Une architecture d'Agent Conversationnel Expressif

Yannick Fouquet — Alexandra Berger — Sylvie Pesty

Laboratoire Leibniz-IMAG
46, Avenue Félix Viallet
38031 Grenoble Cedex

RÉSUMÉ. *Dans cet article, nous décrivons une architecture d'Agent Conversationnel Expressif (ACE). Plus particulièrement, nous montrons la dynamique du dialogue engendré par l'agent. Le modèle de dialogue de l'agent s'appuie sur un* Langage de Conversation Expressif *dérivé de la Théorie des Actes de Discours, associé à une gestion d'attentes, de buts et de stratégies. Ce modèle offre une séparation entre le contrôle du dialogue et celui de la tâche à réaliser, ce qui permet d'obtenir une architecture d'agent conversationnel générique instanciable en fonction de la tâche (par exemple une tâche d'aide à la vente sur un site de e-commerce). L'objectif, à terme, est d'obtenir des Agents Conversationnels Animés (ACA) Expressifs leur permettant d'exprimer des attitudes et des émotions aux niveaux verbal (forme de l'énoncé, intonation, etc.) et non-verbal (gestes, expressions, etc.).*

ABSTRACT. *This paper describes an architecture for expressives conversational agents. The dynamics of dialog is shown. The dialog model uses an* Expressive Conversation Language *derived from Speech Acts Theory, associed to the management of expectations, goals and strategies of the interlocutors. This model separate dialog management and task management, which permits a generic architecture with adaptation to the task (for instance, the task of helping someone buying on a e-commerce website. The final purpose is to obtain Expressive Embodied Conversational Agents (ECA) wich could show attitudes and emotions at verbal level (sentence, intonation, . . .) and non-verbal level (gestures, expressions, . . .).*

MOTS-CLÉS : *agent conversationnel, actes de discours, interaction homme-machine, ACA*

KEYWORDS: *conversational agent, speech acts, human-machine interaction, ECA*

1re soumission à *Wokshop sur les Agents Conversationnels Animés - WACA'2006*, le 3 octobre 2006.

1. Introduction

Les langages «classiques» de communication entre agents, comme KQML (*Knowledge Query Manipulation Language*) et FIPA ACL (*Foundation for Intelligent Physical Agent - Agent Communication Language*) sous-tendent que les agents du système, (de la *communauté*), sont des agents artificiels dont l'activité principale est d'échanger des connaissances. Cependant, si nous considérons les systèmes multi-agents étendus aux agents humains (*communautés mixtes*), les agents artificiels ont besoin de plus d'expressivité et donc d'un langage de communication suffisament riche pour augmenter leurs capacités de dialogue et ainsi prendre part à des *délibérations, marchandages, négociations, etc.*

Nous avons ainsi défini, lors de travaux précédents un *Langage de Conversation Expressif* pour les agents artificiels de communautés mixtes [BER 06]. Ce langage s'appuie sur une adaptation récente la Théorie des Actes de Discours [SEA 69, SEA 85, VAN 90] proposée par Chaïb-draa et Vanderveken [CHA 98] : *la sémantique récursive fondée sur les conditions de succès et de satisfaction*. Trente-deux *actes de conversation* adaptés à la communication avec l'humain ont ainsi été définis dans toutes les catégories d'actes, offrant à un agent artificiel la possibilité d'informer, demander, mais aussi de promettre, se plaindre, féliciter, exiger, regretter...

Du point de vue des modèles et architectures d'agents artificiels, les agents cognitifs de type BDI (*Belief Desire Intention*), fondés sur les travaux philosophiques de Bratman [BRA 87] sur l'intégration des intentions au sein d'une théorie de l'action, sont actuellement un paradigme crucial pour la modélisation des actions des agents artificiels. Cependant, ils n'intègrent pas à propremement parler de modèles de dialogue du type de ceux que l'on développe en dialogue homme-machine, adapté à la communication avec l'humain.

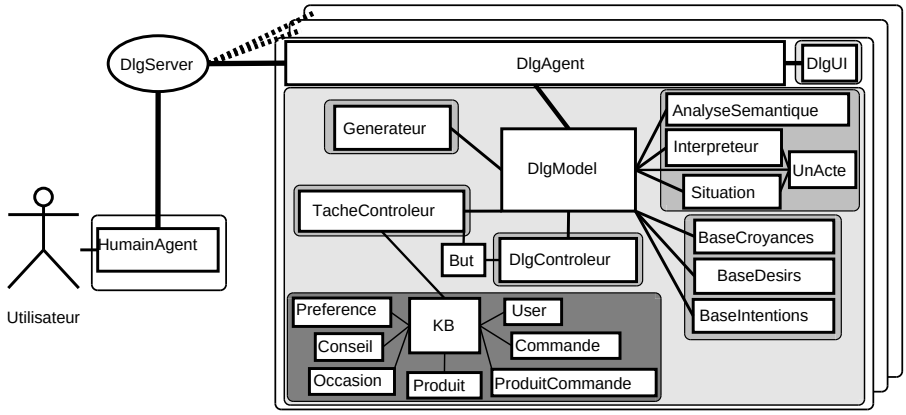
Cet article propose une architecture d'Agent Conversationnel Expressif (ACE) pour des communautés mixtes qui repose, d'une part sur le *Langage de Conversation Expressif* et d'autre part sur un modèle de dialogue explicite qui sépare le contrôle du dialogue de celui de la tâche. Nous montrerons, dans un premier temps, l'architecture globale de l'ACE. Dans un deuxième temps, le modèle de dialogue développé sera approfondi, mettant en relation le *Langage de Conversation Expressif*, la gestion BDI, la gestion des attentes, buts et stratégies et l'interaction entre le contrôle du dialogue (interaction avec l'utilisateur) et celui de la tâche (interaction avec les connaissances).

2. Architecture globale de l'Agent Conversationnel Expressif (ACE) en communauté mixte

Pour présenter l'architecture de l'ACE, nous prenons le cas d'un ACE «vendeur en ligne» dont la tâche est d'aider, par le dialogue, un internaute à acheter un produit sur un site de e-commerce (ici des fleurs). Le dialogue entre l'ACE et l'humain est finalisé et à but délibératif [VAN 01]. L'implantation de l'ACE est réalisée sur la plateforme

Core-DMS (http ://core.imag.fr) qui offre un «middleware» de développement d'applications multi-agents.

Afin d'appréhender globalement l'architecture de l'agent, nous commençons par une brève description de ces modules :



– **DlgServer** : serveur 'relais' pour la mise en place des différents agents-clients et leurs communications.

– **HumainAgent** : agent-client d'interface utilisateur.

– **DlgAgent** : agent-client de l'ACE.

– **DlgUI** : interface du système de dialogue.

– **DlgModel** : noyau du modèle de dialogue.

– **AnalyseSemantique** : analyse sémantique de l'énoncé

– **Interpreteur** : interprétation à partir du schéma sémantique de l'énoncé

– **UnActe** : définition d'un acte de langage

– **BaseCroyances** : gestion d'une base de croyances

– **BaseDesirs** : gestion d'une base de désirs

– **BaseIntention** : gestion d'une base d'intentions

– **Situation** : gestion d'une situation selon l'acte interprété et les bases BDI

– **DlgControleur** : contrôle du dialogue à partir de la situation courante

– **TacheControleur** : contrôle de la tâche à partir de la situation courante

– **But** : gestion des buts

– **Generateur** : génération de l'énoncé réponse à partir du type de réponse et des éléments déterminés par le système

– **KB** : base de connaissances

– **User, Commande, etc.** : modules d'interaction avec la base de données associée

D'après ce schéma, on voit que l'architecture de l'ACE s'articule autour d'un agent-serveur **DlgServer** sur lequel se connectent les agents-clients, permettant ainsi une communication multi-partie. Cet agent-serveur aura en charge la mise en place des autres agents et l'acheminement de leur communication.

Ici, nous nous contenterons de deux interlocuteurs. Deux agents-clients seront donc introduits sur la plateforme. L'un des interlocuteurs est l'utilisateur qui interagit par le biais de l'agent-client spécifique **HumainAgent**. Cet agent-client joue le rôle d'interface vers l'utilisateur. Il présente les messages reçus et envoie les énoncés de l'utilisateur à l'ACE. Il se présente actuellement comme une interface graphique minimale composée de l'historique des messages, d'une zone de saisie du texte à fournir au système, d'un bouton pour envoyer ce texte et d'une liste des agents présents sur le système. A moyen terme, un système de reconnaissance vocale (Sphinx 4[1]), adapté à la langue française, pourra remplacer ou compléter la zone de saisie. De même, un système de synthèse de parole (FreeTTS[2]) pourra remplacer ou compléter l'historique du dialogue.

Le deuxième agent-client introduit sur la plateforme, l'agent-client **DlgAgent** est la partie de l'ACE qui est en charge de répondre à l'utilisateur humain. Il propose ici une interface **DlgUI** de type trace pour observer l'évolution du traitement des énoncés. Cette interface peut également consister en un magicien d'Oz pour des besoins expérimentaux. Un avatar représentant l'agent conversationnel pourra enrichir le système sur les aspects non-verbaux (gestes, expressions, etc.)[FOU 05]. Comme nous allons le voir dans la suite, le coeur de l'ACE se situe dans le modèle de dialogue du module **DlgModel**.

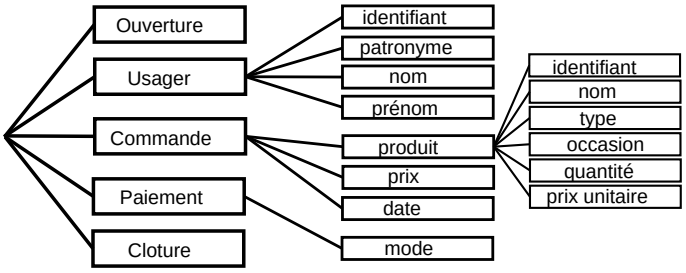
3. Modèle de dialogue pour Agent Conversationnel Expressif

Le modèle de dialogue est issu de travaux de recherche en dialogue homme-machine [CAE 02, CAE 03, CAE 04, FOU 04]. Il repose sur le principe de séparation du contrôle du dialogue (interaction avec l'utilisateur) de celui de la tâche (interaction avec les connaissances) permettant ainsi de garder une certaine généricité à l'agent. Il définit également deux niveaux de traitement des énoncés. Un niveau local permet à l'agent de raisonner en termes de croyances, désirs et intentions. Un niveau global permet à l'agent de raisonner en termes de buts dialogiques et stratégies.

Le module de gestion de la tâche **TacheControleur** concerne l'application (ici la tâche de vente en ligne). Il gère les tâches que l'agent peut effectuer ainsi que ses connaissances. Il offre des fonctions permettant d'interagir avec la **base de connaissances** et la base de données associée[3] (**KB** et ses sous-modules : user, commande,

1. http ://cmusphinx.sourceforge.net/sphinx4/
2. http ://freetts.sourceforge.net/
3. La base de données sert à stocker les différentes informations utiles à l'application (ici, les différents types de bouquets, de fleurs, les prix, l'occasion pour laquelle l'utilisateur souhaite acquérir son bouquet - un anniversaire par exemple, etc.).

etc.), et permet de dégager une organisation des tâches (un arbre) que l'agent est en mesure de traiter. L'arbre des tâches sert à établir un plan pour résoudre chacune d'elle, et définit les thèmes que l'on peut aborder au cours du dialogue. Par exemple pour l'application visée, l'arbre des tâches est le suivant :



Le traitement des tâches consiste à déterminer le but de tâche à résoudre (module **But**), à tenter de l'atteindre puis de le satisfaire. La base de connaissances (module **KB** et ses sous-modules) permet d'en atteindre certains (*e.g.* le prix d'un produit). Les autres nécessiteront l'intervention d'un (ou plusieurs) autre agent (*e.g.* l'agent humain pour déterminer la quantité voulue).

Le module de gestion du dialogue **DlgControleur** doit coordonner les interactions entre les agents-clients, et assurer l'avancement et la cohérence du dialogue. Pour cela, il s'appuie sur les tours de parole précédents. Il doit d'abord déterminer le thème et le but souhaité par l'utilisateur à partir du module de la tâche. Pour résoudre le but, une gestion des attentes associée à un calcul des stratégies tente de fournir la réponse la plus adaptée possible, comme nous allons le voir par la suite.

En interface entre les deux modules **TacheControleur** et **DlgControleur**, le modèle de dialogue **DlgModel** analyse l'énoncé de l'utilisateur et appelle les différents modules pour répondre. Voici un exemple d'énoncé de l'humain :

Je voudrais commander un bouquet de fleurs.

Le module **DlgModel** commence par faire appel à un module d'analyse sémantique **AnalyseSemantique** qui renvoie un schéma sémantique correspondant à l'énoncé. Ce schéma est envoyé au module d'interprétation **Interpreteur** qui renvoie le ou les actes correspondant au schéma. Ces deux modules sont actuellement dans une version très simplifiée et procèdent par identification sur une ontologie. L'énoncé est ainsi formalisé par un acte de la forme Acte-de-Conversation(Contenu-Propositionnel), ce qui donne pour l'exemple précédent :

demander (commande(bouquet))

L'acte de conversation (demander ici) est ensuite analysé en regard de sa définition formelle donnée exprimée en calcul des situations et donnée ici à titre d'illustration (pour plus de détails, se référer à [BER 06]). Soient i et j les deux interlocuteurs et p le contenu propositionnel :

demander i j p : *force illocutoire : directif*
mode d'atteinte : not-oblig(j,i,a)[s']
condition préparatoire : wish(i,do(i,a))[s']

$$
\begin{array}{rrl}
\text{Soit} & & (\forall p)(\forall i,j) \\
& s = & do(says.to(i,j,\langle demander,p\rangle),s_u,0,0) \\
\text{Soit} & & (\forall p')(\forall a)(p \Rightarrow a)(\forall s')(s' \succ s) \\
& s_u = & bel(i,can(j,a,p')[s] \wedge bel(i,Possible(j,a))[s] \\
& & \wedge\, wish(i,do(j,a))[s] \wedge not-oblig(j,i,a)[s'] \\
\text{Et} & s' = & a[s'] \wedge p[s']
\end{array}
$$

L'acte est alors instancié avec la situation courante réelle (module **Situation**). Cette situation se compose de l'acte (module **UnActe**), ainsi qu'une base de croyances (module **BaseCroyances**), une base de désirs (module **BaseDesirs**) et une base d'intentions (module **BaseIntention**) associées. La situation est ainsi modélisée comme un acte de conversation associé à une base BDI telle qu'elle est au moment de l'énonciation. Cela donne, pour notre exemple :

demander bob lea commander(bouquet) :

$$
\begin{array}{rrl}
& s = & do(says.to(bob,lea,\langle demander,commander(bouquet)\rangle),s_u,0,0) \\
\text{Soit} & & (\forall p')(\forall a)(commander(bouquet) \Rightarrow a)(\forall s')(s' \succ s) \\
& s_u = & bel(bob,can(lea,a,p')[s] \wedge bel(bob,Possible(lea,a))[s] \\
& & \wedge\, wish(bob,do(lea,a))[s] \wedge not-oblig(lea,bob,a)[s'] \\
\text{Et} & s' = & a[s'] \wedge commander(bouquet)[s']
\end{array}
$$

Cette situation est alors transmise au module de contrôle du dialogue **DlgControleur** et au module de contrôle de la tâche **TacheControleur** pour déterminer la réponse à fournir (type de réponse et éléments). La cohérence de l'acte est ensuite testée en fonction des actes précédents.

Nous vérifions par unification les conditions de succès de l'acte en regard des situations précédentes, de la situation présente et/ou des situations futures (qu'il faudra donc vérifier plus tard).

Cela revient à vérifier :

$$
\begin{array}{rl}
& bel(bob,can(lea,a,p')[s] \wedge bel(bob,Possible(lea,a))[s] \\
& \wedge\, wish(bob,do(lea,a))[s] \wedge not-oblig(lea,bob,a)[s'] \\
\text{Et} & s' = a[s'] \wedge commander(bouquet)[s']
\end{array}
$$

Les bases de croyances, désirs et intentions sont mises à jours en rajoutant les nouvelles données tirées de l'acte.

Nous vérifions ensuite, par unification également, les conditions de satisfaction[4] :

$$satis_{wd}^{wl}(says.to(bob, lea, \langle demander, commander(bouquet)\rangle), s) \equiv$$
$$\exists(s', s'')(s' \succ s \succ s'')\ Possible(a, s'), ..., Possible(a, s'') \wedge$$
$$success(says.to(bob, lea, \langle demander, commander(bouquet)\rangle), s) \supset$$
$$commander(bouquet)[do(a, do(a, do(a, s''))]$$

Les conditions de satisfaction de cet acte sont : les conditions de succès de celui-ci et le fait de commander effectivement (avec paiement, etc.) un bouquet dans les situations suivantes (immédiatement future ou suivantes).

Les conditions de satisfaction nous orientent vers une position dans l'arbre des tâches. Nous déterminons ainsi l'acte à fournir en réponse par l'intermédiaire de différents sous-systèmes, fonctionnant en parallèle : l'arbre des tâches (et notre position actuelle dans cet arbre), le calcul des attentes [FOU 04], le calcul des buts [CAE 04], le calcul des stratégies [CAE 03] et la base de connaissances. Pour notre exemple[5] :

Attentes	*:*	*<faire, commander(bouquet)>*
		<questionner, bouquet>
Stratégie	*:*	*coopératif*
Arbre des tâches	*:*	*thème = commande(bouquet)*
		sous-buts = type , occasion
		but b1 = commande(bouquet)
		but b2= occasion
		b1 mis en attente . . .

a = <questionner, occasion(bouquet)>

Lorsque l'acte est déterminé, celui-ci est envoyé au **Generateur** qui aura en charge de lui donner sa forme finale. Comme pour l'interprétation, la génération est volontairement simplifiée pour le moment. Celle-ci se fonde sur une ontologie pour fournir l'énoncé de réponse que le **DlgModel** enverra au **DlgAgent** afin que ce dernier la publie en l'envoyant au serveur **DlgServer** qui le redistribuera à tous les clients.

Ex : *a = <questionner, occasion(bouquet)>*
M : Pour quelle occasion ?

Ainsi, le modèle de dialogue de l'ACE s'appuie sur le *Langage de Conversation Expressif* (permettant de représenter les expressions et attitudes, verbales ou non-verbales, de l'agent), sur une gestion des attentes, des buts et des stratégies, et sur une gestion des croyances, désirs et intentions. Il sépare au mieux le contrôle du dialogue (interaction avec l'utilisateur) de celui de la tâche (interaction avec les connaissances), permettant ainsi son adaptation rapide à d'autres tâches sans changer le «moteur d'interaction» de l'ACE avec l'utilisateur.

4. do(a,do(a, do(a,s")) = attente de a
5. Un agent *coopératif* tentera toujours de répondre et de compléter au maximum ses réponses (*e.g.* en donnant des précisions sur le produit).

4. Conclusion et perspectives

Cet article a présenté une architecture pour les Agents Conversationnels Expressifs. Cette architecture inclue un *Langage de Conversation Expressif* et un modèle de dialogue explicite. Elle tente d'intégrer à la fois les richesses de la Théorie des Actes de Discours *via* le langage et les travaux sur le contrôle du dialogue. Nous obtenons ainsi un modèle d'agent capable de gérer la conversation à ses deux niveaux : local, par la dynamique du langage, et global, par la gestion globale de la tâche. L'implantation d'une maquette pour le *e-commerce* est en cours. Elle permet déjà de représenter ces deux niveaux fondamentaux du dialogue, et valide les choix faits pour la définition du *Langage de Conversation Expressif*.

Le noyau développé permettra d'incarner (module **DlgUI**) des Agents Conversationnels Animés (ACA ou ECA pour *Embodied Conversational Agents*) possèdant une représentation graphique, dont le profil ou les attitudes (expressions faciales ou gestuelle) peuvent orienter la façon de «parler». L'objectif est ainsi d'obtenir des Agents Conversationnels Animés Expressifs. Ce noyau permet de représenter l'expression des attitudes et des émotions. Les ACA visés pourront ainsi avoir le «désir» de remercier leur interlocuteur en y mettant les formes : intonation, sourire, gestuelle, etc.

5. Bibliographie

[BER 06] BERGER A., PESTY S., FOUQUET Y., « Un Langage de Conversation Expressif pour agents de communautés mixtes », *Actes de JFSMA*, 2006.

[BRA 87] BRATMAN M. E., *Intention, Plans, and Practical Reason*, H. U. P., 1987.

[CAE 02] CAELEN J., « Modèles formels du dialogue », rapport, 2002, Assises du GdR I3.

[CAE 03] CAELEN J., « Stratégies de dialogue », A. HERZIG B. C.-D., MATHIEU P., Eds., *Modèles Formels d'Interaction, Actes des secondes journées francophones*, Cépaduès, 2003, p. 29-39.

[CAE 04] CAELEN J., NGUYEN H., « Gestion de buts de dialogue », *Actes de TALN, XIème Conférence sur le Traitement Automatique du Langage Naturel*, vol. ISBN 2-9518235-5-5, Fès, avril 2004, p. 345-350.

[CHA 98] CHAIB-DRAA B., VANDERVEKEN D., « Agent Communication Language : A Semantics based on the Success, Satisfaction and Recursion », *Proceedings of ATAL*, 1998.

[FOU 04] FOUQUET Y., « Modélisation des attentes en dialogue oral », PhD thesis, Université Joseph Fourier (Grenoble), octobre 2004.

[FOU 05] FOUQUET Y., CAELEN J., « Deux plates-formes pour l'expérimentation d'un agent conversationnel animé », *Actes de WACA*, Grenoble, juin 2005, p. 23-30.

[SEA 69] SEARLE J., *Speech Acts*, Cambridge U. P., 1969.

[SEA 85] SEARLE J. R., VANDERVEKEN D., *Foundation of Illocutionary Logic*, C. U. P., 1985.

[VAN 90] VANDERVEKEN D., *Meaning and Speech Acts*, vol. 1 et 2, C. U. P., 1990.

[VAN 01] VANDERVEKEN D., « Illocutionnary Logic and Discourse Typology », *Revue Internationale de Philosophie*, vol. 55, n° 2, 2001, p. 243-255.