Greta : Une Plateforme d'Agent Conversationnel Expressif et Interactif

Etienne de Sevin[*] — Radosław Niewiadomski[*] — Elisabetta Bevacqua[*] — André-Marie Pez[*] — Maurizio Mancini[**] — Catherine Pelachaud[*]

[*] *CNRS - Telecom ParisTech*
37/39, rue Dareau
75014 Paris, France
email : {de-sevin, niewiado, bevacqua, pez, pelachaud}@telecom-paristech.fr

[**] *InfoMus Lab, DIST - Università di Genova*
via Causa 13
I-16145, Genova - Italy
email : maurizio@infomus.org

RÉSUMÉ. *Ce papier présente une architecture générique, modulaire et interactive pour un agent conversationnel animé appelé Greta. Celle-ci est un agent capable de communiquer avec des utilisateurs de façon verbale et non-verbale comme le regard, les mouvements de la tête et du torse, les expressions faciales et les gestes. Notre architecture respecte le standard SAIBA qui définit la modularité, les fonctionnalités et les protocoles de communication pour les ACAs. Dans cet article, nous présentons notre architecture, les tests de performance ainsi que plusieurs applications interactives*

ABSTRACT. *This paper presents a generic ,modular and interactive architecture for embodied conversational agent called Greta. It is 3D agent able to communicate with users using verbal and non verbal channels like gaze, head and torso movements, facial expressions and gestures. Our architecture follows the SAIBA framework that defines modular structure, functionalities and communication protocols for ECA systems. In this paper, we present our architecture, performance tests as well as several interactive applications.*

MOTS-CLÉS : *Agent conversationnel animé, architecture, SAIBA, FML, BML*

KEYWORDS: *Embodied conversational agent, architecture, SAIBA, FML, BML*

1re soumission à *Technique et science informatiques*, le 28 octobre 2009

1. Introduction

Les récents progrès technologiques rendent possible la création d'humains virtuels expressifs et interactifs. Les agents conversationnels animés (ACAs) sont des caractères animés virtuels qui sont capables d'avoir une conversation naturelle avec un utilisateur. Dans ce but, des architectures d'agents ont été développées pour simuler des comportements de communication verbale et non verbale. Les ACAs peuvent dialoguer avec les utilisateurs en utilisant des voix synthétiques, des gestes, le regard et des animations faciales.

Construire un ACA nécessite l'implication de plusieurs domaines de recherche. Les problèmes comme la reconnaissance de la parole, la capture de mouvement, la gestion du dialogue ou le rendu de l'animation requièrent différentes compétences pour leurs concepteurs. Cette complexité rend difficile la création d'un ACA par un seul groupe de recherche. Rapidement, il est devenu évident qu'il est nécessaire de partager les expertises et les modules d'un ACA entre les chercheurs. Il est donc important de développer des standards qui permettront une inter-opérationnalité entre les différents modules d'un ACA (Rickel *et al.,* 2002). La standardisation assure aussi bien l'interchangeabilité des modules que l'extension de l'ACA (Kopp *et al.,* 2006), et finalement donne l'opportunité de comparer différentes technologies d'agent (Marriott *et al.,* 2002). Les architectures, qui sont construites sur une structure commune et utilisent des protocoles standardisés de communication, peuvent être facilement étendues, permettant de développer une large variété d'applications (Thiébaux *et al.,* 2008). On peut aussi facilement imaginer que les modules (par exemple le moteur d'animation) peuvent être remplacés par d'autres. Finalement, en utilisant les mêmes entrées standardisées, l'évaluation et la comparaison de différents agents peuvent être possibles.

Cet article présente l'architecture de notre agent conversationnel animé Greta. Elle suit la méthodologie de conception proposée par (Thórisson *et al.,* 2005) et est compatible avec le standard SAIBA (Situation Agent Intention Behavior Animation) (Vilhjálmsson *et al.,* 2007). Elle est générique et modulaire. Chaque module échange des informations et des données (grâce à des langages de balisage) à travers un système central de messages. On utilise le concept de tableau blanc (whiteboard) (Thórisson *et al.,* 2005) qui permet aux modules internes et externes d'être facilement intégrés. L'architecture est suffisamment générique pour être utilisée dans plusieurs applications interactives quasi temps réel.

La prochaine section est dédiée à une vue d'ensemble des différentes architectures existantes pour les agents conversationnels animés. La section 3 présente l'architecture standardisée SAIBA qui nous a servie de modèle. Ensuite, dans la section 4, nous décrivons les différents modules de l'architecture de notre agent Greta, compatible SAIBA, ainsi que les langages de balisage utilisés. Finalement, nous détaillerons les résultats de tests de performance dans la section 5 et des exemples d'application interactive avec notre agent Greta dans la section 6.

2. L'état de l'art

L'agent Rea était un des premiers ACAs. Il a été développé par (Cassell *et al.,* 1999) et était conçu pour travailler comme un agent immobilier virtuel. Ce personnage entièrement modélisé et animé en 3D est capable de comprendre les comportements multimodaux de l'utilisateur et d'y répondre avec un discours et des intonations appropriés, accompagnés par différents types de comportements non-verbaux. Rea est un système temps réel qui nécessite la reconnaissance de la parole et des mouvements de l'utilisateur, un gestionnaire de dialogue et la planification des comportements. Ensuite, le but de l'équipe de Cassel a été de créer un outil qui pouvait être utilisé dans différentes architectures et applications. La boîte à outil BEAT (the Behavior Expression Animation Toolkit) (Cassell, 2001) est un outil d'animation modulaire et extensible fonctionnant en temps réel qui sélectionne et planifie les comportements non-verbaux d'un personnage virtuel. Il extrait les informations linguistiques et contextuelles du texte en entrée, puis choisit les gestes appropriés, le regard, et les autres comportements non verbaux. BEAT permet d'avoir une synchronisation entre tous les comportements en utilisant un ensemble de règles provenant de la recherche en psychologie sur les comportements non verbaux. BEAT produit en sortie un ensemble d'instructions dans un format propriétaire qui peut alors être interprété par un moteur d'animation ou peut être édité par un infographiste 3D.

Max (Kopp *et al.,* 2003) est un autre exemple d'agent en 3D multimodal et interactif. Il permet une communication multimodale bidirectionnelle et peut être intégré à différents périphériques d'entrée. Il a été utilisé, entre autres, pour aider les utilisateurs dans une tâche de construction (Kopp *et al.,* 2003). Max est capable de communiquer avec des utilisateurs en utilisant un discours prosodique, des gestes variés, le regard et des expressions faciales. L'utilisateur communique avec Max en utilisant le langage naturel et des gestes (par l'intermédiaire des gants de données). Max est capable d'avoir une conversation réactive et délibérée avec un utilisateur. Il utilise aussi les expressions faciales pour montrer ses émotions et est capable de produire des comportements de réponse (feedback) comme le suivi du regard de l'utilisateur. Finalement Max planifie et exécute en synchronie tous les comportements verbaux et non verbaux.

Les architectures présentées ci-dessus sont efficaces et robustes mais elles utilisent encore des protocoles de communication et des architectures propriétaires. Huang et *al.* proposent GECA (Generic Embodied Conversational Agent Framework) (Huang *et al.,* 2006), une architecture générique pour construire des ACAs. Elle est en temps réel, distribuée et indépendante du langage de programmation. Elle peut être utilisée pour créer des ACAs capables de capturer et interpréter différentes données en entrée ainsi que produire des comportements verbaux et non verbaux en sortie. Cette architecture est composée de trois couches : un système de communication, un protocole de communication basé sur un langage de balises (XML) et une interface de programmation pour la création de modules.

L' architecture GECA utilise un tableau noir (blackboard) qui intègre les différents modules de l'ACA comme celui de reconnaissance de la parole ou de la capture du mouvement. Elle est capable d'échanger à la fois les flux de données des capteurs et les messages de commandes entre tous les modules. Huang et *al.* (Huang *et al.,* 2006) proposent aussi un protocole de communication de haut niveau entre les modules basé sur un langage de balises (XML). Les messages sont ordonnés dans une structure hiérarchique (par exemple, *entrée.parole.texte*, ou *sortie.corps.geste*) et chaque type de message a un ensemble spécifique d'éléments et d'attributs, par exemple *intensité* ou *durée*. Chaque module peut souscrire à un certain type de message et peut ainsi y accéder facilement quand les messages sont envoyés par les autres modules. L'implémentation de l'architecture est faite en java, utilise le protocole de communication OpenAir et produit des animations au standard MPEG-4. OpenAir[1] est un protocole de communication et de routage basé sur un système de publication/souscription à des tableaux blancs. Il permet à des modules logiciels indépendants de communiquer entre eux pour produire des comportements globaux coordonnés. L'architecture a servi à construire l'agent *guide accompagnateur multiculturel* (Cerekovic *et al.,* 2007). Onze modules ont été utilisés, parmi lesquels, celui de la reconnaissance de la parole et de la capture de mouvement en entrée et le module d'animation de personnages 3D en sortie.

3. Architecture standardisée SAIBA

SAIBA (Situation Agent Intention Behavior Animation)[2] (Vilhjálmsson *et al.,* 2007) est une initiative de recherche internationale dont le principal but est de définir une architecture standard pour la génération de comportements interactifs des agents virtuels animés. Elle définit un certain nombre de niveaux d'abstraction (voir figure 1) de la planification des intentions communicatives de l'agent à la planification des comportements et à leur génération.

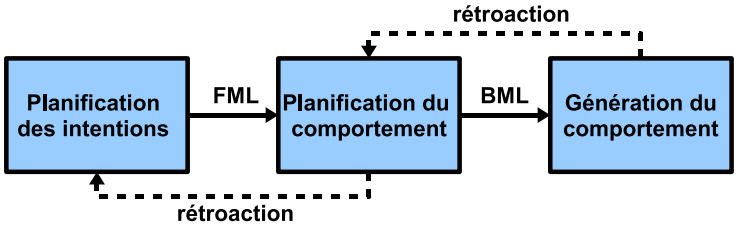

Figure 1. *Architecture standardisée SAIBA*

Le module de la **Planification des intentions** définit les buts courants, l'état émotionnel et les croyances de l'agent. Il les encode en FML[3] (Function Markup Lan-

1. http ://www.mindmakers.org/
2. http ://wiki.mindmakers.org/projects :SAIBA :main
3. http ://wiki.mindmakers.org/projects :FML :main

guage) qui est un langage de balisage pour spécifier des intentions communicatives (Heylen *et al.,* 2008). Pour transmettre les intentions communicatives de l'agent, le module de **Planification du comportement** détermine à partir du FML un certain nombre de signaux communicatifs (par exemple parole, expressions faciales, gestes) qui sont encodés en BML[4] (Behavior Markup Language). Ce langage de balisage standard permet de spécifier les comportements non verbaux de l'ACA (Vilhjálmsson *et al.,* 2007). Finalement la fonction du troisième module de l'architecture SAIBA, la **Génération du comportement**, est de réaliser les comportements générés par le module de planification du comportement. Il reçoit des données au format BML en entrée et en calcule les fichiers d'animation de l'agent virtuel en FAP-BAP.

Il existe plusieurs implémentations compatibles SAIBA comme SmartBody (Thiébaux *et al.,* 2008) et BMLRealizer (Árnason *et al.,* 2008). SmartBody [5] est une architecture modulaire, distribuée et libre de droit pour animer des ACAs en temps réel. SmartBody correspond au module de Génération du comportement de l'architecture de SAIBA. Il prend en entrée du code BML (incluant des données minutées de la parole et la mise à jour de l'état du monde) ; il génère plusieurs comportements et produit des fichiers d'animation du personnage, synchronisée avec le son. Le contenu verbal est généré à partir de texte écrit par un système externe TTS (Text-To-Speech). Le BML de SmartBody correspond seulement à une partie du standard BML (Kopp *et al.,* 2006) ; mais il offre aussi des extensions. SmartBody a été utilisé avec un générateur de comportements non verbaux (Lee *et al.,* 2006) qui correspond au planificateur du comportement de l'architecture SAIBA. C'est un module à base de règles qui génère des annotations BML pour les comportements non verbaux de l'agent à partir de ses intentions communicatives et du texte du discours de l'utilisateur. De plus, SmartBody a été utilisé avec différents personnages animés et moteurs de rendu.

BMLRealizer[6] (Árnason *et al.,* 2008) crée dans le laboratoire CADIA est une autre implémentation du module de génération du comportement de l'architecture SAIBA. C'est une boîte à outil d'animation (libre de droit) pour visualiser des personnages virtuels dans un environnement en 3D. Elle est partiellement basée sur l'architecture SmartBody. Elle utilise aussi du code BML en entrée ; elle génère des fichiers d'animation en sortie et les envoie au moteur de rendu du logiciel Panda3D.

4. Implémentation de l'architecture Greta

Notre système générique d'agent conversationnel animé Greta respecte l'architecture standardisée SAIBA. Il propose en effet une solution complète pour le module de planification du comportement et celui de la génération du comportement ainsi qu'une implémentation partielle du module de planification des intentions pour l'agent interlocuteur. Nous nous sommes concentrés sur le module de planification

4. http ://wiki.mindmakers.org/projects :BML :main
5. http ://smartbody.wiki.sourceforge.net/
6. http ://cadia.ru.is/projects/bmlr/

du comportement plutôt que sur celui de la génération du comportement. Notre module de planification du comportement produit une large palette de comportements non verbaux à partir d'une seule intention communicative. Nous avons aussi réalisé un module avancé pour la planification des intentions pour un agent interlocuteur qui inclue des rétroactions (backchannels) réactives et cognitives ainsi que des imitations de l'utilisateur sans interrompre celui-ci.

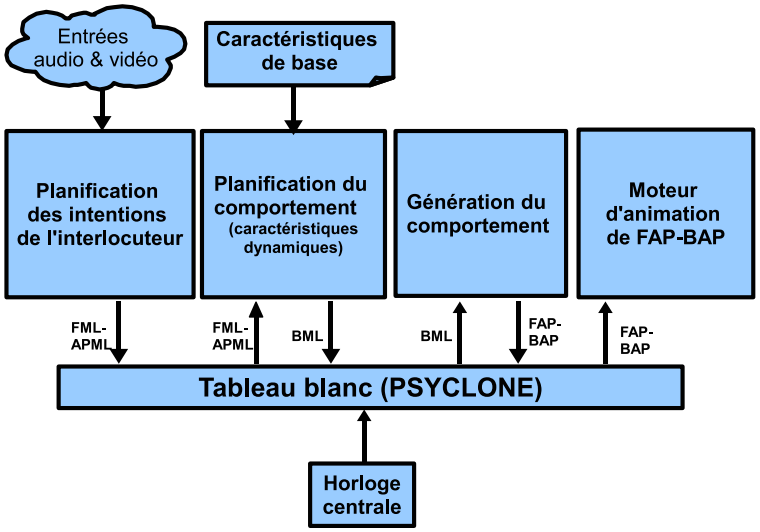

Figure 2. *Architecture de Greta avec ses différents modules selon le standard SAIBA*

La figure 2 illustre l'architecture de l'agent Greta. Comme dit auparavant, elle respecte le standard SAIBA et est composée de trois principaux modules. Dans SAIBA, la planification des intentions est réservée à l'agent locuteur. Pour être capables de contrôler un agent interlocuteur, nous avons introduit le module de **Planification des intentions de l'interlocuteur** (voir section 4.1). Ce module génère automatiquement les intentions communicatives de l'agent interlocuteur. Dans l'état courant de notre architecture, les intentions de l'agent locuteur sont définies manuellement dans un fichier d'entrée FML-APML (voir section 4.6.1). Dans le futur, ils seront générés par le module de planification des intentions de l'agent locuteur. Le module de **Planification du comportement** (voir section 4.2) reçoit en entrée les intentions communicatives de l'agent, qui peut être locuteur ou interlocuteur, écrites en FML-APML. Le module génère en sortie une liste de signaux en langage BML (voir section 4.6.2. Ces signaux sont envoyés au module de **Génération du comportement** (voir section 4.3) qui produit des fichiers MPEG-4 de FAPs et BAPs (voir section 4.4). Finalement l'animation est jouée dans le **Moteur d'animation 3D de FAP-BAP**. Tous les modules sont synchronisés par l'**Horloge centrale** (voir section 4.5) du système et communiquent entre eux à travers un **tableau blanc** appelé Psyclone[7] (Thórisson *et al.,* 2005).

7. http ://www.mindmakers.org/projects/Psyclone/

4.1. *Planification et sélection des intentions de l'interlocuteur.*

Le module de planification des intentions de l'interlocuteur s'occupe de calculer les comportements d'un agent interlocuteur quand il discute avec un utilisateur (ou avec un autre agent). Il englobe trois modules appelés rétroactions réactives, cognitives et d'imitation.

4.1.1. *Planification*

Les recherches dans le domaine ont montré qu'il y a une forte corrélation entre les signaux de rétroactions et les comportements verbaux et non verbaux exécutés par le locuteur (Maatman *et al.,* 2005, Ward *et al.,* 2000). Les rétroactions (backchannels) décrivent les intentions communicatives de l'agent interlocuteur vis-à-vis du discours du locuteur (s'il y croit ou non, aime ou non, accepte ou refuse ce qui a été dit) (Allwood *et al.,* 1993). Des modèles ont été élaborés qui prédisent quand un signal de rétroaction peut être déclenché, basés sur une analyse statistique des comportements du locuteur (utilisateur) (Maatman *et al.,* 2005, Morency *et al.,* 2008, Ward *et al.,* 2000). A partir de la littérature (Maatman *et al.,* 2005, Ward *et al.,* 2000), nous avons établi des règles probabilistes pour provoquer un signal de rétroaction quand notre système reconnait certains comportements du locuteur ; par exemple un mouvement de la tête ou une variation dans le ton de la voix de l'utilisateur peuvent déclencher une rétroaction de l'agent interlocuteur avec une certaine probabilité.

Le *module de rétroactions réactives* s'occupe de ce modèle prédictif. Le *module de rétroactions cognitives* calcule quand et quelle rétroaction doit être exhibée utilisant des informations à propos des croyances de l'agent envers le discours du locuteur. Nous utilisons la taxonomie d'Allwood pour les intentions communicatives des rétroactions (Allwood *et al.,* 1993) : interaction, perception, compréhension, réactions comportementales. A partir d'études précédentes (Bevacqua *et al.,* 2007), nous avons élaboré un lexique de rétroactions. Celui-ci sélectionne quels signaux sont à effectuer à partir du lexique en fonction des réactions de l'agent, envers le discours du locuteur (utilisateur). Le troisième module est le *module de rétroactions d'imitation*. Lorsque des personnes sont complètement engagées dans une interaction, des comportements d'imitation entre elles peuvent arriver (Lakin *et al.,* 2003). Ce module détermine quand et quels signaux l'agent devra imiter. Jusqu'à présent, nous avons considéré seulement les mouvements de la tête du locuteur pour les signaux à imiter.

4.1.2. *Sélection*

Après avoir détecté les possibles rétroactions que l'agent peut faire en réponse au discours de l'utilisateur, un algorithme de sélection détermine quelles rétroactions doivent être effectuées parmi tous les signaux potentiels produits par les modules de rétroactions (voir figure 3). En effet, notre agent ne peut pas, par exemple, imiter un hochement de la tête de l'utilisateur et faire, en même temps, "non" de la tête pour montrer son désaccord. L'algorithme de sélection reçoit en entrée :

– toutes les propositions de rétroactions potentielles avec leurs priorités et leurs intentions communicatives en FML ;

– le niveau d'intérêt de l'utilisateur estimé par l'agent. Ce niveau est calculé en évaluant le regard, la tête et la direction du torse de l'utilisateur à l'intérieur d'une fenêtre temporelle (Peters *et al.,* 2005, Peters *et al.,* 2008) ;

– les intentions communicatives de l'agent qu'il veut montrer à l'utilisateur.

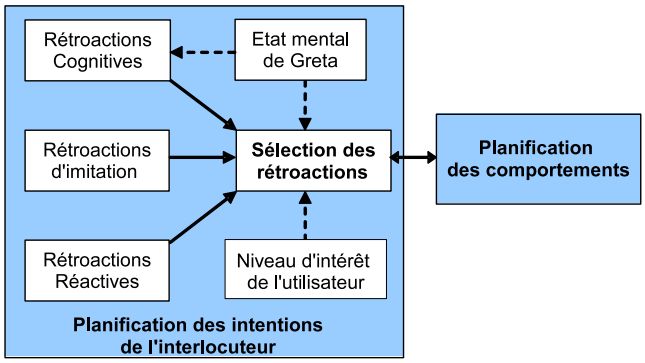

Figure 3. *Algorithme de sélection des rétroactions de notre architecture*

Inspiré du concept de hiérarchie à libre flux utilisé dans (de Sevin *et al.,* 2005), aucun choix n'est effectué au niveau des modules qui génèrent les rétroactions. Le but de l'algorithme de sélection est tout d'abord de recevoir toutes les rétroactions potentielles et d'ajuster leurs priorités en fonction du contexte de l'interaction qui évolue en permanence. Dans notre cas, le contexte est le niveau d'intérêt de l'utilisateur (estimé par l'agent) ainsi que les intentions communicatives et l'état interne de l'agent. Cela permet d'adapter le choix des rétroactions aux comportements de l'utilisateur et à ce que l'agent veut communiquer. Ce contexte peut être beaucoup plus complexe en intégrant par exemple les traits de personnalités ou les motivations de l'agent. Nous avons décidé, dans un premier temps, de réduire sa complexité pour permettre une meilleure évaluation du fonctionnement de notre algorithme de sélection.

L'ajustement des priorités des rétroactions est calculé selon :

– la probabilité d'afficher les rétroactions de l'ACA ; Certaines rétroactions peuvent être ignorées malgré le fait que les modules de détection les ont générées et envoyées à l'algorithme de sélection. Cette probabilité varie en fonction de l'évolution dynamique du contexte de l'interaction.

– le type de rétroactions ; Les imitations seront choisies préférentiellement lorsque l'utilisateur est très intéressé. L'agent montre ainsi qu'il est très engagé dans l'interaction. Les rétroactions réactives seront plus utilisées lorsque l'agent détecte que l'utilisateur perd de l'intérêt pour l'interaction. Enfin les rétroactions cognitives ont, dans tous les cas, une forte priorité car elles prennent en compte l'état mental de l'agent et sont donc très importantes pour l'interaction.

– la phase de l'interaction (début, maintien, fin) (Peters *et al.,* 2005) ; Le choix de la rétroaction sera différent si l'utilisateur commence ou finit l'interaction avec l'agent. Dans le cas où il n'est plus intéressé, notre agent va tenter d'attirer son attention pendant un petit moment en générant des rétroactions puis va considérer que l'interaction est finie.

– les intentions communicatives ; L'algorithme de sélection prend en compte ce que l'agent veut communiquer à l'utilisateur.

– l'évolution moyenne du niveau d'intérêt de l'utilisateur ; Celle-ci correspond à la moyenne du niveau d'intérêt sur un certain intervalle de temps permettant de ne pas tenir compte des variations non significatives. Cela permet d'anticiper sa valeur, utile pour la sélection des rétroactions.

Ensuite, en fonction de cet ajustement des priorités des rétroactions, l'algorithme de sélection doit choisir celles qui sont les plus appropriées en temps réel à un moment donné. En effet les rétroactions qui ont les priorités les plus hautes, sont les plus adaptées au contexte de l'interaction, aux comportements de l'utilisateur et à ce que l'agent veut communiquer. Si des rétroactions sont conflictuelles, celle qui a la priorité la plus haute est choisie prioritairement. Toutes les rétroactions sélectionnées sont envoyées en FML au module de planification du comportement pour être traduites en BML.

4.2. *Planification du comportement.*

Le module de planification du comportement de l'agent prend en entrée, à la fois, les intentions communicatives spécifiées par le langage FML et les *caractéristiques de base* de l'agent (baseline). La tache principale de ce module est de sélectionner une séquence appropriée de comportements à effectuer pour chacune des intentions communicatives. La sortie de la planification du comportement est générée en langage BML. Il contient la séquence de comportements avec leurs marqueurs de temps qui sera jouée par notre agent virtuel Greta.

Les *caractéristiques de base* de l'agent contiennent les informations sur ses préférences à utiliser certaines modalités (tête, regard, visage, geste et mouvement du torse) et sur la qualité de l'expressivité de chacune de ces modalités. L'expressivité est définie par un ensemble de paramètres qui affectent, de façon globale, la qualité des comportements de l'agent : par exemple, faire des gestes larges vs. étriqués ou des mouvements amples vs. saccadés. En fonction des caractéristiques de base de l'agent, le système calcule ses *caractéristiques dynamiques* (dynamicline) associées à chaque intention (par exemple son état émotionnel). Elles affectent localement la qualité des comportements de l'agent. Les caractéristiques dynamiques et l'intention communicative courante de l'agent sont utilisées pour sélectionner le comportement multimodal qui exprime le mieux l'intention déterminée : par exemple, un agent qui est rarement excité (caractéristiques de base), dans un état de joie (intention communicative) pourra produire juste un léger sourire alors qu'un agent très expressif (caractéristiques dyna-

miques) avec la même intention communicative pourra produire une combinaison de signaux comme sourire et étendre ses bras en même temps. Le processus de calcul est expliqué plus en détail dans (Mancini *et al.,* 2008).

4.3. *Génération du comportement.*

Ce module génère l'animation de notre agent en suivant le format MPEG-4 (Ostermann, 2002). Il reçoit en entrée du langage BML qui contient le texte à dire et/ou un ensemble de signaux non verbaux à effectuer. Les expressions faciales, le regard, les gestes, les mouvements du torse sont décrits symboliquement dans des fichiers de données. Chaque étiquette BML (voir section 4.6.2) est instanciée comme un ensemble d'image-clés finement interpolées. Le module de génération du comportement synchronise les comportements à travers les différentes modalités. Il résout aussi d'éventuels conflits entre les signaux qui utilisent les mêmes modalités. La parole est générée par un TTS OpenMary[8] (Schröder *et al.,* 2001) et les mouvements des lèvres sont ajoutés à l'animation. Quand le module de génération du comportement ne reçoit pas de données en entrée, l'agent ne reste pas immobile. Il génère des mouvements d'attente. Périodiquement un morceau de l'animation est calculé et envoyé au moteur d'animation 3D.

4.4. *Moteur d'animation 3D de FAP-BAP*

Le moteur d'animation de FAP-BAP reçoit une animation générée par le module de génération du comportement et la joue dans une fenêtre graphique en 3D. Le moteur d'animation est compatible MPEG-4. L'animation est définie par les paramètres de l'animation faciale (FAPs) et les paramètres de l'animation du corps (BAPs) (Ostermann, 2002). Les FAPs définissent la déformation et les mouvements d'un ensemble de soixante huit points fondamentaux sur un visage synthétique par rapport à leurs positions neutres ; les BAPs représentent les rotations des parties du corps autour d'articulations définies. Les configurations faciales et corporelles sont décrites par, respectivement, les images FAP et BAP. Chaque image FAP ou BAP reçue par le moteur d'animation 3D contient aussi le marqueur de temps pour sa visualisation, calculé par le module de génération du comportement. Ce temps est évalué à partir de l'horloge centrale du système (voir la section 4.5).

4.5. *Synchronisation*

De la même façon que (Huang *et al.,* 2006, Thórisson *et al.,* 2005), notre système échange différents types de messages. Chaque module peut envoyer à un tableau blanc des messages de un ou plusieurs types. Par exemple, le module de génération

8. http ://mary.dfki.de/

du comportement envoie des messages du type *Agent.Data.FAP* pour l'animation faciale, *Agent.Data.BAP* pour l'animation du corps et *Agent.Data.Parole* pour la parole. Chaque module peut s'enregistrer au tableau blanc pour recevoir différents types de messages. Par exemple le module de génération du comportement reçoit deux types de messages : *Agent.Data.BML* contenant les fichiers BML et *Agent.Data.Horloge* utilisé pour la synchronisation de tous les modules entre eux. Celle-ci est assurée par l'horloge centrale du système qui diffuse régulièrement des marqueurs de temps à travers le tableau blanc appelé Psyclone (voir figure 2). Tous les modules sont enregistrés au tableau blanc pour recevoir les marqueurs de temps pour leur synchronisation.

4.6. *Langages de balisage FML et BML*

Dans cette section nous décrivons deux langages de balises BML et FML (basés sur le XML) que nous avons implémentés dans notre système. Ces langages sont définis dans l'architecture standardisée SAIBA[9]par un groupe de chercheurs internationaux. Jusqu'ici ils ont surtout travaillé sur la standardisation du langage BML alors que la spécification du FML en est encore à ses débuts.

4.6.1. *FML : Function Markup Language*

Le langage FML[10] encode les intentions communicatives et émotionnelles que l'agent a l'intention de communiquer. Il est généré par le module de planification des intentions de l'interlocuteur et doit être traduit en BML dans le module de planification du comportement (voir figure 2). Notre version du FML, le FML-APML **??**, est un langage de balisage (basé sur le XML) pour représenter les intentions communicatives, les communications non-verbales et le texte dit par l'agent. Les intentions communicatives de l'agent correspondent à son état émotionnel, ses croyances et ses buts. Le FML-APML est une extension du langage APML (Affective Presentation Markup Language)(Carolis *et al.,* 2004) qui utilise la théorie des actes communicatifs d'Isabella Poggi (Poggi, 2003). Le FML-APML utilise une syntaxe similaire à celle du BML (voir section 4.6.2) et permet de faciliter la génération du dialogue des ACAs. Il a une structure linéaire qui permet de définir la durée de chaque intention communicative de façon explicite permettant sa synchronisation. Chaque étiquette du FML-APML représente une intention communicative ; plusieurs intentions communicatives peuvent être imbriquer en même temps.

Dans notre FML-APML, nous considérons les étiquettes du APML (avec des extensions) suivantes :

– *certitude* : utilisée pour spécifier le degré de certitude que l'agent veut exprimer.

– *performativité* : représente les actions performatives de l'agent, par exemple suggérer, approuver ou être en désaccord.

---

9. http ://wiki.mindmakers.org/projects :SAIBA :main
10. http ://wiki.mindmakers.org/projects :fml :main ?s=fml

– *thème/rhème* : représente le sujet/propos de la conversation ; c'est respectivement la partie du discours qui est déjà connue ou nouvelle dans la conversation des participants.

– *métadiscours* : représente le but de l'état de la relation entre les différentes parties du discours.

– *tour de parole* : modèle les échanges de la prise de parole des locuteurs.

– *émotion* : décrit l'état émotionnel de l'agent. Nous pouvons définir des émotions simples en utilisant des étiquettes émotionnelles (par exemple colère ou tristesse). Dans le FML-APML, nous pouvons définir des états émotionnels complexes comme le masquage (c'est-à-dire l'agent a une certaine émotion mais la cache en montrant une fausse émotion) ou la superposition de deux émotions (Niewiadomski *et al.,* 2007).

– *emphase* : utilisée pour mettre l'accent sur un message verbal ou non verbal de l'agent.

– *rétroaction (backchannel)* : décrit les intentions communicatives de l'interlocuteur. Cela correspond à sa volonté et ses capacités à continuer, percevoir, comprendre l'interaction ainsi que son attitude vis-à-vis du discours du locuteur (s'il y croit ou non, aime ou non, accepte ou refuse ce qui a été dit) (Allwood *et al.,* 1993).

– *monde* : se rapporte aux objets du monde (location, description).

Nous pouvons remarquer que ce langage nous permet de décrire les intentions communicatives de l'agent quand il est soit le locuteur ou l'interlocuteur.

Nous avons ajouté d'autres étiquettes pour obtenir un FML-APML plus complet pour communiquer les intentions communicatives de l'agent et pour une meilleure compatibilité avec le BML :

– *nom* : le nom de l'étiquette, représentant l'intention communicative modélisée par l'étiquette.

– *identifiant* (id) : un unique identifiant est associé à l'étiquette ; il permet d'y faire référence sans ambigüité.

– *type* : cet attribut spécifie la signification communicative de l'étiquette. Par exemple, une étiquette performative a plusieurs valeurs possibles pour le type d'attribut par exemple suggérer, proposer, approuver, etc... En fonction du nom de l'étiquette (performativité) et du type (une des valeurs au dessus), notre module de planification du comportement détermine les comportements non verbaux que l'agent va exécuter.

– *début*(start) : représente l'instant où l'intention spécifiée dans l'étiquette commence à être communiquée, en secondes. Peut être absolu (le temps 0 correspond au début du message FML-APML) ou relatif à une autre étiquette.

– *fin* (end) : durée de l'étiquette. Peut être une valeur numérique (en secondes) relative au début de l'étiquette ou au commencement ou à la fin d'une autre étiquette. Elle représente la durée de l'intention communicative modélisée par l'étiquette.

– *importance* : une valeur entre 0 et 1. Elle représente le coefficient de pondération que l'intention communicative, encodée par l'étiquette, soit communiquée à travers

des comportements non verbaux.

– *intensité* : les émotions peuvent être exprimées avec différentes intensités (Ekman, 1975). L'intensité d'un état émotionnel est décrite par une valeur comprise dans un intervalle entre [0,1].

Voici un exemple d'un FML-APML utilisé dans les tests avec trois différentes intentions communicatives : prévenir, suggérer ou proposer (voir section 2).

```
<?xml version="1.0"?>
<!DOCTYPE fml-apml SYSTEM "fml-apml.dtd" []>
<fml-apml>
  <fml>
    <performative id="p1" type="warn" start="1" end="1.5" importance="1"/>
    <performative id="p2" type="suggest" start="2.5" end="2.5" importance="1"/>
    <performative id="p3" type="propose" start="5" end="2.5" importance="1"/>
  </fml>
</fml-apml>
```

4.6.2. *BML : Behavior Markup Language*

Le langage BML[11] n'est pas encore un standard, cependant les chercheurs se sont mis d'accord sur une spécification "commune" de la syntaxe du BML permettant d'échanger des fichiers BML entre différents modules des architectures, comme décrit dans (Kopp *et al.,* 2006, Vilhjálmsson *et al.,* 2007). Le BML est généré par le module de planification du comportement (à partir du FML) et doit être traduit en FAP-BAP dans le module de génération du comportement pour pouvoir être jouer par le moteur d'animation 3D (voir figure 2). Le langage BML (basé sur le XML) nous permet de spécifier des signaux non verbaux qui peuvent être exprimés à travers des modalités de communication de l'agent. Chaque étiquette BML haut niveau correspond à un comportement que l'agent doit produire sur une modalité donnée : tête, torse, visage, regard, corps, jambes, geste, parole, lèvres. Dans le standard BML, un signal peut être choisi pour chaque modalité à partir d'une petite liste fixe. Chaque signal a un temps de départ et une durée définie. Ces informations temporelles peuvent être absolues (en secondes) ou relatives (par rapport aux autres signaux verbaux ou non verbaux).

Nous avons implémenté des extensions pour le standard BML pour notre agent. Celles-ci nous permettent de définir des étiquettes pour utiliser un ensemble plus large de signaux produits par l'agent et pour spécifier l'expressivité de chaque signal (Hartmann *et al.,* 2005).

Etiquette du signal. Dans la syntaxe du BML, il est possible de spécifier seulement un petit ensemble de signaux pour l'agent. Par exemple, nous pouvons définir juste quatre formes de la bouche : aplatie, sourire, rire et grimace. C'est une limitation puisque des agents, capables d'accomplir des actions complexes, peuvent ne pas être entièrement exploités. Nous avons donc introduit un paramètre appelé *référence* pour spécifier le nom du signal facial que le module de génération du comportement doit produire. Ainsi, dans notre version du BML, nous avons deux types d'informations à

11. http ://wiki.mindmakers.org/projects :BML :main

propos du signal : l'attribut *type*, qui est obligatoire et réfère à un petit ensemble de signaux définis dans le standard BML, et l'attribut *référence*, qui est utilisé par notre agent pour exécuter des comportements non verbaux à partir d'un ensemble plus large de signaux. Dans notre architecture, le module de génération du comportement préfère toujours le signal spécifié par l'attribut *référence*, si il est présent. Mais nous pouvons aussi envoyer le code BML calculé par notre système à un autre module de génération du comportement. En effet il contient l'attribut obligatoire *type* et peut donc être interprété par d'autres modules de génération du comportement.

Paramètres d'expressivité. Notre agent peut dynamiquement moduler les signaux multimodaux en utilisant un petit ensemble de paramètres de haut niveau, appelés *paramètres d'expressivité* (Hartmann *et al.,* 2005). Ils influencent la qualité du mouvement : par exemple, le geste de lever une main peut être accompli rapidement ou doucement, avec plus ou moins d'énergie, atteignant un point dans l'espace plus ou moins proche, et ainsi de suite. Les paramètres d'expressivité ne sont pas inclus dans la dernière syntaxe du BML mais peuvent être interprétés dans notre module de génération du comportement. Ainsi nous pouvons spécifier non seulement *quels* signaux l'agent doit exécuter mais aussi *comment* il doit le faire. Par exemple, battre le rythme avec la main peut être exécuté de différentes façons : rapidement ou doucement, souplement ou brusquement, etc...

Voici un exemple d'une partie d'un BML utilisé dans les tests qui permet à l'ACA d'exprimer une expression faciale de surprise et des gestes de joie (voir section 1).

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE bml SYSTEM "bml.dtd" []>
<bml>
  <face id="emotion-1" start="0.1" end="2.5" stroke="1">
    <description level="1" type="gretabml">
    <reference>affect=surprise</reference>
    <intensity>12</intensity>
      <FLD.value>-0.60</FLD.value>
      <OAC.value>0.90</OAC.value>
      <PWR.value>0.40</PWR.value>
      <REP.value>0.85</REP.value>
      <SPC.value>0.55</SPC.value>
      <TMP.value>0.45</TMP.value>
      <preference.value>0.50</preference.value>
    </description>
  </face>
  <gesture id="emotion-0" start="0.5" end="0.9" stroke="1">
    <description level="1" type="gretabml">
    <reference>emotion=joy_signal_10</reference>
    <intensity>1</intensity>
      <FLD.value>-0.50</FLD.value>
      <OAC.value>0.75</OAC.value>
      <PWR.value>0.60</PWR.value>
      <REP.value>0.70</REP.value>
      <SPC.value>0.35</SPC.value>
      <TMP.value>0.55</TMP.value>
      <preference.value>0.50</preference.value>
    </description>
  </gesture>
</bml>
```

5. Evaluation de l'architecture Greta

Nous avons effectué des tests de performance de notre architecture Greta. Nous avons mesuré deux types de performances définis par rapport à l'architecture SAIBA : du planificateur du comportement au moteur d'animation 3D FAP-BAP et du planificateur des intentions au moteur d'animation 3D FAP-BAP. Nous n'avons pas testé toute l'architecture (c'est-à-dire des comportements de l'utilisateur à la génération de l'animation) car cela dépend de l'efficacité de modules externes de capture et d'analyse du comportement. De la même façon, dans ces tests de performance, nous n'avons pas utilisé de système externe TTS (Text To Speech) pour générer la parole. Tous les tests ont été effectués sur un seul ordinateur avec tous les modules SAIBA en état de fonctionnement et connectés entre eux. Nous avons utilisé un ordinateur avec un processeur Intel Core2 Extreme X7900 à 2.8 GHz et Windows Vista 32 bits. D'après le logiciel de tests de performance Sisoftware Sandra 2009[12], le processeur a des performances globales de 21 Gops (milliard d'opérations par seconde).

Pour la première série de mesures, nous avons préparé des séquences de fichiers BML contenant différents signaux à effectuer par notre agent. Les fichiers sont envoyés au module de génération du comportement où les étiquette BML sont traduites en fichiers d'animation FAP et BAP. Nous avons mesuré le temps de génération pour les animations de durées différentes : 7.5, 15, 30 et 60 secondes. Pour chacune des conditions de temps, nous avons utilisé six fichiers BML : trois fichiers BML composé de, respectivement, trois (F1), sept (F2) ou quinze (F3) différentes expressions faciales et trois fichiers BML composé de, respectivement, trois (G1), sept (G2) ou quinze (G3) différents gestes. Ces fichiers BML permettaient à Greta d'exprimer la surprise (voir un exemple dans la section 4.6.2), la joie, la colère... De plus chacun des six fichiers, F1-3 et G1-3, ont été testés quatre fois, donc au total nous avons fait quatre-vingt seize tests.

Durée des animations	7.5 sec	15 sec	30 sec	60 sec
F1 (3 expressions faciales)	0.247	0.319	0.495	0.829
F2 (7 expressions faciales)	0.266	0.352	0.522	0.863
F3 (15 expressions faciales)	0.299	0.374	0.620	0.980
G1 (3 gestes)	0.291	0.439	0.631	0.997
G2 (7 gestes)	0.324	0.462	0.732	1.136
G3 (15 gestes)	0.399	0.573	0.798	1.354

Tableau 1. *temps de génération de l'animation (en secs) pour un fichier BML.*

Le Tableau 1 présente les valeurs moyennes de temps de génération de l'animation pour un fichier BML. Le temps de génération varie de 0.247 secondes dans le cas F1 (trois expressions faciales avec un durée totale de 7.5 secondes), à 1.354 secondes dans le cas G3 (quinze gestes avec un durée totale de 30 secondes). Nous en avons

12. http ://www.sisoftware.net/index.html

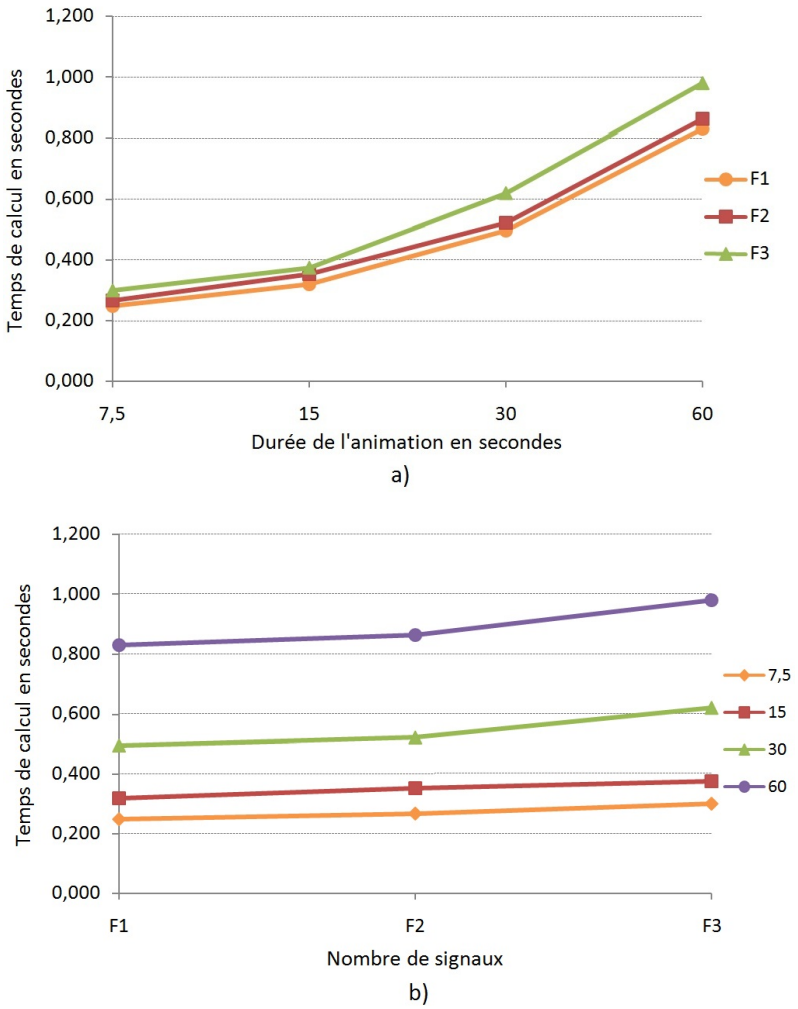

Figure 4. *Tests de performance de l'architecture pour l'animation du visage : a) la relation entre la durée des animations et le temps de génération, b) la relation entre le nombre de signaux et le temps de génération.*

déduit que le temps moyen de génération d'une image diminue quand la durée totale de l'animation augmente. Ce temps diminue aussi avec l'augmentation de la quantité de signaux utilisés dans Fi-Gj. A partir des tests que nous avons effectués (voir les figures 4 et 5), nous pouvons conclure que le temps de calcul du BML jusqu'aux FAP-BAP dépend linéairement, à la fois, de la durée et du contenu de l'animation (c'est-à-dire modalités expressives et nombre de signaux utilisés).

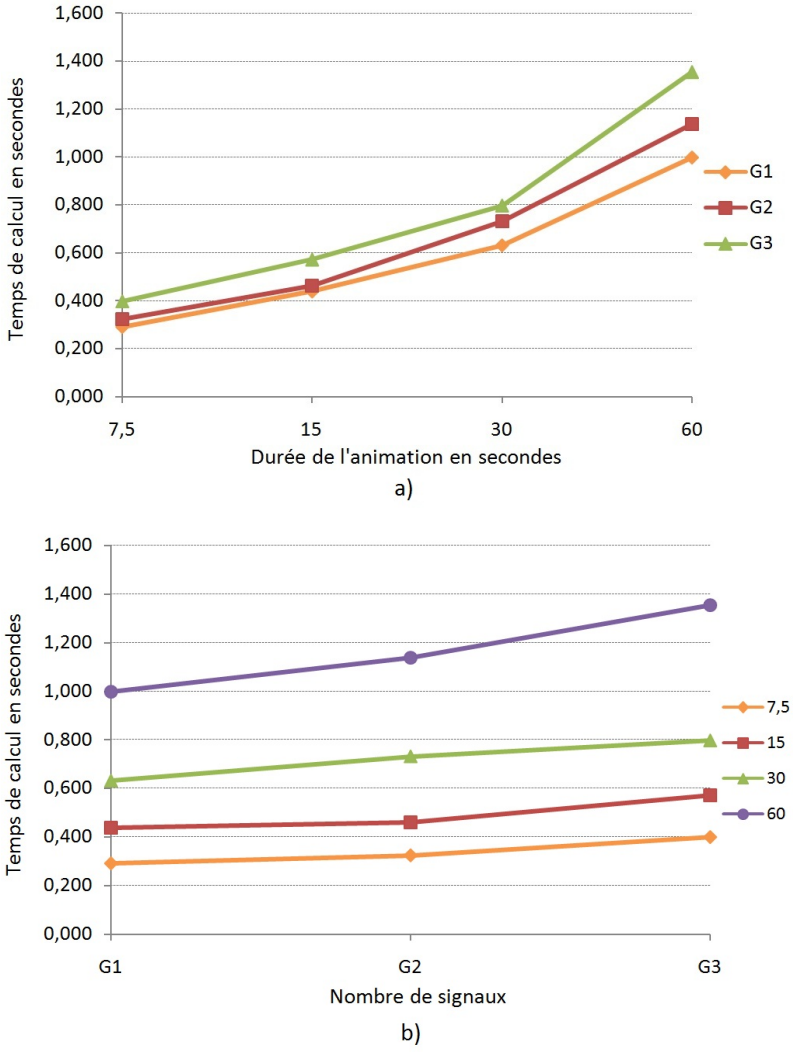

Figure 5. *Tests de performance de l'architecture pour l'animation des gestes : a) la relation entre la durée des animations et le temps de génération, b) la relation entre le nombre de signaux et le temps de génération.*

Dans une seconde phase, nous avons testé la performance, à la fois, des modules de la planification du comportement et de la génération du comportement. Nous avons préparé des séquences de fichiers FML-APML (voir un exemple 4.6.1) contenant différentes intentions communicatives à effectuer par notre agent comme le fait de préve-

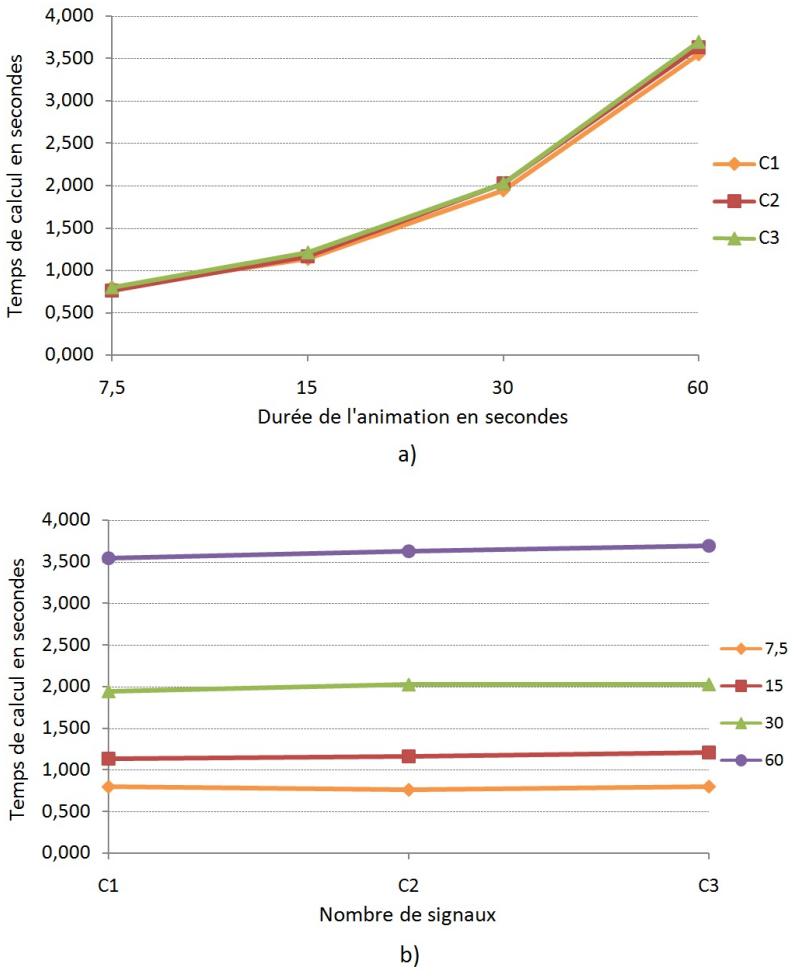

Figure 6. *Tests de performance de l'architecture pour l'animation à partir des intentions communicatives : a) la relation entre la durée des animations et le temps de génération, b) la relation entre le nombre de signaux et le temps de génération.*

nir, suggérer ou proposer (voir un exemple dans la section 4.6.1). Comme pour le premier test, nous avons considéré quatre conditions différentes qui conduisent à quatre animations de différentes durées : 7.5, 15, 30, et 60 secondes. Dans chaque condition, nous avons utilisé une séquence de trois fichiers FML composés par un (C1), deux (C2) ou trois (C3) différentes intentions communicatives. Chaque fichier C1-3 a été testé quatre fois, donc au total nous avons fait quarante huit tests.

Durée des animations	7.5 sec	15 sec	30 sec	60 sec
C1 (1 intention)	0.795	1.132	1.942	3.546
C2 (2 intentions)	0.759	1.161	2.025	3.634
C3 (3 intentions)	0.798	1.212	2.029	3.695

Tableau 2. *Temps de génération de l'animation (en secs) pour un fichier FML-APML.*

Le tableau 2 présente les valeurs moyennes du temps de génération de l'animation. Celui-ci varie 0.795 secondes dans le cas C1, à 3.695 secondes dans le cas C3. Chaque intention communicative est instanciée comme un ou plusieurs signaux par le module de planification du comportement. Par conséquent, le nombre de signaux utilisés dans l'animation finale est à priori inconnu. Comme précédemment, nous en avons déduit que le temps moyen de génération pour une seule image diminue lorsque la durée de l'animation augmente. A partir des tests que nous avons effectués (voir la figure 6), nous pouvons conclure que le temps de calcul du FML-APML aux FAP-BAPs est linéairement dépendant de la durée de l'animation.

Tous les tests de performance convergent pour dire que la relation entre le temps de génération et la durée totale de l'animation est linéaire. Nous pouvons en conclure que les tests de performance montrent que notre système peut être utilisé dans des applications interactives quasi temps réel (impliquant un petit délai). Lors de l'interaction avec un utilisateur, les FML et BML gérés par le système correspondent à des animations de courte durée (inférieures ou égales à 7.5 secondes). Ils sont envoyés continuellement en fonction des comportements de l'utilisateur. Le délai résultant de l'architecture n'est donc pas suffisamment important pour empêcher l'utilisateur d'interagir normalement avec Greta. Dans les prochaines sections nous allons illustrer cela en décrivant plusieurs applications interactives qui ont été implémentées avec notre architecture d'agent.

6. Applications interactives de Greta

Notre agent conversationnel animé Greta a été conçu pour être capable de gérer une interaction naturelle avec des utilisateurs ou d'autres agents. Notre première étape dans cette direction a été de faire un agent qui perçoit le monde extérieur. Les modules d'analyse détectent les caractéristiques de la voix et les comportements non verbaux de l'utilisateur. Les informations envoyées par ces modules grâce au tableau blanc sont alors utilisées par notre architecture pour planifier la réponse de l'agent.

6.1. *Agent interlocuteur interactif*

Notre agent conversationnel animé a été utilisé pour construire un agent interlocuteur interactif (Bevacqua *et al.,* 2008). Son rôle est de montrer à l'utilisateur qu'il

l'écoute sans l'interrompre. Les rétroactions non verbales générées par l'agent interlocuteur sont basées sur ses intentions communicatives et sont mémorisées dans son état mental. Celui-ci décrit comment l'agent interlocuteur est en train de réagir au discours de l'utilisateur (voir section 4.1).

6.1.1. *Agent SAL.*

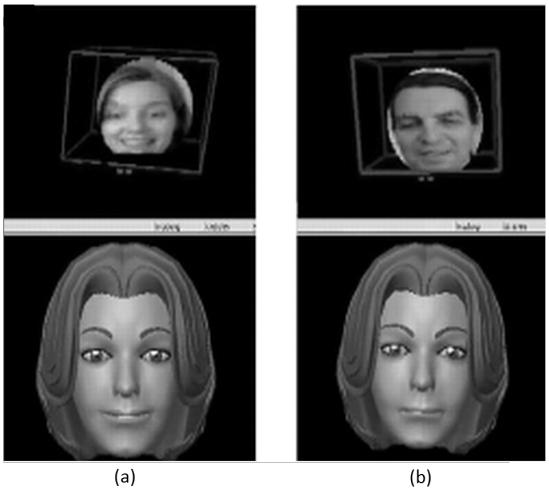

Figure 7. *Interaction entre des utilisateurs et deux agents SAL : a) Poppy et b) Spike.*

Dans le projet européen SEMAINE [13], nous développons un agent interlocuteur artificiel et sensistif (Sensitive Artificial Listener, SAL) (Douglas-Cowie *et al.,* 2008). Notre but est de doter un agent interlocuteur de capacité pour maintenir une interaction avec un utilisateur en utilisant certaines émotions spécifiques et en montrant des rétroactions appropriées. Dans ce projet, nous allons considérer quatre agents SAL définis chacun par des caractéristiques de base spécifiques (Bevacqua *et al.,* 2008). En fonction de ses caractéristiques de base et de son état mental, l'agent SAL va montrer des rétroactions très différentes. Les utilisateurs devaient raconter une histoire à l'agent qui devait montrer sa participation à travers la production de rétroactions. Dans la figure 7a, l'agent est Poppy, un agent extraverti et optimiste. L'agent montre des rétroactions positives en utilisant le sourire et le hochement de tête. A coté, la figure 7b illustre Spike en interaction avec un utilisateur. Spike est mécontent et provocateur. Il utilise principalement le froncement de sourcils pour signaler des rétroactions négatives comme le désaccord. Dans cette application, nous avons interfacé notre système avec Watson (Morency *et al.,* 2005) qui traque les mouvements de tête de l'utilisateur en temps réel.

Figure 8. *Deux interlocuteurs pour l'utilisateur : notre agent et un robot Aibo.*

6.1.2. *Agents interlocuteurs virtuels et robotiques interactifs.*

Dans l'atelier d'été eNTERFACE 2008 (Moubayed *et al.,* 2008), un atelier européen dédié à la recherche sur les interfaces multimodales [14], le module de planification des intentions de l'interlocuteur a été connecté avec Pure Data (Puckette, 1988), un environnement de programmation graphique pour le traitement du son en temps réel, et avec un module de détection du visage qui détecte les sourires, les hochements verticaux et horizontaux de tête de l'utilisateur (Baklouti *et al.,* 2008) (voir figure 8). Les signaux de rétroactions générés par le système sont joués à la fois par notre agent Greta en 3D et par un robot Aibo de Sony. Comme l'Aibo est un robot chien, un lexique de rétroactions a été élaboré spécifiquement pour lui. Par exemple, quand le signal de rétroaction demandé est un sourire, le robot chien remue sa queue et des lumières clignotent sur sa tête et son dos. Le contrôle des comportements de l'agent et du robot est fait par les mêmes fichiers BML. Le module de génération du comportement donne en sortie des FAPs pour l'agent et des commandes Aibo pour le robot (Moubayed *et al.,* 2008). Nous avons commencé un projet national GV-Lex[15] sur ce thème mais avec un robot Nao[16]

6.2. *Conteur Interactif Virtuel*

Une autre application de notre agent Greta est d'avoir implémenté un Conteur Interactif Virtuel (The Interactive Storyteller). Elle a été développée dans le cadre du projet européen CALLAS[17]. Dans ce scenario, une application informatique présente le contenu d'une histoire et affiche une séquence d'images fixes avec un fort impact

13. http ://www.semaine-project.eu/
14. http ://enterface08.limsi.fr/project/7
15. http ://www.gvlex.com/index.html
16. http ://www2.aldebaran-robotics.com/
17. http ://www.callas-newmedia.eu/

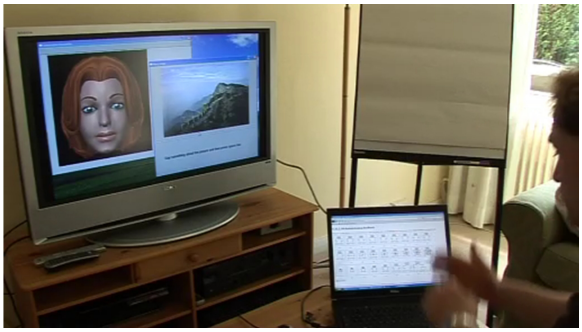

Figure 9. *Un utilisateur commente une image à impact émotionnel élevé. L'agent s'adapte en temps réel aux émotions de l'utilisateur*

émotionnel (voir figure 9). Le rôle de l'agent Greta est d'interagir avec l'utilisateur en lui demandant de commenter les images observées. La parole et les gestes de l'utilisateur sont analysés pour détecter son état émotionnel. Ensuite l'agent explique les images et adapte ses comportements à l'état émotionnel de l'utilisateur. Le but est d'adapter le comportement de l'agent aux émotions de l'utilisateur.

7. Conclusion

Notre architecture générique et modulaire d'agent conversationnel animé interactif Greta est capable de communiquer avec des utilisateurs de façon verbale et non-verbale. Notre agent est une implémentation complète de l'architecture standard SAIBA pour les modules de planification et de génération du comportement, ainsi qu'une implémentation partielle du module de planification des intentions (seulement pour l'agent interlocuteur). Notre système utilise à la fois des extensions des langages FML et BML en voie de standardisation. D'après les résultats de nos tests de performance que nous avons obtenus, notre architecture est appropriée pour les applications interactives quasi temps réel. En effet lors de l'interaction avec l'utilisateur, les BML et FML gérés par l'architecture sont de courte durée. Le délai résultant n'est donc pas suffisamment important pour empêcher l'utilisateur d'interagir normalement avec Greta. De plus notre architecture peut être facilement connectée avec des modules internes et externes. En effet chaque module peut échanger des informations et des données à travers un système central de messages très flexible. Ceci est bien illustré par les différents exemples d'intégration de notre architecture dans des applications interactives comme les agents interlocuteurs artificiels ou le conteur interactif.

Dans le futur, nous prévoyons de construire un module de planification des intentions pour le locuteur. Pour cela, nous allons devoir intégrer un module de planification du dialogue. Nous allons aussi continuer à améliorer le module de planification des intentions de l'interlocuteur, en particulier l'algorithme de sélection des rétroactions.

Remerciements

Ce travail a été partiellement financé pat le projet STREP SEMAINE IST-211486 [18], le projet IP-CALLAS IST-034800 [19], le projet ANR-MyBlog-3D [20], et l'atelier d'été sur les interfaces multimodales eNTERFACE 2008 [21].

8. Bibliographie

Allwood J., Nivre J., Ahlsén E., « On the Semantics and Pragmatics of Linguistic Feedback », *Semantics*, 1993.

Árnason B., Porsteinsson A., « The CADIA BML Realizer », 2008. http ://cadia.ru.is/projects/bmlr/.

Baklouti M., Couvet S., Monacelli E., « Intelligent Camera Interface (ICI) : A Challenging HMI for Disabled People », *First International Conference on Advances in Computer-Human Interaction*, p. 21-25, 2008.

Bevacqua E., Heylen D., Tellier M., Pelachaud C., « Facial Feedback Signals for ECAs », *AISB'07 Annual convention, workshop "Mindful Environments"*, Newcastle, UK, p. 147-153, April, 2007.

Bevacqua E., Mancini M., Pelachaud C., « A Listening Agent Exhibiting Variable Behaviour », *in* , H. Prendinger, , J. C. Lester, , M. Ishizuka (eds), *Proceedings of 8th International Conference on Intelligent Virtual Agents*, vol. 5208 of *LNCS*, Springer, Tokyo, Japan, p. 262-269, 2008.

Carolis B. D., Pelachaud C., Poggi I., Steedman M., « APML, a Mark-up Language for Believable Behavior Generation », *in* , H. Prendinger, , M. Ishizuka (eds), *Lifelike Characters. Tools, Affective Functions and Applications*, 2004.

Cassell J., « BEAT : The behavior expression animation toolkit », *SIGGRAPH '01 : Proceedings of the 28th annual conference on Computer Graphics and Interactive Techniques*, ACM Press, p. 477-486, 2001.

Cassell J., Bickmore T., « Embodiment in Conversational Interfaces : Rea », *Conference on Human Factors in Computing Systems*, Pittsburgh, PA, 1999.

Cerekovic A., Huang H.-H., Zoric G., Tarasenko K., Levacic V., Pandzic I. S., Nakano Y. I., Nishida T., « Towards an Embodied Conversational Agent Talking in Croatian », *The 9th Intenational Conference on Telecommunications (ConTEL 2007)*, Zagreb, Croatia, 2007.

de Sevin E., Thalmann D., « A motivational Model of Action Selection for Virtual Humans », *Computer Graphics International (CGI)*, IEEE Computer SocietyPress, New York, p. 213-220, 2005.

Douglas-Cowie E., Cowie R., Cox C., Amir N., Heylen D., « The Sensitive Artificial Listener : an induction technique for generating emotionally coloured conversation », *LREC2008 - Workshop on Corpora for Research on Emotion and Affect*, Marocco, May, 2008.

18. http ://www.semaine-project.eu
19. http ://www.callas-newmedia.eu
20. https ://picoforge.int-evry.fr/cgi-bin/twiki/view/Myblog3d/Web
21. http ://enterface08.limsi.fr/project/7

Ekman P., *Unmasking the Face. A guide to recognizing emotions from facial clues*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1975.

Hartmann B., Mancini M., Pelachaud C., « Implementing Expressive Gesture Synthesis for Embodied Conversational Agents », *The 6th International Workshop on Gesture in Human-Computer Interaction and Simulation*, VALORIA, University of Bretagne Sud, France, 2005.

Heylen D., Kopp S., Marsella S., Pelachaud C., Vilhjalmsson H., « Why Conversational Agents do what they do ? Functional Representations for Generating Conversational Agent Behavior. The First Functional Markup Language Workshop », 2008. The Seventh International Conference on Autonomous Agents and Multiagent Systems Estoril, Portugal.

Huang H.-H., Masuda T., Cerekovic A., Tarasenko K., Pandzic I. S., Nakano Y. I., Nishida T., « Toward a Universal Platform for Integrating Embodied Conversational Agent Components. », *in* , B. Gabrys, , R. J. Howlett, , L. C. Jain (eds), *Proceedings of 10th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems*, vol. 4252 of *Lecture Notes in Computer Science*, Springer, p. 220-226, 2006.

Kopp S., Jung B., Leßmann N., Wachsmuth I., « Max - A Multimodal Assistant in Virtual Reality Construction », *KI*, vol. 17, n° 4, p. 11-18, 2003.

Kopp S., Krenn B., Marsella S., Marshall A. N., Pelachaud C., Pirker H., Thórisson K. R., Vilhjalmsson H., Badler N., Johnson L., « Behavior Markup Language », http ://www.mindmakers.org/projects/BML, 2006.

Lakin J. L., Jefferis V. A., Cheng C. M., Chartrand T. L., « Chameleon Effect as Social Glue : Evidence for the Evolutionary Significance of Nonconsious Mimicry », *Nonverbal Behavior*, vol. 27, n° 3, p. 145-162, 2003.

Lee J., Marsella S., « Nonverbal Behavior Generator for Embodied Conversational Agents », *in* , J. Gratch, , M. Young, , R. Aylett, , D. Ballin, , P. Olivier (eds), *Proceedings of 6th International Conference on Intelligent Virtual Agents*, vol. 4133 of *LNCS*, Springer, Marina Del Rey, CA, USA, p. 243-255, 2006.

Maatman R., Gratch J., Marsella S., « Natural Behavior of a Listening Agent », *in* , T. Panayiotopoulos, , J. Gratch, , R. Aylett, , D. Ballin, , P. Olivier, , T. Rist (eds), *Proceedings of 5th International Working Conference on Intelligent Virtual Agents*, vol. 3661 of *LNCS*, Springer, Kos, Greece, p. 25-36, 2005.

Mancini M., Pelachaud C., « Distinctiveness in multimodal behaviors », *in* , L. Padgham, , D. C. Parkes, , J. Müller, , S. Parsons (eds), *Proceedings of 7th Conference on Autonomous Agents and Multi-Agent Systems*, 2008.

Marriott A., Pelachaud C., Rist T., Ruttkay Z., Vilhjalmsson H., « Embodied conversational agents - let's specify and evaluate them ! », 2002. http ://www.vhml.org/workshops/AAMAS.

Morency L., de Kok I., Gratch J., « Predicting Listener Backchannels : A Probabilistic Multimodal Approach », *in* , H. Prendinger, , J. C. Lester, , M. Ishizuka (eds), *Proceedings of 8th International Conference on Intelligent Virtual Agents*, vol. 5208 of *LNCS*, Springer, Tokyo, Japan, 2008.

Morency L., Sidner C., Lee C., Darrell T., « Contextual recognition of head gestures », *Proceedings of the 7th International Conference on Multimodal Interfaces*, ACM New York, NY, USA, p. 18-24, 2005.

Moubayed S. A., Baklouti M., Chetouani M., Dutoit T., Mahdhaoui A., Martin J.-C., Ondas S., Pelachaud C., Urbain J., Yilmaz M., « Generating Robot and Agent Backchannels During a Storytelling Experiment », *ICRA*, Pasadena, California, 2008.

Niewiadomski R., Pelachaud C., « Model of Facial Expressions Management for an Embodied Conversational Agent », *Proceedings of the 2nd Conference on Affective Computing and Intelligent Interaction*, vol. 4738 of *Lecture Notes in Computer Science*, Springer, p. 12-23, 2007.

Ostermann J., « Face Animation in MPEG-4 », *in* , I. Pandzic, , R. Forchheimer (eds), *MPEG-4 Facial Animation - The Standard Implementation and Applications*, Wiley, England, p. 17-55, 2002.

Peters C., Asteriadis S., Karpouzis K., de Sevin E., « Towards a Real-time Gaze-based Shared Attention for a Virtual Agent », *Workshop on Affective Interaction in Natural Environments (AFFINE)*, Crete, Greece, 2008.

Peters C., Pelachaud C., Bevacqua E., Mancini M., Poggi I., « A model of attention and interest using Gaze behavior », *in* , T. Panayiotopoulos, , J. Gratch, , R. Aylett, , D. Ballin, , P. Olivier, , T. Rist (eds), *Proceedings of 5th International Working Conference on Intelligent Virtual Agents*, vol. 3661 of *Lecture Notes in Computer Science*, Springer-Verlag, Kos, Greece, p. 229-240, 2005.

Poggi I., « Mind markers », *in* , N. Trigo, , M. Rector, , I. Poggi (eds), *Gestures. Meaning and use*, University Fernando Pessoa Press, Oporto, Portugal, 2003.

Puckette M., « Pure Data », 1988. http ://www.puredata.info.

Rickel J., Marsella S., Gratch J., Hill R., Traum D., Swartout B., « Towards a New Generation of Virtual Humans for Interactive Experiences », *IEEE Intelligent Systems*, vol. 17, n° 4, p. 32-38, 2002.

Schröder M., Trouvain J., « The MARY Text-to-Speech System », 2001. http ://mary.dfki.de/.

Thiébaux M., Marsella S., Marshall A., Kallmann M., « SmartBody : behavior realization for embodied conversational agents. », *in* , L. Padgham, , D. C. Parkes, , J. Müller, , S. Parsons (eds), *Proceedings of 7th Conference on Autonomous Agents and Multi-Agent Systems*, p. 151-158, 2008.

Thórisson K. R., List T., Pennock C., Dipirro J., « Whiteboards : Scheduling Blackboards for Semantic Routing of Messages & Streams », *AAAI-05 Workshop on Modular Construction of Human-Like Intelligence*, p. 8-15, 2005.

Vilhjálmsson H., Cantelmo N., Cassell J., Chafai N. E., Kipp M., Kopp S., Mancini M., Marsella S., Marshall A., Pelachaud C., Ruttkay Z., Thórisson K., van Welbergen H., van der Werf R., « The Behavior Markup Language : Recent Developments and Challenges », *in* , C. Pelachaud, , J.-C. Martin, , E. André, , G. Chollet, , K. Karpouzis, , D. Pelé (eds), *Proceedings of 7th International Conference on Intelligent Virtual Agents*, vol. 4722 of *LNCS*, Springer, Paris, France, p. 99-111, 2007.

Ward N., Tsukahara W., « Prosodic features which cue back-channel responses in English and Japanese », *Journal of Pragmatics*, vol. 23, p. 1177-1207, 2000.