités de Muse-logique. Quelles preuves avons-nous pour affirmer que le système serait capable de raisonner logiquement dans différents contextes ? Est-ce que le système est capable de modéliser correctement et précisément l'état cognitif d'un apprenant ? Est-ce qu'il est capable de prédire le comportement d'un apprenant ? Est-il capable de fournir des feedbacks (retroactions) constructifs ? Est-ce que l'approche pédagogique est efficace et aide à l'apprentissage du raisonnement logique? Toutes ces questions sont importantes et doivent trouver réponse à travers une évaluation sérieuse du système.

L'évaluation d'un système est importante. Elle permet de juger de sa validité et de son efficacité. Dans le cas de MUSE-Logique, la validité fait référence aux connaissances du domaine enjeu de l'apprentissage (le raisonnement logique), les structures et mécanismes utilisées pour permettre la prise de décision du système (formalisme de représentation des connaissances; structures et algorithmes d'inférence, etc.). L'efficacité réfère à l'efficacité pédagogique en termes de gain d'apprentissage que procure MUSE-Logique.

Nous présenterons dans ce chapitre une évaluation détaillée de Muse-Logique. L'accent sera mis sur le protocole d'évaluation dans les cas où les données réelles n'ont pas encore été récoltées.

6.1. Approches d'évaluation des systèmes tutoriels intelligents

L'évaluation des systèmes tutoriels intelligents est un processus nécessaire pour la validation de ces derniers. Cependant, peu d'attention est portée sur cette thématique. (Woolf, 2010) mentionne que l'évaluation est une garantie de la qualité du système et permet de tester l'efficacité des outils, la capacité d'adaptation ou encore la pertinence des stratégies d'apprentissage utilisées dans un domaine. L'objectif est de vérifier que le système correspond aux attentes des utilisateurs.

Il y a plusieurs avantages à faire une évaluation d'un STI. Parmi ces avantages, nous pouvons citer l'amélioration des résultats de l'apprentissage (gain d'apprentissage), l'amélioration de la durée de vie du système, l'accroissement d'acceptation de l'utilisateur, la vente du logiciel et l'aide à développer le domaine en partageant les expériences (Iqbal, Oppermann, & Patel, 1999). Il existe plusieurs méthodes d'évaluation pour les STI, mais la littérature ne suggère pas de ligne directrice claire pour juger quelle méthode est appropriée selon un usage particulier (Jeremić, Jovanović, & Gašević, 2009; Mark & Greer, 1993). Néanmoins, une évaluation demande du temps et est couteuse. Une évaluation est très complexe, car un STI est la fusion de plusieurs disciplines (sciences de l'éducation, didactique, psychologie, sciences cognitives, informatique et ergonomie) (Perrine, 2015).

Deux dimensions peuvent être considérées dans l'évaluation : le moment de l'évaluation et son contexte. Selon le moment on a l'évaluation formative et l'évaluation sommative. Selon le contexte, on a l'évaluation menée dans le monde réel et l'évaluation menée en laboratoire.

6.1.1. Évaluation formative

Dans l'évaluation formative, les chercheurs évaluent le système en cours de développement, pour identifier les problèmes et procéder à des modifications (Mark & Greer, 1993). Cette évaluation permet de tester l'efficacité du système, les capacités du tuteur à mener à bien l'enseignement et la connaissance du système. Il est ainsi possible d'améliorer le fonctionnement du système à partir des erreurs ou d'une mauvaise conception du système. L'évaluation formative est réalisée soit avec les experts et les développeurs, soit avec un petit groupe d'étudiants. C'est une évaluation subjective et qualitative (obtenue des observations, questionnaires, entretiens et analyse du protocole). L'évaluation formative peut être conduite sur des données descriptives.

6.1.2. Évaluation sommative

L'évaluation sommative permet d'appuyer les demandes formelles, le comportement ou les résultats associés à un système complet. L'évaluation se fait lorsque la conception du système est terminée. L'évaluation sommative permet de fournir l'objectivité et la reproductibilité d'une étude à partir de l'efficacité globale du système. De plus, elle détermine si les buts du système sont accomplis (enseignement, apprentissage, etc.). Ces évaluations sont quantitatives (basées sur les mesures des variables, études empiriques, expériences et analyses statistiques). L'évaluation implique de tester le système avec un grand nombre d'étudiants, minimum trente allant jusqu'à une centaine. Cependant, il y a des raisons de s'engager dans une évaluation formative plutôt que sommative. Par exemple, l'apprenant peut faire des erreurs, ce qui peut avoir un impact sur les résultats d'apprentissage (Mark & Greer, 1993; Woolf, 2010).

6.1.3. L'évaluation menée dans le monde réel

Les évaluations menées dans le monde réel étudient en profondeur le système dans un contexte naturel en utilisant des sources multiples de données. Les évaluations conduites par exemple dans les salles de classe sont plus probables de valider les résultats pour un certain nombre d'étudiants et des situations d'enseignement. L'évaluation va fournir plus de conditions variées que ne le font les tests en laboratoire. Une véritable expérience en classe appelle à une répartition aléatoire des sujets aux conditions fixées par le protocole d'évaluation (Woolf, 2010).

6.1.4. L'évaluation menée en laboratoire

La réalisation d'évaluations en laboratoire a l'avantage d'adresser des questions de recherche et permet une plus grande maitrise des expériences. En effet, il est possible de sélectionner un grand nombre de sujets et d'avoir des connaissances préalables à tester (Woolf, 2010). Dans ce type d'évaluation, les évaluateurs (en général les concepteurs) vont utiliser des procédures spécifiques pour tester les hypothèses et certains contrôles pour garantir la validité du système (Ainsworth, 2005). Les procédures statistiques pourraient être utilisées pour contrôler les effets de non-randomisation. Les caractéristiques personnelles des élèves (compétences cognitives ou intérêts du sujet) ne peuvent pas être comparées dans les quasi-expériences (Woolf, 2010).

6.2. Évaluation du système

De ce qui a été dit plus haut, il est préférable dans notre cas de faire une évaluation formative en laboratoire. Elle est utile pour guider la conception du système

et valider certaines hypothèses et choix. Cette évaluation interne aura pour but d'étudier la relation entre l'architecture du STI et son comportement. L'évaluation sommative qui se déroulera dans le monde réel (évaluation externe) viendra dans un deuxième temps, soit à la fin du développement du système. Elle permettra de confirmer l'efficacité du système (Aïmeur & Frasson, 2000). N'ayant pas encore des données réelles, nous allons nous concentrer sur l'évaluation formative du système. Cependant, nous nous pencherons également sur la description du protocole de validation sommative qui sera exécuté une fois que les données réelles seront collectées. Toutefois, ces données viendront juste confirmer que le système est bel et bien efficace, ce que nous essayerons de démontrer dans cette partie.

Pour une évaluation plus précise de Muse-logique, nous avons opté pour l'évaluation par composants. Cette démarche d'évaluation a été proposée par (Mark & Greer, 1993) dont le principal but est l'évaluation spécifique de l'architecture et du comportement du STI pour chacun de ses composants. Dans cette section, nous décrivons l'évaluation de chaque composant du système Muse-logique selon les objectifs qui ont été fixés dès le début de ce mémoire.

6.2.1. Évaluation du module expert

L'expert (ou module expert) dans Muse-logique a pour rôle de stocker et manipuler les connaissances du domaine. Étant donné que le domaine en jeu est le raisonnement logique, l'expert Muse-logique doit être capable de faire un raisonnement logique tout en expliquant son cheminement. Il doit également être capable de reconnaître les erreurs de raisonnement et de les corriger. Les capacités de l'expert doivent être vérifiées avant que le STI ne soit terminé. Dans un STI, les informations sur un sujet ou un domaine sont présentées à l'apprenant sous la forme d'exemples, de problèmes à résoudre, de questions, de commentaires, etc. Cette composante doit être

à cet effet, testée au cours du développement du STI pour avoir un expert complet. Les méthodes d'évaluation pour cette composante sont une inspection des experts ou le test de Turing. Étant donné que dans notre équipe, nous avons des experts du domaine de la logique, nous opterons pour l'inspection des experts. Le protocole de validation de l'expertise de l'expert est le suivant :

- A partir d'une banque de problèmes de raisonnement fourni par les experts en logique, nous avons testé les capacités de raisonnement de l'expert Muse-logique. Nous avons comparé les réponses fournies par les experts humains avec celles que fournira l'expert Muse. Nous vérifierons également si le système est capable de prouver son raisonnement en donnant le type et le mode de raisonnement en jeu.
- À partir de cette même banque de problèmes, nous donnons au système des problèmes de raisonnement dont la réponse est erronée et nous vérifions que le système est bien capable de reconnaître un raisonnement erroné de l'expliquer et de donner une solution remédiative.

Pour ce qui est du premier volet du protocole ci-dessus, la banque de problèmes fournis par les experts contient 16 problèmes de raisonnement en situation concrète, 4 problèmes en situation abstraite et 4 problèmes en situation symbolique, pour un total de 24 problèmes diversifiés. Nous avons écrit un script qui extrait ces problèmes de la base de données et qui le fournit à l'expert Muse-logique. Une partie des résultats de l'expert Muse peut être visualisée à la Figure VI.1. Les réponses de l'expert sont les lignes en gras. Nous constatons qu'en plus de donner la réponse au problème qui lui est posé, le système est capable de dire quel type et mode de raisonnement il s'agit. La comparaison des résultats obtenus sur les 24 types de problèmes de raisonnement implicatif, avec ceux des experts humains, montre que l'expert codé dans Muse-logique a fourni une réponse valide à chaque problème. Nous concluons que le système est capable de raisonner logiquement tout en détectant le type et le mode de raisonnement en jeu.

```
14- Règle: Si Louise se brosse les dents régulièrement, alors elle aura des caries
   Information: Louise n'a pas de caries
Expert : Louise ne se brosse pas les dents régulièrement, c'est un raisonnement de type Implicatif
15- Règle: Si Louise se brosse les dents régulièrement, alors elle aura des caries
   Information: Louise a des caries
Expert : On ne peut pas conclure, c'est un raisonnement de type Implicatif dont le mode est le AC
16- Règle: Si Louise se brosse les dents régulièrement, alors elle aura des caries
   Information: Louise ne se brosse pas les dents régulièrement
Expert : On ne peut pas conclure, c'est un raisonnement de type Implicatif dont le mode est le DA
17- Règle: Si une personne morp, alors elle deviendra plede
   Information: Pierre morp
Expert : Pierre est devenu plede, c'est un raisonnement de type Implicatif dont le mode est le MPP
18- Règle: Si une personne morp, alors elle deviendra plede
   Information: Pierre n'est pas devenu plede
Expert : Pierre ne morp pas, c'est un raisonnement de type Implicatif dont le mode est le MTT
19- Règle: Si une personne morp, alors elle deviendra plede
   Information: Pierre est devenu plede
Expert : On ne peut pas conclure, c'est un raisonnement de type Implicatif dont le mode est le AC
20- Règle: Si une personne morp, alors elle deviendra plede
   Information: Pierre ne morp pas
Expert : On ne peut pas conclure, c'est un raisonnement de type Implicatif dont le mode est le DA
21- Règle: A>B
   Information: A
Expert : B, c'est un raisonnement de type Implicatif dont le mode est le MPP
22- Règle: A>B
   Information: ¬B
Expert : ¬A, c'est un raisonnement de type Implicatif dont le mode est le MTT
```

Figure VI.1 : Extrait des réponses du système à 24 problèmes de raisonnement

Pour l'évaluation de la capacité du système à identifier une erreur de raisonnement à l'expliquer et à donner des éléments correctifs, nous avons testé le système en lui donnant un ensemble de 48 erreurs de raisonnement (2 erreurs de raisonnement pour chacun des syllogismes de la banque des exercices). Un extrait des 3 premières erreurs ainsi que les réponses du système à ces erreurs de raisonnement est présenté à la Figure VI.2. Une comparaison des résultats du système et ceux des experts humains nous amènent à conclure que l'expert dans Muse-logique est capable de desseller le type d'erreur de raisonnement, de donner une explication s' il y a lieu et de proposer une solution afin d'y remédier.

```
1- Règle: Si on lance une roche dans une fenêtre, alors la fenêtre brisera
   Information: La fenêtre n'est pas brisée
   Réponse: On a lancé une roche dans la fenêtre
Expert : C'est une réponse mauvaise atypique, donc invalide
2- Règle: Si on lance une roche dans une fenêtre, alors la fenêtre brisera
   Information: La fenêtre est brisée
   Réponse: On a lancé une roche dans la fenêtre
Expert : C'est une mauvaise réponse typique, donc invalide.
   Explication: cet erreur est faite lorseque L'antécédent est considéré comme seul antécédent possible, comme cause unique.
L'implication est traité comme une équivalence : le AC est considéré comme valide.
Nous augmentons indûment l'information de la prémisse majeure (de 3 cas vrais à 2 cas vrais) et allégeons notre mémoire de travai
   Solution:  La correction passe par la sensibilisation aux antécédents alternatifs (et aux causes multiples).
3- Règle: Si on lance une roche dans une fenêtre, alors la fenêtre brisera
   Information: On n'a pas lancé de roche dans la fenêtre
   Réponse: La fenêtre brisera
Expert : C'est une mauvaise réponse atypique, donc invalide.
   Explication: cet erreur est faite lorseque L'antécédent est considéré comme seul antécédent possible, comme cause unique.
L'implication est traité comme une équivalence : le DA est considéré comme valide.
Nous augmentons indûment l'information de la prémisse majeure (de 3 cas vrais à 2 cas vrais) et allégeons notre mémoire de travai
   Solution:  La correction passe par la sensibilisation aux antécédents alternatifs (et aux causes multiples).
BUILD SUCCESSFUL (total time: 2 seconds)
```

Figure VI.2 : Détection des erreurs de raisonnement par le système

6.2.2. Évaluation du modèle cognitif de l'apprenant (réseau bayésien)

Le modèle de l'apprenant plus précisément le modèle cognitif dans notre cas (les autres modèles étant à venir) permet de réaliser le diagnostic et la modélisation de l'état de connaissances de l'apprenant. Le diagnostic étudie le comportement de l'apprenant et la modélisation apporte les renseignements passés et présents de l'apprenant, ce qui permet d'obtenir des informations significatives. Les connaissances de l'apprenant dans Muse-logique sont mesurées grâce au réseau bayésien. Les caractéristiques importantes de cette modélisation sont la validité, la fiabilité, l'objectivité et la normalisation. Le diagnostic doit être valide et fiable. Le diagnostic doit être fiable en donnant des résultats d'évaluation constants comparables entre les apprenants. L'objectivité et la normalisation sont moins applicables aux STI (Mark & Greer, 1993). La modélisation de l'apprenant est valable que s'il reflète fidèlement l'apprenant dans le temps. Il est difficile d'avoir une objectivité et un ensemble de normes pour le modèle de l'apprenant. Ces caractéristiques ne sont pas adaptées à la modélisation de l'apprenant (Perrine, 2015). Nous allons pour ce faire nous concentrer sur la validité et la fiabilité de notre modèle cognitif. La validation est un facteur important pour la

composante apprenant qui peut être évaluée soit pendant ou à la fin de la conception du système (Mark & Greer, 1993).

Le protocole d'évaluation du modèle cognitif est le suivant : pour l'évaluation de la validité (structure du réseau, liens de causalités, probabilités à priori) nous utilisons les résultats des experts combinés aux résultats que nous avons obtenus grâce au CDM. L'évaluation de la fiabilité se fait par une procédure de prédiction (Pardos & Heffernan, 2010) dans laquelle l'on essaie de prédire les réponses des apprenants sur certaines questions à l'aide du réseau bayésien. Étant donné que nous n'avons pas encore collecté des données réelles, ces évaluations ont été conduites sur des données générées aléatoirement à partir de l'IDE Netbeans. Le but est de présenter la faisabilité du protocole d'évaluation dont il est question, et de prouver la valeur ajoutée de notre modélisation dont les données réelles ne viendront que confirmer.

Protocole d'évaluation de la validité : Rappelons que les aspects à évaluer pour la validité du modèle cognitif (réseau bayésien) sont la structure du réseau, les liens de causalité, les probabilités à priori. Une validation externe a été conduite avec les experts qui se sont prononcés sur la pertinence des nœuds, liens et tables de probabilités constituant le réseau bayésien. Une analyse cognitive des différentes tâches de raisonnement par les experts du domaine a permis de mettre en place la structure et les liens de causalité entre les nœuds définis. Nous avons présenté les résultats de ces analyses au 0 du présent mémoire. Pour ce qui est des probabilités à priori, leur choix s'est fondé sur les connaissances des experts en raisonnement logique quant aux dépendances et influences entre les variables du réseau. Les tables de probabilités permettent de quantifier ces influences et de contrôler le poids de chaque nœud en fonction de ses parents. On peut modéliser différents phénomènes observés dans la littérature du domaine du raisonnement logique. Généralement, pour un étudiant, lors de l'apprentissage du raisonnement conditionnel, le raisonnement en contexte causal est le premier à être maitrisé, vient ensuite le raisonnement en contexte contrefactuel, en abstrait puis en symbolique (H. Markovits, 2013). D'un point de vue des nœuds et

des influences, l'influence de chaque nœud sur le nœud compétence à l'implication peut être résumé de la façon suivante : Influence Symbolique > Abstrait > Contrefactuel > Causal. D'autre part, la compétence Inhiber P&nonQ est plus facile à maitriser que la compétence à avoir une bonne gestion des 3 modèles (H. Markovits, 2013). Le nœud Inhiber P&nonQ a donc une influence moins importante sur le nœud de compétence dans un contexte particulier que le nœud Avoir une bonne gestion des 3 modèles. En contexte familier, la compétence pour des exercices de type MPPFD (MPP en contexte factuel ou causal avec peu d'alternatives) et MTTFD (MTT en contexte factuel ou causal avec peu d'alternatives) est maitrisée plus tôt dans l'apprentissage de la logique que les exercices de type MPPMD (MPP *many disablers*) et MTTMD (MTT *many disablers*) (H. Markovits, 2013). Les nœuds MPPFD et MTTFD vont donc avoir une influence moins importante sur leurs parents que les nœuds MTTMD et MPPMD. Les probabilités conditionnelles ont été définies a priori en se basant sur les influences entre les variables.

Nous pensons que le fait de disposer du CDM nous permettra de nous prononcer sur une validation tant structurale que sémantique du réseau, issue d'une observation et analyse de données. Par exemple, l'analyse de la matrice de corrélation entre les nœuds compétence produite par le CDM permettra de confirmer l'existence des liens et même la validité des tables de probabilités à priori. A défaut d'avoir des données réelles, nous avons simulé cette évaluation avec des données générées aléatoirement sachant que nous avons une garantie que les données réelles ne viendront que confirmer les résultats. Les données aléatoires sont utilisées ici plus pour illustrer la faisabilité du protocole présenté. Pour ce faire, nous avons conduit une expérience avec une matrice DINA (voir 0 et 0 pour les détails concernant cette matrice) créée aléatoirement pour simuler les résultats des étudiants au test de niveau pour pouvoir effectuer les tests. Elle contient en ligne des étudiants et en colonne des items (exercices). Dans notre cas, nous avons choisi d'avoir 400 lignes pour 400 étudiants. En colonne, ce sont les 16 lignes de la Q-Matrix contenant les items. La Q-Matrix contient en ligne les items et en

colonne les compétences (voir annexe B). À chaque item, elle fait donc correspondre des compétences. Elle a donc 16 lignes pour 16 items et 12 colonnes pour 12 compétences. Ces deux matrices sont remplies uniquement de 0 et de 1. Lors des tests effectués, puisque le résultat était inexploitable (voluminosité), nous avons partitionné la matrice DINA et la Q-Matrix de façon à effectuer les calculs pour 4 compétences au lieu de 12. Nous avons donc obtenu trois matrices DINA et trois Q-Matrix. Avec ce partitionnement, les résultats étaient donc exploitables. Le premier résultat visible à la Figure VI.3, représente les valeurs des paramètres GUESS (probabilité de donner une bonne réponse sachant qu'on n'a pas la compétence) et SLIP (probabilité de donner une mauvaise réponse sachant que l'on a la compétence requise). La matrice de corrélation entre les nœuds de compétences visible à la Figure VI.4 (les 4 premières compétences sont liées à la Génération du modèle interprétation), nous permet d'estimer en matière de probabilités le lien entre ces différentes compétences. Grace à cette matrice, il sera possible de valider la structure (liens causaux entre les nœuds) du réseau. Par exemple, le fait que le familier dans le premier modèle (modèle ET) soit fortement corrélé avec le familier dans le second modèle ($\Leftrightarrow$) implique qu'il doit exister un lien de causalité entre ces deux compétences. En suivant ce protocole avec les données réelles, nous pouvons donc valider notre modèle cognitif.

```
Item parameters
   item guess  slip
1    V1 0.521 0.494
2    V2 0.337 0.434
3    V3 0.049 0.367
4    V4 0.535 0.559
5    V5 0.385 0.275
6    V6 0.473 0.503
7    V7 0.478 0.383
8    V8 0.466 0.617
9    V9 0.500 0.654
10  V10 0.497 0.081
11  V11 0.464 0.054
12  V12 0.467 0.000
13  V13 0.462 0.005
14  V14 0.514 0.000
15  V15 0.514 0.221
16  V16 0.527 0.873
```

Figure VI.3 : Paramètres Guess et Slip pour chaque items de la Q-Matrix.

```
Tetrachoric correlations among skill dimensions
                Familier.ET Contrefactuel.ET Abstrait.ET Formel.ET
Familier.ET         1.00000          -0.36225    -0.10055   0.03929
Contrefactuel.ET   -0.36225           1.00000    -0.22390   0.08719
Abstrait.ET        -0.10055          -0.22390     1.00000   0.06552
Formel.ET           0.03929           0.08719     0.06552   1.00000
```

Figure VI.4 : Matrice de corrélation entre les items de compétences

Protocole d'évaluation de la fiabilité : Le point à évaluer ici est la fiabilité du réseau bayésien. Est-ce que cette modélisation rend effectivement compte de l'état des connaissances de l'apprenant ? Dans la littérature, la fiabilité est souvent évaluée par la capacité de prédiction du modèle de l'apprenant (Pardos & Heffernan, 2010). Pour ce faire, puisqu'il existe 20 types de connaissances (les feuilles du réseau bayésien se trouvant à l'annexe) à l'implication et directement évaluables (au moins 5 exercices pour chaque type), il nous faut au minimum un patron de réponses à 60 exercices. Parmi ces 60 exercices, l'apprenant doit avoir répondu à au moins 5 exercices pour chaque élément de connaissances. Ainsi, nous utilisons les 4 premières réponses d'un élément de connaissances pour initialiser le réseau, et nous prédisons la réponse de l'apprenant aux autres exercices du même élément. Par exemple, si un apprenant obtient 4 bonnes

réponses sur 4 dans le MPP en contexte abstrait, alors sa probabilité de maîtrise de cet élément de connaissance, calculée à travers le réseau bayésien de Muse-logique (utilisation de l'algorithme d'inférence *junction tree*) est de 0.998 ce qui veut dire qu'il y a 99,8% de chance que ses autres réponses soient également correctes. Ce qui peut être vérifiée de manière intuitive comme ceci : si sur 5 problèmes, une personne réussit les 4 premiers problèmes qui lui sont présentés alors il y a de fortes chances qu'il réussisse au dernier problème. S'il arrive qu'il réussisse 3 sur les 4 alors ses chances de trouver le dernier exercice diminuent. C'est en fait ce raisonnement qui est déduit du réseau à la seule différence qu'elle n'est pas faite d'une manière intuitive. Comme on l'a vu dans l'évaluation de la validité, le raisonnement codé dans le réseau a été étudié et validé par les experts. Il peut cependant arriver que cet apprenant rate le prochain exercice par erreur ou distraction, la probabilité qu'une telle situation se produise est appelée le SLIP. Cette probabilité est calculée à l'aide de données collectées sur les apprenants et grâce aux algorithmes du CDM (voir Figure VI.3). Ainsi, si le réseau parvient à prédire les prochains résultats de l'apprenant, alors la modélisation de l'état de connaissance de l'apprenant est validée.

6.2.3. Évaluation du modèle pédagogique

Le modèle pédagogique ou encore le tuteur est un composant très important dans un STI, car il permet de favoriser l'acquisition de l'expertise par l'apprenant. Ce modèle doit être conçu par les experts en s'appuyant sur les méthodes d'instruction et les conditions d'utilisation du système. L'objectif est d'avoir un enseignement personnalisé (qui se fait grâce au réseau bayésien) par rapport aux caractéristiques des apprenants (les connaissances antérieures, le niveau de développement cognitif et le style d'apprentissage) et efficace (stratégies d'interactions efficaces) pour que l'apprentissage soit effectif.

Selon (Mark & Greer, 1993), l'évaluation du modèle pédagogique peut se faire par une comparaison aux théories d'enseignement ou à un enseignant expert humain. Il est à noter que le comportement du modèle pédagogique dépend de l'apprenant et du domaine de connaissance. L'évaluation du caractère adaptatif du comportement du tuteur repose sur l'évaluation du modèle cognitif de l'apprenant qui a été effectuée ci-dessus. De même l'évaluation des connaissances du tuteur repose sur l'évaluation de l'expert qui a été également fait. Il ne reste plus qu'à évaluer la pertinence des interventions du tuteur dans l'apprentissage. Dans le domaine de l'apprentissage et de l'enseignement, la rétroaction a été considérée comme un principe fondamental pour un apprentissage efficace (Andre, 1997), ou au moins comme un élément important de l'enseignement (Collis, De Boer, & Slotman, 2001; Narciss, 2008). Généralement, les interactions du tuteur dans les systèmes d'apprentissage sont conçues de manière intuitive (Narciss, 2008). Cependant, Il existe dans la littérature un ensemble de grandes lignes à respecter pour construire des stratégies d'interactions efficaces dans les environnements informatiques d'apprentissage (Narciss & Huth, 2004). Nous évaluons à cet effet, notre modèle pédagogique par rapport à ces lignes directrices.

Dans (Narciss, 2008; Narciss & Huth, 2004) il est dit qu'en général, un feedback doit être constitué de 2 parties. La première partie appelée évaluation ou composant de vérification se rapporte au résultat de l'apprenant par rapport à l'exercice en cours. Elle doit indiquer le niveau de performance réalisé (par exemple, la réponse est correcte / incorrecte, pourcentage de réponses correctes, etc.). La deuxième partie appelée partie informative, se compose d'informations relatives à la question, la tâche, les erreurs, ou les solutions. Le tableau 6.1 tiré de (Narciss, 2008), présente pour chaque partie d'un feedback les différents cas qui peuvent se présenter avec des exemples à l'appui. Un feedback élaboré doit donc contenir au minimum une partie évaluative et une partie informative. Le respect de cette structure a pour avantage de (Nicol & Macfarlane-Dick, 2006) :

- aider à clarifier ce qui est une bonne performance (objectifs, critères, etc.);

- faciliter le développement de l'auto-évaluation (réflexion) dans l'apprentissage;
- fournir des informations de haute qualité pour les étudiants au sujet de leur apprentissage;
- encourager les croyances motivationnelles positives et l'estime de soi;
- fournir des occasions de combler l'écart entre la performance actuelle et souhaitée;
- fournir des informations aux enseignants qui peuvent être utilisés pour aider à façonner l'enseignement.

Les interventions du tuteur dans notre système ont été conçues grâce à l'aide des experts et ceci, en se basant sur l'analyse cognitive des différents comportements qui peuvent survenir pendant l'apprentissage du raisonnement. Par exemple, une analyse cognitive des erreurs de type AC et DA a permis de conclure que pour éviter de faire ces erreurs, le tuteur doit les inhiber en sensibilisant l'apprenant aux causes multiples. Ce résultat est transcrit dans les règles tutorielles de Muse-logique sous forme d'interventions du tuteur. Nous avons vu au chapitre III (règles tutorielles) et au chapitre V (captures d'écrans des interventions du tuteur) que, tout commentaire du tuteur commence par un message rendant compte à l'apprenant du résultat de l'évaluation de sa réponse (partie évaluative) tout en le motivant. Ensuite vient la partie informative dans laquelle le tuteur rappelle des notions de cours concernant l'exercice encours s'il y a lieu, donne des informations sur le type d'exercice (voir Figure VI.5) ou encore explique l'erreur commise et propose des solutions. Le tuteur dans Muse-logique ne donne la réponse qu'après 2 erreurs consécutives sur le même exercice (voir règles tutorielles).

Figure VI.5 : Commentaires du tuteur après un échec

Tableau VI.1 : Classification du contenu des feedbacks

Catégorie	Exemples
Partie évaluative	
Performance du sujet	15/20, 85% de réussite, etc.
Evaluation de la réponse	Correct, Incorrect
Explications	Description ou indications de la bonne réponse
Partie informative	
Contraintes de l'exercice	Conseils /explications sur les règles de traitement de la tâche Aides / explications des sous-tâches, explications sur les exigences de la tâche, explications sur le type de la tâche.
Informations sur les concepts	Conseils / explications sur les termes techniques, explications sur le contexte conceptuel Exemples illustrant le concept

	Aides /explications sur les attributs de concept
Informations sur les erreurs	Nombre d'erreurs, Zones où l'erreur s'est produite Aides / explications sur le type d'erreurs, explications sur les sources d'erreurs
Comment procéder pour l'exécution de l'exercice	Les astuces pour la correction d'erreur Conseils / explications sur les stratégies spécifiques à la tâche, explications sur les étapes de traitement de la tâche, montrer des exemples
Informations métacognitives	Conseils / explications sur les stratégies métacognitives, questions d'orientation Métacognitive.

En dehors des feedbacks élaborés, le modèle pédagogique doit intégrer des tâches d'apprentissage interactives (Narciss, 2008). Ces dernières sont des tâches qui comprennent plusieurs étapes de résolution ou plusieurs essais. A travers ces tâches, le système cherche à mieux comprendre le pourquoi de certaines réponses. Dans Muse-logique, il y a des situations qui amènent le tuteur à demander plus d'informations à un apprenant par rapport à sa réponse. Le tuteur réagit ainsi afin d'aller chercher les raisons du choix de cette réponse, identifier le problème et proposer des solutions spécifiques. Ce cas de figure se présente lorsque l'apprenant est devant un exercice de type AC ou DA (voir Figure VI.6Figure VI.3).

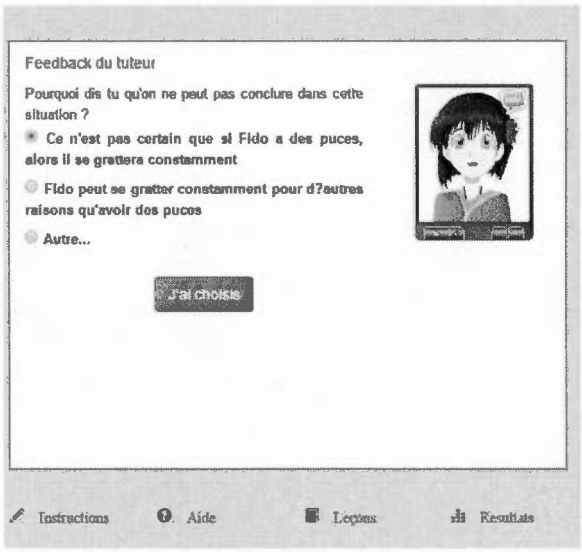


Figure VI.6 : Feedback pour comprendre la réponse de l'apprenant à un AC

En somme, nous voyons que le modèle pédagogique dans Muse-logique est conçu sur des bases solides et permettra ainsi un apprentissage effectif.

Ce chapitre s'est concentré sur une évaluation interne du système Muse-Logique. Nous avons pu valider le système expert du domaine en prouvant qu'il estcapable de mener un raisonnement valide dans différentes situations de raisonnement, identifier une erreur de raisonnement tout donnant les causes et des solutions remédiatives.

Concernant le modèle cognitif, nous pouvons déjà nous prononcer positivement sur la validité structurale et même sémantique du réseau, car sa construction s'est faite sur une base théorique et validée par les experts. Les probabilités à priori dans le réseau vont être affinées lors de l'utilisation du prototype par les étudiants. Nous avons présenté les protocoles qui seront utilisés une fois que nous aurons des données réelles. On conclut que c'est vraiment les données réelles qui nous donnerons des réponses effectives pour le modèle de l'apprenant.

Le modèle pédagogique a été évalué suivant une base théorique. L'évaluation qui a été positive, nous rend déjà compte de la valeur ajoutée d'un tel système dans

l'apprentissage du raisonnement logique. Une évaluation sur une base pratique (observation des sujets pendant l'utilisation du système) pourrait être conduite par la suite pour confirmer ces résultats théoriques.

La prochaine étape serait de confirmer grâce à une évaluation externe que le système tutoriel intelligent Muse-logique est effectif dans la prédiction des connaissances de l'apprenant, sa modélisation cognitive et dans l'apprentissage du raisonnement logique. Cette évaluation consistera en une évaluation des performances des apprenants réels avant et après l'usage du STI. Cette évaluation pourra être conduite après que le système aura récolté un certain nombre de données. Néanmoins, grâce à l'évaluation interne que nous avons mené dont les résultats sont satisfaisants, et au protocole défini pour une évaluation externe du modèle cognitif, nous prévoyons des résultats également satisfaisants après une évaluation sommative en monde réel.

CHAPITRE VII

CONCLUSION

Dans ce mémoire, nous avons présenté le développement d'un système tutoriel intelligent Muse-logique pour l'apprentissage du raisonnement logique. Un début d'évaluation des composants du système a permis de prouver ses capacités et son utilité.

Muse-logique est constitué d'un expert capable de raisonner, démontrer son raisonnement, détecter des erreurs de raisonnement logique et les corriger. Son modèle de l'apprenant est capable d'estimer l'état de connaissances d'un apprenant en tout temps. Son modèle pédagogique est capable de fournir un enseignement effectif en se basant sur une interaction adaptative et une rétroaction riche, constructive et favorisant un apprentissage effectif. Enfin, son composant interface est développé de sorte à pouvoir être accessible à tout le monde sans besoin d'installation.

Muse-logique se veut être un système générique pour tout type de raisonnement et de logique. Sa conception a été faite de manière à permettre une prise en charge facile de n'importe quel type de raisonnement et de logique. L'interdisciplinarité de l'équipe à jouer un grand rôle dans le développement de ce STI. Les résultats des études menés par les experts logiciens ont permis de nourrir le modèle expert. Grâce aux résultats des analyses des tâches de raisonnement par les experts en sciences cognitives, le modèle cognitif de l'apprenant et le modèle pédagogique ont pu être élaborés. Finalement, avec l'appui des experts dans le domaine des systèmes tutoriels intelligents, Muse-logique a pu être développé et mis en œuvre.

Une évaluation préliminaire du système a été effectuée. Cependant, bien que des conclusions intéressantes aient été tirées de cette évaluation formative qui s'est déroulée en laboratoire, il reste nécessaire de les valider par une évaluation sommative dans un contexte réel. A cet effet, nous avons présenté le protocole à suivre pour mener cette évaluation. Nous sommes tout de même confiants que les résultats qui seront obtenus dans cette évaluation ne viendront que confirmer nos conclusions. Muse-logique devrait être utilisé durant la session d'automne 2015 par des étudiants du baccalauréat à l'UQAM, dans le cadre d'un cours de logique. Les données pourront être collectées à ce moment et le gain d'apprentissage pourra être évalué si ces données sont suffisantes.

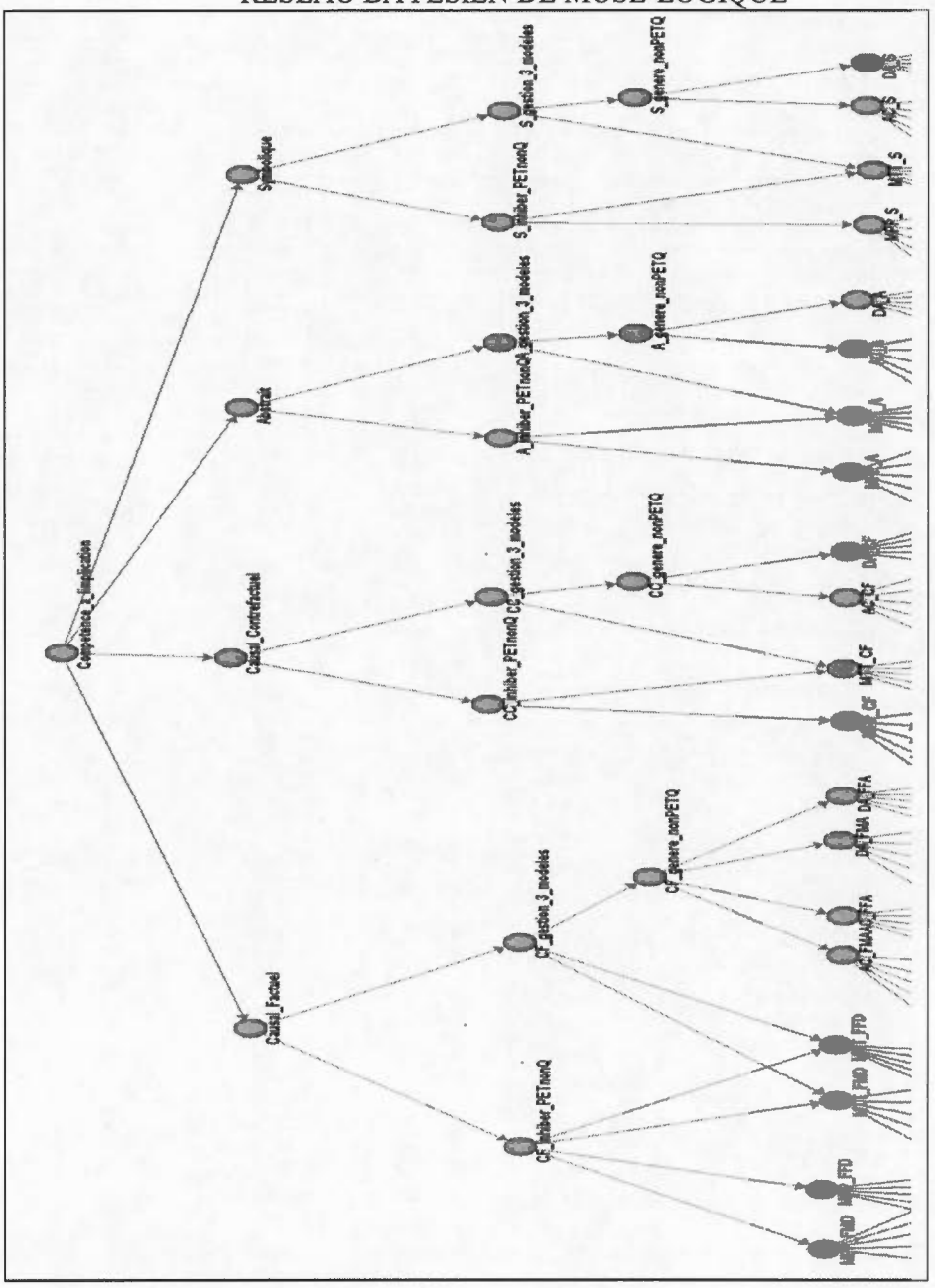
L'ensemble des objectifs visés dans le cadre de ce mémoire, notamment le développement d'un STI capable de raisonner logiquement et d'enseigner ce raisonnement, ont été atteints. Cependant, dans la perspective d'un développement futur, des ajouts ou améliorations pourraient être envisagés tant au niveau conceptuel qu'au niveau de l'implémentation de l'outil. Dans les prochaines versions du système, nous nous pencherons sur l'intégration d'autres types de raisonnement et de logique. On pourrait envisager à ce moment, l'amélioration du caractère adaptatif du système. En effet, le tuteur dans Muse-logique présente les exercices en fonction du niveau de connaissances de l'apprenant. Cependant, ses commentaires ne changent pas en fonction du niveau de l'apprenant, mais en fonction du type d'exercice. Ce que nous proposons est la construction à l'aide des experts pédagogue, d'un ensemble de feedback adapté aux différents états de connaissance. Nous nous pencherons également sur l'intégration des autres modules (module psychologique et affectif) du composant apprenant grâce à l'utilisation des outils tels que le *Tobbi* ou l'*eyestraker*. A travers ces améliorations, nous visons un système complet, encore plus adaptatif et mieux informé sur l'état général de l'apprenant afin de mieux contribuer à son apprentissage.

Ce projet de recherche a été une expérience très enrichissante. Il nous a permis de construire un système tutoriel pour l'apprentissage de la logique. Ce résultat vient

combler les manques relevés dans la littérature sur ce domaine. Il innove du fait que toute son architecture a été construite sur une base solide et avec la participation active des experts (ce qui n'est généralement pas le cas). Bien que la première version de Muse-logique soit un succès, il y a encore quelques points à travailler dont la stratégie pédagogique et sa contribution dans l'apprentissage. Est-ce que la stratégie pédagogique dont la sélection des exercices se base sur le niveau de l'apprenant permet de le faire apprendre plus rapidement ? Le fait de présenter à l'apprenant les exercices qu'il ne maîtrise pas tant qu'il n'obtient pas de bonnes réponses à ces exercices n'entraînerait pas plutôt un découragement ? Une piste de réponses à ces questions pourrait se trouver dans la fouille de données collectées lors de l'usage du système, un thème intéressant pour des travaux futurs.

APPENDICE A

RESEAU BAYESIEN DE MUSE-LOGIQUE



APPENDICE B

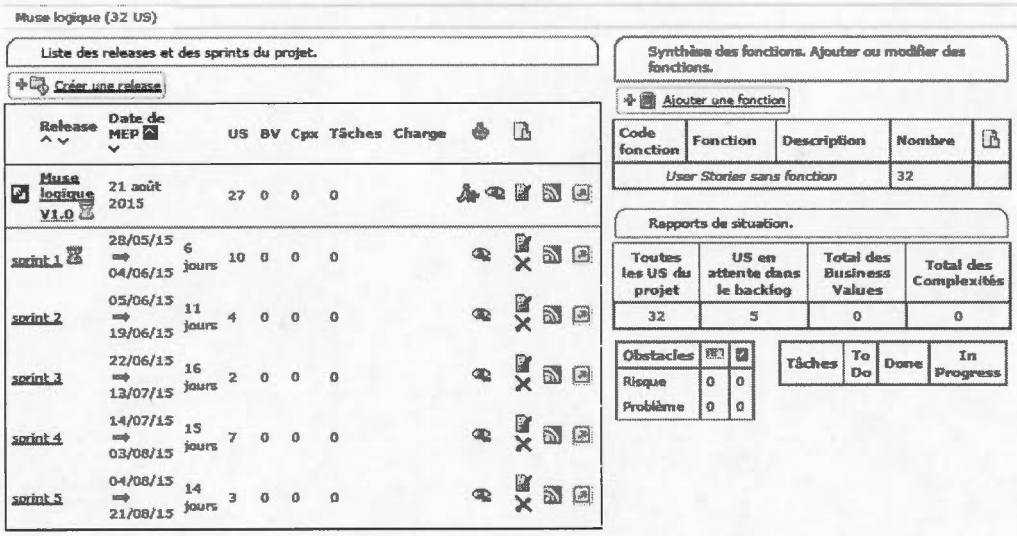
Q-MATRICE DE MUSE-LOGIQUE

	Génération du 1° model (interprétation &)				Génération du 2° modèle (interprétation <=>)				Génération du 3° modèle (interprétation =>)			
	Familier	Contrefactuel	Abstrait	Formel	Familier	Contrefactuel	Abstrait	Formel	Familier	Contrefactuel	Abstrait	Formel
1	1	0	0	0	0	0	0	0	0	0	0	0
2	1	0	0	0	1	0	0	0	0	0	0	0
3	1	0	0	0	1	0	0	0	1	0	0	0
4	1	0	0	0	1	0	0	0	1	0	0	0
5	1	1	0	0	0	0	0	0	0	0	0	0
6	1	1	0	0	1	1	0	0	0	0	0	0
7	1	1	0	0	1	1	0	0	1	1	0	0
8	1	1	0	0	1	1	0	0	1	1	0	0
9	1	1	1	0	0	0	0	0	0	0	0	0
10	1	1	1	0	1	1	1	0	0	0	0	0
11	1	1	1	0	1	1	1	0	1	1	1	0
12	1	1	1	0	1	1	1	0	1	1	1	0
13	1	1	1	1	0	0	0	0	0	0	0	0
14	1	1	1	1	1	1	1	1	0	0	0	0
15	1	1	1	1	1	1	1	1	1	1	1	1
16	1	1	1	1	1	1	1	1	1	1	1	1

Légende			
Familier	CF	Abstrait	Formel
1- MPP	5- MPP	9- MPP	13- MPP
2- MTT	6- MTT	10- MTT	14- MTT
3- AC	7- AC	11- AC	15- AC
4- DA	8- DA	12- DA	16- DA

APPENDICE C

SCRUM : PLANNING DU PROJET



Muse logique (32 US)

Liste des releases et des sprints du projet.

Créer une release

Release	Date de MEP		US	BV	Cpx	Tâches	Charge
Muse logique V1.0	21 août 2015		27	0	0	0	
sprint 1	28/05/15 ⇒ 04/06/15	6 jours	10	0	0	0	
sprint 2	05/06/15 ⇒ 19/06/15	11 jours	4	0	0	0	
sprint 3	22/06/15 ⇒ 13/07/15	16 jours	2	0	0	0	
sprint 4	14/07/15 ⇒ 03/08/15	15 jours	7	0	0	0	
sprint 5	04/08/15 ⇒ 21/08/15	14 jours	3	0	0	0	

Synthèse des fonctions. Ajouter ou modifier des fonctions.

Ajouter une fonction

Code fonction	Fonction	Description	Nombre
User Stories sans fonction			32

Rapports de situation.

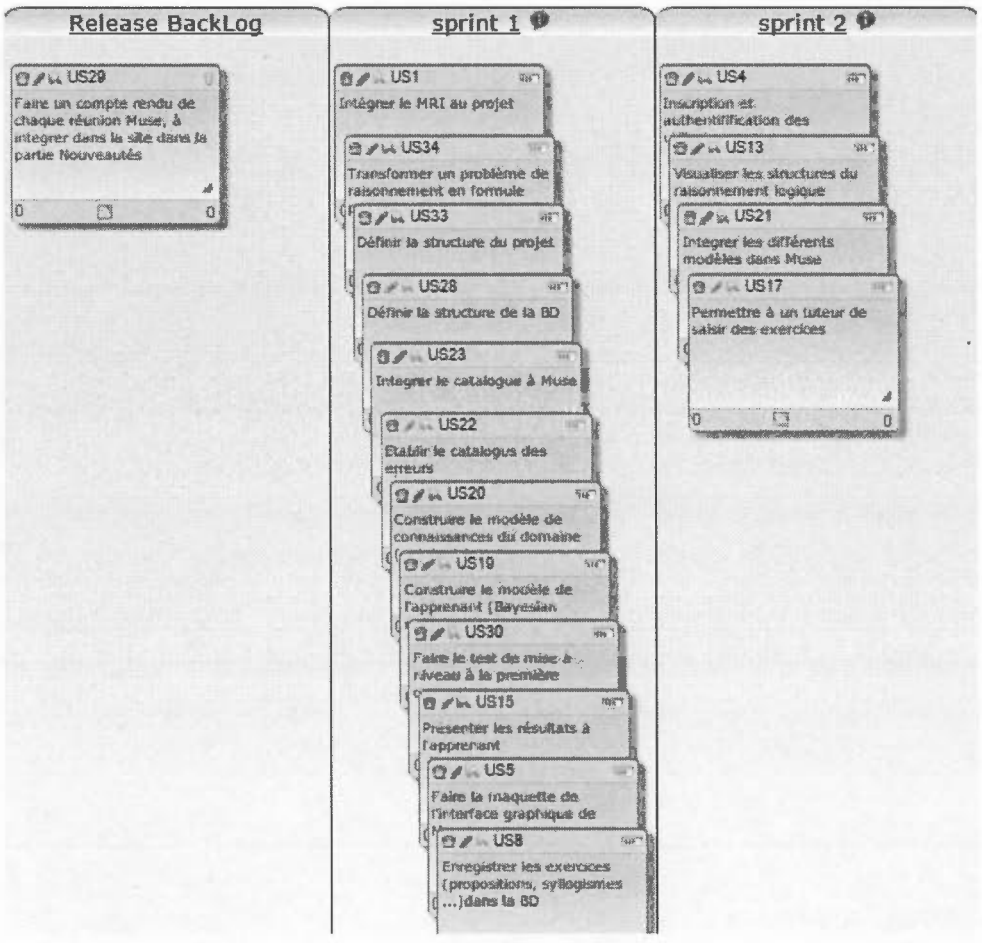
Toutes les US du projet	US en attente dans le backlog	Total des Business Values	Total des Complexités
32	5	0	0

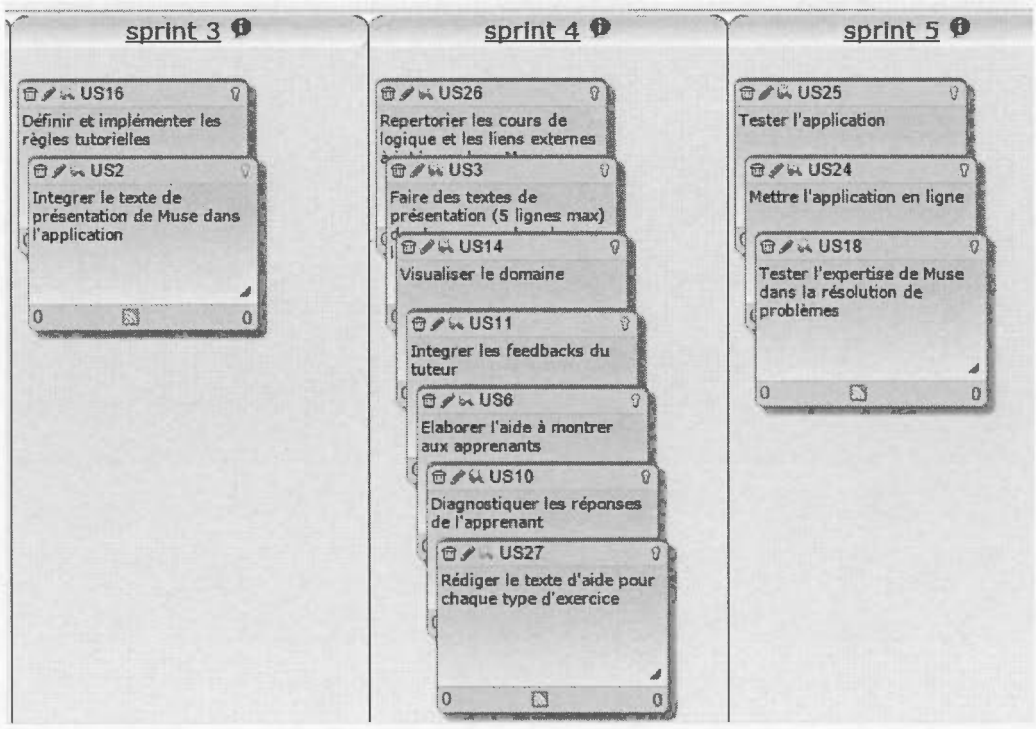
Obstacles		
Risque	0	0
Problème	0	0

Tâches	To Do	Done	In Progress

APPENDICE D

SCRUM : LISTE DES SPRINTS

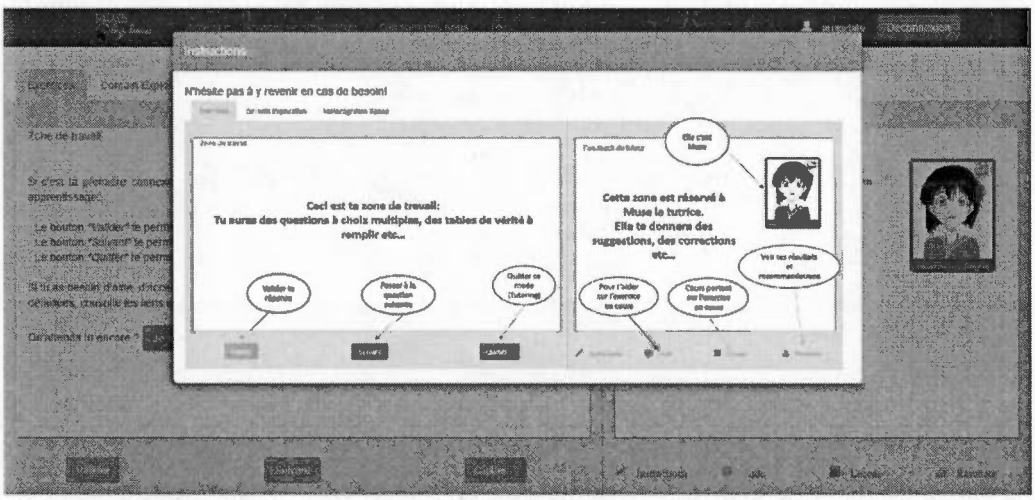



sprint 3
US16
Définir et implémenter les règles tutorielles
US2
Integrer le texte de présentation de Muse dans l'application
0
0
sprint 4
US26
Repertorier les cours de logique et les liens externes
US3
Faire des textes de présentation (5 lignes max)
US14
Visualiser le domaine
US11
Integrer les feedbacks du tuteur
US6
Elaborer l'aide à montrer aux apprenants
US10
Diagnostiquer les réponses de l'apprenant
US27
Rédiger le texte d'aide pour chaque type d'exercice
0
0
sprint 5
US25
Tester l'application
US24
Mettre l'application en ligne
US18
Tester l'expertise de Muse dans la résolution de problèmes
0
0

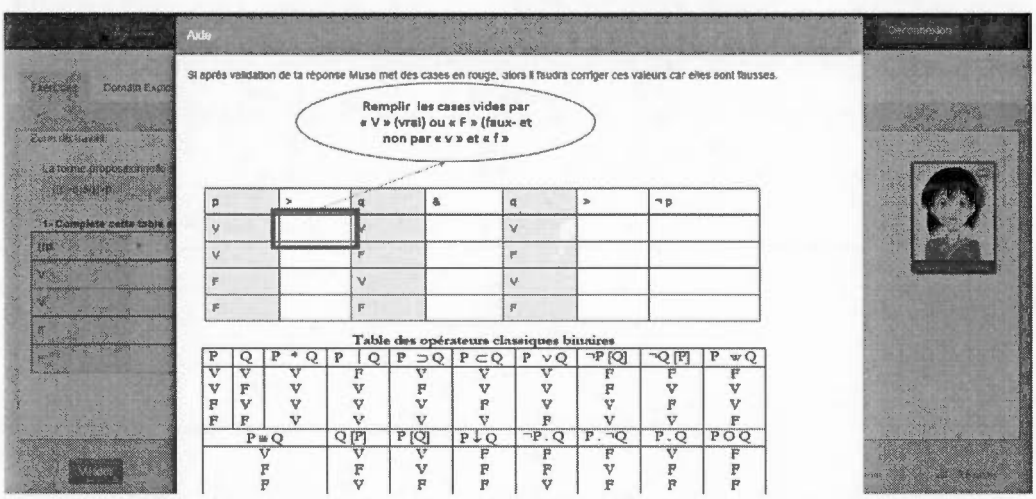
APPENDICE E

CAPTURES D'ECRAN

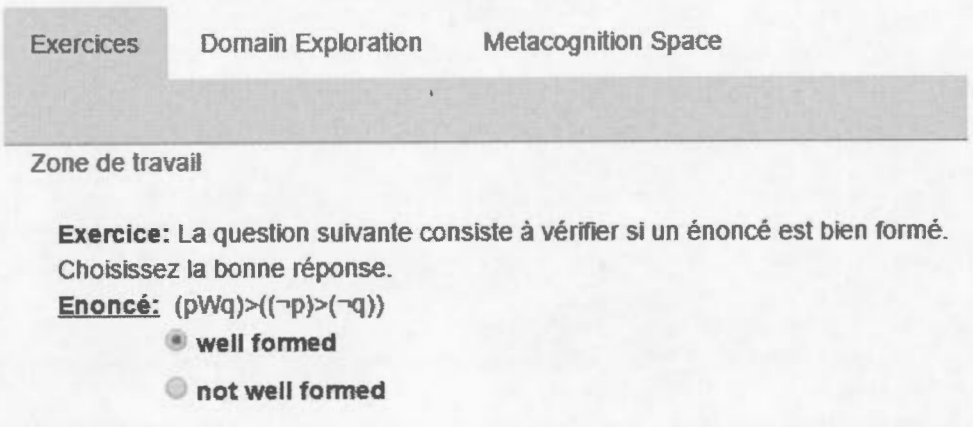
Interface apprenant - Instructions



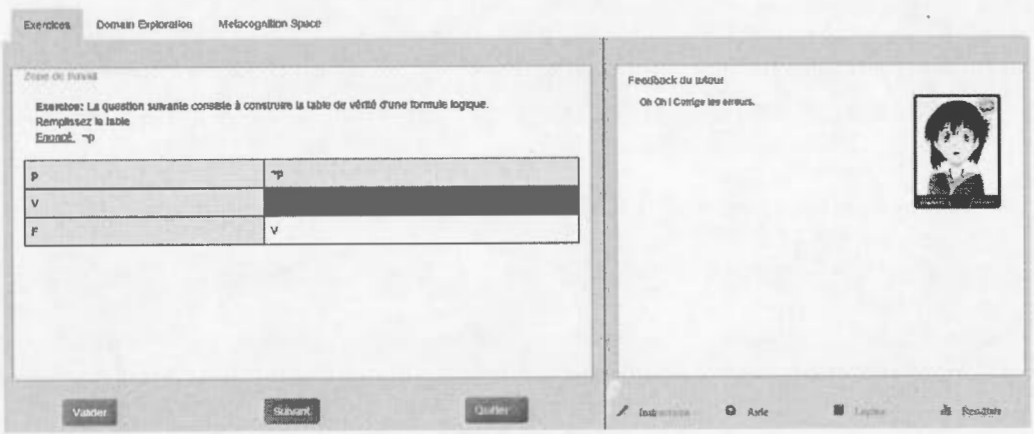
Interface apprenant – Aide de construction de table de vérité



Interface apprenant – Exercice de vérification d'EBF



Interface apprenant – Exercice de construction de table de vérité



BIBLIOGRAPHIE

01net. Logiciel R Retrieved from http://www.01net.com/telecharger/windows/Programmation/creation/fiches/106260.html

Aïmeur, E., & Frasson, C. (2000). Reference model for evaluating intelligent tutoring systems. *Université de Montréal, TICE*.

Ainsworth, S. (2005). *Evaluation methods for learning environments.* Paper presented at the AIED.

Aleven, V. (2010a). *Intelligent Tutoring Systems: 10th International Conference, ITS 2010, Pittsburgh, PA, USA, June 14-18, 2010, Proceedings*: Springer Science & Business Media.

Aleven, V. (2010b). Rule-based cognitive modeling for intelligent tutoring systems *Advances in intelligent tutoring systems* (pp. 33-62): Springer.

Alonso, J. A., Aranda, G. A., & Martn–Mateos, F. J. (2007). KRRT: Knowledge representation and reasoning tutor system *Computer Aided Systems Theory–EUROCAST 2007* (pp. 400-407): Springer.

Amiguet, M. Raisonnement automatique: L'approche « logique » https://www.matthieuamiguet.ch/media/documents/MA-IARTI-04-RaisonnAutom.pdf Retrieved from https://www.matthieuamiguet.ch/media/documents/MA-IARTI-04-RaisonnAutom.pdf

Anderson, J. R. (1983). A spreading activation theory of memory. *Journal of verbal learning and verbal behavior, 22*(3), 261-295.

Andler, D. (2005). Les neurosciences cognitives: une nouvelle "nouvelle science de l'esprit"? *PSN, 3*(2), 74-87.

André, R. (2005). *Le syllogisme : notes de cours et illustrations. Cégep de Lévis-Lauzon, Département de mathématiques*.

Andre, T. (1997). Construction: Implications for Instructional Design. *Instructional Design: International Perspectives. Theory, research, and models. Vol. 1*, 243.

Anellis, I. (2004). The genesis of the truth-table device. *Russell: the Journal of Bertrand Russell Studies, 24*(1).

Angeles, P. A. (1981). *Dictionary of philosophy*: HarperCollins Publishers.

Barnes, J. (1969). Aristotle's theory of demonstration. *Phronesis*, 123-152.

Barnes, T. (2006). *Evaluation of the q-matrix Method in Understanding Student Logic Proofs.* Paper presented at the FLAIRS Conference.

Belghith, K., Nkambou, R., Kabanza, F., & Hartman, L. (2012). An intelligent simulator for telerobotics training. *Learning Technologies, IEEE Transactions on, 5*(1), 11-19.

Brusilovsky, P., & Millán, E. (2007). *User models for adaptive hypermedia and adaptive educational systems.* Paper presented at the The adaptive web.

Bull, S. (2004). Supporting learning with open learner models. *Planning, 29*(14), 1.

Bull, S., & Kay, J. (2010). Open learner models *Advances in intelligent tutoring systems* (pp. 301-322): Springer.

Carolane, M. (2015). *Modélisation des connaissances de l'apprenant par un réseau bayésien dans le cadre d'un système tutoriel intelligent : MUSE logique : rapport de stage. [Document non publié]. Université du Québec à Montréal.*

Christophe, A. (2015). *Participation au développement du Système Tutoriel Intelligent Muse Logique : rapport de stage. [Document non publié]. Université du Québec à Montréal.*

Clancey, W. J. (1992). Model construction operators. *Artificial Intelligence, 53*(1), 1-115.

Codetrails. An introduction to Bayesian Networks with Jayes. Retrieved from http://www.codetrails.com/blog/introduction-bayesian-networks-jayes

Collis, B., De Boer, W., & Slotman, K. (2001). Feedback for web-based assignments. *Journal of Computer Assisted Learning, 17*(3), 306-313.

Conati, C., Gertner, A., & Vanlehn, K. (2002). Using Bayesian networks to manage uncertainty in student modeling. *User modeling and user-adapted interaction, 12*(4), 371-417.

Croy, M., Barnes, T., & Stamper, J. (2008). *Towards an Intelligent Tutoring System for propositional proof construction*: na.

Cummins, D. D., Lubart, T., Alksnis, O., & Rist, R. (1991). Conditional reasoning and causation. *Memory & Cognition, 19*(3), 274-282.

Documents, D.-D. Bibliothèque graphique JavaScript d3js. Retrieved from http://d3js.org/

Eduscol, p. n. d. p. d. l. é. e. F. Gestion de projet agile. Retrieved from http://eduscol.education.fr/sti/sites/eduscol.education.fr.sti/files/ressources/techniques/2517/2517-gestion-de-projet-agile.pdf

Elayne, D. M. B. (2009). Étude des effets de supports didactiques numériques, médiateurs dans la conceptualisation en statistique. *(Thèse de doctorat). Université de Lyon 2.* Retrieved from http://theses.univ-lyon2.fr/documents/getpart.php?id=lyon2.2009.de_moura_braga_e&part=224275

Evans, J. S. B. (2002). Logic and human reasoning: an assessment of the deduction paradigm. *Psychological bulletin, 128*(6), 978.

Evans, J. S. B., Newstead, S. E., & Byrne, R. M. (1993). *Human reasoning: The psychology of deduction*: Psychology Press.

formation, M. Tutoriel JEE. Retrieved from https://www.mistra.fr/tutoriels-java/tutoriel-jee.html

Fournier, V. (2010). *Un modèle hybride pour le support à l'apprentissage dans les domaines procéduraux et mal définis. (Thèse de doctorat) . Université du Québec à Montréal. Récupéré d'Archipel, l'archive de publications électroniques de l'UQAM http://www.archipel.uqam.ca/3606.*

Heckerman, D. (2008). A tutorial on learning with Bayesian networks *Innovations in Bayesian Networks* (pp. 33-82): Springer.

Huertas, A., Humet, J. M., López, L., & Mor, E. (2011). The sell project: a learning tool for e-learning logic *Tools for Teaching Logic* (pp. 123-130): Springer.

Iqbal, M. A., Oppermann, R., & Patel, M. A. (1999). A Classification of Evaluation Methods for Intelligent Tutoring Systems *Software-Ergonomie'99* (pp. 169-181): Springer.

java, D. e. Hibernate. Retrieved from http://www.jmdoudoux.fr/java/dej/chap-hibernate.htm

Jeremić, Z., Jovanović, J., & Gašević, D. (2009). Evaluating an intelligent tutoring system for design patterns: The DEPTHS experience. *Journal of Educational Technology & Society, 12*(2), 111-130.

JESS. Jess the rule engine Retrieved from http://herzberg.ca.sandia.gov/

Legrand, J. (1992). *Le langage Prolog: Exemples en Turbo Prolog*: Editions Technip.

Lesta, L., & Yacef, K. (2002). *An intelligent teaching assistant system for Logic.* Paper presented at the Intelligent Tutoring Systems.

logic, T. p. o. Pol web tutor Retrieved from http://www.poweroflogic.com/cgi/menu.cgi?chapter=1

Mark, M. A., & Greer, J. E. (1993). Evaluation methodologies for intelligent tutoring systems. *Journal of Artificial Intelligence in Education, 4*, 129-129.

Markovits, H. (2013). *The Developmental Psychology of Reasoning and Decision-Making*: Taylor & Francis.

Markovits, H. (2014). On the road toward formal reasoning: Reasoning with factual causal and contrary-to-fact causal premises during early adolescence. *Journal of experimental child psychology, 128*, 37-51.

Markovits, H., Brunet, M.-L., Thompson, V., & Brisson, J. (2013). Direct evidence for a dual process model of deductive inference. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 39*(4), 1213.

Marquis, P., Papini, O., & Prade, H. (2009). Turing et l'Intelligence Artificielle. *CRIL, Lens; LSIS, Marseille; IRIT Toulouse.*

Narciss, S. (2008). Feedback strategies for interactive learning tasks. *Handbook of research on educational communications and technology, 3*, 125-144.

Narciss, S., & Huth, K. (2004). How to design informative tutoring feedback for multimedia learning. *Instructional design for multimedia learning*, 181-195.

Nicol, D. J., & Macfarlane-Dick, D. (2006). Formative assessment and self-regulated learning: A model and seven principles of good feedback practice. *Studies in higher education, 31*(2), 199-218.

Nkambou, R. (2010a). *Advances in intelligent tutoring systems* (Vol. 308): Springer Science & Business Media.

Nkambou, R. (2010b). *Introduction aux systèmes tutoriels intelligents: notes de cours et illustrations, INF7470. Université du Québec à Montréal, Département informatique.*

Nkambou, R., Kenfack, C., Robert, S., & Brisson, J. (2015). The Design Rationale of Logic-Muse, an ITS for Logical Reasoning in Multiple Contexts. In C. Conati, N. Heffernan, A. Mitrovic, & M. F. Verdejo (Eds.), *Artificial Intelligence in Education* (Vol. 9112, pp. 738-742): Springer International Publishing.

Noveck, I., Van der Henst, J., Rossi, S., & Mercier, H. (2007). Psychologie cognitive et raisonnement. *Psychologies du raisonnement. Bruxelles: DeBoeck.*

Ochs, M. (2004). *Systèmes tutoriels émotionnellement intelligents.* Université de Montréal.

Ohlsson, S. (1987). *Some principles of intelligent tutoring.* Paper presented at the Artificial intelligence and education.

Openclassrooms. Organiser son code selon l'architecture MVC Retrieved from https://openclassrooms.com/courses/concevez-votre-site-web-avec-php-et-mysql/organiser-son-code-selon-l-architecture-mvc

Paquette, G. (2011). *Les environnements d'apprentissage intelligents: notes de cours et illustrations, INF5100. Télé-université au Québec, Département informatique*

Pardos, Z. A., & Heffernan, N. T. (2010). Modeling individualization in a bayesian networks implementation of knowledge tracing *User Modeling, Adaptation, and Personalization* (pp. 255-266): Springer.

Perrine, R. (2015). *Etat de l'art, évaluation des systèmes tutoriels intelligents : rapport. [Document non publié]. Université du Québec à Montréal.*

Protégé. Logiciel de création d'ontologies. Retrieved from http://protege.stanford.edu/

Qumer, A., & Henderson-Sellers, B. (2008). An evaluation of the degree of agility in six agile methods and its applicability for method engineering. *Information and software technology, 50*(4), 280-295.

Rialle, V. (1996). IA et sujet humain: entre physis et sémiosis. *Intellectica, 2*(23), 121-153.

Rips, L. J. (2002). Reasoning. *Stevens' Handbook of Experimental Psychology*.

Robert, S. (2005). Chapter 31 - Categorization, Reasoning, and Memory from a Neo-logical Point of View. In H. C. Lefebvre (Ed.), *Handbook of Categorization in Cognitive Science* (pp. 699-717). Oxford: Elsevier Science Ltd.

Robert, S. (2013). *Introduction à la logique, cahier d'exercices et de problèmes: note de cours et illustrations, PHI-1007*. Notes de cours et illustrations. Université du Québec à Montréal.

Robert, S. (2014). *Raisonnement valides et sophismes : notes de cours. Université du Québec à Montréal, Département de philosophie*.

Schwaber, K. (1997). Scrum development process *Business Object Design and Implementation* (pp. 117-134): Springer.

Schwaber, K. (2004). *Agile project management with Scrum*: Microsoft Press.

Stanovich, K. E. (2009). Distinguishing the reflective, algorithmic, and autonomous minds: Is it time for a tri-process theory. *In two minds: Dual processes and beyond*, 55-88.

tartarus, S. French stemming algorithm. Retrieved from http://snowball.tartarus.org/algorithms/french/stemmer.html

Tatsuoka, K. K. (1983). Rule space: An approach for dealing with misconceptions based on item response theory. *Journal of educational measurement, 20*(4), 345-354.

Tchetagni, J., Nkambou, R., & Bourdeau, J. (2007). Explicit reflection in prolog-tutor. *International Journal of Artificial Intelligence in Education (IJAIED), 17*, 169-215.

Tchouanto, P. (2014). *Modularisation des ontologies. (Mémoire de maîtrise). Université du Québec à Montréal. Récupéré d'Archipel, l'archive de publications électroniques de l'UQAM http://www.archipel.uqam.ca/6544/*.

Templin, J. L., & Henson, R. A. (2006). Measurement of psychological disorders using cognitive diagnosis models. *Psychological methods, 11*(3), 287.

Thompson, V. A. (1995). Conditional reasoning: The necessary and sufficient conditions. *Canadian Journal of Experimental Psychology/Revue canadienne de psychologie expérimentale, 49*(1), 1.

Tsamardinos, I., Brown, L. E., & Aliferis, C. F. (2006). The max-min hill-climbing Bayesian network structure learning algorithm. *Machine learning, 65*(1), 31-78.

Vanlehn, K. (2006). The Behavior of Tutoring Systems. *Int. J. Artif. Intell. Ed., 16*(3), 227-265.

Vanlehn, K., Lynch, C., Schulze, K., Shapiro, J. A., & Shelby, R. (2005). *The Andes physics tutoring system: Lessons learned*. Retrieved from

Varin, C. (2007). L'enseignement du raisonnement conditionnel : de la logique aux neurosciences. *(Mémoire de maîtrise). Université du Québec à Montréal.* Retrieved from d'Archipel, l'archive de publications électroniques de l'UQAM www.archipel.uqam.ca/4748/1/M10009

Wenger, E. (2014). *Artificial intelligence and tutoring systems: computational and cognitive approaches to the communication of knowledge*: Morgan Kaufmann.

Wikipedia. Logique. Retrieved from http://fr.wikipedia.org/w/index.php?title=Logique&oldid=116092728

Wikipedia, l. e. l. Protégé. Retrieved from https://fr.wikipedia.org/wiki/Prot%C3%A9g%C3%A9_(logiciel)

Woolf, B. P. (2010). *Building intelligent interactive tutors: Student-centered strategies for revolutionizing e-learning*: Morgan Kaufmann.