الجمهورية الجزائرية الديمقراطية الشعبية

الجمهورية الجزائرية الديمقراطية الشعبية

République Algérienne Démocratique et Populaire

Ministère de l'enseignement supérieur et de la recherche scientifique

Université de 8 Mai 1945 – Guelma -

Faculté des Mathématiques, d'Informatique et des Sciences de la matière

Département d'Informatique

Mémoire de Fin d'études Master

Filière : Informatique

Option : Master Académique

Thème :

Un système tutoriel intelligent pour l'apprentissage du langage Python

Encadré Par :

Farek Lazhar

Présenté par :

Diarra Salimata

Oumou Koulthoum

Juin 2016

Remerciements

Nous tenons tout d'abord à remercier Allah le tout puissant et miséricordieux, qui nous a donné la force et la patience d'accomplir ce Modeste travail.

En second lieu, nous tenons à remercier notre encadreur Mr FAREK Lazhar ses précieux conseils, sa patience et son aide durant toute la période du travail.

Nos vifs remerciements vont également aux membres du jury pour l'intérêt qu'ils ont porté à notre recherche en acceptant d'examiner notre travail et de l'enrichir par leurs propositions.

Ces remerciements vont à l'endroit du corps professoral et administratif de la faculté **des mathématiques, d'informatique et des sciences de la matière.**

Enfin, nous tenons également à remercier toutes les personnes qui ont participé de près ou de loin à la réalisation de ce travail.

MERCI

Dédicaces

Louange à Allah le tout miséricordieux, le très miséricordieux qui nous a permis de réaliser ce travail en toute sérénité.

Ce travail je le dédie principalement à la mémoire de mon père qui m'a toujours encouragé et motivé dans mes études, j'espère avoir été à la hauteur des vos espérances, les mots me manquent pour exprimer ma reconnaissance pour tous les sacrifices que vous avez dû faire pour m'offrir le meilleur. Je prie Dieu de vous accorder une place au paradis.

A ma très chère mère pour son soutien infaillible, son amour, ses bénédictions puisse Dieu vous accordez santé, bonheur et une longue vie.

A mes adorables frères et sœurs, Madou, Bako, Papou, Sékou, Fanta, Mai, Aiché, Bébé... qui m'ont toujours épaulé.

A ma tante et homonyme pour son assistance, sa bonne volonté et d'avoir toujours cru en moi.

Une pensée pour mon binôme Oumy qui a supporté mes petits caprices durant toutes ces années, tu resteras à jamais dans mon cœur.

Je tiens également à remercier en la personne de Hamadi Camara pour sa disponibilité et son soutien dans les moments difficiles.

A tous mes amis, qu'ils trouvent ici l'expression de ma reconnaissance.

Merci

Diarra Salimata

Dédicaces

Je dédie ce travail à :

A ma très chère mère,

Une femme gentille, patiente, forte, dévouée : les mots ne suffisent guère pour exprimer l'attachement, l'amour et l'affection que je porte pour vous et jamais je ne saurais vous remercier quant à votre détermination et aux sacrifices envers mon éducation. Vous êtes et vous serez toujours un modèle pour moi et c'est à vous que je dois cette réussite.

A mon cher père,

A mon grand frère Habchi qui m'a toujours soutenu et cru en moi.

A ma grande sœur Moinadi,

Une fille extraordinaire avec un grand cœur. Appart d'être ma sœur, tu es une amie avec laquelle je peux toujours compter.

A ma très chère sœur et amie Elia Bencheik,

A Abdourahman Ali Salim,

A Fahad, tu es un ami génial et merci de ton soutien malgré la distance.

A Mebarki Nouria,

A toutes les personnes qui m'ont aidé de près ou de loin :

Mes tantes, mes cousins et cousines.

Mes amis(es) : Onzairou Abdallah, Said Assoumani, Faoulat, Toilhati...

A ma très chère binôme et amie Diarra Salimata, qui m'a toujours soutenu, le parcours n'était pas facile mais on peut dire Alhamdouli'lah. Encore une fois merci.

A toutes la promotion 011 de Guelma...

Oumou Koulthoum Mbae Mmadi

Résumé

Ce mémoire a été rédigé dans le cadre d'un mémoire de fin d'étude en master 2 recherche informatique.

Le projet intitulé « Un système tutoriel intelligent pour l'apprentissage du langage Python » consiste à la conception et la réalisation d'une plateforme web destinée pour l'apprentissage du langage Python.

En premier lieu il nous incombait de faire un état de l'art des STI (Systèmes Tutoriels Intelligents) en général.

En second lieu nous avons effectué une étude détaillée sur la conception du dit système partant des fonctions de base du langage Python, du modèle de l'apprenant et celle du tuteur.

La dernière phase consiste à l'implémentation ou le codage de l'application en question. L'élaboration de différentes interfaces et la gestion de la base de données des utilisateurs.

Abstract

This brief was written as part of an end of study in memory of master 2 computer research.

The project "An intelligent tutoring system for learning Python" consists in the design and implementation of a web platform designed for learning Python.

First it was for us to make a state of the art of ITS (Intelligent Tutoring Systems) in general.

Secondly we conducted a detailed study on the design of said system starting from the basic features of the Python language, the learner model and that of the guardian.

The final phase consists in the implementation or the coding of the application in question. The development of various interfaces and management of the user database.

Sommaire

Chapitre 3 : Implémentation du système

Liste des figures

Liste des tableaux

Liste des abréviations

STI : Système Tutoriel Intelligent

EAO : Enseignement Assisté par Ordinateur

OWL : Web Ontology Language

TIC : Technologie de l'Information et de la Communication

QCM : Questionnaire à Choix Multiple

ENT : Environnement Numérique de Travail

HTML: HyperText Mark-up Language

IA: Intelligence Artificielle

BFO: Basic Formal Ontology

SUMO: Suggested Upper Merged Ontology

EC : Expérience Concrète

OR : Observation Réfléchie

CA: Conceptualisation Abstraite

EA : Expérimentation Active

EIAO : Enseignement Intelligemment Assisté par Ordinateur

RDF : Resource Description Framework

SGBD : Système de Gestion de Base de Données

Introduction générale

L'ordinateur est utilisé à des fins pédagogiques depuis son apparition. En tant que dispositif interactif, il peut offrir des possibilités permettant de dispenser des cours adaptés à un grand nombre d'apprenants sur une base individualisée. En effet chaque apprenant a des connaissances antérieures différentes et assimile à sa manière, en conséquence il a des besoins cognitifs différents.

L'enseignement doit alors être adapté à chaque apprenant individuellement. C'est ainsi qu'au fil des années, apparaissent des logiciels éducatifs tentant de fournir des enseignements individualisés.
Si pendant longtemps l'apprenant est ignoré au profit des matières à inculquer, dans ces systèmes il devient un objet de préoccupation.

L'objectif de notre mémoire est de concevoir un système tutoriel intelligent pour l'apprentissage du langage de programmation Python ; à cet effet nous avons organisé le travail en 3 chapitres :

- Dans le premier chapitre nous ferons un point sur l'état de l'art des STI en général, leurs structures et leurs évolutions ;
- Le second chapitre s'articulera autour de la conception du système proprement dit, l'élaboration des différents modèles et des diagrammes en utilisant les ontologies ;
- Enfin le troisième chapitre sera consacré à l'implémentation du système.

Ces chapitres se terminent par une conclusion et des perspectives possibles pour la poursuite de notre travail.

Problématique

Il est vrai que les systèmes tutoriels intelligents sont de plus en plus répandus de nos jours. En effet, les STI se retrouvent dans des secteurs d'activités variés tels : l'armée (exemple Sherlock pour apprendre à réparer les avions F16), la médecine (Guidon pour apprendre à établir de bons diagnostics) …

L'idée est développer une plateforme d'enseignement entièrement dédiée aux étudiants et aux personnes s'intéressant à l'apprentissage du langage Python.
Ce système devra répondre aux attentes des utilisateurs en matière d'ergonomie et disposera de l'apport d'un tuteur intelligent pour une meilleure orientation de l'apprenant dans son cursus.

La plateforme disposera de cours théoriques sous forme de fichiers PDF, vidéos des séances d'exercices selon les gouts et les préférences de l'apprenant.

Le tuteur intelligent sera en mesure de guider l'apprenant tout au long de son cycle d'apprentissage, de l'aider à corriger d'éventuelles erreurs survenues relatives à la syntaxe, à la sémantique et au lexique du langage Python.

Chapitre 1 : Etat de l'art sur les systèmes tutoriels intelligents

1.1. Introduction

L'idée de systèmes d'apprentissage automatisés remonte au début des années soixante (1960), avec les espoirs suscités par le domaine de l'intelligence artificielle, qui a donné lieu aux systèmes d'EAO (Enseignement Assisté par Ordinateur). Les premiers STI sont apparus au début des années 80 à la convergence de plusieurs disciplines définissant le domaine émergent de l'intelligence artificielle en éducation, comme les sciences cognitives, l'informatique ou les sciences de l'éducation. SCHOLAR est le tout premier système du genre. Le livre *Intelligent Tutoring Systems* paru en 1982 est le premier livre sur les STI. C'est une collection d'articles sur des travaux représentant une percée dans la mise en œuvre de concepts et de techniques d'intelligence artificielle pour l'apprentissage. Toutefois, c'est la publication du livre *Artificial Intelligence and Tutoring Systems* de Wenger qui établit réellement les grandes lignes du nouveau domaine. C'est donc en 1987 que débute le développement des STI modernes.

Les Systèmes Tutoriels Intelligents (STI) sont censés afficher un comportement proche de celui d'un tuteur humain. Ils sont capables de gérer l'interaction avec leurs utilisateurs. De plus, leurs stratégies tutorielles permettent de déterminer quand et comment organiser les interventions dans le processus d'apprentissage afin de réorienter la démarche de l'utilisateur.

1.2. Définition

Les systèmes tutoriels intelligents (STI) sont des environnements d'apprentissage informatisés qui visent à imiter le comportement d'un tuteur humain dans ses capacités d'expert pédagogue et d'expert du domaine. Ainsi, tout comme un tuteur, les logiciels de ce type ont le potentiel d'amener l'apprenant à réaliser une tâche et de fournir des rétroactions pertinentes sur leurs actions.

Les STI sont essentiellement des environnements de résolution de problèmes ou d'exercices. Ils favorisent l'apprentissage dans un domaine précis en guidant et en assistant l'apprenant. Parfois, ils exposent d'abord le contenu du domaine à l'apprenant ; parfois, ils présentent directement les exercices qui permettront d'assimiler les connaissances.

1.3. Architecture générale d'un STI

De manière générale, les STI se composent de quatre modules principaux : le modèle du domaine, le modèle de l'apprenant, le modèle pédagogique et le modèle d'interface.

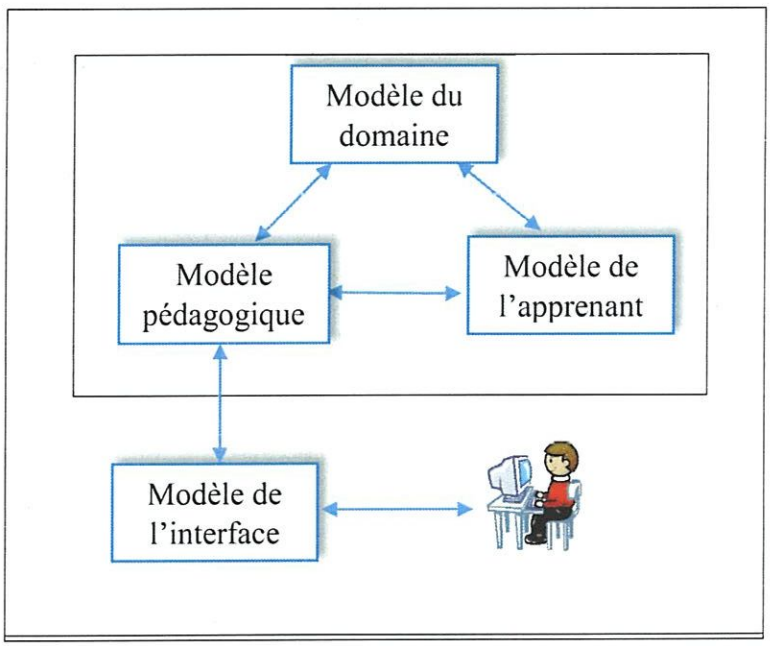

Figure 1.1 : *Composants d'un STI.*

1.3.1. Modèle du domaine

Le modèle du domaine est une représentation de la connaissance que l'apprenant doit acquérir, il représente l'ensemble des connaissances expertes sur le domaine d'apprentissage. Ce modèle offre au reste du système la possibilité, d'une part de vérifier si les actions effectuées par l'apprenant sont correctes, et d'autre part de proposer des solutions ou des explications aux problèmes rencontrés par l'apprenant. Il contient également des informations permettant d'interpréter ces connaissances, afin de permettre l'exécution des actions pour « *faire à la place* » de l'apprenant. Le modèle du domaine contient alors des connaissances qui

sont comparables aux connaissances déclaratives et procédurales misent en jeu lors de l'apprentissage de compétences en psychologie [4].

1.3.2. Modèle pédagogique (tuteur)

Le modèle pédagogique sert à instrumenter l'intervention pédagogique du système. Il utilise les informations du modèle de l'apprenant pour planifier et déterminer les aspects du modèle de connaissances (ou modèle du domaine) à présenter à l'apprenant de façon adéquate. En conséquence, ce modèle doit être capable de : modéliser l'interaction entre le tuteur et l'apprenant ; concevoir les tâches exposées à l'apprenant de façon adaptative ; fournir des rétroactions et de l'aide adaptée à l'apprenant (p. ex., il pourrait fournir de la rétroaction de façon proactive ou réactive) ; de mener des activités pédagogiques telles qu'accorder des explications ou des conseils et/ou faire passer des tests.

L'objectif final visé par ce modèle est de minimaliser les différences entre les connaissances de l'expert du domaine et l'apprenant. Pour ce faire, les interventions potentielles du système s'appuient sur des stratégies pédagogiques dont les principales sont les suivantes :

- Le « coaching » : il consiste à conseiller et guider l'apprenant lorsqu'il s'éloigne de la solution [12]. Ce mécanisme détermine le moment convenable pour interrompre l'apprenant et intervenir. Cette intervention peut avoir des formes aussi variées que : un avertissement pour signaler à l'apprenant qu'il a pris une mauvaise décision ou un encouragement lorsqu'il répond correctement à une question, un conseil pour remédier à une situation précise, des renseignements détaillés, etc.

- L'apprentissage par perturbation : il peut se faire en employant un agent perturbateur qui va parfois fournir des réponses correctes et d'autres fois, essayer d'induire l'apprenant en erreur en lui donnant de mauvaises réponses. Le but ici est de vérifier et d'améliorer la confiance et l'estime de soi chez l'apprenant.

- L'apprentissage socratique : il consiste à questionner l'apprenant sur ses erreurs pour le faire progresser. La figure ci-après nous donne un exemple de règle socratique [12].

SI L'étudiant donne une explication d'un ou plusieurs facteurs qui est insuffisante ALORS Formuler une règle générale pour démontrer que les facteurs sont suffisants. Demander à l'étudiant si la règle est vraie.

Figure 1.2 : *Exemple de règle socratique* [12]

- Le guidage : cette stratégie correspond aux environnements de type micromondes [12].

Il faut noter que l'implantation des stratégies pédagogiques pose des difficultés non négligeables. Par ailleurs, des recherches sont en cours pour que les STI puissent utiliser plusieurs stratégies pédagogiques. Toutefois, ces recherches sont confrontées au problème du fort lien qui existe entre la stratégie pédagogique et la représentation des connaissances incluse dans le modèle du domaine et utilisée par le modèle de l'apprenant.

1.3.3. Modèle Apprenant

Ce modèle contient les connaissances sur la matière (les connaissances théoriques et les connaissances pratiques (les exercices)). Il est conçu de façon à garder les informations sur les processus cognitifs de l'apprenant et de stocker ses préférences et/ou ses expériences. C'est a partir du modèle de l'apprenant que le tuteur adapte l'environnement d'apprentissage afin de répondre aux besoins, objectifs et intérêts de l'apprenant et Il permet aussi de surveiller et d'évaluer le rendement de l'apprenant et également de mettre à jour son modèle.

Le modèle apprenant qui permet de sauvegarder les informations sur les connaissances et les compétences actuelles s'appelle *modèle de recouvrement*, ce modèle utilise la méthode « overlay » et peut évaluer les connaissances en les comparants à celles du tuteur. La méthode de recouvrement représente l'ensemble des connaissances de l'apprenant comme un sous-ensemble des connaissances de l'expert du domaine.

Cette méthode a été utilisée par Carbonell dans son système SCHOLAR pour modéliser l'usager. Ce modèle présente toutefois un désavantage parce qu'il ne peut pas déterminer la cause d'une erreur. Le système peut seulement conclure que le comportement incorrect de l'apprenant est dû à la maîtrise insuffisante du contenu enseigné.

Pour corriger cet inconvénient, Brown et Burton ont proposé en 1978 le modèle des erreurs qui utilisent la méthode Buggy. Ce modèle peut montrer à l'apprenant les connaissances, détecter les erreurs, et indiquer pourquoi l'apprenant a commis ces erreurs. D'une manière générale, il y a deux types de modèles de l'apprenant : *Le modèle de recouvrement (modèle overlay)* et *le modèle des erreurs (modèle Buggy)*.

Les deux modèles peuvent évaluer les connaissances par leurs interactions avec le système ou en utilisant des travaux d'évaluation.

La modélisation de la connaissance de l'étudiant et de son comportement d'apprentissage utilise fondamentalement deux procédures : inventorier les domaines que les étudiant ont maîtrisés ou qu'ils ont tentés d'apprendre ; 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. On peut distinguer quatre grandes sources d'information pour la mise à jour du modèle de l'étudiant :

1. Le comportement de résolution de problèmes de l'étudiant (ou le progrès de sa performance) tel qu'observé par le système via son module d'interface ;
2. Les questions directes posées par l'étudiant ;
3. Les hypothèses basées sur l'expérience d'apprentissages de l'élève ;
4. Les hypothèses basées sur certaines mesures de difficulté de la matière présentée.

Cependant, la plupart des systèmes n'utilisent en général que les deux premières sources pour mettre à jour le modèle étudiant. [1.9]

Le modèle de l'apprenant doit être *dynamique* parce qu'il répond à chaque étape de l'apprentissage de l'apprenant humain.

A) La théorie overlay

Le système SCHOLAR, développé par Carbonell (1970), est le premier STI à employer la théorie *overlay* pour modéliser son usager. Ce système, dont l'objectif est d'obtenir la représentation la plus fidèle possible de la connaissance détenue par l'étudiant, génère d'abord la représentation d'un étudiant idéal qui maîtrise tout le contenu pédagogique. Ensuite, l'étudiant réel est progressivement identifié grâce à une démarche de comparaison entre sa performance et celle de l'étudiant idéal.

La théorie *overlay* a également été proposée par Carr et Goldstein (1977), pour la modélisation de l'usager du système WUSOR-II. Ces chercheurs ont retenu comme données de modélisation de l'étudiant les indices cognitifs suivants : la réceptivité aux interventions

tutoriels, le besoin de répétition et le degré d'oubli. L'action de l'étudiant est donc toujours analysée par rapport à ces trois indices. Il y a donc tout un ensemble de règles responsables de l'établissement d'un lien entre les rétroactions de l'étudiant et ces indices cognitifs.

Il est possible d'identifier deux limitations de la théorie *overlay*. Premièrement, les règles de modélisation de l'étudiant ne constituent pas un module indépendant, mais un sous-ensemble des règles de codification de l'expertise. Le processus de modélisation doit donc se limiter aux techniques de codification de connaissances choisies par le concepteur du système tutoriel. Deuxièmement, il n'y a pas de règles relatives aux erreurs commises par l'étudiant, le système peut seulement conduire à la conclusion que le comportement incorrect de l'apprenant est dû à la maîtrise insuffisante du contenu. Dans ce contexte, les déviations et les distorsions de connaissances ne sont pas considérées dans la modélisation de l'étudiant.

B) La théorie de la réparation (Repair Theory)

D'après Burns et Capps (1988), il y a deux niveaux de complexité dans la question de la modélisation de l'étudiant. Le premier niveau correspond à la vérification des écarts entre la connaissance de l'étudiant et celle du tuteur. Cette vérification, à travers laquelle le système peut mettre au point une représentation de son usager, constitue la méthode *overlay*', discutée ci-avant. Pour représenter l'étudiant, cette théorie n'utilise que des informations relatives à ses connaissances incomplètes. Le deuxième niveau de complexité correspond à la représentation des connaissances incomplètes et incorrectes, impliquant l'organisation d'un catalogue d'erreurs. Ce niveau correspond aussi à la représentation des points de vue ou des opinions que l'étudiant a sur les connaissances qui lui sont communiquées.

Pour le développement de mécanismes de modélisation de l'étudiant plus fidèles aux comportements pris en compte, les recherches les plus récentes mettent l'accent sur l'utilisation d'un catalogue regroupant les erreurs les plus communément commises par l'étudiant. Ces erreurs sont à la base de la théorie de la réparation, proposée par Brown et VanLehn (1980). Les modules de l'étudiant, développés selon cette théorie, tentent d'identifier chez l'élève des déviations de la connaissance transmise par le système. Dès que ces déviations sont identifiées, des actions pédagogiques sont déclenchées par le système dans le but de réparer les erreurs sous-jacentes. La théorie de la réparation a été employée dans la conception du système DEBUGGY (Burton, 1982). Le modèle de l'étudiant produit par ce système est mis au point selon le principe de la décomposition de la rétroaction de l'usager,

identifiant et isolant ses procédures incomplètes et incorrectes. Une fois identifiées, les erreurs sont classées dans un catalogue, qui devient un paramètre de modélisation.

Selon la théorie de la réparation, lorsqu'une erreur n'est pas cataloguée, le module de l'étudiant essaie de combiner d'autres erreurs afin d'identifier celle en question. Brown et VanLehn (1980) mentionnent trois techniques d'identification d'erreurs et d'obtention d'un catalogue :

1. les erreurs peuvent être identifiées dans la documentation ;
2. les erreurs peuvent être identifiées à travers l'analyse du comportement d'un groupe d'étudiants ;
3. s'il y a une théorie d'apprentissage sur le contenu pédagogique à être enseigné, les erreurs peuvent être prévues en fonction de cette théorie.

C) Aspect du modèle apprenant

Le rôle de la modélisation de l'apprenant était initialement clairement différencié : l'apprenant était l'objet modélisé. Mais pour que ce rôle soit bien rempli, le modèle de l'apprenant peut être caractérisé par un certain nombre de critères relatifs aux méthodes de construction de ce modèle (Holt & al., 1994 ; Kass, 1987 ; Dillenbourg & Self, 1991). Donc, il peut être :

1. **implicite** lorsque les informations décrivant le comportement de l'apprenant et influençant le déroulement de l'interaction avec le système sont incorporés dans ce dernier ;
2. **explicite** lorsque les informations sur l'apprenant sont intégrées et codées dans le système de manière explicite dans le but de gérer l'interaction avec l'apprenant ;
3. **statique** lorsque les connaissances de l'apprenant sont déterminées avant toute utilisation et ne peuvent être l'objet d'une modification en cours de session ;
4. **dynamique** lorsqu'on peut ajouter ou modifier les données en cours de session ;
5. **spécifique** lorsqu'il peut être adapté à une catégorie d'apprenants ;
6. **de surface** lorsqu'il contient des informations limitées qui ne peuvent expliquer l'état cognitif de l'apprenant ;
7. **profond** dans le cas où il contient des informations plus représentatives de l'état cognitif de l'apprenant.

1.3.4. Modèle d'Interface

L'interface sert de couche de communication entre l'apprenant et le STI. Son élaboration est basée sur des principes d'ergonomie de l'interaction afin de simplifier la communication entre le système et l'utilisateur. De nos jours beaucoup d'efforts sont mis dans le raffinement des interfaces en fonction de l'environnement d'apprentissage. Ainsi, il existe plusieurs types d'interfaces parmi lesquels nous pouvons citer : le micromonde, la simulation et l'hypermédia.

1.4. Représentation des connaissances

Une définition claire des termes et des notions définissant le domaine que l'on veut enseigner par un STI est primordiale, et il est impératif de s'assurer que le STI puisse manipuler aisément ces notions. Si pour un logiciel fonctionnant sur le principe des questions à choix multiples, cela se limite à énoncer clairement les questions à poser et de les diversifier pour couvrir l'ensemble du curriculum, la représentation des connaissances pour l'enseignement intelligent implique plus de contraintes. L'enseignement par un STI se fait en utilisant les mêmes notions qu'un tuteur humain. Un STI raisonne en termes du domaine et fournit les étapes d'une solution, des suggestions, des explications et des justifications de façon similaire à ceux d'un tuteur humain. À noter que par connaissances relatives au domaine, on entend aussi bien la définition des notions théoriques (**connaissances sémantiques**, ex. « résistance », « courant », état initial du problème) que la représentation des méthodes pratiques de résolution de problèmes et des actions qui y sont associées (**connaissances procédurales**, ex. « Application de la loi d'Ohm », « Recherche de la valeur de la résistance équivalente »).

1.4.1. Connaissances sémantiques

Il pourrait sembler que les connaissances sémantiques aient une importance moindre que les procédurales dans les STI parce que ces derniers offrent une aide à la résolution de problèmes à des étudiants censés avoir un savoir suffisant sur les notions théoriques du domaine. Par ailleurs, des STI dédiés à un domaine particulier n'ont pas forcément besoin d'un formalisme de représentation de connaissances sémantiques riche et polyvalent. Les concepteurs de logiciels commerciaux et les STI dédiés n'ont pas à se soucier de la

réutilisabilité du formalisme et choisissent, ou développent des formalismes spécifiquement adaptés au domaine enseigné. En analyse de circuits par exemple, la représentation est souvent sous forme de « *frames* ».

Le calcul des prédicats est un puissant outil de représentation. Les inférences représentent des processus de raisonnement permettant la résolution de problèmes entiers. Cependant, sa rigidité et son expressivité restreinte en font un outil peu utilisé en STI. La description d'un vaste domaine, ainsi que des notions abstraites ne serait pas à la portée d'un expert dudit domaine qui n'aurait pas une grande expérience avec ce formalisme. Des raisonneurs artificiels existent pour le calcul des prédicats de premier ordre. Cependant, le raisonnement sur du calcul d'ordre supérieur, souvent nécessaire dans des domaines complexes, est beaucoup plus problématique.

Les logiques de description sont une combinaison de la richesse expressive des réseaux sémantiques et des possibilités d'inférence du calcul des prédicats. Ce formalisme, associé au langage OWL (pour « *Web Ontology Language* »), a été utilisé pour représenter les connaissances sémantiques de plusieurs domaines dans le cadre de notre projet.

1.4.2. Connaissances procédurales

Les domaines enseignés par les STI auxquels nous nous intéressons nécessitent, dans la plupart des cas, l'acquisition d'habiletés menant à des changements dans l'état d'un problème. Ces changements peuvent être physiques, comme l'apprentissage de pilotage d'aéronefs, ou symboliques comme la résolution d'un problème mathématique (transformation d'inconnues en connues). L'enseignement de la géographie ou de l'histoire de l'art, par exemple, quoique faisable, est d'un intérêt limité du point de vue procédural.

La quasi-totalité des STI enseignant des habiletés cognitives travaille avec des connaissances procedurales ecrites sous forme de règles de production. Nous citerons brièvement quelques autres techniques de représentation utilisables.

1.4.3. Connaissances erronées

Les connaissances sémantiques et procédurales ne correspondent pas automatiquement à des connaissances correctes, c'est-à-dire des connaissances qu'un expert du domaine

utiliserait. Étant donné que les connaissances représentées sont également utilisées pour modéliser les connaissances d'un étudiant, représenter des connaissances relatives à des erreurs fréquemment commises est d'un réel intérêt. Le modèle de représentation dit par recouvrement ignore ces connaissances erronées et définit les connaissances d'un apprenant comme incluses dans les connaissances d'un expert du domaine, tandis que le modèle de représentation par perturbation stipule que les connaissances de l'apprenant sont constituées de connaissances correctes, faisant partie des connaissances de l'expert, et de connaissances erronées. Un exemple de connaissance erronée récurrente en matière d'analyse de circuits pourrait être de confondre les lois des résistances équivalentes série et parallèle. À ce sujet, notons la nécessité de différencier les erreurs d'inattention des erreurs dues à un vrai manque de connaissances.

Certains STI, permettent la représentation de connaissances erronées afin d'y associer des connaissances et des comportements pédagogiques propres.

1.5. Exemples de systèmes tutoriels intelligents

Voici quelques exemples de STI aux domaines d'applications et particularités disparates :

Projets	Caractéristiques
SCHOLAR (Carbonnell, 1973)	Tuteur sur la géographie de l'Amérique du sud mettant en œuvre des dialogues à initiative mixte.
BUGGY(Brown, 1978)	Tuteur en arithmétique sur la soustraction, basé sur les erreurs possibles de l'apprenant représentées explicitement.
PROUST (Johnson & Soloway, 1984)	Tuteur commercialisé d'analyse des erreurs de l'élève en programmation Pascal, par comparaison de sa solution avec des dérivations de schémas de solutions.
GEOMETRY TUTOR (Anderson, 1985)	Tuteur qui fournit une interaction originale permettant une visualisation du raisonnement géométrique. Il utilise un modèle de l'élève incorporant des connaissances fausses.
GUIDON (Clancey, 1987)	Tuteur sur les maladies infectieuses du sang. L'élève joue le rôle d'un médecin consultant. L'expertise pédagogique et le

	modèle de l'élève y sont explicites.

Tableau 1.1. *Exemples des STI.*

1.6. Conclusion

Dans ce chapitre nous avons présenté les différents modèles qui composent un STI (Système Tutoriel Intelligent), les connaissances qui vont servir de base au raisonnement pédagogique de notre système. Ces connaissances sont contenues dans des modèles inspirés de ceux qui sont utilisés classiquement dans un STI.

Nous avons détaillé les modèles du domaine, de l'apprenant, des erreurs et de l'interface et également on s'est fait une idée sur la dynamique entre les modèles, qui permet de mettre à jour les connaissances de chacun et définit un processus amenant à une décision pédagogique.

Dans ce qui suit nous entamerons la conception proprement dite de notre système tutoriel.

Chapitre 2 : Conception du système

2.1. Introduction

Une nouvelle génération de systèmes avancés d'apprentissage doit intégrer des nouvelles approches pédagogiques donnant à l'apprenant un rôle actif pour apprendre et construire ses connaissances. Ces systèmes doivent être plus interactifs, mais surtout intégrer une vision plus centrée sur l'utilisateur. L'objectif de ces systèmes est d'adapter la présentation de la connaissance et aider l'apprenant à naviguer à travers le graphe composé par l'ensemble des pages et des liens.

L'ingénierie ontologique est devenue un thème récurrent dans les milieux de recherche du domaine des systèmes tutoriels intelligents. L'ingénierie des connaissances de type ontologique serait une piste de travail à privilégier pour l'analyse, la conception, et le développement d'environnements interactifs d'apprentissage à distance.

2.2. L'ontologie en Intelligence Artificielle

Situées au carrefour de plusieurs grands domaines informatiques (ingénierie des connaissances, intelligence artificielle, etc.) mais aussi en psychologie ou en cognition, les ontologies sont aujourd'hui considérées comme l'un des paradigmes clés pour l'interopérabilité sémantique. Ainsi, de l'intelligence artificielle au web sémantique l'information n'est plus considérée uniquement au travers du prisme de la donnée, mais aussi au travers de la représentation des connaissances.

2.3. Origines et définitions

2.3.1. Contexte d'apparition des ontologies

Bien qu'originellement rattachée au domaine de la philosophie, la notion d'ontologie est considérée dans ce manuscrit sous l'angle de l'Intelligence Artificielle (IA). Conçues comme une réponse aux problèmes posés par l'intégration des connaissances au sein des systèmes

informatiques, les ontologies apparaissent comme une clé essentielle pour la manipulation automatique de l'information au niveau sémantique.

Historiquement, l'origine des ontologies remonte aux années 80 lors de la réalisation des premiers systèmes experts. Les tentatives de modélisation des processus cognitifs, sous la forme de bases de connaissances, s'avérèrent être un processus complexe nécessitant un temps considérable. Dès lors, afin de favoriser le partage des connaissances, il apparait nécessaire d'implémenter de nouveaux formalismes capables de connaître à la fois les termes utilisés, mais aussi la sémantique associée. De ce fait, et notamment sous l'impulsion d'un groupe de chercheurs américains participant au projet Knowledge Sharing Effort, une nouvelle forme de représentation des connaissances répondant aux contraintes observées a été définie : l'ontologie.

2.3.2. Définitions

D'une vision purement philosophique à son application en IA, de multiples définitions et interprétation de la notion d'ontologie ont été proposées. Ainsi, Guarino note que ce terme est rattaché à sept différentes notions.

Originellement la première définition, et aussi la plus couramment citée, est celle donnée par Gruber pour qui une ontologie est :

« Unc spécification explicite d'une conceptualisation » ?? Ref

Néanmoins, de nombreux auteurs lui reprochent son caractère trop vague et générique. C'est pourquoi c'est celle donnée par Studer qui est habituellement préférée. Celle-ci ajoute à la définition de Gruber la notion de partage. Une ontologie se définit alors comme : « une spécification Formelle et explicite d'une conceptualisation partagée »

Cette définition fait ressortir trois notions particulièrement importantes dans le domaine des ontologies :

•**Formel** : signifie que la conceptualisation et la représentation du domaine doivent être standardisées et utilisables par un système informatique

•**Explicite** : spécifie que les concepts utilisés tout comme les contraintes sont définis de façon déclarative

•**Conceptualisation** : souligne le fait qu'une ontologie n'est qu'une abstraction du monde réel et que les termes utilisés ainsi que leurs relations doivent être décrit sans ambiguïté.

•**Partage** : implique que les ontologies favorisent une connaissance consensuelle

D'une manière plus formelle, une ontologie est constituée d'un vocabulaire spécifique permettant une description du domaine sous la forme d'un graphe constitué de concepts et de relations. Celui-ci peut alors être décrit de la manière suivante : 0 : {C, R, HC, rel, A0} où c'est un ensemble de concepts, R un ensemble de relation entre ces concepts, HC une hiérarchie entre ces concepts, rel une fonction spécifiant la relation de r et enfin A0 un ensemble d'axiomes exprimé dans un langage ontologique.

2.4. Les composants d'une ontologie [2]

Comme cela a été mentionné, les ontologies fournissent un vocabulaire du domaine et définissent, de façon plus ou moins formelle, le sens des termes et des relations entre ces derniers. Pour cela, les connaissances intégrées dans une ontologie sont formalisées en s'appuyant sur cinq types de composants : les concepts, les relations, les fonctions, les instances et les axiomes :

•**Concept** : terme que l'on retrouve également dans la littérature sous le nom de classe, il fait référence à la représentation d'un objet matériel, d'une notion ou d'une idée dans l'ontologie.

Il s'agit, d'abstractions pertinentes d'un fragment du monde réel en fonction d'un domaine d'application. Un concept peut être classé selon plusieurs dimensions : le niveau d'abstraction (concret ou abstrait), l'atomicité (élémentaire ou composé) et le niveau de réalité (réel ou fictif). Selon Uschold, un concept se compose de trois parties : (1) un ou plusieurs termes, (2) une notion et (3) un ensemble d'objets. Le ou les termes permettent d'identifier le concept. La notion, aussi appelée intention du concept, désigne la sémantique du concept défini au travers de ses propriétés et de ses attributs. Enfin, l'ensemble d'objets forme l'extension du concept, il s'agit de toutes les instances du concept. Lors de la création d'une ontologie, le problème qui se pose est la sélection de ces concepts. Quels sont ceux à intégrer ? Comment définir les hiérarchies ? Comment surmonter la variabilité des représentations ? Pour répondre à ces questions, Bachimont propose de recourir à une normalisation sémantique des concepts en s'appuyant sur le paradigme différentiel. Celui-ci permet d'éviter toute ambiguïté de sens en définissant chacun des concepts en fonction des quatre principes différentiels que sont : (1) le principe de communauté avec le père, (2) le principe de différence avec le père, (3) le principe de différence avec les frères et (4) le principe de communauté avec les frères. Enfin, un concept se définit également au travers du lien qu'il peut entretenir avec les autres concepts de l'ontologie. De ce fait, Guarino propose de qualifier

un concept en se basant sur des propriétés formelles de rigidité, d'identité, d'unité et de dépendance.

•**Relations** : Dans une ontologie, les relations correspondent aux types d'interaction possibles entre les concepts d'un domaine. De ce fait, elles se définissent à la fois par leur signature mais aussi par le contenu sémantique entre les concepts qu'elles unissent.

Par exemple, deux relations peuvent porter sur deux mêmes concepts (signature sémantique égale) sans toutefois avoir le même sens (contenu sémantique différent). Dans une ontologie, les relations permettent ainsi de structurer les connaissances et de définir les interrelations entre les concepts. Par exemple si l'on souhaite définir une relation de spécialisation entre les concepts d'un domaine, on utilise pour cela une relation de type sous-classe-de (subClassOf). Ces relations peuvent également être utilisées afin de faire référence à des propriétés spécifiques afin d'exprimer une valeur qui peut être textuelle ou encore algébrique. Ainsi, deux grands types de relations peuvent être distingués : les relations taxonomiques et les relations associatives. Les premières, également appelées subsomption ou encore relation de spécificité/généricité permettent d'organiser hiérarchiquement un ensemble de concepts. Les secondes désignent toutes les relations entre les concepts qui ne sont pas des relations de taxonomiques. Ces relations peuvent être enrichies par des notions de symétrie, de transitivité ou encore de non-réflexivité.

•**Fonctions** : ce sont un cas particulier de relations dans lesquelles un élément est défini en fonction des éléments précédents.

•**Instances** : elles sont utilisées pour représenter les individus d'une ontologie. Ces individus correspondent à une instance concrète de la classe à laquelle ils appartiennent. Dans le cas d'une ontologie contenant des instances, celles-ci devient alors une base de connaissances.

•**Axiomes** : sont des assertions toujours vraies à propos des abstractions (concepts et relations) du domaine modélisé. Ils permettent de combiner des concepts, des relations et des fonctions pour définir des règles d'inférence. L'ajout d'axiomes dans une ontologie peut avoir notamment pour objectif de définir la signification des composants, de restreindre la valeur des attributs ou encore de vérifier la validité des informations spécifiées.

2.5. Typologie des ontologies

Si les définitions de la notion d'ontologie sont très nombreuses, leurs typologies ne le sont pas moins. Plusieurs d'entre elles ont en effet été proposées selon par exemple le degré de

conceptualisation, le niveau de représentation des connaissances ou encore le niveau d'expressivité.

Ce document n'a pas pour ambition de faire une liste exhaustive des typologies existantes, mais plutôt d'en présenter deux qui semblent traduire concrètement les différentes représentations d'une ontologie.

2.5.1 Typologie selon l'objet de conceptualisation

La typologie proposée par Guarino [2] est l'une des représentations les plus couramment citées. Celle-ci se décompose en quatre niveaux dépendant du degré de conceptualisation :

•**Ontologies de haut-niveau** : Ces ontologies que l'on retrouve également sous le terme de top level ontologies, ontologies de sens commun/général ou encore ontologies génériques modélisent les concepts les plus généraux. Elles sont réutilisables d'un domaine à un autre et sont conçues pour réduire les incohérences des termes définis en aval de la hiérarchie. Parmi les ontologies de haut niveau les plus connus, citons notamment le projet Basic Formal Ontology (BFO) ou encore le projet Suggested Upper Merged Ontology (SUMO).

•**Ontologies de tâches** : elles décrivent les concepts utilisés pour résoudre un ou des problèmes d'une activité spécifique indépendamment d'un domaine quelconque. Elles fournissent un ensemble de termes et de relations au moyen duquel il est possible de décrire la manière de résoudre un problème. Selon Mizoguchi, une ontologie de tâches est la structure computationnelle d'un système à base de connaissances.

•**Ontologies de domaine** : elles sont spécialisées pour un certain type d'artefact et s'attachent à décrire le vocabulaire relatif à un domaine donné. Elles permettent de spécialiser les termes et les notions des ontologies de haut niveau.

•**Ontologies d'application** : c'est l'ontologie la plus spécifique, elle est utilisée pour modéliser les concepts d'un domaine particulier dans le cadre d'une activité spécifique. Elles sont à la fois une union et une spécialisation des ontologies de tâches et de domaines.

2.5.2. Typologie selon le niveau de formalisation

À la proposition de Guarino de classifier les ontologies selon l'objet de conceptualisation, ajoutons également celle d'Uschold et Gruninger qui porte sur le niveau de formalisation. En effet, selon le langage ou les formalismes de représentation utilisés, le degré

de formalisation de l'ontologie peut être variable. Cette variabilité est exprimée sous la forme d'une typologie en quatre niveaux :

•**Ontologies informelles** : exprimées en langue naturelle elles sont de ce fait facilement compréhensibles par les utilisateurs. Mais l'absence de formalisation rend leur utilisation dans un système informatique et leur validation problématiques.

•**Ontologies semi-informelles** : la sémantique du langage est plus structurée et limitée

•**Ontologies semi-formelles** : exprimée formellement dans un langage artificiel

•**Ontologies formelles** : exprimée dans un langage artificiel disposant d'une sémantique formelle permettant d'effectuer des vérifications. C'est le cas par exemple des logiques de description qui seront présentées dans le chapitre.

Cette typologie met en évidence plusieurs niveaux de représentation des ontologies.

L'élaboration de l'ontologie permettra d'une part au tuteur de se faire une idée sur la façon d'agir avec l'apprenant selon son style et ses préférences d'apprentissage pour mieux le guider et de minimiser le cout d'apprentissage.

2.6. Les styles d'apprentissage [7]

Les recherches dans le domaine de l'enseignement nous démontrent qu'on a tendance à enseigner en se basant sur notre propre style d'apprentissage. Or, si notre apprenant n'a pas le même style d'apprentissage que nous, il y aura des difficultés à l'horizon. Il serait donc pertinent pour tout tuteur de se familiariser avec les différents styles d'apprentissage pour la simple et bonne raison que cela les aidera à devenir des tuteurs plus efficaces.

2.6.1. Le modèle de Kolb [3]

Le modèle de Kolb a permis de mettre en rapport les styles d'apprentissages et l'orientation aux études.

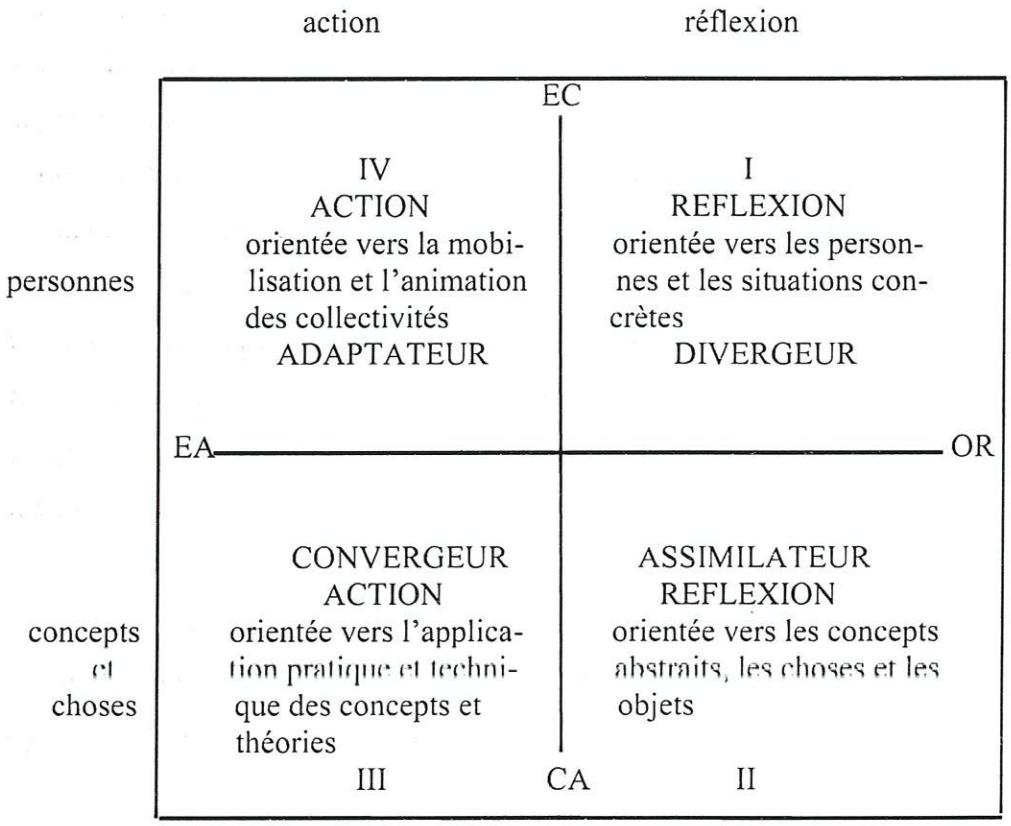

Figure 2.1 : *Modèle de Kolb.*

2.6.2. L'apprentissage expérientiel et ses composantes

Il est constitué de quatre phases successives qui forment un cycle. Qui dit cycle, dit implicitement répétition à l'infini. La personne qui apprend de façon efficace passe plus ou moins consciemment par chacune des phases illustrées dans ce qui suit :

A) Le mode EC (expérience concrète, implication)

C'est le point de départ du processus d'apprentissage. Le sujet s'implique dans la démarche d'apprentissage selon une motivation intrinsèque. Ce type de motivation est provoqué par une situation, des faits ou événements qui font problème et auxquels il a le désir de trouver une solution. Ainsi, le sujet devra être engagé directement dans une expérience véritable, intéressé personnellement par la situation à l'étude ou le problème, stimulé par la situation présentée et enfin initiera lui-même la démarche d'apprentissage pour passer à la seconde étape et aux étapes ultérieures.

B) Le mode OR (observation réfléchie, analyse)

Le sujet passe à l'analyse des données de l'expérience concrète. Après avoir observé des faits de l'expérience vécue, il réfléchit à leur signification en les considérants sous différentes perspectives. Pour ce faire, il peut utiliser ses sources internes de données expérientielles (exploration de l'expérience immédiate et référence aux expériences antérieures) ou ses sources externes (expérience immédiate des autres personnes placées dans la même situation, expériences analogues présentées dans des documents écrits ou audio-visuels). Cela va lui permettre de réfléchir, analyser, déduire et induire adéquatement. La pensée organise, comprend et interprète le matériel reçu.

C) Le mode CA (conceptualisation abstraite, synthèse)

Le sujet s'applique à faire la synthèse des éléments dégagés par l'analyse entreprise à l'étape précédente, en établissant un ordre d'importance parmi les éléments identifiés, en faisant ressortir les liens entre les éléments de même nature et finalement en cherchant à identifier les causes de la problématique observée. Il cherche à cerner le « nœud du problème » en dégageant des concepts et en les schématisant ou en appliquant des modèles théoriques susceptibles de représenter le fruit de ses observations et réflexions. La personne devient apte à comprendre et expliquer certains phénomènes, à tirer des conclusions précises. Ceci constitue un effort de créativité personnelle qui rend plus significatif l'apprentissage

D) Le mode EA (expérimentation active, application)

Le sujet qui apprend confronte ses conceptions théoriques avec la réalité. Il cherche à appliquer ses connaissances de façon à résoudre des problèmes pratiques. Ainsi, il vérifie la validité de ses conclusions théoriques quant aux causes des problèmes, aux caractéristiques des processus et aux conséquences prévisibles. L'apprenant doit se montrer attentif aux résultats observables de son expérimentation et être disposé à réviser ses conceptions lorsque son expérimentation ne produit pas les résultats attendus. L'expérimentation active le prépare à envisager de nouvelles expériences concrètes et à se sensibiliser à de nouvelles facettes des phénomènes impliqués, de manière à faire des cycles d'apprentissage plus profitables. Il s'agit donc d'un processus qui lorsque complété à travers ses 4 étapes déborde le cadre scolaire et permet d'apprendre non seulement à l'école, mais dans la vie à travers les différents événements qui la ponctue.

2.6.3. Les styles d'apprentissage de Kolb

David Kolb est un éducateur qui a beaucoup contribué à l'étude des styles d'apprentissage. Ses recherches sur les styles d'apprentissage des adultes ont démontré que les adultes, bien qu'ils apprennent continuellement, ont tendance à privilégier un mode d'apprentissage. Cela ne veut pas dire qu'ils n'ont qu'une seule façon d'apprendre mais plutôt qu'ils apprennent mieux avec une approche en particulier. Kolb a identifié, à partir des divers modes d'apprentissage, quatre styles d'apprentissage :

Les quatre styles d'apprentissage :

- Le style accommodateur
- Le style divergent
- Le style assimilateur
- Le style convergent

A) Le style accommodateur (le manipulateur)

Ses modes privilégiés sont l'expérimentation active (EA) et l'expérience concrète (EC). Il a des aptitudes particulières pour l'exécution et la réalisation et aime mener à terme des projets complexes impliquant la participation de plusieurs personnes. Il excelle dans la prise de décision rapide, l'adaptation sur le champ et l'improvisation commandée par des circonstances particulières. Il subordonne les idées aux personnes et tend à solutionner les problèmes de façon empirique plutôt que rationnelle. Il tient plus compte des opinions des autres que de ses propres compétences. Bien qu'à l'aise avec les personnes, il tend à y être dépendant et à les manipuler pour arriver à ses fins. Ce sont des touches à tout qui sont destinés au monde des affaires, à la gestion et la coordination et aux relations publiques.

B) Le style divergent (l'observateur)

Ses deux modes privilégiés sont sa capacité de participation sociale (EC) et d'observation réflexive (OR). L'imagination constitue sa principale ressource et lui permet d'analyser les faits selon différentes perspectives, d'exceller dans les remue-méninges, d'inventorier avec beaucoup de créativité les diverses utilisations d'un objet par exemple. Les divergeurs vont d'abord s'intéresser aux personnes et situations sociales car ils ont besoin d'interagir avec d'autres. Ils ont des intérêts culturels variés et sont attirés par les sciences

humaines. C'est le style typique des conseillers, agents de développement, administrateurs du personnel, professionnels aidants.

C) Le style assimilateur (le conceptualisateur)

Ses deux modes privilégiés sont l'observation réfléchie (OR) qui l'apparente au divergeur et la conceptualisation abstraite (CA). Sa capacité de conception de modèles théoriques constitue sa principale ressource. Il peut élaborer des notions à partir des phénomènes observés et intégrer plusieurs observations disparates. Il assimile plusieurs théories qu'il utilise pour donner des explications aux phénomènes qui l'intéressent. Les concepts abstraits l'intéressent davantage que les personnes et en plus il se soucie peu de l'applicabilité de ses modèles théoriques. Pour lui, seule compte une théorie vraisemblable et cohérente. Il est intéressé par les sciences pures, spéculatives, fondamentales plutôt que les par les sciences appliquées et la technologie.

D) Le style convergent (le penseur-expérimentateur)

Ses deux modes privilégiés sont la conceptualisation (CA) et l'expérimentation active (EA). Il est aux antipodes du divergeur et sa plus grande force est l'application pratique des notions théoriques de manière à en vérifier la validité et/ou à en exploiter l'utilité. Il est très à l'aise dans les problèmes qui font appel à une seule réponse exacte et dans les examens objectifs. Il s'engage dans des raisonnements hypothético-déductifs de façon à organiser ses connaissances dans le but de résoudre des problèmes spécifiques.

Les convergeurs préfèrent travailler seuls plutôt qu'en équipe, et la technologie les intéresse particulièrement. Les choses les intéressent plus que les personnes et ils s'orienteront habituellement vers les sciences appliquées, le génie ou les cours techniques.

2.7. Le langage Python [13]

2.7.1. Généralités

Python est un langage de programmation objet, multi-paradigme et multiplateformes. Il favorise la programmation impérative structurée, fonctionnelle et orientée objet. Il est doté d'un typage dynamique fort, d'une gestion automatique de la mémoire par ramasse-miettes et d'un système de gestion d'exceptions ; il est ainsi similaire à Perl, Ruby, Scheme, Smalltalk et Tcl.

Il est également apprécié par les pédagogues qui y trouvent un langage où la syntaxe, clairement séparée des mécanismes de bas niveau, permet une initiation aisée aux concepts de base de la programmation.

Python est un langage qui peut s'utiliser dans de nombreux contextes et s'adapter à tout type d'utilisation grâce à des bibliothèques spécialisées. Il est cependant particulièrement utilisé comme langage de script pour automatiser des tâches simples mais fastidieuses, comme un script qui récupérerait la météo sur Internet ou qui s'intégrerait dans un logiciel de conception assistée par ordinateur afin d'automatiser certains enchaînements d'actions répétitives (voir la section Adoption). On l'utilise également comme langage de développement de prototype lorsqu'on a besoin d'une application fonctionnelle avant de l'optimiser avec un langage de plus bas niveau. Il est particulièrement répandu dans le monde scientifique, et possède de nombreuses extensions destinées aux applications numériques.

Le langage Python est placé sous une licence libre proche de la licence BSD3 et fonctionne sur la plupart des plates-formes informatiques, des supercalculateurs aux ordinateurs centraux4, de Windows à Unix en passant par GNU/Linux, Mac OS, ou encore Android, iOS, et aussi avec Java ou encore .NET. Il est conçu pour optimiser la productivité des programmeurs en offrant des outils de haut niveau et une syntaxe simple à utiliser.

Concrètement, voilà ce qu'on peut faire avec Python :

- de petits programmes très simples, appelés **scripts**, chargés d'une mission très précise sur votre ordinateur ;
- des programmes complets, comme des jeux, des suites bureautiques, des logiciels multimédias, des clients de messagerie…
- des projets très complexes, comme des progiciels (ensemble de plusieurs logiciels pouvant fonctionner ensemble, principalement utilisés dans le monde professionnel).

Voici quelques-unes des fonctionnalités offertes par Python et ses bibliothèques :

- créer des interfaces graphiques ;
- faire circuler des informations au travers d'un réseau ;
- dialoguer d'une façon avancée avec votre système d'exploitation ;
- …

2.7.2. Les différents types d'erreurs et exceptions générées par Python [8]

La programmation est un processus très complexe, et tel est le cas dans toute activité humaine. Les erreurs de programmation sont appelés "bugs", et l'ensemble des techniques qui sont utilisées pour détecter et corriger les erreurs de thèses, appelé "debug" (ou "débogage"). Sachant que les compétences les plus importantes à acquérir au cours du processus d'apprentissage est de "debug" un programme efficacement.

Pour comprendre le contexte de notre travail, nous expliquons d'abord quelques concepts liés :

A) Types d'erreur de programmation

En effet, comme dans tous les langages de programmation, il peut se produire dans un programme Python, trois types d'erreurs :

• Les erreurs de syntaxe : Python ne peut exécuter un programme que si la syntaxe est parfaitement correcte. Dans le cas contraire, le processus est interrompu et un message d'erreur est affiché. Le terme syntaxe fait référence aux règles de syntaxe de la langue que les auteurs ont mise en place pour la structure du programme.

• Les erreurs sémantiques : Le deuxième type d'erreur est l'erreur sémantique ou erreur logique. S'il y a une erreur de ce type dans l'un de vos programmes, il fonctionne parfaitement, en ce sens que vous obtenez aucun message d'erreur, mais le résultat est pas ce que vous attendiez : Les instructions de programme de votre séquence ne correspond pas à l'objectif désiré. La sémantique (logique) est incorrecte.

• Les erreurs d'exécution : Le troisième type d'erreur est l'erreur d'exécution, qui apparaît uniquement lorsque votre programme est déjà en cours d'exécution, mais que les circonstances spéciales (par exemple, votre programme tente de lire un fichier qui n'existe plus.). Ces erreurs sont également appelées exceptions car ils indiquent généralement que quelque chose d'exceptionnel est arrivé (qui n'a pas été prévu).

B) Différence entre erreur et Exception

Une erreur est généralement une réponse au problème posé par le système. Les exceptions sont des alertes dont l'emplacement et le comportement sont définis par le développeur. Les exceptions sont utilisées pour gérer les erreurs, l'inverse n'a pas de sens.

C) La gestion des erreurs et des exceptions

La gestion des erreurs est généralement résolu en enregistrant l'état d'exécution au moment de l'erreur et interrompre le flux normal du programme à exécuter une fonction spéciale ou un morceau de code, qui est connu comme le gestionnaire d'exception. Selon le type d'erreur ("division par zéro", "fichier erreur d'ouverture» et ainsi de suite) qui avait eu lieu, le gestionnaire d'erreurs peut résoudre le problème et le programme peut se poursuivre ensuite avec les données précédemment enregistrées.

La gestion des exceptions est une construction dans certains langages de programmation pour manipuler ou traiter automatiquement avec des erreurs. De nombreux langages de programmation tels que C ++, Objective-C, PHP, Java, Ruby, Python, et beaucoup d'autres ont un support intégré pour la gestion des exceptions. Lorsqu'une exception se produit le flux normal du programme est perturbé et le programme / Application se termine de façon anormale, qui ne sont pas recommandés, par conséquent, ces exceptions doivent être traitées.

Malheureusement, en utilisant le support intégré pour les besoins de gestion des exceptions pour une connaissance profonde débutant sur la façon dont ces exceptions se produisent. En outre, tous les détails peuvent être donnés par le compilateur pour guider et aider les apprenants à attraper ces exceptions. Quand un programme de l'apprenant se bloque lors de l'exécution, l'apprenant reçoit une rétroaction lui disant qu'il y a une exception qui est arrivé. Cependant, le niveau cognitif d'un débutant ne peut pas lui permettre de rattraper ces exceptions pour éviter les erreurs éventuelles. Python (version 3.4) a environ 29 exceptions intégrées.

Non de l'exception	Description
ArithmeticError	base pour toutes les erreurs qui se produisent pour le calcul numérique.
FloatingPointError	Déclenchée quand un calcul en virgule flottante échoue.
ZeroDivisonError	Déclenchée quand division ou modulo par zéro a lieu pour tous les types numériques.
EOFError	Déclenchée quand il n'y a pas d'entrée soit de la raw_input () ou entrée () et la fin du fichier est atteinte.
IndexError	Déclenchée quand un index ne se trouve pas dans une séquence.
IOError	Déclenchée quand une opération d'entrée / sortie échoue, comme la déclaration d'impression ou la fonction open () lorsque vous essayez d'ouvrir un fichier qui n'existe pas.
OSError	pour les erreurs liées au système d'exploitation.
ValueError	lorsque la fonction intégrée pour un type de données a le type valide des arguments, mais les arguments ont des valeurs non valides spécifiées.

Tableau 2.1 : *Quelques exceptions intégrées standard Python*

D) Utilisation des exceptions pour la gestion des erreurs

Dans ce paragraphe, nous expliquons la manière dont un système intelligent peut aider un apprenant dans la gestion des exceptions à améliorer son code en évitant les erreurs d'exécutions éventuelles.

Nous croyons que la capture intelligemment « exceptions » va simplifier et améliorer la gestion des erreurs et donc d'augmenter la qualité du code en général, et rendre la programmation agréable et en même temps en donnant une rétroaction intelligente pour optimiser l'effet d'apprentissage.

La figure 2 montre comment gérer manuellement une exception en utilisant la syntaxe Python:

```
Try:
    ....................
except Exception_type:
    ....................
else:
    ....................
```

Figure 2.2 : *Gestion manuelle d'une exception en utilisant la syntaxe Python*

Comme le montre la figure 2, Python peut gérer toute exception de programmation en utilisant un *try...except...else*.

Exemple :

```
try:
    f = open("file")
except IOError:
    sys.exit("could not open file")
```

Figure 2.3 **:** *Exemple de gestion d'une exception Python*

Le code exemple représenté sur la figure 2 peut produire une rétroaction montrant un nom erreur s'il est impossible d'ouvrir le fichier et il sera immédiatement apparent ce qui doit être corrigé. Cependant, un grand et complexe script Python peut générer plusieurs exceptions au cours de l'exécution, ce qui rend difficile pour un programmeur novice à gérer manuellement toutes les exceptions éventuelles, ce qui augmente considérablement les erreurs de programmation. L'exemple sur la figure 3, montre un code Python simple qui est capable de provoquer beaucoup d'exceptions d'exécution :

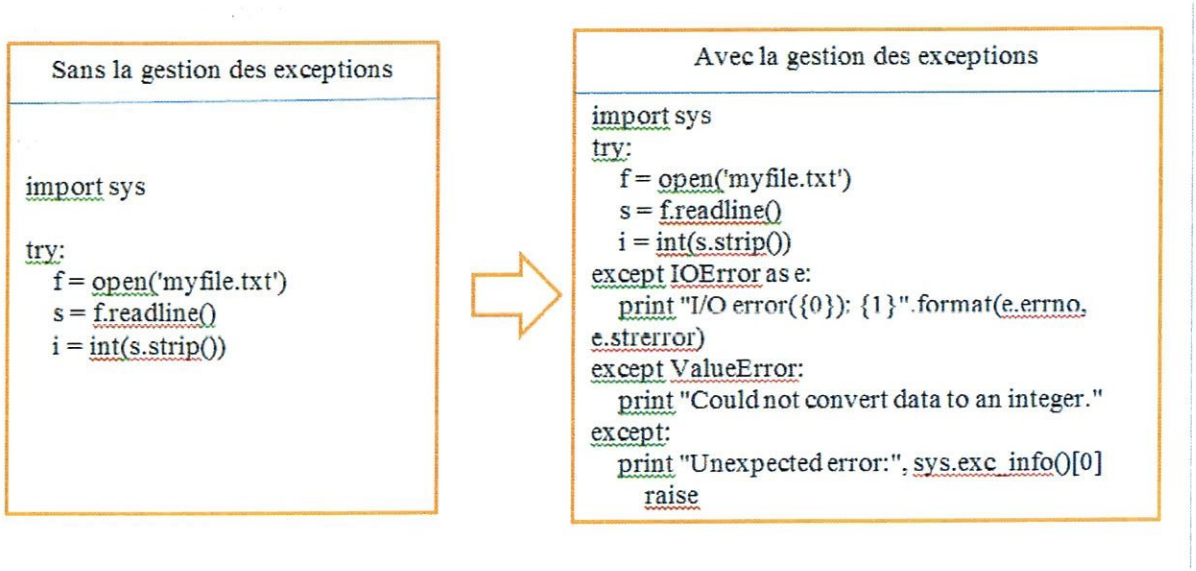

Figure 2.4 : *Un code Python simple avec gestion des exceptions multiples*

2.8. Conception du modèle tuteur

Le fonctionnement [5] décisionnel de plusieurs STI est basé sur une analyse cognitive de l'enseignement. De ce point de vue, l'apprenant possède des connaissances limitées ou manquantes qui l'amèneront à commettre des erreurs. L'analyse de ces erreurs permet au tuteur virtuel de diagnostiquer les concepts qui sont méconnus de l'apprenant. Quand la difficulté a été diagnostiquée, c'est au tuteur d'enseigner les connaissances nécessaires pour remédier à la situation

Pour une meilleure compréhension de ces fonctions on a l'exprimé sous forme de pseudo-algorithme dans le schéma ci-dessous. Cet algorithme explique la manière dont le tuteur pourra interagir avec l'apprenant à travers l'ontologie de connaissance du système.

Nous allons évaluer la capacité du tuteur d'analyser les scripts de l'apprenant, la correction des erreurs et de donner des actions pédagogiques appropriées (à savoir de rétroaction), sachant que l'efficacité de tuteur est fortement affectée par son comportement.

En fait, comme dans tous les langages de programmation, il peut se produire dans un programme Python, trois types d'erreurs : les erreurs syntaxiques, sémantiques et les erreurs d'exécution (ou exceptions).

Par exemple si l'apprenant au cours de son apprentissage commet une erreur syntaxique le tuteur à travers l'ontologie de base identifie la source du problème et oriente ensuite l'apprenant vers le lien qui lui permettra de corriger son erreur et bien sûr selon son style d'apprentissage et ses préférences.

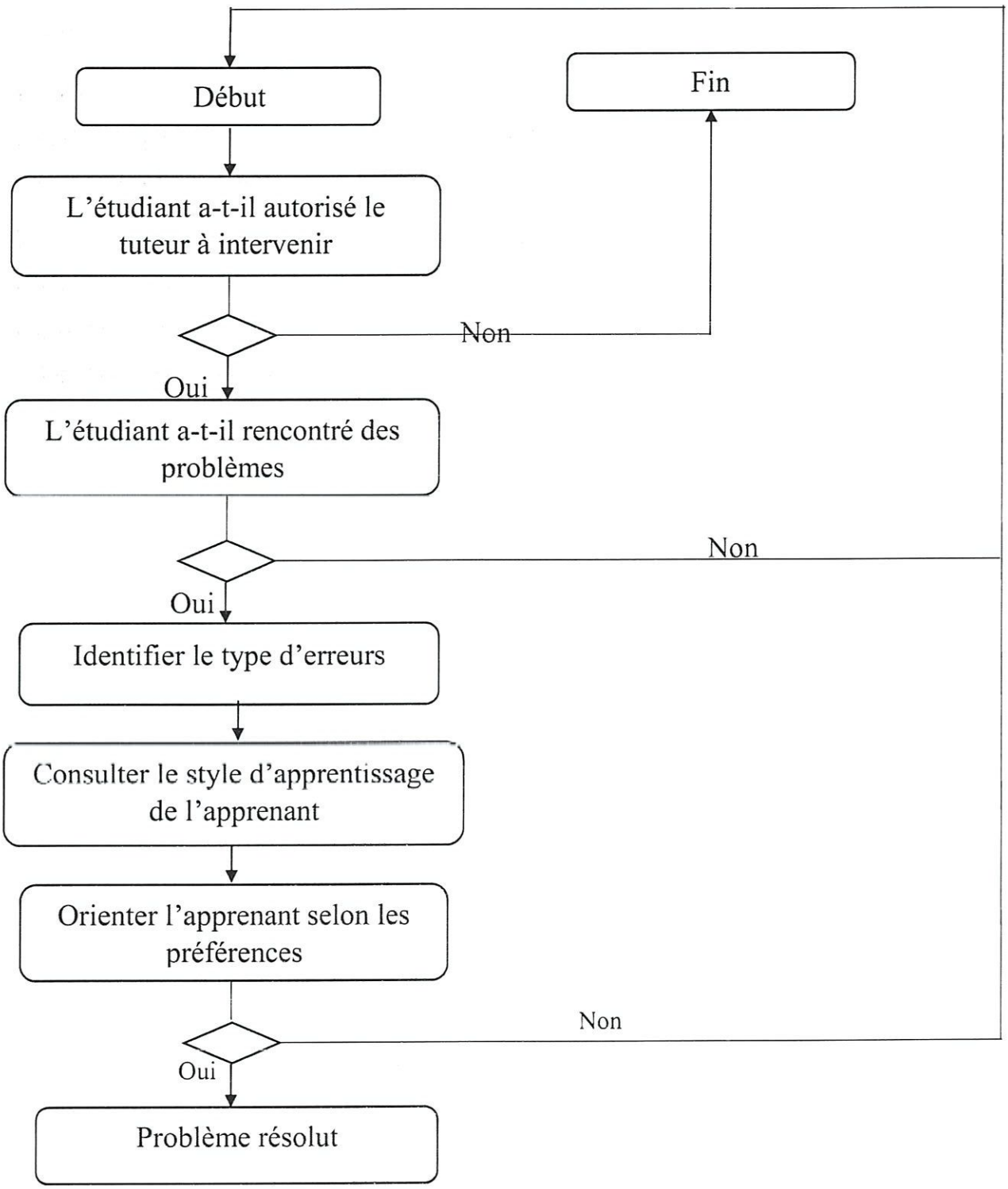

Figure 2.5 : *Modèle du tuteur.*

2.9. Conception du modèle apprenant

Le modèle apprenant comme son nom l'indique, sert à représenter un apprenant. Le caractère binaire et rigoureux de l'informatique ne permet pas de représenter toutes les configurations possibles des connaissances et des habilités d'un apprenant. Cependant le modèle de l'apprenant doit être conçu de manière à exprimer la plus grande variété de comportements, de traits cognitifs et de profils psychologiques. De plus le modèle de l'apprenant peut servir de source d'information pour l'enseignant qui s'interrogerait sur les compétences et les connaissances d'un de ses élèves.

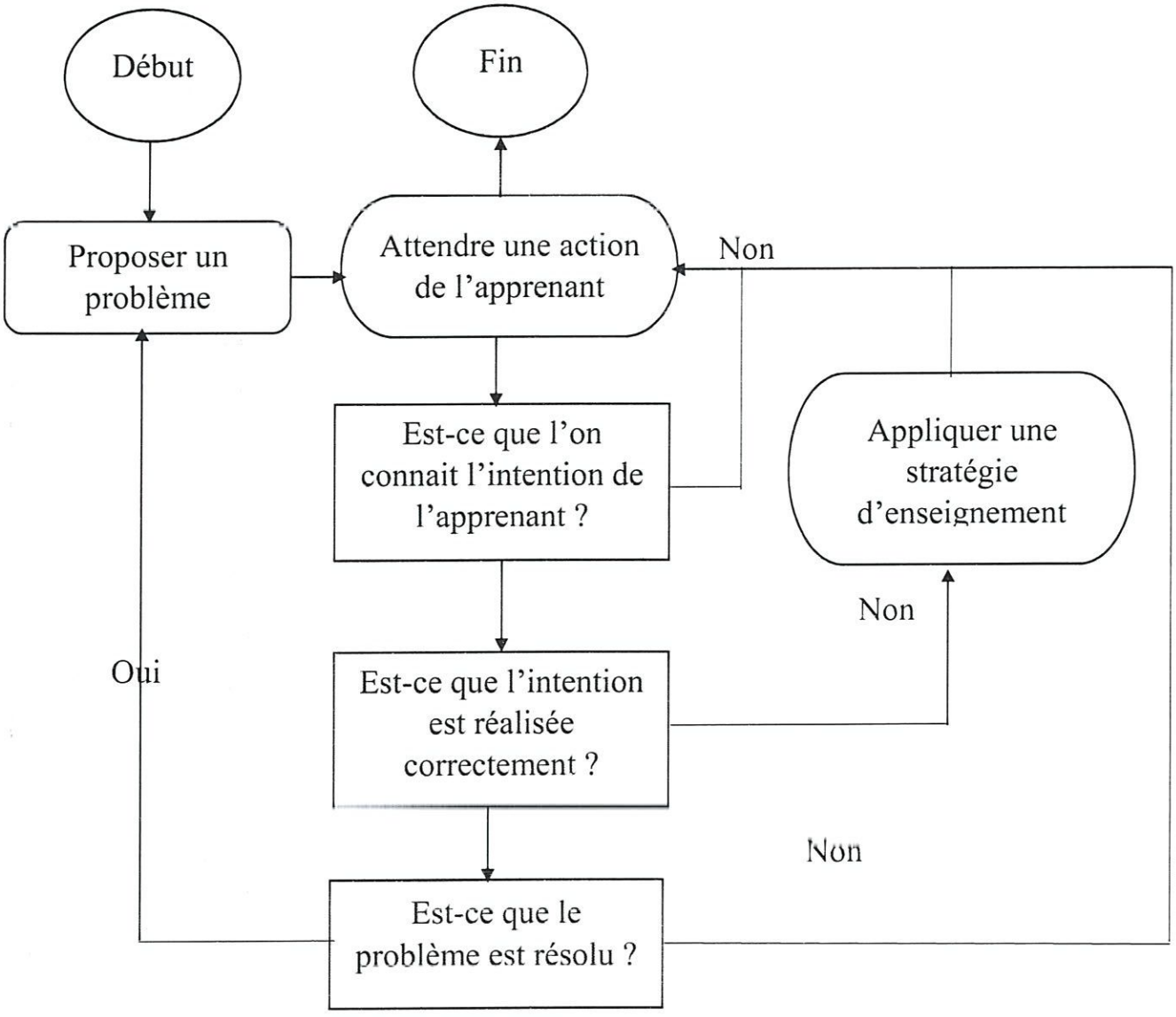

***Figure 2.6**: Modèle de l'apprenant.*

2.10. Conception du modèle du domaine

Afin de représenter au mieux le savoir du domaine, nous avons attribué, à chaque classe, un ensemble de propriétés permettant de les définir plus précisément en leur associant une valeur particulière. Cette valeur peut être une chaine de caractères, un nombre, une date, un booléen, etc. Comme nous l'avons déjà précisé plus haut, ces attributs sont héréditaires : ceux de la classe-mère sont automatiquement transmis aux classes-filles.

L'ontologie [11] que nous venons de décrire constitue le modèle de connaissances qui servira de guide aux différentes étapes d'extraction d'information associées.

2.11. Conclusion

L'ontologie et l'ingénierie ontologique possèdent un potentiel riche et multiple pour les domaines de la formation à distance.

L'appropriation du concept d'ontologie par le monde de l'ingénierie des connaissances a donné lieu à l'émergence d'une multiplicité de méthodes et d'outils, qui devraient se stabiliser et se raffiner progressivement.

Dans le prochain chapitre nous réaliserons une étude détaillée de l'implémentation de notre système.

Chapitre 3 : Implémentation du système

3.1. Introduction

L'objectif des applications de l'intelligence artificielle en éducation n'est évidemment pas d'amener les apprenants à penser comme des machines. Au contraire, en travaillant à rendre l'ordinateur moins rigide, moins algorithmique et plus intelligent : d'une part, on dégage des modèles plus opérationnels de l'intelligence et on distingue mieux ce qui sépare l'être humain de la machine ; d'autre part, on peut construire des outils pédagogiques qui respectent mieux l'intelligence des apprenants.

On peut distinguer deux grands pôles autour desquels se regroupent ces applications pédagogiques de l'intelligence artificielle. À une extrémité, les systèmes tutoriels intelligents (STI) enseignent ou guident l'apprenant souvent de très près dans sa démarche d'apprentissage. À l'autre extrémité, des outils intelligents sont fournis à l'élève pour qu'il explore et apprenne à maîtriser un domaine grâce à son activité personnelle de construction des connaissances.

Selon l'importance de l'aide apportée par le système, on peut classifier les environnements intelligents d'apprentissage à l'aide de l'ordinateur (EIAO) comme suit :

1. À une extrémité se trouvent les logiciels qui se contentent de donner une simple Rétroaction à l'apprenant. C'est le cas notamment de la plupart des progiciels, des hypermédias et des micro-mondes avec lesquels l'apprenant explore un domaine de connaissances. C'est également le cas des systèmes experts, lorsqu'ils ne contiennent pas de véritable module d'explication. L'apprenant utilise les outils du système pour réaliser une tâche ou pour résoudre un problème. Le système affiche certaines conséquences des actions de l'apprenant d'une façon qui devrait l'aider à cheminer vers une solution.

2. Un niveau interventionniste moyen consiste à prévoir une aide interactive intelligente. Un tel mécanisme affichera, sur demande de l'apprenant, une explication relative à la composante de l'interface où il se trouve ou à l'outil qu'il utilise, le plus possible en relation avec la tâche qu'il est en train de réaliser. Un bel exemple d'une telle approche se retrouve dans le module d'explication d'un système expert. Celui-ci peut être plus ou moins sophistiqué, mais il offre toujours une explication directement reliée aux interventions de l'usager sur le cas présentement à l'étude. On peut aussi greffer des systèmes d'aide interactive à des progiciels ou à des micro-mondes pour éviter de laisser l'apprenant trop démuni dans l'univers des actions possibles.

3. Un niveau interventionniste plus élevé est présent dans les systèmes « coach » ou les conseillers actifs. Dans ces systèmes, non seulement l'apprenant obtient sur demande une aide ciblée sur ses activités, mais le système peut également décider d'intervenir pour afficher un conseil lorsqu'il lui semble que l'apprenant a des difficultés trop grandes avec la tâche en cours. Cependant, le conseil n'est pas ici impératif, il peut être suivi ou non par l'apprenant.

4. Enfin, dans certains systèmes tutoriels intelligents, l'initiative est entièrement laissée au système qui exerce un guidage serré de l'apprenant. Ici les interventions du système sont impératives, les erreurs sont soulignées et demandent correction de la part de l'apprenant.

Notre projet s'identifie plus à la première approche de par sa conception et son mode de fonctionnement. L'implémentation étant la dernière étape dans l'élaboration de notre projet, il s'agit du codage proprement dit du système, de l'exploitation de l'ontologie développée antérieurement durant la conception, du développement de l'interface utilisateur afin d'obtenir le logiciel répondant aux attentes.

L'objectif de ce chapitre étant d'implanter le système nous étalerons dans ce qui suit les moyens mis en œuvre et les outils nécessaires pour la réalisation de notre travail.

.3.2. Technologie utilisée

3.2.1. L'Editeur d'Ontologies Protégé 2000 [14]

Protégé-2000 [Prot] est un éditeur qui permet de construire une ontologie pour un domaine donné, de définir des formulaires d'entrée de données, et d'acquérir des données à l'aide de ces formulaires sous forme d'instances de cette ontologie. Protégé est également une librairie Java qui peut être étendue pour créer de véritables applications à bases de connaissances en utilisant un moteur d'inférence pour raisonner et déduire de nouveaux faits par application de règles d'inférence aux instances de l'ontologie et à l'ontologie elle-même (méta-raisonnement).

Dans le contexte du web sémantique des « plugin » pour les langages RDF, DAML+OIL et OWL ont été développés pour Protégé. Ces « plugin » permettent d'utiliser Protégé comme éditeur d'ontologies pour ces différents langages, de créer des instances et les sauver dans les formats respectifs.

Il est également possible de raisonner sur les ontologies en utilisant un moteur d'inférence général tel que JESS [JES], ou des outils d'inférence spécifiques au web sémantique basés sur des logiques de description [DL] tels que RACER [RAC]. Ces deux outils peuvent être facilement intégrés à Protégé. Les logiques de description permettent de définir les bases logiques des différents formalismes de représentation de la connaissance tant sur le plan de la représentation que sur le raisonnement. Dans les formalismes de représentation de la connaissance, il est souvent nécessaire de restreindre l'expressivité pour rendre certains types de raisonnement, tels que la classification automatique, faisable (« tractable »).

3.2.2. Le système de gestion de base de données MYSQL [16]

Une base de données est une collection de données organisées de façon à être facilement accessibles, administrées et mises à jour. Les bases de données peuvent être classées par le type de contenu qu'elles renferment : bibliographique, full text, images ou des nombres....

En informatique, les bases de données sont parfois classées selon la façon dont elles organisent les données. L'approche la plus courante est la base de données relationnelle, une base de type tabulaire dans laquelle les données sont définies pour qu'elles puissent être réorganisées et rendues accessibles à travers plusieurs méthodes. Une base de données distribuée en est une autre qui peut être distribuée ou répliquée sur plusieurs points du réseau.

Une base de données orientée objet est cohérente avec la donnée définie dans une classe ou une sous-classe objet.

Un **Système de Gestion de Base de Données** (SGBD) est un logiciel qui permet de stocker des informations dans une base de données. Un tel système permet de lire, écrire, modifier, trier, transformer ou même imprimer les données qui sont contenus dans la base de données.

Parmi les logiciels les plus connus il est possible de citer : MySQL, PostgreSQL, SQLite, Oracle Database, Microsoft SQL Server, Firebird ou Ingres.

MySQL est un SGBD parmi les plus populaires au monde. Il est distribué sous double licence, une licence publique générale GNU et une propriétaire selon l'utilisation qui en est faites. La première version de MySQL est apparue en 1995 et l'outil est régulièrement entretenu. Il fonctionne sur de nombreux systèmes d'exploitation (dont Linux, Mac OS X, Windows, Solaris, FreeBSD...) et qui est accessible en écriture par de nombreux langages de programmation, incluant notamment PHP, Java, Ruby, C, C++, .NET, Python ...

L'une des spécificités de MySQL c'est qu'il inclut plusieurs moteurs de bases de données et qu'il est par ailleurs possibles au sein d'une même base de définir un moteur différent pour les tables qui composent la base. Cette technique est astucieuse et permet de mieux optimiser les performances d'une application. Les 2 moteurs les plus connus étant **MyISAM** (moteur par défaut) et **InnoDB**.

La réplication est possible avec MySQL et permet ainsi de répartir la charge sur plusieurs machines, d'optimiser les performances ou d'effectuer facilement des sauvegardes des données.

3.2.3 Le langage PHP [15]

PHP (officiellement, ce sigle est un acronyme récursif pour *PHP Hypertext Preprocessor*) est un langage de scripts généralistes et Open Source, spécialement conçu pour le développement d'applications web. Il peut être intégré facilement au HTML.

Ce qui distingue PHP des langages de script comme le Javascript, est que le code est exécuté sur le serveur, générant ainsi le HTML, qui sera ensuite envoyé au client. Le client ne reçoit

que le résultat du script, sans aucun moyen d'avoir accès au code qui a produit ce résultat. Vous pouvez configurer votre serveur web afin qu'il analyse tous vos fichiers HTML comme des fichiers PHP. Ainsi, il n'y a aucun moyen de distinguer les pages qui sont produites dynamiquement des pages statiques.

Le grand avantage de PHP est qu'il est extrêmement simple pour les néophytes, mais offre des fonctionnalités avancées pour les experts. Ne craignez pas de lire la longue liste de fonctionnalités PHP. Vous pouvez vous plonger dans le code, et en quelques instants, écrire des scripts simples.

Bien que le développement de PHP soit orienté vers la programmation pour les sites web, vous pouvez en faire bien d'autres usages.

3.2.4. La plateforme EasyPHP

EasyPHP fut le premier package WAMP à voir le jour (1999). Il s'agit d'une plateforme de développement Web, permettant de faire fonctionner localement (sans se connecter à un serveur externe) des scripts PHP. EasyPHP n'est pas en soi un logiciel, mais un environnement comprenant deux serveurs (un serveur web Apache et un serveur de bases de données MySQL), un interpréteur de script (PHP), ainsi qu'une administration SQL phpMyAdmin. Il dispose d'une interface d'administration permettant de gérer les alias (dossiers virtuels disponibles sous Apache), et le démarrage/arrêt des serveurs. Il permet donc d'installer en une seule fois tout le nécessaire au développement local du PHP. Il peut tout à fait être comparé aux logiciels WAMP5 ou XAMPP, qui sont des équivalents. Par défaut, le serveur Apache crée un nom de domaine virtuel (en local) 127.0.0.1 ou localhost. Ainsi, quand on choisit « Web local » dans le menu d'EasyPHP, le navigateur s'ouvre sur cette URL et affiche la page index.php de ce site qui correspond en fait au contenu du dossier www d'EasyPHP.

EasyPHP peut être utilisé comme une application portable, c'est-à-dire lancé sur une clé USB.

Versions récentes

- EasyPHP DevServer 14.1 VC11 avec PHP 5.5.x (supporte PHP 5.6.x / 5.5.x / 5.4.x / 5.3.x VC9/10/11 - voir composants), Apache 2.4.7 VC11, MySQL 5.6.15, phpMyAdmin 4.1.4

- EasyPHP DevServer 14.1 VC9 avec PHP 5.4.x (supporte PHP 5.4.x / 5.3.x / 5.2.x VC9 - voir composants), Apache 2.4.7 VC9, MySQL 5.6.15, phpMyAdmin 4.1.4
- EasyPHP WebServer 14.1 beta 2

3.2.5. Le JavaScript [10]

JavaScript est un langage de programmation de scripts principalement employé dans les pages web interactives mais aussi pour les serveurs3. C'est un langage orienté objet à prototype, c'est-à-dire que les bases du langage et ses principales interfaces sont fournies par des objets qui ne sont pas des instances de classes, mais qui sont chacun équipés de constructeurs permettant de créer leurs propriétés, et notamment une propriété de prototypage qui permet d'en créer des objets héritiers personnalisés. En outre, les fonctions sont des objets de première classe.

Ce langage, créé en 1995 par Brendan Eich, est actuellement à la version 1.8.2. C'est une implémentation de la troisième version de la norme ECMA-262 qui intègre également des éléments inspirés du langage Python. La version 1.8.5 du langage est prévue pour intégrer la cinquième version du standard ECMA.

JavaScript et java

Il ne faut pas confondre le JavaScript et le Java. En effet contrairement au langage Java, le code est directement écrit dans la page HTML, c'est un langage qui ne permet aucune confidentialité au niveau des codes (ceux-ci sont effectivement visibles).D'autre part l'applet Java (le programme) a été préalablement compilée, et une machine virtuelle permettant d'interpréter le pseudo-code doit être chargée en mémoire (du côté du client) à chaque chargement de la page, d'où un important ralentissement pour les applets Java contrairement au JavaScript qui est directement interprété.

3.3. Interfaces de l'application

3.3.1. Interface principale

L'interface principale de notre l'application varie selon la préférence de l'utilisateur. De ce fait on peut avoir plusieurs modes d'affichage selon le gout de l'utilisateur à savoir les fichiers PDFs, les vidéos, les pages web.

✓ **L'interface pour les cours PDFs**

Voici l'interface apprenant pour les cours en format Pdf :

***Figure 3.1** : interface apprenant pour les fichiers pdf.*

✓ **L'interface pour les cours vidéo**

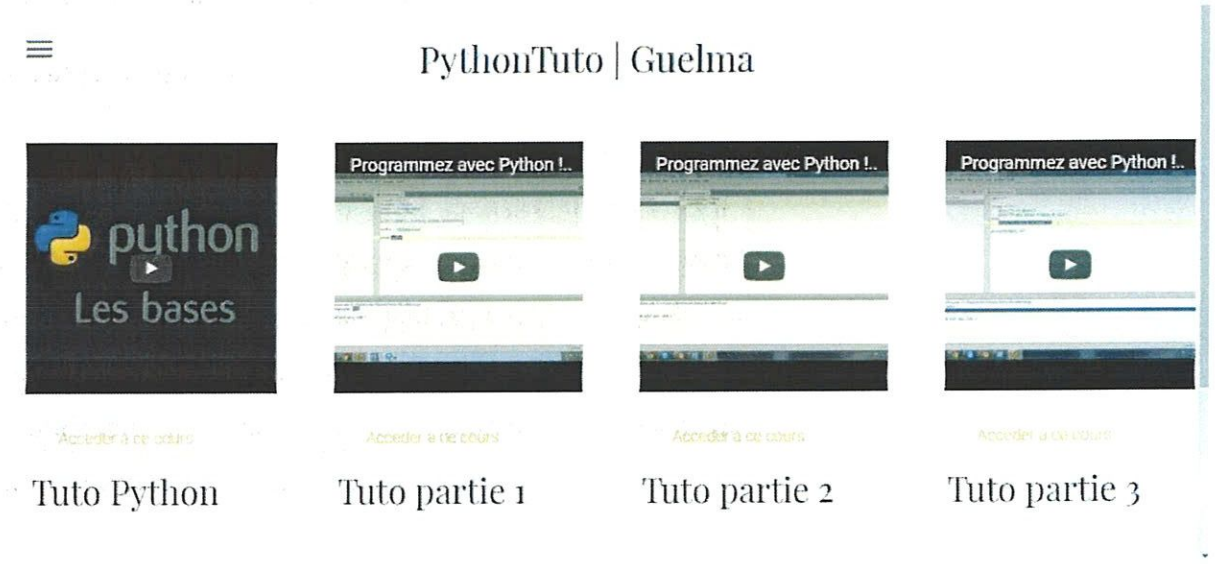

***Figure 3.2** : Interface de l'apprenant pour les vidéos.*

3.3.2. Menus principaux

A) La page d'authentification

Pour pouvoir accéder au système il faut un pseudo et un mot de passe pour les utilisateurs déjà inscrits sinon vous pouvez cliquer sur le lien « S'inscrire » se trouvant sur la même page.

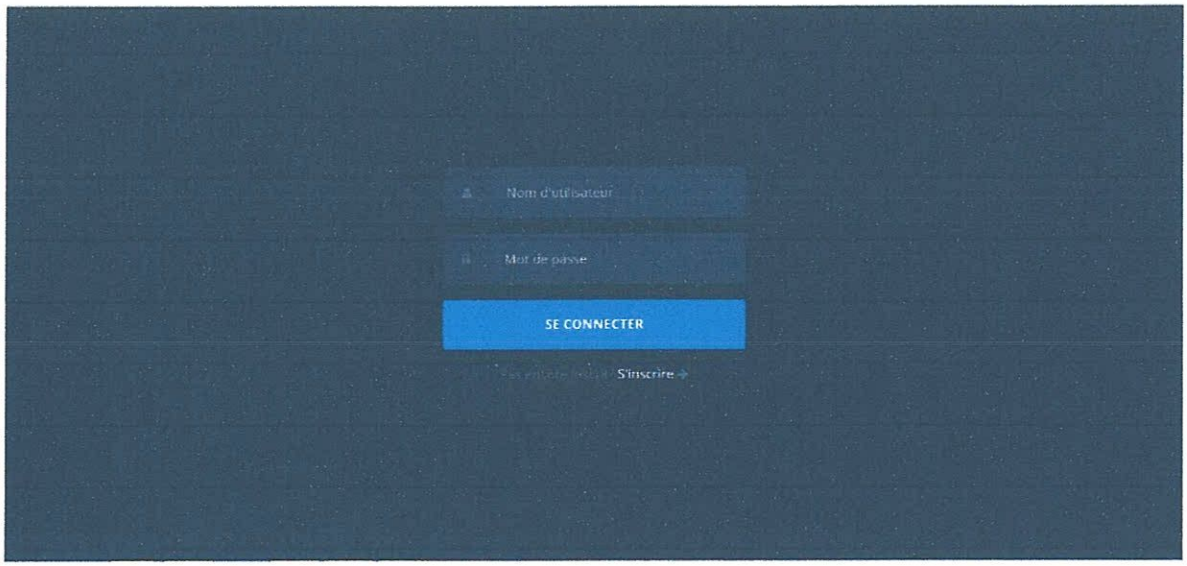

Figure 3.3 *: interface de connexion.*

B) La page d'inscription

Pour s'inscrire il faut remplir le formulaire d'inscription contenant les informations sur le nom, le prénom, un nom d'utilisateur, un mot de passe et la préférence d'apprentissage.

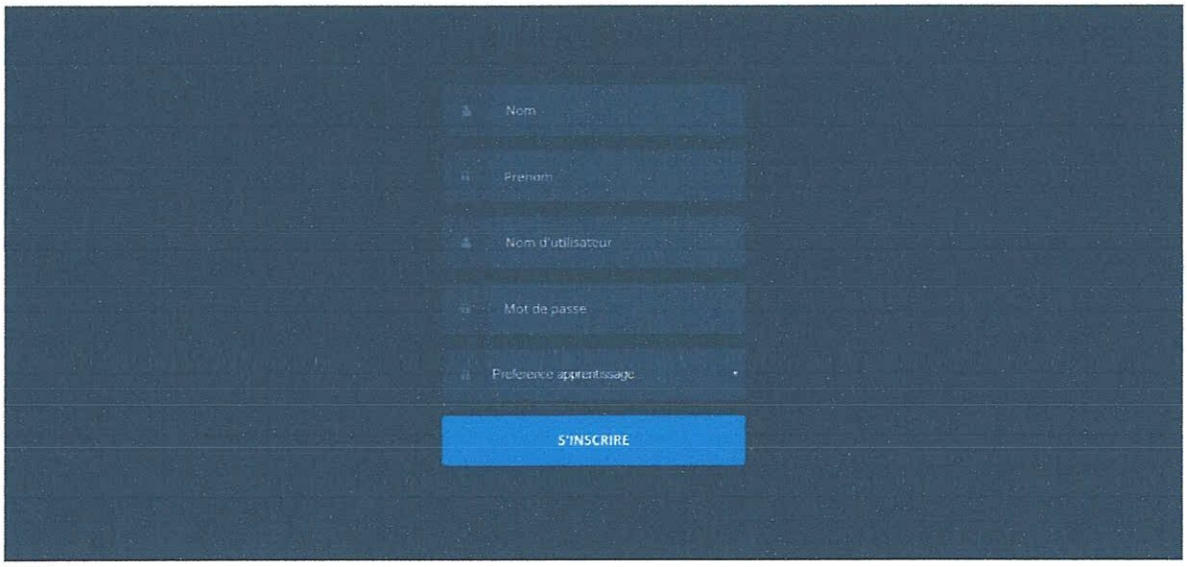

Figure 3.4 : *interface de la page d'inscription.*

C) L'interface pour le test de Kolb

Une fois inscrit l'utilisateur est redirigé automatiquement sur une autre page contenant une série de neuf questions afin de découvrir son style d'apprentissage.

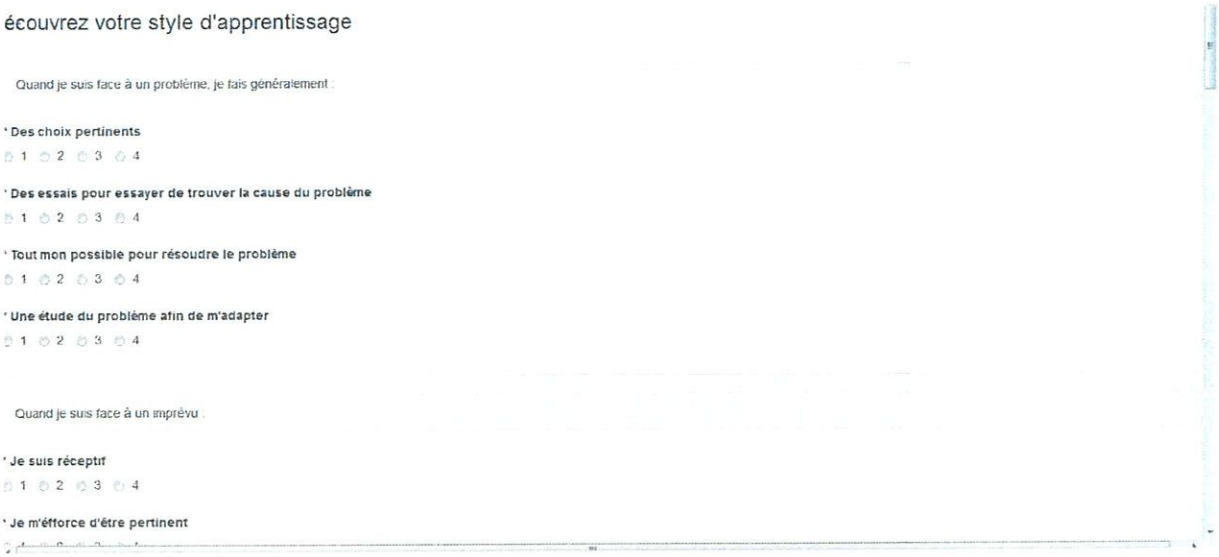

Figure 3.5 : *Interface du questionnaire de Kolb*

D) Le profil de l'utilisateur

Une fois connectée l'apprenant peut accéder au cours et séances d'exercices, passer le test de niveau s'il le souhaite. IL pourra voir son profil avec son nom, son prénom, son style, sa préférence d'apprentissage et son niveau qui est défini par défaut à « débutant ».

Figure 3.6 : *interface du profil de l'utilisateur.*

E) Diagnostic des erreurs par le tuteur intelligent

Les utilisateurs pourront effectuer des séances d'exercices et seront orientés à travers des messages s'ils commettent des erreurs et le type de l'erreur commis.

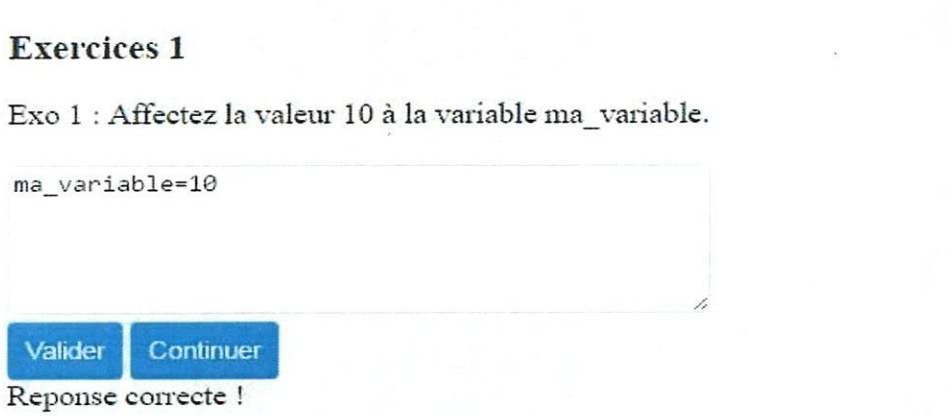

Figure 3.7 : *Interface du diagnostic des erreurs*

F) L'interface pour passer le test de niveau

Un test de niveau est disponible pour évaluer ses performances au cours de l'apprentissage. Selon le score le statut de l'apprenant peut prendre l'un des trois valeurs : Débutant, moyen ou expert.

Figure 3.8 : *Interface du test de niveau*

G) L'interface des corrections pour le test de niveau

Après le test niveau l'apprenant pourra consulter le corrigé des questionnaires pour pouvoir se situer sur les erreurs qu'il l'a commis et connaitre ses points faibles.

Résultat du QCM Langage Python

Note : 6/20
Niveau : Débutant

Note détaillée

Question n°	Votre réponse	La bonne réponse	Point(s)	Cumul
1	je ne sais pas	(3)	0	0
2	2	(4)	-1	-1
3	2	(2)	2	1
4	2	(2)	2	3
5	1	(1)	2	5
6	3	(5)	-1	4
7	3	(4)	-1	3
8	2	(1)	-1	2
9	3	(3)	2	4
10	2	(2)	2	6

Figure 3.9 : *Interface du corrigé du test de niveau*

3.4. Conclusion

Le développement d'un système tutoriel intelligent pour l'apprentissage en utilisant les ontologies nécessite la maitrise de plusieurs outils et technologies de développement. C'est un système complexe qui engendre plusieurs difficultés en développement ; la première difficulté rencontrée est la transcription du fichier ontologique pour qu'elle puisse être exploitée en Php, s'en suit le manque de ressources en ligne par rapport à ce sujet et les contraintes à respecter pour pouvoir répondre au mieux aux attentes des utilisateurs et respecter les spécifications du système.

Conclusion générale et perspectives

L'enseignement est l'une des activités parmi les plus difficiles, exigeant une constante remise en question. Il n'existe et ne peut exister aucune méthode pédagogique parfaite, aucune technique infaillible. Et l'enseignement assisté par ordinateur n'est pas une panacée. Mais, dûment considérée, l'utilisation de l'ordinateur peut contribuer à un notable enrichissement de la pédagogie.

L'élaboration de didacticiels peut s'avérer profitable pour tout enseignant, tout apprenant et pour l'enseignement, dans la mesure où on est conduit à se poser de nouveaux problèmes didactiques ; où on doit s'interroger non seulement sur la matière à enseigner, mais aussi sur les activités pédagogiques, sur les élèves, les modes d'acquisition des connaissances, les styles d'apprentissage les représentations.

L'objectif principal de notre travail était de développer un système tutoriel intelligent visant à faciliter l'apprentissage du langage Python. Malgré le fait qu'il existe de nombreuses plateformes permettant l'apprentissage en ligne notre système n'est pas en reste de ces derniers parce qu'en plus de la formation proposée nous mettons plus l'accent sur le style d'apprentissage et les préférences propres à chaque apprenant, ce qui lui procure une certaine ergonomie.

L'élaboration de ce projet nous a ainsi permis d'approfondir nos connaissances sur le langage PHP et la combinaison avec la base de données MYSQL et la modélisation ontologique qui nous était encore inconnue jusqu'au début de notre étude.

Il est urgent de signaler que ce mode d'enseignement ne va pas se substituer au mode classique d'enseignement mais il vient le compléter.

Comme perspective de ce travail on se propose :

-d'intégrer l'interpréteur Python pour un plus grand large choix dans les séances d'exercices

-possibilité de créer des forums entre les membres du site.

Références bibliographiques :

[1] Alicia Heraz, (Octobre 2009) Processus cérébraux adaptés aux systèmes tutoriels intelligents vers un système tutoriel intelligent cérébro-sensible, [en ligne]. Disponible : https://papyrus.bib.umontreal.ca/xmlui/bitstream/handle/1866/4310/heraz_alicia_ah_201_these.pdf. (Consulté 17 février 2016).

[2] Arnaud Vandecasteele, (le 30 Octobre 2012), Modélisation ontologique des connaissances expertes pour l'analyse de comportements à risque : application à la surveillance maritime, [en ligne]. Disponible : https://halshs.archives-ouvertes.fr/pastel-00819259/document. (Consulté le 04 avril 2016)

[3] Bouchekouf Asma, (2013), Perception du comportement de l'apprenant dans un environnement d'apprentissage, [en ligne]. Disponible : http://biblio.univ-annaba.dz/wp-content/uploads/2014/09/Bouchekouf-Asma.pdf. (Consulté le 19 avril 2016).

[4] Cédric Buche, Pierre De Loor, Ronan Querrec, Système tutoriel intelligent pour l'apprentissage de travail procédural et collaboratif, [en ligne]. Disponible : http://www.enib.fr/~buche/article/MFI_05.pdf. (Consulté le 17 février 2016).

[5] Collège Frontière : Manuel de Tutorat, Les styles d'apprentissage, [en ligne]. Disponible : http://savoirs.usherbrooke.ca/bitstream/handle/11143/4756/MR43004.pdf. (Consulté le 14 avril 2016).

[6] Cours de programmation en langage python, [en ligne]. Disponible : http://fsincere.free.fr/isn/python/cours_python.php. (Consulté le 22 mai 2016).

[7] Dominic Paradis, (Juillet 2007), Conception d'un modèle de l'apprenant utilisant les réseaux bayésiens pour le système tutoriel intelligent ASTUS, [en ligne]. Disponible : http://www.frontiercollege.ca/french/ressources/APPRENT.pdf. (Consulté le 19 mars 2016).

[8] Farek Lazhar, Tlili-Guiassa Yamina, Ontology-based Intelligent Tutor for Python Learning, 2015.

[9] Gilbert Paquette, (Avril 2012), Télé-université, [en ligne]. Disponible : http://www.teluq.uquebec.ca/expl_inf5100/pdf-doc/txt17.pdf. (Consulté le 17 février 2016).

[10] Javascript - Introduction au langage Javascript (2016), [en ligne]. Disponible : http://www.commentcamarche.net/contents/577-javascript-introduction-au-langage-javascript. (Consulté le 03 juin 2016).

[11] Laurie Serrano, (Année universitaire 2009-2010) Modélisation d'une ontologie de domaine et des outils d'extraction de l'information associés pour l'anglais et le français, [en ligne]. Disponible : http://dumas.ccsd.cnrs.fr/dumas-00569002/document. (Consulté le 06 avril 2016).

[12] Nkoa Antoine Nama TSimj, (Février 2007), Elaboration d'un cadre d'évaluation des systèmes tutoriels intelligents: application à Andes2, [en ligne]. Disponible : http://www.archipel.uqam.ca/1113/1/M10239.pdf. (Consulté le 05 avril 2016).

[13] Openclassrooms, Qu'est-ce que Python [en ligne]. Disponible : https://openclassrooms.com/courses/apprenez-a-programmer-en-python/qu-est-ce-que-python. (Consulté le 22 mai 2016).

[14] Philippe Massonet, (2003), La Création d'Ontologies Web Sémantique avec Protégé-2000, [en ligne]. Disponible : https://www.cetic.be/La-Creation-d-Ontologies-Web. (Consulté le 20 mai 2016).

[15] Qu'est-ce que PHP ?, [en ligne]. Disponible : http://php.net/manual/fr/intro-whatis.php. (Consulté le 20 mai 2016).

[16] Que signifie base de données, [en ligne].

Disponible : http://www.lemagit.fr/definition/Base-de-donnees. (Consulté le 20 mai 2016).