

# Outils pour la spécification de stratégies pédagogiquespilotant des interfaces graphiques réactives

Mahmoud Boufaida, Patrick Barril

▶ **To cite this version:**

Mahmoud Boufaida, Patrick Barril. Outils pour la spécification de stratégies pédagogiquespilotant des interfaces graphiques réactives. Sciences et Techniques Educatives, 1994, 1 (4), pp.503-520. edutice-00001452

**HAL Id: edutice-00001452**
**https://edutice.hal.science/edutice-00001452**

Submitted on 25 Oct 2006

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Outils pour la spécification de stratégies pédagogiques pilotant des interfaces graphiques réactives

# Tools for the specification of tutoring strategies which drive reactive graphical interfaces

**Mahmoud BOUFAIDA**
Institut d'Informatique, Université de Constantine,
route d'Aïn-el-Bey,
25000 Constantine, Algérie

**Patrick BARRIL**
M.A.S.I., Institut Blaise Pascal,
Université Pierre et Marie Curie - C.N.R.S.,
4, place Jussieu, 75252 Paris Cedex 05, FRANCE
Patrick.Barril@masi.ibp.fr

**Résumé :**

Nous décrivons un ensemble d'outils de conception de dialogues pédagogiques, destiné à la spécification, la compilation et à l'exécution aussi bien d'éléments d'interaction graphiques réactifs que de stratégies pédagogiques. Il conduit à un développement incrémental et itératif de didacticiels, permettant de vérifier la fidélité conceptuelle et l'ergonomie de l'interface élève sans avoir à programmer. Il produit des didacticiels mieux équilibrés entre le tutorat planifié et l'exploration libre de micro-mondes. L'architecture de ce système rend la conduite du dialogue indépendante de la représentation de la matière à enseigner et fait du prototype un noyau de tuteur générique. Nous discutons en conclusion les perspectives d'évolution vers un générateur de tutoriels intelligents.

**Abstract :**

We describe a tool-box of utilities for designing pedagogical dialogues, intended for specifying, compiling and executing reactive dialogue elements as well as tutoring strategies. It leads to an incremental and iterative development of courseware and allows an author to validate the conceptual faithfulness and the ergonomics of the student interface without programming. It produces tutorials with a better balance between planified tutoring and free exploration of micro-worlds. The architecture of the system makes the control of dialogue independent from the representation of the domain matter, and turns the prototype into a kernel for a generic tutor. We then discuss some perspectives for evolving the system into a generator for intelligent tutoring systems.

**Mots clés** : interfaces graphiques, stratégies pédagogiques, outils auteur, tuteurs intelligents, dialogue pédagogique, Smalltalk.

**Key words :** graphical interfaces, tutoring strategies, authoring tools, intelligent tutoring systems, pedagogical dialogue, Smalltalk.

## 1. Introduction

Les travaux de recherche en EAO (**E**nseignement **A**ssisté par **O**rdinateur) traditionnel ont conduit à des systèmes auteurs [Madaule 87] commercialisés mais aboutissant à des didacticiels difficiles à maintenir et limités dans leur possibilité d'adaptation à un apprenant. Ceci est lié à un style de programmation impératif, à la parcellisation et au mélange des connaissances de la matière à enseigner, des prévisions faites par les auteurs sur les réponses possibles de l'apprenant, et enfin, des enchaînements. Dans le cadre de l'EIAO (**E**nseignement **I**ntelligemment **A**ssisté par **O**rdinateur) [Sleeman 82], les générateurs de tuteurs intelligents [Elsom-Cook 90] [Gavignet 91] [Regourd 88] sont souvent restés au stade du prototype. Nous nous plaçons dans une approche intermédiaire : celle d'outils pour l'auteur, mais produisant des tutoriels n'intégrant pas nécessairement toutes les fonctionnalités d'un tutoriel intelligent, ou encore produisant des Environnements Interactifs d'Apprentissage avec Ordinateur, avec la nouvelle acceptation du sigle EIAO suggérée dans [Balacheff 89].

Deux approches existent pour la conception de tutoriels intelligents : le tutorat planifié d'une part, fondé sur une organisation didactique du curriculum, l'exploration libre d'autre part, basée sur le développement de micro-mondes conceptuellement fidèles. Un souci majeur pour l'auteur d'un système d'enseignement est de mettre au point l'équilibre entre ces deux approches, particulièrement pour les domaines qui intègrent des connaissances abstraites mais nécessitent l'acquisition de savoir-faire avant la compréhension de nouveaux concepts.

Les interfaces graphiques réactives sont mieux adaptées au développement de micro-mondes, tandis que le tutorat, reposant sur des stratégies pédagogiques déclaratives, a surtout été étudié en supposant la synthèse d'énoncés en langage naturel pour présenter des connaissances. La synthèse d'une interaction graphique pose des problèmes difficiles de sémantique et de spécification, cependant, les deux approches ne sont pas inconciliables. Nous avons donc écrit un système permettant de piloter une interface graphique réactive par des stratégies tutorielles. Les règles qui traduisent ces stratégies ne permettent pas de construire une unité d'interaction, mais permettent d'évaluer la pertinence d'une séquence, chaque unité étant une ressource pédagogique conçue par l'auteur.

## 2. Les objectifs pédagogiques d'une interaction

On distingue usuellement dans un tuteur intelligent [Sleeman 82] :
- le composant expert qui représente les connaissances du domaine à enseigner,
- le composant pédagogue qui régit l'interaction,
- le modèle de l'apprenant qui contient les connaissances du tuteur sur celui-ci.

C'est essentiellement l'existence de cette représentation de l'élève qui permet une meilleure adaptation à celui-ci et une plus grande interactivité. Malgré d'importants progrès sur les techniques de représentation des connaissances et sur le difficile problème de la modélisation de l'élève et de son apprentissage, cette architecture est l'objet de deux sortes de critique : les unes de nature épistémologique, les autres sur le primat de l'interactivité et la notion de dialogue.

L'apprentissage conjoncturel (Situated cognition [Brown 88]) soutient que l'apprentissage n'est pas un simple transfert incrémental de connaissances mais une assimilation : l'apprenant construit ou reconstruit des concepts sur la base des faits observés et des expériences antérieures. L'épistémologie qui fonde cette approche est qu'une représentation n'est pas "la connaissance" : les connaissances émergent par l'interaction des processus mentaux et de la réalité [Clancey 91a] [Clancey 91b] et la connaissance est indissociable de la perception, de l'action et de la réflexion.

La plupart des travaux sur le dialogue sont basés sur la compréhension du langage naturel et nécessitent des objectifs de dialogue : l'interprétation d'un énoncé dépend de son intention. Le tutorat négocié (Negociated tutoring [Baker 91]), par exemple, fait reposer l'interaction tutorielle sur une négociation explicite des objectifs du dialogue. Celui-ci est considéré comme un jeu de communication où l'élève coopère activement avec la machine pour définir et résoudre un problème. La mise en oeuvre du tutorat négocié est souvent liée à l'utilisation de points de vue [Moyse 90], c'est à dire d'interprétations multiples d'un même modèle abstrait, et à la négociation d'une de ses représentations.

Ces deux nouvelles approches compliquent considérablement la conception du modèle de l'apprenant et du diagnostic. Pour l'apprentissage conjoncturel, le processus de reconceptualisation dépend du contexte global des connaissances de l'apprenant. Le tutorat négocié nécessite de diagnostiquer les intentions de l'élève, objectifs qui ne sont pas forcément clairs (même pour lui), ou d'adopter une tactique métacognitive pour les faire éliciter[1] .

Ces deux approches ont aussi en commun l'utilisation d'objectifs pédagogiques explicites, que ce soit le choix d'un micro-monde pour illustrer un concept ou le choix d'un point de vue pour réifier[2] ce concept dans le cadre d'un problème spécifique. Or, une interface utilisateur graphique ne gère pas d'objectifs ou de plans, elle assure la réactivité, c'est-à-dire la réaction en "temps réel" aux actions de l'utilisateur. On peut même opposer réactivité et planification,

---

[1] L'élicitation sollicite l'élève pour qu'il explique son raisonnement afin de faire émerger ses intentions.

[2] La réification consiste à rendre concret un concept ou un processus abstrait. Dans le contexte d'un tuteur intelligent, c'est rendre une notion à enseigner observable et interactivement manipulable.

autant pour la complexité des calculs (temps borné - complexité exponentielle quand le problème est décidable) que pour le style de programmation (temps réel - intelligence artificielle) [Boufaida 93b]. Nous utilisons donc une division de l'interaction en deux couches : la couche supérieure, décrite par les stratégies tutorielles, fournit une structure intentionnelle à la couche inférieure, formée d'unités d'interaction réactives. Cette organisation permet de définir des objectifs pédagogiques.

### 3. Conception de l'interface utilisateur

La programmation d'une interface utilisateur relève de la programmation "temps réel" mais aussi du système d'exploitation. Le principe de base de la conception consiste à séparer le noyau fonctionnel de l'application de son interface. Il est appliqué par le système de gestion de l'interface du langage Smalltalk que nous utilisons, avec en plus un soin particulier pour obtenir un système extensible et des interactions réutilisables. Ceci conduit à une architecture spécifique (Modèle-Vue-Contrôleur) [Goldberg 84] qui est un squelette d'application et définit des protocoles de communication explicites. Mais d'une part, les principes de conception et les justifications de l'architecture restent largement implicites, d'autre part, il est tout à fait possible de réaliser des interfaces opérationnelles sans respecter l'architecture, au prix d'une réutilisation ultérieure beaucoup plus difficile. Ainsi, l'ouverture d'une fenêtre composée demande de simples compétences en programmation par objets, alors que la définition d'un nouveau composant exige une compréhension du système Smalltalk et de la notion de squelette d'application, et même une expertise certaine pour un composant réutilisable [Campbell 90].

La séparation entre le modèle, représentant la sémantique de l'application, et ses présentations, qui définissent la syntaxe de l'interaction ne peut bien sûr être complète. C'est le cas par exemple, de la rétroaction sémantique dans les systèmes de fichiers utilisant la métaphore du bureau ; quand on tire un document avec la souris, si ce document passe devant un dossier ou une application capable de l'ouvrir, le dossier ou l'application le signale à l'utilisateur en changeant sa présentation visuelle. A un niveau sémantique plus abstrait, le principe de fidélité conceptuelle conduit à refléter la structure du modèle abstrait dans la structure de l'interaction, par exemple dans un éditeur de graphes ou de graphiques structurés. Ce principe est particulièrement important dans une approche didactique : dans un didacticiel d'informatique par exemple, la différence entre un tableau et une liste, notamment entre les accès indexés et séquentiels, conduit à des présentations et des interactions qui réifient ces mécanismes d'accès. Mais d'autres présentations sont plus adaptées à d'autres points de vue ; par exemple pour illustrer l'ajout dynamique d'un nouvel élément, permis pour une liste, mais interdit pour un tableau, ou encore, une présentation synthétique comme agrégat de valeurs, qui est préférable si la structure de donnée n'est pas le sujet principal de l'interaction. On le

voit, le choix d'une présentation dépend de l'objectif de l'interaction et de la situation d'apprentissage.

Des outils de prototypage d'interfaces utilisateurs ont été développés, par exemple, pour Smalltalk, RAPID [Freburger 87] ou PARTS [Digitalk 92] : ils assurent l'assemblage des composants d'interaction et définissent leur synchronisation. Cependant, ces outils ne gèrent ni d'intentions de dialogue, ni de plans, ni même de contextes d'interaction : chaque fenêtre définit un dialogue autonome et indépendant. Ils ne permettent donc pas directement de mettre au point des présentations en fonction du contexte d'apprentissage.

Si la conception de l'interface élève est une part importante du travail de l'auteur, ses principaux soucis portent sur l'ergonomie, sur la fidélité conceptuelle des présentations, et sur la pertinence d'une interaction par rapport à un objectif pédagogique. Un des buts de notre système est d'éviter à un auteur la programmation explicite des ressources d'interaction.

## 4. Description générale du système

Nous avons donc développé un système auteur, baptisé GEODE, pour **G**énérateur d'**E**nvironnements **O**bjets **D**édiés à l'**E**nseignement, avec deux objectifs. L'un, technique, est d'intégrer la conduite de dialogue (décrite par des stratégies pédagogiques) à l'utilisation d'une interface graphique réactive (représentée par des ressources d'interaction). L'autre, pédagogique, est de permettre la mise au point de didacticiels intégrant à la fois le tutorat planifié et l'exploration de micro-mondes.

Ce travail s'insère dans un projet plus large, nommé FORCE (**F**ormat **O**bjet pour la **R**eprésentation des **C**onnaissances **E**ducatives) qui développe un environnement auteur destiné à des didacticiels spécifiques d'informatique. Au coeur de ce projet, se trouve l'utilisation comme micro-mondes d'outils fournis par des environnements pédagogiques de programmation, tel PASCAL/V [Brette 93].

### 4.1. Modélisation de l'environnement créé par GEODE

La structure d'un tutoriel créé par GEODE se compose de deux parties (figure 1) :

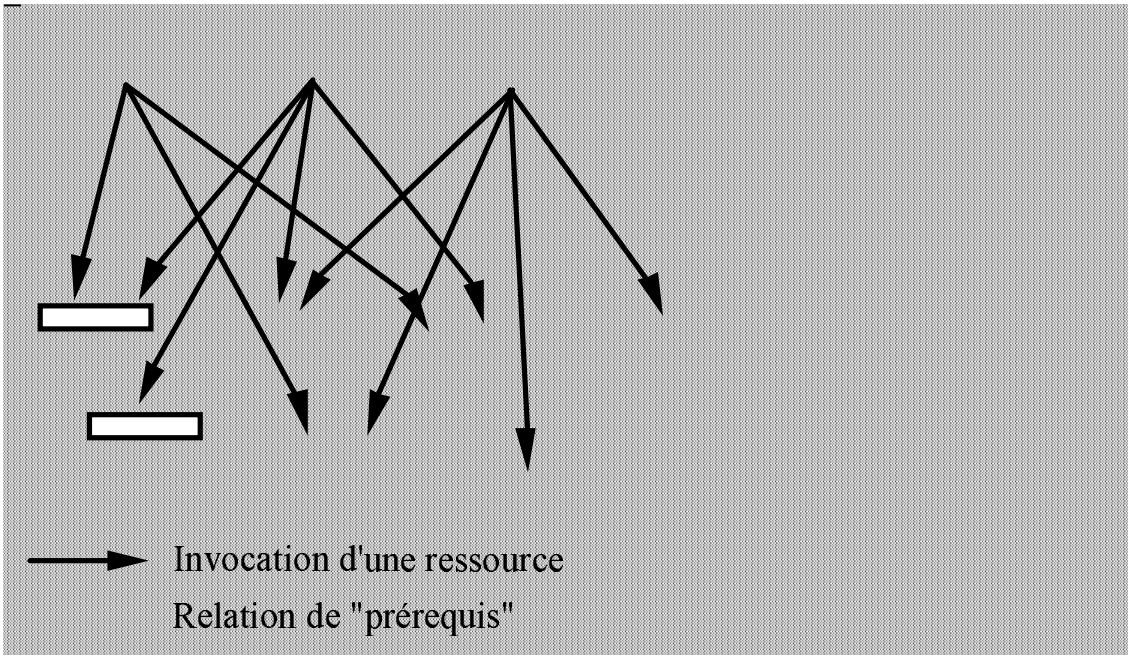



Figure 1. Les deux niveaux d'un tutoriel créé par GEODE

- le niveau **supérieur** correspond aux plans de tutorat et aux objectifs pédagogiques ; chaque plan peut activer des fonctionnalités de tutorat prédéfinies comme l'analyse des réponses ou des tactiques pédagogiques (explication, aide, ...). Les modèles de réponses et l'interaction liée à une tactique sont spécifiques à une ressource d'interaction : l'auteur doit les associer à un noeud du niveau inférieur,

- le niveau **inférieur** contient les ressources d'interaction organisées en réseau de "prérequis". Ces liens définissent des stratégies par défaut consistant à parcourir le graphe et permettent donc de tester des enchaînements avant de concevoir les stratégies définitives. Ils servent aussi de support visuel. Chaque noeud schématise une ressource réactive et ne contient pas d'objectifs pédagogiques (ni de connaissances du domaine).

Un troisième niveau, externe à GEODE, comprend l'ensemble des connaissances du domaine. Les connaissances mises en oeuvre par une interaction sont associées à un noeud spécifique et sont accessibles aux stratégies. Cet accès indirect permet d'articuler souplement un modèle spécifique au problème posé et un modèle général de la matière à enseigner.

Cette division en plans d'enseignement et ressources d'interaction permet de découpler les interactions liées à un exercice de son objectif pédagogique : une même notion peut avoir de multiples présentations, de même qu'une connaissance peut être mise en oeuvre dans plusieurs exercices.

### 4.2. Comportement du système

Le système GEODE fonctionne sous deux modes (figure 2) :
- le mode auteur, divisé en deux sous-modes : Edition et Test,
- le mode élève.

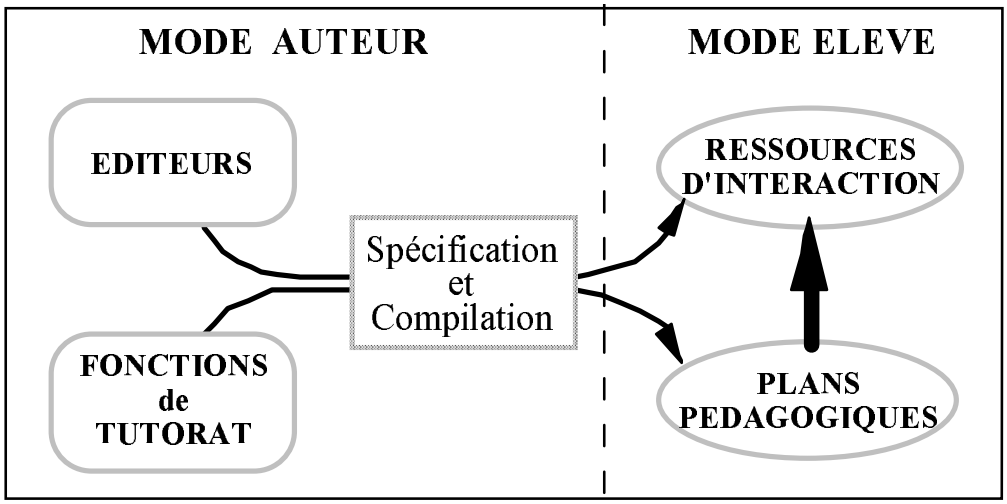



Figure 2. Les deux modes du système GEODE

En sous mode **Edition** du mode auteur, l'interface auteur permet la définition d'une interface élève. Elle utilise deux styles de saisie :
- l'un, par manipulation directe, utilisant la souris et des menus, pour la création d'éléments d'interaction graphique réactifs,
- l'autre, textuel, pour l'édition des stratégies, représentées par des clauses Prolog.

Le sous mode **Test** peut être vu comme un mode élève particulier mais destiné à l'auteur : il évite de sortir de ce mode auteur, et évite donc des opérations automatiques de compilation et d'enregistrement.

En mode **élève**, un "moteur" exécute les plans correspondant au didacticiel produit. Ces plans peuvent invoquer des programmes de diagnostics et prendre ainsi en compte des éléments de réponse fournis par l'apprenant.

## 5. Fonctionnalités offertes aux auteurs

Nous décrivons ici les moyens offerts par GEODE pour la programmation graphique de l'interface élève et la saisie textuelle des règles tutorielles.

### 5.1. La génération des éléments d'interface

L'édition d'un didacticiel commence par la présentation d'une fenêtre graphique (figure 3) associée à un éditeur de réseaux et permettant de spécifier, par manipulation directe, des

noeuds et des liens. Le réseau édité est utilisé comme support pour la définition et l'exécution des stratégies tutorielles. En effet, chaque arc du réseau correspond à une transition possible et permet de définir un enchaînement par défaut (didacticiel "classique"). Un cheminement plus précis et adapté aux réactions de l'élève, spécifie le parcours de ce réseau par des stratégies pédagogiques (didacticiel "intelligent"). A chaque noeud est associé un menu permettant de le nommer et d'ouvrir un éditeur spécialisé. Une option de ce menu permet également d'associer des modèles de réponses qui sont évaluées de manière classique, indépendamment des programmes de diagnostics à caractéristiques intelligentes. Ce menu permet aussi d'associer un dialogue à une tactique pédagogique telle que l'aide ou la remédiation.

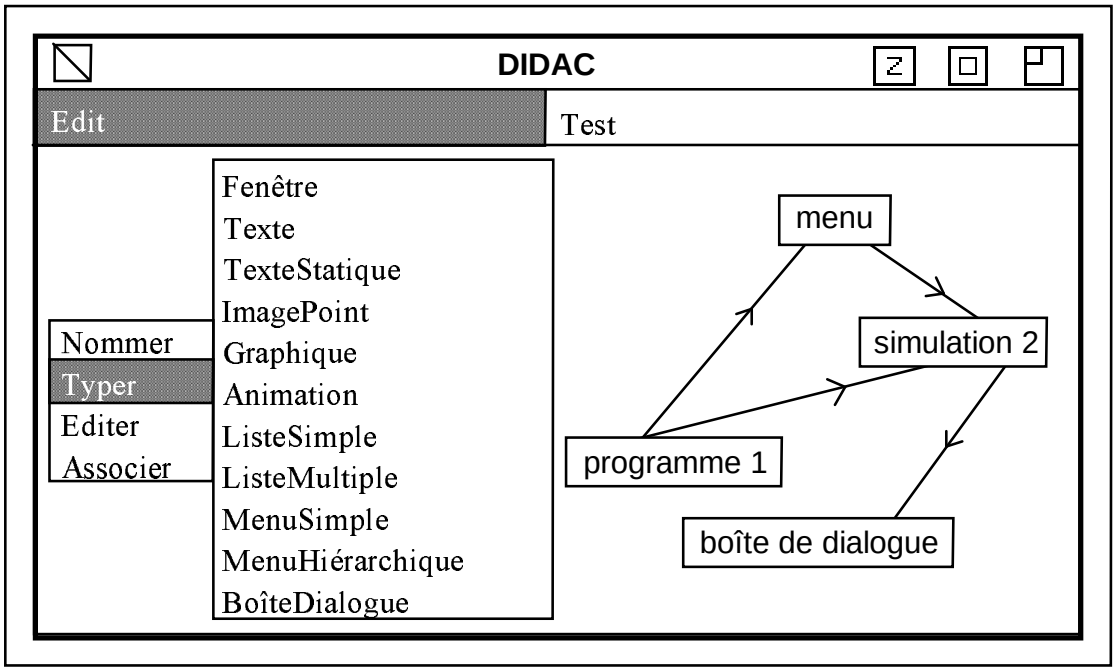



Figure 3. Présentation de la fenêtre liée à l'interface auteur.

Cette interface auteur permet également l'enregistrement et la reprise éventuelle de la mise au point d'un didacticiel ainsi que l'ouverture de la fenêtre correspondant à l'édition des règles pédagogiques.

Actuellement, le système propose onze éditeurs spécialisés tels que ceux pour les listes à sélections multiples, les textes statiques ou les menus hiérarchiques [Boufaida 93a].

### 5.2. La spécification des stratégies pédagogiques

La description et l'exploitation des objectifs et des intentions liées au dialogue font appel à des notions qui relèvent de l'intelligence artificielle, comme des prévisions, des retours en arrière, des analyses d'erreurs, des jeux d'essai, etc … Leur utilisation pédagogique dans un tutoriel nécessite une représentation explicite. Afin de faciliter leur prototypage par l'auteur,

nous avons choisi une représentation déclarative, utilisant le langage Prolog pour lequel il existe un interprète sous Smalltalk. Les stratégies pédagogiques sont donc décrites par des clauses écrites en Prolog. La bonne intégration des deux langages et l'utilisation de la syntaxe de Prolog facilitent aussi l'utilisation par l'auteur du couplage de GEODE avec la structure du domaine et les règles de diagnostic.

La saisie des stratégies pédagogiques est effectuée à l'aide d'une fenêtre divisée en trois panneaux (figure 4) :

a) un premier contenant la ou les classes de stratégies pédagogiques[3] ,

b) un deuxième comprenant les noms des prédicats correspondant aux stratégies pédagogiques[4] ,

c) enfin un troisième permettant d'éditer le texte source d'un paquet de clauses. Chaque prédicat correspond soit à une fonctionnalité prédéfinie de tutorat, soit à une combinaison logique d'autres prédicats.

Au cours du déroulement d'un didacticiel, chaque élément d'interaction peut envoyer des messages aux programmes de diagnostics pour le traitement des réponses de l'apprenant. Les ressources de diagnostics invoquées retournent des résultats aux stratégies tutorielles réparties sur deux couches :

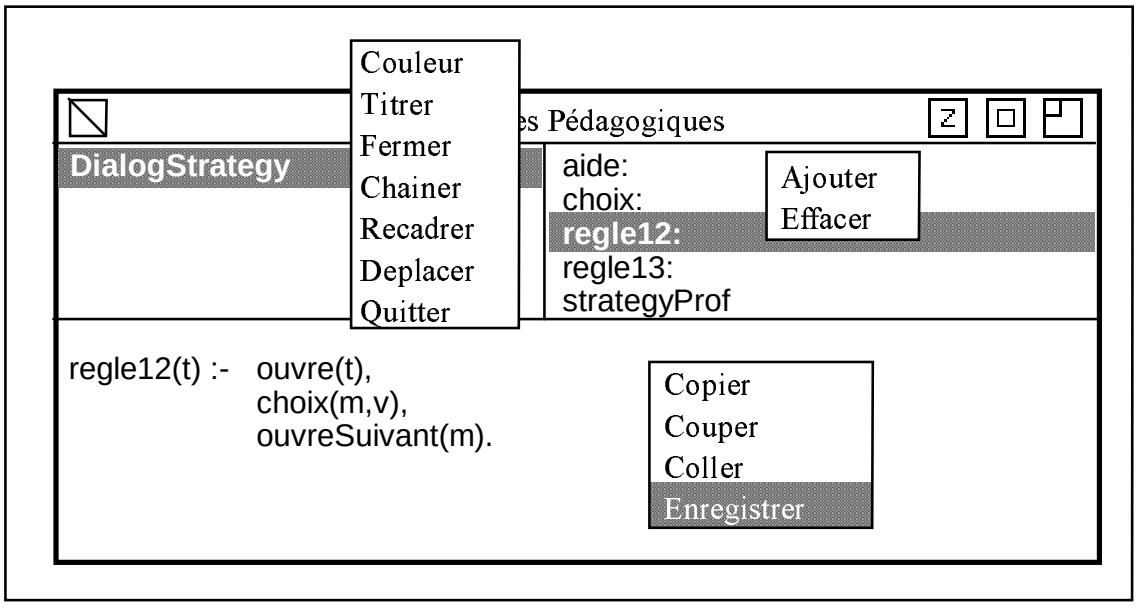



Figure 4. Présentation de la fenêtre correspondant aux stratégies.

---

[3] Grâce à l'héritage de Smalltalk, on peut utiliser plusieurs classes définissant des stratégies pédagogiques.

[4] La compilation de Prolog/V traduit un paquet de clauses par une méthode Smalltalk.

- une couche qui comprend des **stratégies prédéfinies** (reposant sur les liens du réseau ainsi spécifié par l'auteur). Elles sont utilisées comme des stratégies par défaut lorsque l'auteur teste le didacticiel et permettent d'effectuer des parcours "en largeur d'abord" ou en "profondeur d'abord". En mode élève, elles sont exécutées si toutes les règles échouent.

- une couche sur laquelle reposent les stratégies pédagogiques spécifiées par l'auteur et qui sont des combinaisons de primitives prédéfinies de tutorat. Cet ensemble est divisé en deux catégories :

- des **primitives de gestion** qui opèrent sur des ressources d'interaction par exemple ouvrir, suspendre ou fermer une fenêtre. Ces primitives de bas niveau sont indépendantes du domaine et interviennent toujours au cours de l'exécution d'un didacticiel. Elles permettent de revenir sur des noeuds prérequis pour, par exemple, corriger une erreur ou rappeler certaines notions. Elles autorisent aussi à passer aux noeuds postrequis pour présenter un nouvel exercice.

- des **tactiques de dialogue** correspondant à des fonctionnalités plus abstraites dont le système gère l'activation. Grâce à ce mécanisme, l'auteur n'aura qu'à spécifier leurs conditions d'activation et leur contenu sans devoir les programmer. Ce contenu n'est pas synthétisé car il dépend largement des représentations choisies pour les connaissances de la matière et celles de l'élève. Cependant, une tactique peut déléguer des messages et utiliser des composants et des diagnostics spécifiques du domaine. Actuellement, GEODE gère les tactiques suivantes :

- **annonce d'un objectif pédagogique**, activée par défaut à l'ouverture de l'élément d'interaction auquel elle est rattachée. Elle permet à l'élève d'organiser et de structurer ses idées sur la matière avant d'entreprendre la résolution d'un problème.

- **aide**, pouvant être déclenchée dans différentes situations :
  - dans le cas d'un élément évalué par modèle de réponses au bout de trois (par défaut) réponses incomplètes,
  - dans le cas où l'élève la sollicite au moyen d'une option (définie par l'auteur) d'un menu d'une interaction,
  - activation directe par une stratégie.

- **explication**, déclenchée par les stratégies pédagogiques, lors d'une émission, par l'élève, d'une réponse fausse, vraie ou incomplète.

- **remédiation**, activée comme une explication mais liée au diagnostic d'une réponse fausse. En fait, on peut remplacer un écran de remédiation par une séquence de remédiation définie par une stratégie de parcours sur une composante du réseau de ressources.

- **résumé**, déclenchée par la stratégie courante ou par sollicitation de l'élève, utilise le mécanisme de sauvegarde du contexte d'interaction dans un fichier historique.

Toutes les primitives et tactiques peuvent être invoquées par le prédicat de même nom. Par exemple, la règle suivante permet d'abandonner la stratégie courante et d'en activer une autre à l'aide d'un menu :

```
abandonStrategie(currentNode) :- ouvre(currentNode),
                                 menu('Reactif/Logique', 'reactif logic'], #reactif),
                                 suivant(currentNode, suivNode),
                                 changeStrategie(suivNode, 'regle14').
abandonStrategie(currrentNode) :- choix(currentNode, aNode),
                                 ouvre(aNode).
```

La première clause permet d'ouvrir un élément de dialogue présentant l'organisation d'un cours en deux points de vue. L'appel du prédicat **menu()** dans la première clause permet de laisser le choix à l'élève. Le prédicat **suivant()** détermine le premier noeud spécifié par l'auteur. Ensuite, il y a abandon du plan courant et reprise de celui défini par la règle "regle14". Le prédicat **choix()** permet de déterminer le noeud possédant le plus grand nombre de prérequis.

Voici un deuxième exemple portant sur le traitement de réponses :

```
traiter(currentNode) :-      ouvre(currentNode),
                             diagnostiquer(currentNode, model),
                             continuer(model, currentNode).
    continuer(#rep1, currentNode) :-   remedier(currentNode).
    continuer(#rep2, currentNode) :-   choix(currentNode, v),
                                       ouvre(v).
```

La règle **traiter** permet d'ouvrir le noeud courant, de saisir et de traiter la réponse de l'apprenant en émettant un message à un évaluateur qui utilise le modèle de réponses associé, retourné en paramètre, tandis que le prédicat **continuer** choisit le noeud suivant de l'enchaînement.

### 6. Fonctionnalités offertes à l'apprenant

Les interactions en mode élève sont de deux sortes, selon qu'elles nécessitent ou non un accès aux composants externes à GEODE, c'est à dire aux programmes de diagnostics et au modèle de l'élève. Les interactions internes à GEODE correspondent à l'affichage, à la présentation et à la saisie des informations. Le tuteur exécutera dans ce cas précis un plan

pertinent après invocation des diagnostics. La version actuelle de GEODE fournit un certain nombre d'options que l'auteur pourra combiner en un menu lié à une fenêtre de l'interface élève. Les options possibles sont :

**Suite** qui active le noeud postrequis choisi par la stratégie courante,

**Retour Arrière** qui permet de revenir à un noeud prérequis,

**Aide** qui déclenche la tactique de dialogue associée,

**Résumé** qui fabrique une synthèse des notions liées aux noeuds parcourus,

**RepriseDebut** qui reprend à partir de la racine du réseau courant,

**Brouillon** qui ouvre la fenêtre correspondant à un carnet de notes. L'édition, en particulier, les couper-coller, sont gérés par le système Smalltalk.

**Sortie** qui correspond à l'abandon d'une session. Si le noeud courant est terminal[5], le didacticiel est fini et la session se termine sans qu'aucune opération ne s'effectue. Si au contraire, ce noeud de sortie possède des postrequis, son identification est enregistrée dans une variable globale, ainsi que le contexte. Ce mécanisme gère les points d'arrêt et de reprise des didacticiels.

Nous avons choisi de nous conformer au style d'interface standard de Smalltalk pour l'interface auteur. Même si l'ergonomie en est critiquable, ce style est sanctionné par son succès commercial et offre l'avantage d'une meilleure intégration entre GEODE et Smalltalk quand l'auteur doit programmer des modules spécialisés. Une question plus intéressante est de savoir dans quelle mesure l'architecture MVC de Smalltalk d'une part, et l'approche de GEODE d'autre part, contraignent l'auteur pour concevoir l'interface élève. Pour MVC, Brette [Brette 94] décrit des difficultés pour gérer des fenêtres auxiliaires permettant de réifier un aspect spécifique d'un modèle dont dépend la fenêtre principale, mais pouvant devenir autonome. Bouabsa [Bouabsa 94] décrit des techniques de solutions en termes de contraintes. Dans le cadre de GEODE, la gestion structurée de ces fenêtres auxiliaires doit être interfacée par de nouvelles primitives et des tactiques de dialogue.

## 7. Architecture logicielle

L'architecture de GEODE est divisée en deux parties : l'une liée au mode auteur, l'autre au mode élève. Certains de ses éléments sont invoqués seulement dans l'un des deux modes.

---

[5] Un noeud terminal est un noeud ne possédant pas de postrequis.

En **mode auteur,** elle est constituée des composants suivants (figure 5.a) :

- une **boîte à outils spécialisés** pour l'EAO. Elle regroupe un ensemble d'éditeurs d'éléments d'interface graphique prédéfinis et paramétrables permettant de définir les ressources d'interaction,

- un **administrateur de ressources** qui appelle les différents éditeurs et le gestionnaire de dialogue,

- un **gestionnaire de dialogue** qui permet la saisie des stratégies tutorielles, et engendre des plans combinant les fonctions prédéfinies de tutorat,

L'accès aux différents composants externes (modèle de l'élève et connaissances du domaine) est effectué par délégation de messages.

En **mode élève**, il comporte (figure 5.b) :

- l'**administrateur de ressources** (le même que dans le mode auteur) qui gère les interactions et communique avec le "moteur" d'exécution,

- un "**moteur" d'exécution** qui met en oeuvre un didacticiel et sélectionne les stratégies pédagogiques spécifiées dans le gestionnaire de dialogue,

- le **gestionnaire de dialogue** qui active les stratégies sous forme compilée (c'est à dire les plans) et invoque les ressources d'interaction,

- un **évaluateur** des réponses émises par l'élève,

De plus, GEODE met à jour un **historique de l'élève** qui contient un suivi de l'interaction et sert de modèle de l'élève par défaut.

## 8. Implantation

La version actuelle du prototype s'exécute sur une machine de type PC/AT sous le système d'exploitation MS/DOS et est implantée en Smalltalk/V [Digitalk 88]. Ce langage à objets possède une interface utilisateur réactive et graphique et est bien adapté à nos objectifs, d'autant plus qu'il offre des possibilités de programmation déclarative en Prolog/V. Nous avons strictement respecté les protocoles du modèle MVC (**M**odèle-**V**ue-**C**ontrôleur) de l'interface homme-machine.

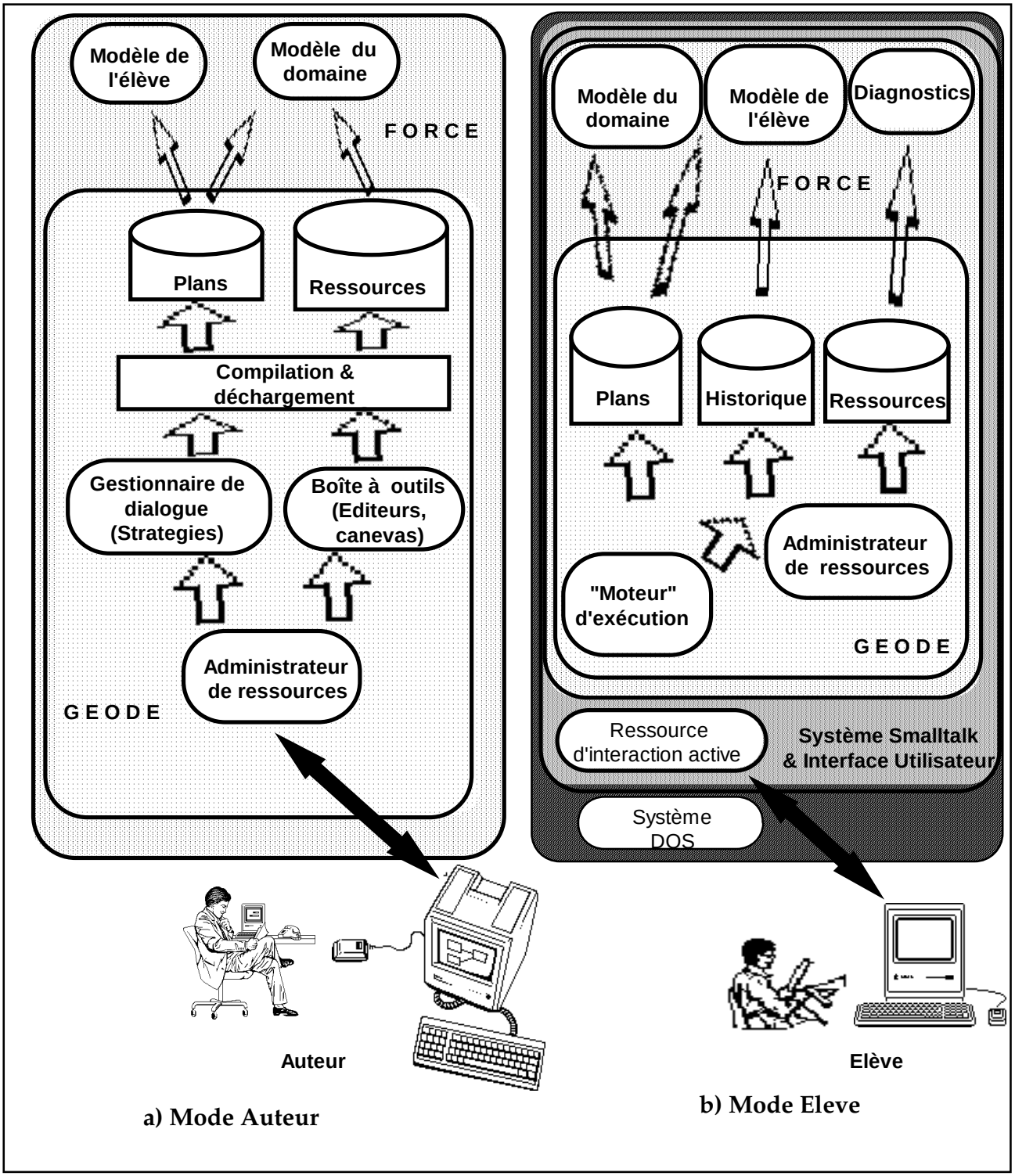



Figure 5. Architecture logicielle de GEODE

Chaque tâche assurée par notre système est distribuée dans plusieurs classes qui coopèrent [Barril 93]. L'implantation d'une tâche par plusieurs classes permet une utilisation plus fine de l'héritage et une meilleure réutilisablité du code. Cette méthodologie permet

l'extension de GEODE à d'autres outils de spécification d'éléments réactifs, ou encore l'incorporation de composants externes spécifiques, par exemple une représentation déclarative de la matière à enseigner ou des programmes de diagnostics spécialisés.

Une fonctionnalité ne correspond donc pas à une classe à cause des collaborations mais aussi à cause du code dynamique. Nous utilisons du code suspendu (des blocs[6] Smalltalk) pour l'ouverture d'une unité d'interaction et des processus[7] pour les plans pédagogiques. Ainsi, le "moteur" d'exécution est distribué entre l'interprète Prolog (une boucle de passage de continuation et l'unification), le système Smalltalk (l'administrateur de processus, l'interface utilisateur et la machine virtuelle) et les structures créées par GEODE (blocs d'ouverture, boîtes aux lettres).

Toutes les primitives et tactiques de dialogue ont été adaptées au format d'appel des prédicats Prolog; l'auteur peut ainsi les invoquer dans les stratégies pédagogiques. L'activation d'une tactique spécifiée dans un noeud est matérialisée par l'évaluation d'un prédicat. Par exemple, **tactique**(**x**) où tactique est un prédicat et l'argument **x** représente le noeud correspondant.

L'articulation entre l'interface graphique (gérée principalement par la classe **Geode** sous classe de **Object**) et l'interface déclarative (gérée par la classe **DialogStrategy** sous classe de **Prolog**) est basée sur des protocoles de coopération.

Le système Smalltalk définit un processus "interface utilisateur", le seul qui ait accès aux évènements créés par les périphériques et puisse ouvrir une fenêtre. Cette ouverture entraîne une perte du contexte de contrôle. Le mode élève, utilise donc deux processus, l'un pour le plan pédagogique courant, l'autre, le processus "interface utilisateur", pour les primitives d'interaction. Les deux sont synchronisés par un mécanisme de boîte aux lettres (utilisant deux sémaphores) dans le cas d'un élément de dialogue non modal (par exemple, une fenêtre). La synchronisation est différente pour un élément d'interaction modal (par exemple un menu ou une boîte de dialogue[8]) et un peu plus complexe dans le cas de l'abandon du plan en cours (prédicat **changeStrategie**()) où l'on doit relancer un troisième processus.

## 9. Perspectives

---

6 Un bloc Smalltalk correspond à du code suspendu. C'est une fonction anonyme que l'on peut affecter à une variable et invoquer autant de fois que nécessaire.

7 Un processus au sens Smalltalk correspond à un fil de contrôle. C'est un objet qui représente une copie de la pile de la machine virtuelle. Les processus permettent le partage de la machine virtuelle en coroutine mais tous ont le même environnement d'objets (image virtuelle).

8 En Smalltalk, un prompteur par exemple.

Avant d'esquisser l'évolution de GEODE vers un générateur de tutoriels intelligents, il est intéressant de le situer par rapport aux architectures proposées pour l'interface homme-machine. La boîte à outils qu'offre GEODE remplit des lacunes du système d'exploitation MS-Dos. Les systèmes plus récents (MS-Windows, OS/2) offrent des services équivalents, ce qui a conduit à l'abandon de MVC et au choix pour Smalltalk de la programmation par instance (utilisée par la boîte à outils de PARTS). L'adaptation de GEODE le rendra plus indépendant de la plateforme et d'autre part, constituera une ouverture vers des applications spécialisées multimédias.

Les architectures d'interfaces dans les systèmes Smalltalk sont de trop bas niveau pour concevoir des interfaces sophistiquées, des simulations par exemple, tout en gérant simplement les dépendances entre les différents composants. Le modèle PAC (Présentation-Abstraction-Contrôle [Coutaz 90]) permet une conception multi-agents, adaptée à des dialogues à plusieurs fils d'activités. Les ressources d'interactions, les plans et le mécanisme de boîtes aux lettres sont analogues aux sommets respectifs d'une triade PAC, bien qu'une partie du contrôle soit assurée par les dépendances et que le modèle de GEODE ne soit pas hiérarchique. Il est donc envisageable de piloter une implantation de PAC en Smalltalk par des stratégies pédagogiques liées à des agents.

La conception de GEODE est beaucoup plus proche du paradigme de la "Surface d'Interaction" [Took 90]. Cette architecture repose sur un médium (la surface d'interaction), l'application et l'utilisateur étant tous deux clients du modèle d'interaction construit sur ce médium. Dans notre cas, le processus "interface utilisateur" correspond à la surface, les éléments réactifs correspondent au modèle. L'évolution de GEODE vers un générateur de tutoriels divise l'application en deux parties, correspondant aux stratégies et à des processus d'espionnage. La structure logique d'un tutoriel distingue d'une part, des processus d'enseignement, pilotés par la structure du domaine, et d'autre part, des processus d'apprentissage, pilotés par les diagnostics. Une fonction essentielle du didacticiel est de les synchroniser.

L'implantation de Prolog sous Smalltalk présente des inconvénients dans cette optique. Le langage doit être étendu pour permettre de synchroniser plusieurs fils d'activité et aussi pour gérer logiquement des états mutables (par exemple la liste des fenêtres contrôlées, ou des assertions dans le modèle de l'élève. L'utilisation des prédicats **assert**() et **erase**() n'est qu'un pis-aller). De même, la représentation d'un plan en cours d'exécution par un processus Smalltalk, au lieu d'un objet plus simple, est peu efficace pour coder le contexte logique. Le remplacement de l'interprète par un moteur de plans, adapté à la gestion de la concurrence et

des effets de bord, mais doté d'une syntaxe déclarative simple, permettant à l'auteur de raisonner sur des agents, est donc un développement prioritaire pour l'évolution de GEODE.

**10. Conclusions**

Nous avons présenté le système GEODE permettant d'engendrer des environnements d'apprentissage reposant sur des représentations en deux niveaux : réactif (interface graphique) et logique (dialogue). C'est cette articulation qui permet la mise au point de tutoriels équilibrés entre le tutorat planifié et l'approche micro-monde.

Ce système offre la possibilité de définir des intentions et des objectifs pédagogiques sur des enchaînements d'objets réactifs, ce qui constitue une étape préliminaire pour l'élaboration et la gestion de dialogues, fondés sur la notion de négociation, par exemple.

Nous avons esquissé des perspectives d'amélioration, liées principalement à la technique ou à l'ingénierie des interfaces homme-machine. Cependant, les développements les plus intéressants ne peuvent provenir que de l'écriture de tutoriels spécifiques. En particulier, une voie à étudier consiste à raffiner le sens de tactiques comme l'aide ou l'explication, en fonction de situations comme la résolution de problème, l'assimilation d'une connaissance ou l'exploration.

**Références**

[Baker 91] M.J. BAKER: "Negotiating Goals in Intelligent Tutoring Dialogues", in Costa, E. (Eds), New Directions in Intelligent Tutoring Systems, Springer, 1991.

[Balacheff 89] N. BALACHEFF : "Exigences épistémologiques des recherches en EIAO", Journées EIAO de Cachan, 18-19 décembre 1989, Rapport IBP-LAFORIA 31-90, p. 317-339, 1989.

[Barril 93] P. BARRIL, M. BOUFAIDA, J-F. BRETTE : "Class cooperation in a dedicated object system : the FORCE authoring environment", 10th Int. Conf. TOOLS Europe 93, Versailles, France, 1993.

[Bouabsa 94] M. BOUABSA, J-F. BRETTE, P. BARRIL : "Les dépendances comme objets de première classe", Rapport IBP-MASI, n° 94-24 , 13 p., 1994.

[Boufaida 93a] M. BOUFAIDA, P. BARRIL : "Situated planning and resources in the design of an intelligent tutorial", Proceedings of the 21th ACM Computer Science Conference '93, Indianapolis, USA, 1993.

[Boufaida 93b] M. BOUFAIDA, P. BARRIL : "Object-Oriented Authoring Tools for Tutoring Strategies in a Graphical Student Interface", IEEE Conference on Computer Aided Engineering Education CAEE'93, Bucharest, Romania, September 1993.

[Brette 93] J-F. BRETTE, P. BARRIL : "Pascal/V : un environnement pédagogique Pascal sous Smalltalk/V", dans Environnements Interactifs d'Apprentissage avec Ordinateur, Troisièmes Journées EIAO de Cachan, France, Février 1993, p. 91-102, Eyrolles, Paris, 1993.

[Brette 94] J-F. BRETTE : "PASCAL/V : Un environnement pour l'apprentissage de le programmation par découverte guidée. Réification et interprétation contextuelle du rôle des variables." Thèse de l'Université Pierre et Marie Curie, PARIS VI, Mai 1994.

[Brown 88] J.S. BROWN: "Towards a new epistemology for learning", in Intelligent Tutoring Systems 88, Montréal, Canada, 1988.

[Campbell 90] R.L. CAMPBELL : "Developmental scenario analysis of Smalltalk programming", ACM Conf. on Human Factor in Computing Systems, CHI'90, p. 269-276, ACM Press, 1990.

[Clancey 91a] W. CLANCEY: interview realized by J. Sandberg, in Artificial Intelligence Communications, Vol. 4, n°1, p. 4-9, March 1991.

[Clancey 91b] W. CLANCEY: "Situated Cognition: Stepping out of Representational FlatLand", in Artificial Intelligence Communications, Vol.4, n°2/3, p.109-112, June/September 1991.

[Coutaz 90] J. COUTAZ : "Interfaces Homme-Ordinateur : Conception et réalisation", Dunod, 1990.

[Digitalk 88] Smalltalk\V286: Object Oriented Programming System (OOPS), Tutorial and Programming Handbook. Digitalk Inc, Los Angeles, 1988.

[Digitalk 92] PARTS Workbench, Parts Assembly and Reuse Tool Set. User's Guide. Digitalk Inc., Los Angeles, 1992.

[Elsom-Cook 90] M.T. ELSOM-COOK, Cl. O'MALLEY: "ECAL: Bridging The Gap Between CAL And Intelligent Tutoring Systems", Computers & Education, Vol. 15, N° 1-3, p. 69-81, 1990.

[Freburger, 1987] K. FREBURGER : "RAPID : Prototyping Control Panel Interfaces", Proceedings of OOPSLA'87, p. 416-422, October 1987.

[Gavignet 91] E. GAVIGNET: "Environnement de conception de systèmes d'apprentissage: Une modélisation de la connaissance pédagogique", Thèse d'Université de Nancy1, 1991.

[Goldberg 84] A. GOLDBERG, D. ROBSON : "Smalltalk-80: The Interactive Programming Environment", Addison-Wesley, 1984.

[Madaule 87] F. MADAULE, P. BARRIL, B. DE LA PASSARDIERE, F. LE CALVEZ, M.M. POC, M. URTASUN: "Quels outils informatiques pour réaliser des didacticiels : une étude comparative", T S I, Vol. 6, n°1, p. 241-257, Dunod, 1987.

[Moyse 90] R. MOYSE: "Implementing Knowledge Negotiation", Proc. of the IFIP TC3 International Conference on Advanced Research on Computers in Education, Tokyo, Japan, 18-20 July 1990.

[Regourd 88] J. P. REGOURD : "GAMETE : un système auteur pour l'EIAO", TSI Vol. 7, n°1, Dunod, 1988.

[Sleeman 82] D. H. SLEEMAN, J. S. BROWN (Eds): "Intelligent Tutoring Systems", Academic Press, (London, NewYork), 1982.

[Took 90] R. TOOK : "Surface Interaction : a paradigm and model for separating application and interface", in Computer Human Interaction, Proc. of CHI'90, p. 35-42, ACM, 1990.

**Mahmoud Boufaida** est enseignant à l'Institut d'Informatique de l'Université de Constantine depuis 1980. Il a séjourné pendant quatre années à Paris, au laboratoire MASI (Méthodologie et Architecture des Systèmes Informatiques) de l'Université Pierre & Marie Curie (Paris VI). Il y a effectué des recherches en EIAO dans le projet FORCE, au sein de l'équipe SIE (Systèmes Informatiques pour l'Enseignement).

**Patrick Barril** est Chargé de Recherches C.N.R.S. au laboratoire MASI. Il a animé le projet FORCE qui a développé des outils pour l'auteur de didacticiels d'informatique. Il travaille sur les interactions cognitives et les architectures d'interfaces homme-machine adaptées à leurs mises au point et plus précisément, sur les méthodes et les formalismes pour les systèmes multi-agents.