Achille Falaise

Premier pas vers une TA interactive pour le tchat

Rapport de stage de seconde année de Master recherche Information, Cognition, Apprentissages, option Sciences cognitives

Sous la direction de :

Christian Boitet
Hervé Blanchon

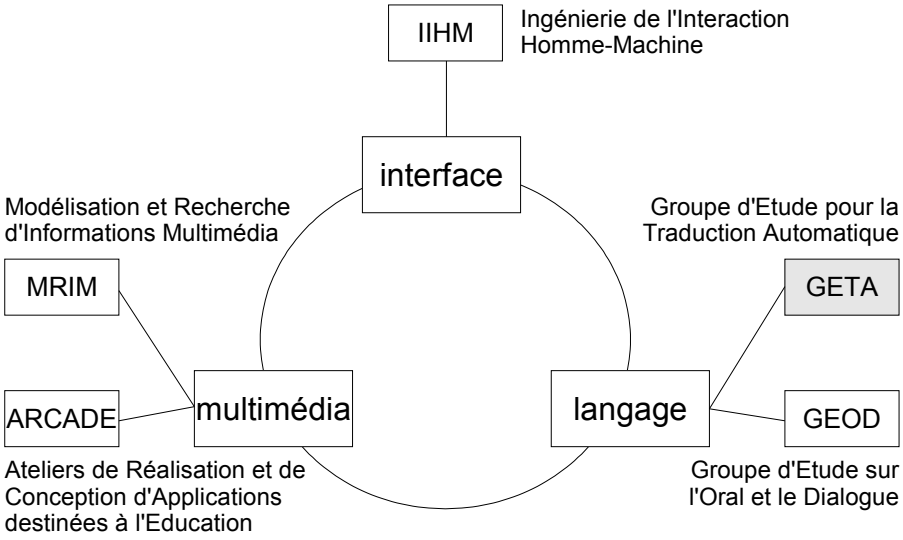
CLIPS-GETA

11 juin 2004

SOMMAIRE

CONTEXTE DU STAGE

Ce stage a été effectué au sein de l'équipe GETA[1] du laboratoire CLIPS[2].



Présentation du CLIPS

Le laboratoire CLIPS conduit des recherches dans les domaines suivants :

- interactions homme-machine,
- interactions humaines médiées par la machine,
- traitement des informations : langue, parole, images

Le GETA est une équipe pluridisciplinaire formée d'informaticiens et de linguistes. Les thèmes de recherche du GETA concernent tous les aspects théoriques, méthodologiques et pratiques de la TAO (Traduction Assistée par Ordinateur), et plus généralement de l'informatique multilingue. Le GETA est issu du CETA (1961- 1971), laboratoire pionnier de la TA en France.

1 Groupe d'Etude pour la Traduction Automatique.

2 Communication Langagière et Interaction Personne-Système.

INTRODUCTION

L'utilisation du tchat et de la messagerie instantanée connaît un fort développement depuis quelques années. Parallèlement, la traduction automatique a fait d'importants progrès. Les stratégies de traduction interactive, en particulier, ont montré qu'il était possible d'obtenir une TA de bonne qualité, pour peu que l'utilisateur accepte de coopérer avec la machine. Or, aucun système de tchat multilingue commercial ne tire aujourd'hui parti des possibilités de la traduction interactive. De plus, ces systèmes se contentent de transposer les méthodes de traduction de documents à la traduction de tchat, sans tenir compte de quelque manière que ce soit des spécificités de ce mode de communication. Dans ces conditions, il n'est pas étonnant que la TA de tchat existante peine à rassembler des usagers.

Qu'est-ce que la traduction interactive pourrait apporter à la TA de tchat ? Avant d'être en mesure de répondre à cette question, on doit être capable d'évaluer ce que la TA « classique » apporte au tchat. On sait que les traductions ainsi obtenues sont mauvaises ; mais qu'en est-il en termes d'utilisabilité ? Après tout, tous les utilisateurs n'ont pas forcément besoin d'une traduction de qualité. Une personne en train d'apprendre une langue par exemple, pourra se servir de la TA comme d'une béquille pour dialoguer avec des correspondants. Tel autre, polyglotte, pourra se reporter à une traduction, même de mauvaise qualité, lorsqu'il bute sur un mot inconnu.

Dans le cadre de ce stage, je me propose de contribuer à la résolution de ces questions, à travers la constitution d'un corpus de tchat multilingue, s'appuyant sur la TA traditionnelle. Après une rapide présentation du tchat, j'exposerai ses principales caractéristiques linguistiques du point de vue du TAL, en m'appuyant sur un corpus de tchat monolingue. Je détaillerai ensuite la façon dont je compte modéliser des sessions de tchat multilingues, ainsi que le système de tchat développé en vue de l'obtention du corpus. Enfin, je proposerai quelques pistes prolongeant cette étude, dans les domaines du tchat écrit et oral.

I. LE TCHAT, QU'EST-CE QUE C'EST ?

1.1 Le tchat

Etant donné le nombre toujours croissant de modes de communication disponibles sur internet, il peut être utile de rappeler ce qu'est exactement le tchat. Par le terme « tchat », également orthographié « chat » ou « t'chat », on entend *un dialogue écrit, en temps réel et médié par la machine*. Le tchat présente de nombreuses similarités avec le dialogue oral ; et c'est sans doute le moyen de communication écrit qui s'en rapproche le plus. Il en est nettement plus proche que d'autre outils, tels que les forums ou les e-mails, que le caractère différé assimile plus au dialogue épistolaire qu'à la conversation parlée. De plus, le tchat est *volatil*. Le contenu d'une session de tchat, à l'instar d'une conversation orale, n'a pas vocation à être mis à disposition du public ; et lorsqu'un utilisateur se connecte, il est dans l'ignorance totale de ce qui a été dit avant son arrivée, de même qu'il ne pourra pas savoir ce qui se dira après sa déconnexion. En outre, comme on le verra plus loin, les tchateurs introduisent dans l'écrit des éléments propres à l'oral.

Par contre, la messagerie instantanée présente exactement les mêmes caractéristiques que le tchat, à ceci près que le dialogue y est limité à deux utilisateurs. On peut donc considérer la messagerie instantanée comme une sous-classe du tchat.

Bien qu'il existe de nombreux logiciels et réseaux de tchat, on peut relever quelques constantes. Tout d'abord, le tchat s'appuie sur une architecture de type client/serveur ; c'est à dire que les utilisateurs ne communiquent pas directement entre eux, mais par l'intermédiaire d'un serveur unique, comme décrit dans la figure 1.

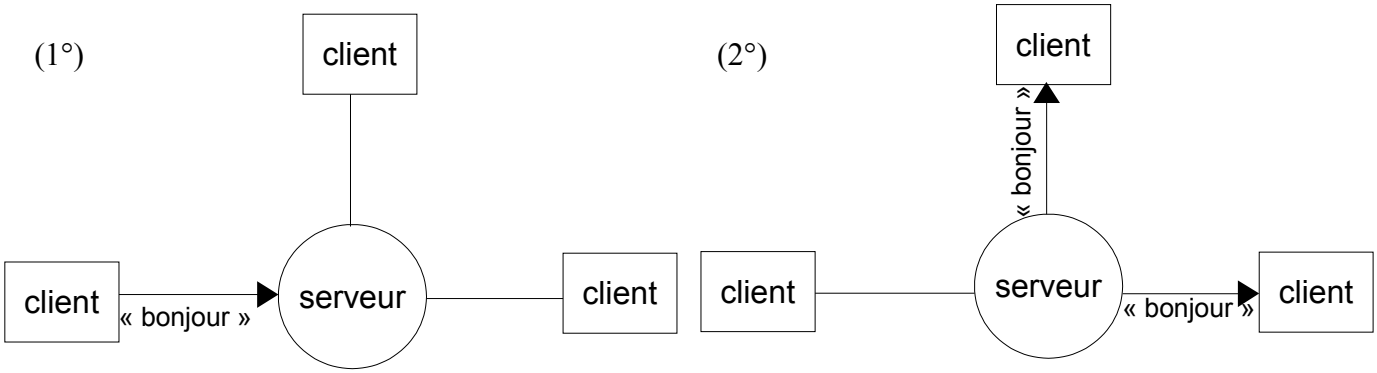


Figure 1: aperçu de l'architecture d'un système de tchat. Chaque message est d'abord envoyé au serveur, puis transmis vers les autres clients.

De plus, l'interface des clients s'organise généralement en trois zones ; l'une exposant la conversation en cours, le second permettant la saisie d'un nouveau message et son envoi, et le troisième présentant les utilisateurs connectés.

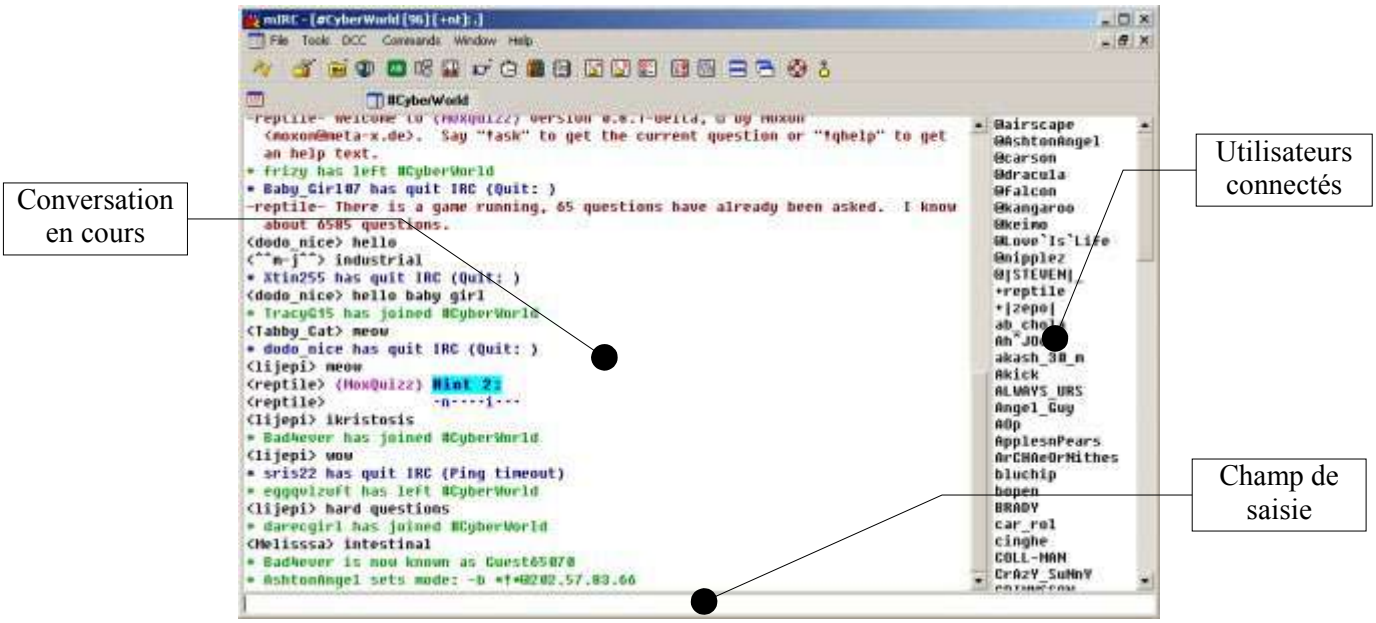


Figure 2 : mIRC, le plus connu des clients de tchat.

L'exemple ci-dessous est tiré d'une session de tchat ; les identifiants des utilisateurs (les *pseudos*) ont été anonymés. L'extrait commence au moment où l'utilisateur **A** se connecte. Comme il a été souligné plus haut, ce dernier ne sait pas ce qui a été dit avant son connexion.

```
[01] A> soir tlm
[02] A> kikooooooooo B :*)
[03] B> kikooo A :)
[04] <C vient de se connecter>
[05] A> yo C
[06] C> yop all
[07] C> yop A
[08] A> o y repond today
[09] C> kikouuu B
[10] C> yop midam B
[11] C> yop D
[12] A> j'ai de la chance loll
```

A dispose de la liste des utilisateurs connectés, et peut donc en saluer certains nominalement dès sa connexion.

Un nouvel utilisateur (**C**) vient de se connecter. Toutes les personnes connectées en sont informées.

D est connecté, mais n'intervient pas dans cet extrait.

Suite de l'énoncé [8]. Comme dans une conversation verbale, il arrive que des énoncés s'entrecroisent.

Exemple 1 : extrait d'une session de tchat sur irc.epiknet.org#francophone.

1.2 Les utilisateurs

Selon [LAT01], on peut estimer le nombre de tchateurs réguliers à au moins 1,5 million de personnes (hors messagerie instantanée). Ce nombre stagne après plusieurs années de croissance, comme l'attestent les données recueillies par [HIN] pour les principaux réseaux de tchat, alors même que le nombre d'internautes est en progression. Ces données ne rendent pas compte toutefois du développement du tchat dans le cadre professionnel. De plus en plus d'entreprises adoptent le tchat

comme outil de communication interne, et les utilisateurs professionnels n'apparaissent pas dans ces statistiques.

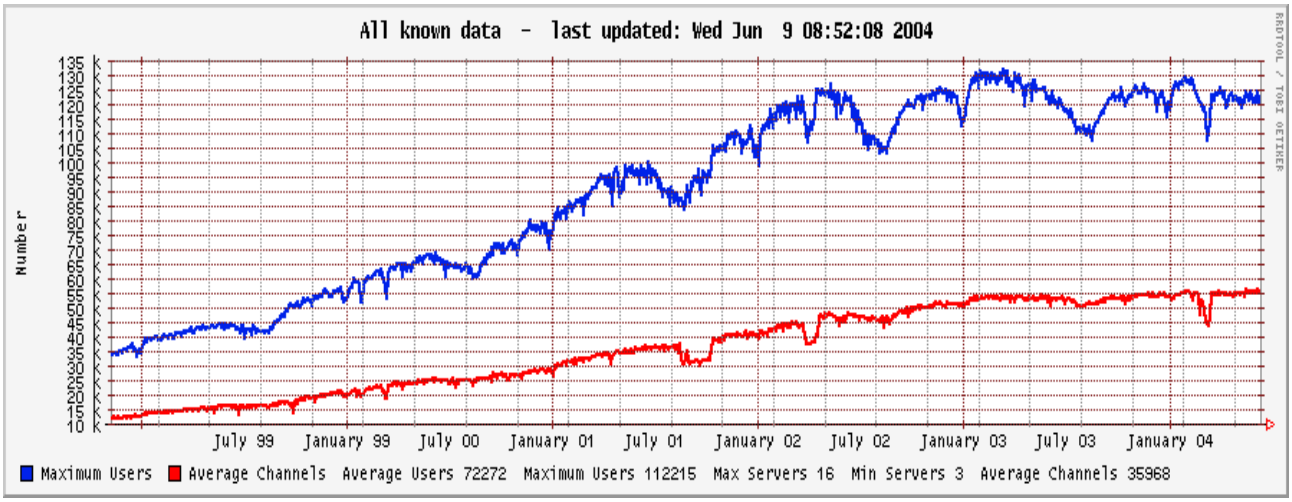


Figure 3 : Progression du nombre d'utilisateurs du réseau IRCNet entre janvier 1999 et janvier 2004. Source [HIN].

Les utilisateurs de messagerie instantanée sont nettement plus nombreux, et son utilisation est toujours en phase de croissance, au point que l'on considère souvent que cet outil a supplanté le tchat *stricto sensu*. Selon [MAG], le nombre d'utilisateurs réguliers de services de messagerie s'élevait à 150 millions en 2001, et était estimé devoir monter à 400 millions en 2004, soit une part très significative de l'ensemble des internautes.

1.3 Les intervenants

Tous les intervenants d'un session de tchat ne sont pas nécessairement humains, et c'est quelque chose que l'on ne peut ignorer lorsque l'on envisage d'étudier la langue du tchat. En effet, certains programmes, appelés robots ou « bots », apparaissent comme de véritables utilisateurs, dotés d'un nom (un « pseudo »), et capables de générer des messages. Il est normalement impossible de les distinguer des utilisateurs humains, autrement que par le caractère mécanique et mal à propos de leurs interventions. Comme exemple de robot, on peut citer certains « gadgets », qui envoient un message pré-enregistré lorsqu'un certain mot-clef est détecté dans un message. D'autres robots réagissent à des commandes, comme les robots de gestion des utilisateurs. Il faut donc bien garder à l'esprit, lorsqu'on étudie le langage du tchat, que certains messages sont générés par, ou destinés à, des machines. Ces messages, bien que faisant partie de la conversation, ne doivent pas être analysés sur le même plan que les autres.

Enfin, certains messages servent à notifier des événements aux utilisateurs. C'est le cas notamment de l'énoncé [04] de l'exemple 1, « *C vient de se connecter* », qui indique qu'un utilisateur vient de se connecter. Contrairement aux messages de robots, ceux-ci ne sont jamais associés à un pseudo (nom d'utilisateur), et sont donc aisés à distinguer.

1.4 Les canaux

Tous les tchateurs connectés sur un serveur ne communiquent pas nécessairement ensemble. En général, un serveur permet l'accès à plusieurs *canaux* (aussi dits *salles*) de discussion, qui sont totalement cloisonnés. On ne voit que ce qui se passe dans le canal auquel on est connecté, et on ne

peut envoyer de messages que vers ce dernier, comme s'il n'y avait que lui sur le serveur. Sur la plupart des serveurs publics, la création de canaux est totalement libre ; n'importe quel utilisateur peut en ouvrir.

1.5 Protocoles et réseaux de tchat

En matière de tchat, il existe un protocole incontournable : IRC (acronyme d'Internet Relay Chat). Créé en 1988, il s'agit d'un protocole libre, et il est utilisé par la plupart des réseaux de tchat. Il existe en effet plusieurs réseaux IRC distincts, chacun comptant plusieurs dizaines de serveurs répartis sur toute la planète. Les plus connus sont *IrcNet*, *EFNet*, *DalNet* et *UnderNet*, qui se partagent environ 150 000 usagers en période de pointe [LAT01]. Tous sont accessibles *via* un client IRC standard (comme *mIRC*, par exemple). Sur IRC, les noms de canaux sont précédés d'un dièse (par exemple, le canal #help est souvent présent sur les serveurs IRC).

Dans le domaine de la messagerie instantanée, les choses sont nettement plus complexes. De nombreux réseaux existent, qui reposent sur des protocoles incompatibles. Ces réseaux ne permettent normalement aucune interopérabilité et les clients sont spécifiques à chaque réseau. Les plus connus d'entre eux sont les réseaux reposant sur des protocoles propriétaires : *ICQ*, *MSN*, *Y!*, *AIM*, etc... Mais il existe aussi d'autres réseaux libres, comme *Jabber* (basé sur le protocole XMPP), qui offre par ailleurs des passerelles vers les autres réseaux, ainsi que des fonctions de tchat.

II. LA LANGUE DU TCHAT

2.1 Représentation d'un corpus de tchat

Afin de pouvoir voir ce qu'est réellement du tchat, et quels problèmes spécifiques se posent dans le cadre du traitement automatique de la langue, il peut être utile de constituer un corpus qui puisse servir de référence. J'en ai donc généré un, à partir de logs de sessions de tchat librement consultables sur internet.

2.1.1 Quelques exemples

Les exemples ci-dessous donnent une idée assez précise du type de document que l'on envisage de coder :

```
[01] A>  coucouuuuuuuuuuuuuuuuuuuuuuuuu
[02] A>  il y a klkun??????????????????????????
[03] A> ou je suis tute seulllllllllllllllll
[04] A> .??????????????????????,,,
[05] A> B tes po loin????
[06] A> lalalalala lallalalalal
[07] A> roooooooooooooooo
[08] A> meme po un bonjour
[09] A> ouinnnnnnnnnnnnn
```

Exemple 2 : monologue sur irc.epiknet.org#18-25ans.

```
[01] A> heureusement que je suis la sinon sa serait presquemort le chan
[02] B> mais il est mort même avec toi !
[03] B> mdr
[04] A> lol
[05] <C vient de se connecter>
[06] C> re
[07] A> put1 sa booste bien ce que j'ecoute
[08] B> c koi?
[09] A>ché pas j'ecoute une radio américaine</message>
```

Exemple 3 : quelques abréviations en provenance de irc.epiknet.org#18-25ans.

2.1.2 Représenter du tchat

Une session de tchat peut être vue comme un ensemble de messages, ordonnés séquentiellement et produits par différents auteurs. Comme il a été montré en 1.3, il faut distinguer plusieurs types de messages, selon qu'ils sont :

1. générés par un utilisateur humain à destination de ses pairs ;

2. générés par un utilisateur humain à destination d'un robot (ce sont des commandes) ;

3. générés par un robot (généralement à destination d'un humain, je ne connais pas de cas de robot émettant des commandes à destination d'autres robots) ;

4. générés par le serveur (pour les notifications d'événements, voir à nouveau 1.3).

Mais si la syntaxe particulière des messages de la seconde catégorie permet une identification aisée, c'est loin d'être le cas avec ceux de la troisième, qui affectent généralement la forme d'énoncés « naturels ». C'est pourquoi j'ai fait le choix de ne pas donner de représentation spécifique aux énoncés générés par des robots, et de les traiter comme des énoncés de la première catégorie.

2.1.3 Formalisme adopté

Le corpus est codé en XML. Une session correspond à l'élément racine <*log*>, qui peut avoir pour fils un ou plusieurs des éléments suivants :

- une balise <*commentaire*>, dont le contenu est un commentaire sur la session ;
- une balise <*message*>, dont le contenu est un message de type 1 ou 3 ;
- une balise <*commande*>, dont le contenu est un message de type 2 (une commande) ;
- une balise <*evenement*>, dont le contenu est un message de type 4 (un événement).

Les éléments <*message*>, <*commande*> et <*evenement*> possèdent des attributs *date* et *heure*, qui correspondent au moment où le message est parvenu au serveur. Les éléments <*message*> et <*commande*> comportent en outre un attribut *auteur*, donnant l'identifiant de l'utilisateur ayant produit le message.

Si je reprends l'exemple 1, j'obtiens le code XML suivant :

	<log langue="fr"> <commentaire>Exemple</commentaire>
[01] A> soir tlm	<message auteur="A" heure="15:13">soir tlm</message>
[02] A> kikooooooooo B :*)	<message auteur="A" heure="15:13">kikooooooooo B :*)</message>
[03] B> kikooo A :)	<message auteur="B" heure="15:25">kikooo A :)</message>
[04] <C vient de se connecter>	<evenement heure="15:25">C vient de se connecter</evenement>
[05] A> yo C	<message auteur="A" heure="15:29">yo C</message>
[06] C> yop all	<message auteur="C" heure="15:57">yop all</message>
[07] C> yop A	<message auteur="C" heure="15:57">yop A</message>
[08] A> o y repond today	<message auteur="A" heure="15:57">o y repond today</message>
[09] C> kikouuu B	<message auteur="C" heure="16:11">kikouuu B</message>
[10] C> yop midam B	<message auteur="C" heure="16:25">yop midam B</message>
[11] C> yop D	<message auteur="C" heure="16:48">yop D</message>
[12] A> j'ai de la chance loll	<message auteur="A" heure="18:00">j'ai de la chance loll</message> </log>

Exemple 4 : code XML correspondant à une session de tchat. Dans le code XML, les attributs date *ont été omis, et les identifiants des utilisateurs ont été anonymés.*

2.2 Constitution d'un corpus de tchat

2.2.1 Sources

Le corpus de tchat est constitué à partir des logs disponibles sur le site http://www.botstats.com. Ce site met à la disposition du public les logs de quelques canaux du serveur EpikNet. Je tiens à souligner que la décision de publier les logs de tchats revient entièrement au propriétaire de chaque canal (généralement son créateur), et que les usagers du canal sont (normalement) informés que leurs conversation son enregistrées. Au total, les archives de 107 canaux sont consultables sur le site. Une archive représente en moyenne 2 mois de conversations sur chaque canal.

2.2.2 Extraction et reformatage

Les logs sont tout d'abord extraits du site à l'aide d'HTTrack, un logiciel d'aspiration de sites web. Puis, les fichiers HTML obtenus sont convertis en XML par un script Perl. Le format HTML n'étant pas conforme à la norme XML, je n'ai pas pu utiliser la librairie XML de Perl, et j'ai donc eu recours à des expression régulières pour effectuer la transformation. Ce programme de reformatage est consultable en annexe ; je détaille le fonctionnement d'un programme similaire destiné au reformatage d'un corpus dans [FAR03].

2.2.3 Exploitation du corpus

Les fichiers XML ainsi obtenus sont associés à une feuille de style XSLT, afin de permettre une visualisation plus conviviale. Ils peuvent ensuite être affichés dans un visualisateur XML, comme Internet Explorer[3].

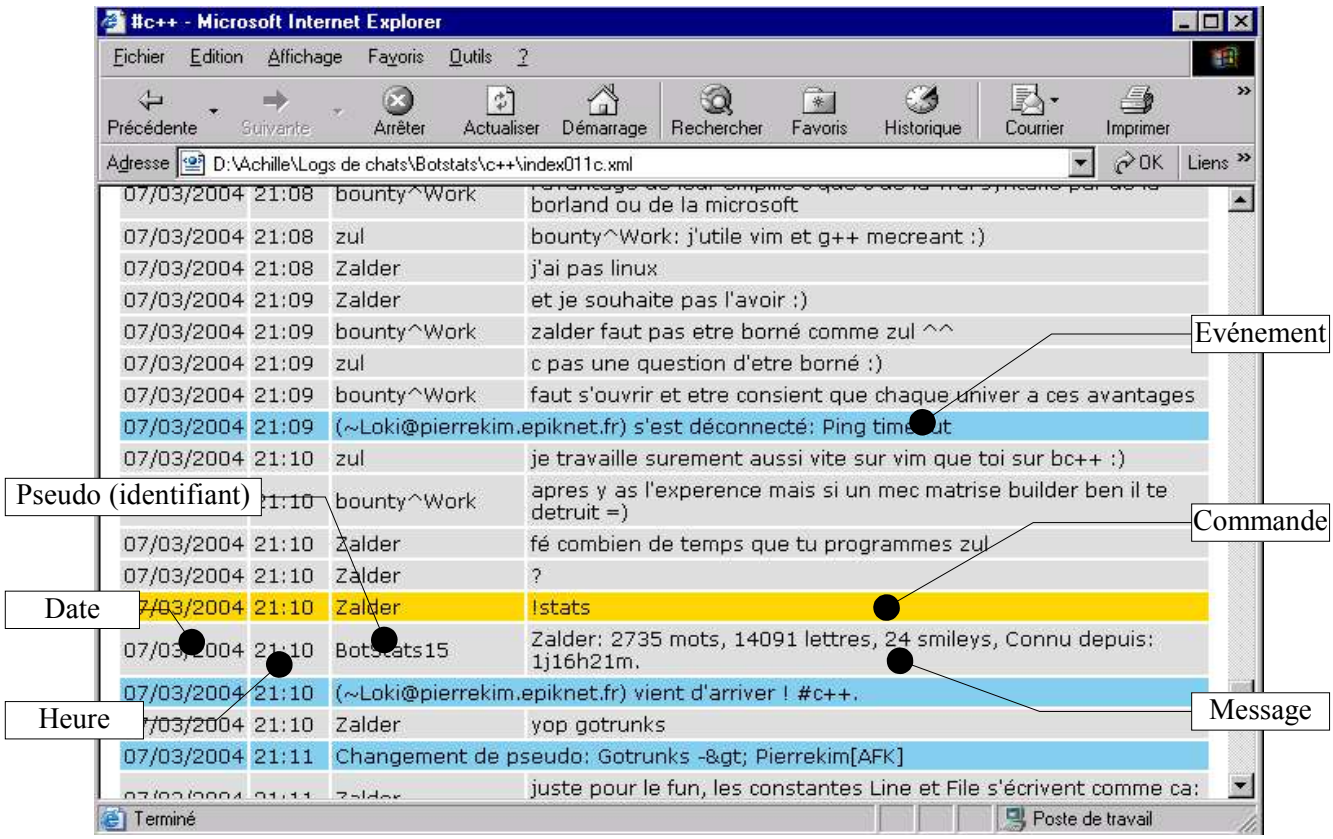


Figure 4 : visulisation d'un extrait du corpus dans MSIE, grâce à une feuille de style XSLT

2.3 Le français « tchaté »

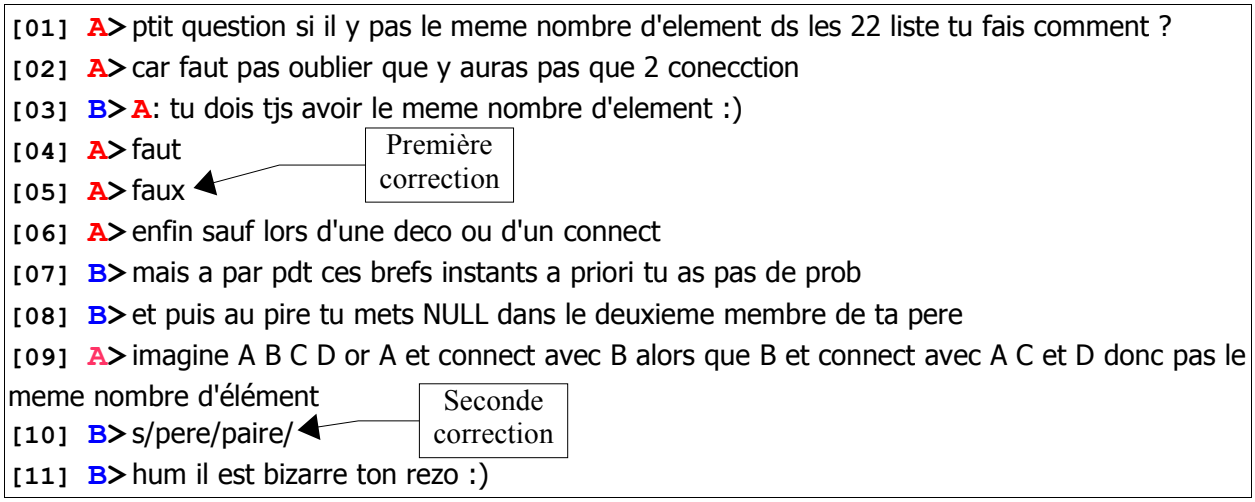
Dans cette étude, je me limiterai aux messages en langue « naturelle », c'est à dire ceux produits par les utilisateurs humains à destination de leurs pairs. J'essaierai de relever les caracéristiques de la langue « tchatée » qui peuvent poser problème dans le cadre du traitement automatique de la langue, tant au niveau lexical qu'au niveau syntaxique.

2.3.1 Niveau lexical

En premier lieu, le tchat est tributaire de sa médiation par l'écrit. Les messages d'une session de tchat sont rédigés très rapidement, généralement sans relecture ; de nombreux tchateurs s'enorgueillissent d'ailleurs de leur vitesse de frappe. La conséquence, on s'en doute, consiste en une profusion de coquilles et autres fautes de frappe. Au delà de l'aspect moteur, au niveau de la planification linguistique, l'orthographe pâtit également de cette précipitation. Même lorsque l'utilisateur la maîtrise suffisament, ce qui est loin d'être toujours le cas, la phase que l'on pourrait qualifier de « planification orthographique » se trouve malmenée. De plus, les diacritiques ont une tendance à disparaître dans la langue « tchatée ».

3 Après installation de la librairie MSXML 3.0+. Celle-ci est présente sur toutes les versions de Windows depuis Windows 98, et peut être télécharger pour les autres systèmes d'exploitation supportant MSIE.

Détail intéressant, les utilisateurs corrigent parfois leurs fautes après coup. L'extrait suivant en donne deux exemples :



Exemple 5 : extrait du canal irc.epiknet.org#c++, les italiques signalent les éléments corrigés.

Mais un grand nombre d'autres phénomènes linguistiques, qui pourraient passer pour des « fautes », se révèlent en fait parfaitement volontaires ; c'est pourquoi je les qualifierai plutôt de « divergences ». On pense bien entendu aux abréviations, fréquentes en tchat, et parfois spécifiques à ce mode de communication (comme *lol*, *mdr*, *tlm*, etc.). Mais un autre phénomène émerge, aussi fréquent que surprenant ; je le qualifierai de « divergence phonologique » : il s'agit d'une modification de la graphie d'un mot, destinée à lui donner une prononciation légèrement différente de la norme. On rencontre ainsi des expressions telles que « *kikoo* », « *oki* », ou encore « *salut les zamis* ». Parfois, ces divergences phonologiques se manifestent par des allongements, comme dans le « *kikooooooooo* » de l'exemple 1. D'autres expressions jouent un rôle d'ordre plus pragmatique, comme les célèbres émoticons (aussi appellées « smileys »). Enfin, certaines graphies ne se justifient que d'un point de vue graphique, comme par exemple le remarquable « ¯`°2º□œ=",,__ [AmAnDiNe] !_ _,,._"=œº□œ2º`¯ »[4]. D'après [PIE03], ces divergences volontaires découleraient d'une « ludogénèse », destinée à introduire, en jouant avec la graphie, des émotions et de la personnalité dans la langue écrite.

Nombre de ces divergences volontaires sont aujourd'hui devenues des classiques du tchat, bien connus des tchateurs, y compris des utilisateurs de messageries instantanées, si bien que [PIE03] parle de « créolisation » pour décrire ce processus d'émergence d'une norme : le français « tchaté » (l'ensemble des conventions lexicales considérées comme connues de tous) serait né des apports de chaque utilisateur, pérennisés ou non par l'usage. Dans ce contexte, certains canaux, fréquentés par une importe population « d'habitués », pourraient ce comporter comme de petites communautés linguistiques, avec leur lot d'idiomatismes.

2.3.2 Niveau syntaxique

Je ne reviendrai pas sur les problèmes orthographiques, si ce n'est pour dire, puisque nous considérons ici le niveau syntaxique, que les fautes d'accord se révèlent relativement plus nombreuses en tchat que les fautes d'usage. Cela est très frappant lorsque l'on considère que les fautes d'accord ont rarement l'occasion de se réaliser : en général par l'absence d'une flexion (donc

4 *in* [PUJ01].

guère plus d'une fois par mot), et ce uniquement dans un contexte fléchi. Même des énoncés par ailleurs parfaitement normés, et que l'on peut donc supposer produits par des personnes compétentes en orthographe, présentent ce type de divergence orthographique. On pourrait voir là le signe que lors de la phase de « planification orthographique », la planification lexicale intervient avant la planification syntaxique (l'accord requiert une vue d'ensemble de l'énoncé, plus coûteuse).

Comme le relève [PIE03], la syntaxe des énoncés de tchat tient plus de l'écrit que de l'oral. Entre autres phénomènes propres à la langue parlée, les topicalisations y sont fréquentes, ainsi que les constructions du type *situation* + *thème* + (*rhème*). Enfin, pour éviter les messages trop longs, les usagers découpent souvent leurs messages en propositions. On en peut en avoir un aperçu dans l'exemple 5, où les messages [07] et [08] constituent deux propositions d'un même énoncé. Cela n'est pas sans rappeler le découpage en groupes prosodiques. Ce dernier parallèle entre langue parlée et langue tchatée peut toutefois sembler assez accidentel, dans la mesure où il paraît motivé par la mécanique dans la langue parlée (il faut bien reprendre sa respiration), et plutôt par un critère ergonomique dans la langue tchatée (afin d'éviter les messages longs) ; mais le fait que le découpage, quelle que soit le type de contrainte, s'opère au niveau des frontières propositionnelles n'est pas inintéressant à observer.

III. TRADUIRE DU TCHAT

3.1 Objectifs

Comme on l'a vu précédemment, la langue tchatée présente des caractéristiques qui la rendent particulièrement complexe à traiter automatiquement. Toutefois, le contexte d'un tchat *multilingue* est différent. En effet, dans le cas d'un tchat monolingue, rien n'incite l'utilisateur à la rigueur ; la seule limite de sa fantaisie, de sa « ludogénèse », provient de la nécessité de se faire comprendre par les interlocuteurs. Dans un tchat multilingue par contre, il s'agit de se faire comprendre d'un logiciel de TA; on peut donc supposer que l'utilisateur ajustera la « normalité » de son discours en concéquence. Et le fait que sa rigueur se voie récompensée par une traduction de meilleure qualité ne peut que le motiver d'avantage. On peut donc supposer que les caractéristiques du tchat multilingue se révèleront plus proches de la norme de l'écrit que ne le sont celles du tchat monolingue.

De plus, il faut bien avoir à l'esprit qu'on ne recherche pas une traduction parfaite, mais une traduction simplement *utile*. La qualité d'un service intégrant de la traduction ne peut pas se mesurer à l'aune de la qualité absolue de la traduction obtenue. Par exemple, des utilisateurs possédant quelques notions des langues de leurs interlocuteurs pourraient se contenter d'un simple service de « sous-titrage » assez approximatif ; tandis qu'une négociation de contrat requiert une qualité de traduction optimale.

On cherche aussi à mesurer la tolérance des utilisateurs aux contraintes, et des moyens permettant de réduire ces dernières autant que possible sans nuire à la qualité du service. Elles sont de deux ordres :

- d'abord, on souhaite contraindre l'utilisateur à une certaine rigueur orthographique et stylistique, à laquelle les tchateurs sont peu accoutumés, comme on l'a constaté ;
- ensuite, on souhaite proposer à l'utilisateur d'assister le traducteur automatique, d'une part en levant certaines ambiguïtés à la demande du système du traducteur (cf. [BLA94]), et d'autre part en évaluant lui-même la traduction obtenue, par exemple au vu d'une rétro-traduction ; il s'agit d'une traduction *interactive* (voir aussi 4.2.2, au sujet de l'introduction de l'interaction dans la traduction de tchat).

C'est à cette fin qu'on se propose d'élaborer un corpus de tchat multilingue. Un tel corpus permettrait d'évaluer l'utilisabilité d'un système de tchat multilingue intégrant intéraction et contrôle de l'utilisateur, au vu des performances actuelles des traducteurs automatiques.

Il existe actuellement quelques outils de tchat multilingue, comme *WordLingo Chat*, *AmiChat*, et *The Pacific Grove Multilingual Chat Room* (cf. [ROS01] pour une comparaison entre ces trois systèmes). Ceux-ci sont généralement soit coûteux (car destinés à des entreprises), soit trop limités dans leur versions d'essai (par exemple, impossibilité de générer des logs), et surtout ne permettent pas de contrôler la traduction.

Il m'a donc fallu réaliser un système de tchat multilingue basique, capable de générer un corpus, et qui puisse servir de base pour l'expérimentation de fonctions de traduction interactive. Dans le cadre du master, je me suis limité à l'ajout de fonctions de traduction automatique classiques à un système monolingue libre préexistant.

3.2 Etude d'un premier outil pour la collecte de tchat multilingue avec TA classique

3.2.1 Choix du type d'implémentation

Différents types d'implémentation ont été envisagés pour traduire du tchat. Le premier consistait en un robot (cf. 1.3 pour plus de précisions sur les robots de tchat). Cette implémentation présente plusieurs avantages. D'une part, la programmation de robots se fait très simplement grâce à des langages de script. D'autre part, le système de TA ainsi réalisé peut être introduit sur n'importe quel canal de tchat compatible ; il suffit d'obtenir l'accord des usagers du canal sur lequel on souhaite l'introduire pour trouver des testeurs. Mais, outre l'incertitude quant à la disponibilité et à la fiabilité des API mises à disposition par ces langages de scripts[5], leur statut de clients leur interdit d'aiguiller les messages. Toutes les traductions sont donc nécessairement adressées à tous les utilisateurs du système, y compris à ceux qui ne souhaiteraient par y avoir recours du tout. Dès lors, imaginons une discussion entre locuteurs du français, de l'anglais, de l'allemand et de l'espagnol : chaque message serait traduit dans les trois autres langues, ce qui multiplierait par quatre le nombre de messages affichés en même temps. Le déchiffrement d'une telle conversation risquerait de devenir rapidement, sinon totalement impossible, du moins suffisamment pénible pour que les usagers du canal ne souhaitent pas prolonger l'expérience.

C'est finalement une implémentation de type passerelle qui a été retenue. Se comportant comme un serveur vis à vis des clients, un tel système permet de centraliser les informations, ce qui simplifie la gestion des logs, tout en permettant de ne transférer aux clients que ce qui leur est nécessaire (se reporter en 3.4.1 pour plus de détails sur le fonctionnement de la passerelle).

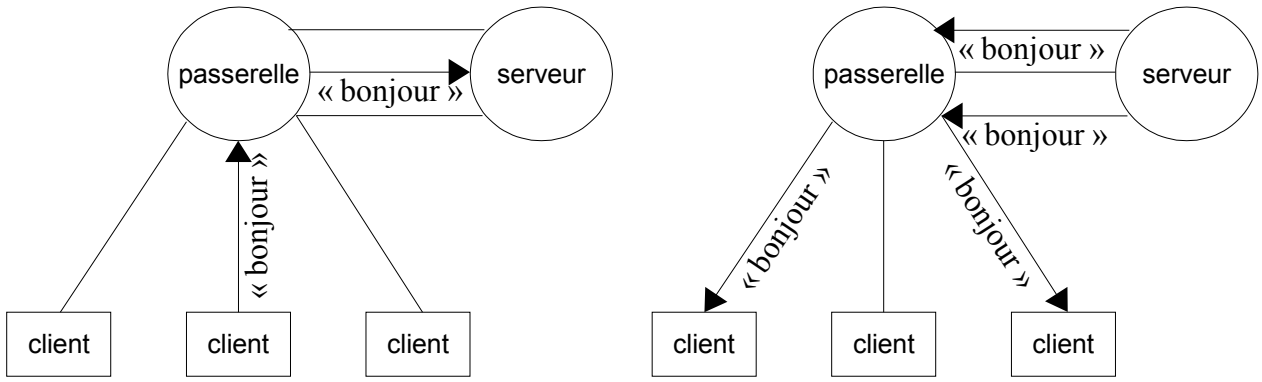


Figure 5 : tous les messages passent par la passerelle, que ce soit dans le sens client-serveur ou dans le sens serveur-client.

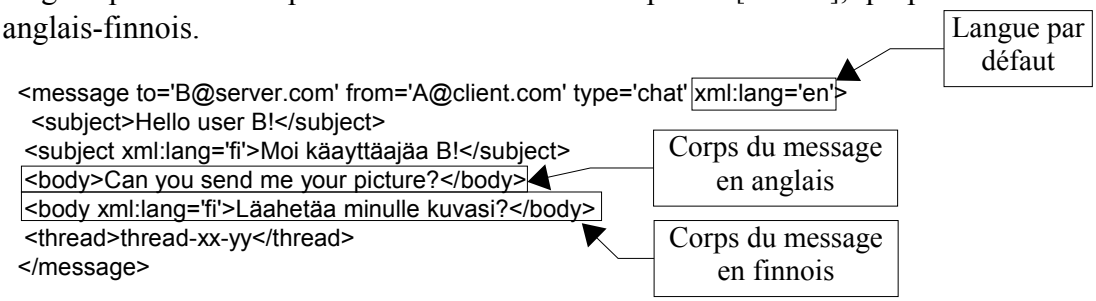
5 On a en particulier besoin d'expressions régulières et de gestion du XML.

3.2.2 Choix du protocole et du réseau

Afin de maximiser le nombre d'usages et d'utilisateurs, on souhaitait pouvoir proposer aux utilisateurs aussi bien du tchat que de la messagerie instantanée. Bien qu'étant un protocole ouvert, IRC n'était donc pas satisfaisant, car limité au seul tchat. En effet, la possibilité offerte par IRC d'envoyer des messages privés ne saurait entrer en concurrence avec la messagerie instantanée. De ce point de vue, le meilleur protocole était XMPP[6].

Il s'agit d'un protocole ouvert, client-serveur et serveur-serveur, basé sur XML. Il s'applique aux systèmes de messagerie instantanée, indicateurs de présence, et, plus intéressant pour nous, au tchat. Bien que peu connu, ce protocole, soutenu par la société *Jabber*, a été en majeure partie approuvé par l'IETF[7] comme proposition de standard. Les spécifications complètes du protocole XMPP sont disponibles sur le site de la *Jabber Software Foundation*[8], dont [LAU04] a tiré une présentation plus synthétique.

Le protocole XMPP dispose d'un autre atout indéniable dans le cadre de ce stage, puisqu'il permet de décrire des conversations multilingues, grâce à l'attribut optionnel *xml:lang*. Cet attribut, relevant de l'espace de noms standard de XML, peut être appliqué à un message pour indiquer sa langue par défaut, dont chacun des éléments peut ensuite être accompagné de sa traduction dans une langue spécifiée. Je reproduis ci-dessous un exemple de [LAU04], qui présente un message bilingue anglais-finnois.



Exemple 6 : message de messagerie instantannée bilingue anglo-finnois codé en XMPP; exemple tiré de [LAU04].

Malheureusement, aucun serveur XMPP n'implémente cette fonctionnalité à l'heure actuelle.

Le réseau basé sur XMPP se nomme Jabber (éponyme de la société *Jabber*). Ce réseau est ouvert. En effet, n'importe qui peut installer un serveur chez soi, qui communiquera alors avec les autres serveurs du réseau Jabber. De ce fait, un nombre important de serveurs et de clients est disponible, et beaucoup sont distribués sous licence GPL. Ces derniers pourront donc servir de base pour des développement futurs, visant par exemple à améliorer l'ergonomie des clients de tchat multilingue.

6 *Extensible Messaging and Presence Protocol*

7 L'IETF (*Internet Engineering Task Force*) est le groupe de normalisation des protocoles Internet (http://www.ietf.org/).

8 JSF (http://www.jabber.org/).

3.3 Modélisation d'une session de tchat multilingue

3.3.1 Représenter du tchat multilingue

Une session de tchat multilingue est, comme une session monolingue, constituée de messages. Et de la même manière, on distinguera des messages « normaux », des commandes, et des notifications d'événements. L'aspect multilingue tient au fait que chaque message pourra comporter plusieurs versions traduites en plus de l'originale. Précisons aussi que l'on compte traiter de la même manière tchat et messagerie instantanée, qui, comme on l'a montré en 1.1, ne diffèrent pas fondamentalement. Enfin, nous souhaitons modéliser certaines informations concernant la configuration des clients, à savoir la langue choisie par l'utilisateur en émission, ainsi que les langues choisies en réception (voir 3.5.2 pour une description des possibilités de configuration offertes à l'utilisateur).

3.3.2 Formalisme adopté

Le format de représentation est à nouveau basé sur XML. A la racine <*log*> sont attachés deux types d'éléments : des <*message*>, <*commande*>, et <*evenement*> d'une part, et des éléments <*client*> d'autre part.

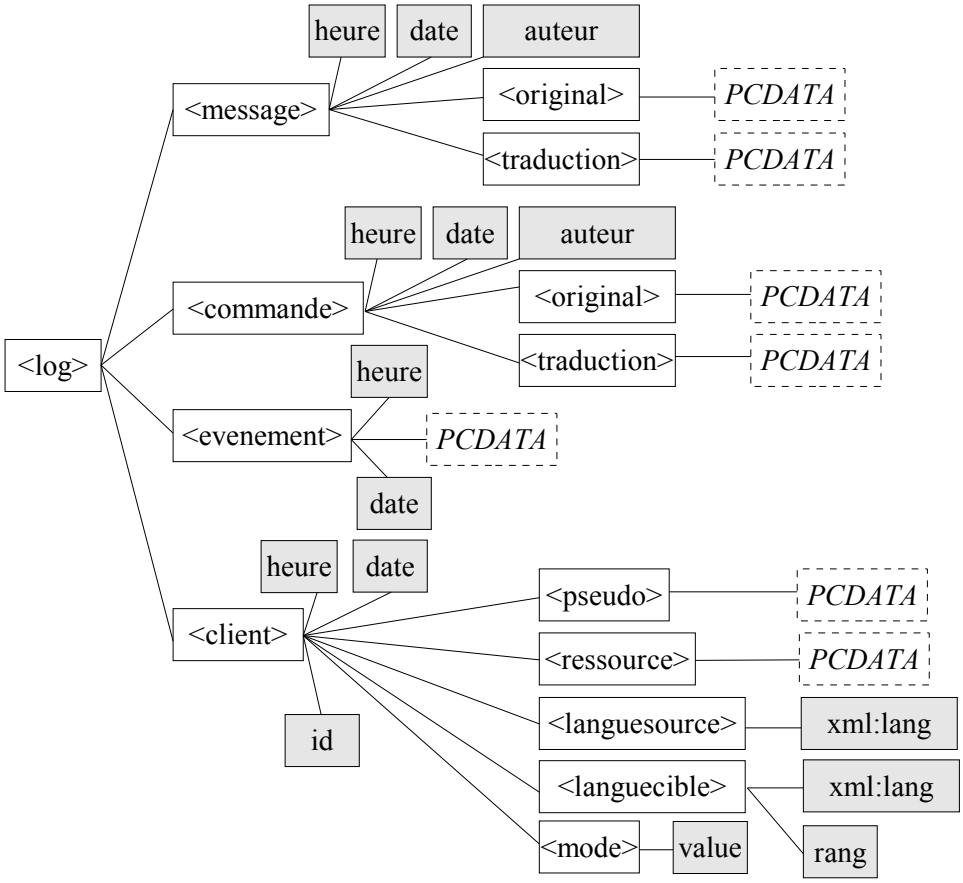


Figure 6 : hiérarchie des éléments du corpus multilingue.

Les <*message*>, <*commande*> sont semblables à ceux employés pour le corpus monolingue (*cf.* 2.1.2), à ceci près que le message n'y est plus directement accessible, mais se situe un niveau plus bas, à l'intérieur d'éléments <*original*> et <*traduction*>, respectivement destinés à l'original du message (un message possède toujours un original et un seul) et à ses éventuelles traductions. Les

éléments <*original*> et <*traduction*> ne possèdent qu'un attribut *xml:lang*. Le formatage du texte étant possible en XMPP, le message pourra contenir lui-même des balises (en particulier une balise <*body*>). Les <*evenement*> contiennent simplement un code de l'événement décrit.

Les <*client*>, quant à eux, servent à représenter la configuration des clients. Comme l'utilisateur peut changer la configuration de son client au cours d'une conversation, chaque élément <*client*> est caractérisé, outre par un attribut *id* identifiant le client, par des attributs *date* et *heure*, permettant d'associer chaque message avec les configurations clientes correspondantes. Un élément <*client*> peut en outre contenir les éléments suivants (leurs rôles respectifs sont détaillés en 3.5.2) :

- <*pseudo*>, qui donne le pseudonyme de l'utilisateur en PCDATA ;
- <*ressource*>, qui indique le client utilisé, lui aussi en PCDATA ;
- <*languesource*>, dont l'attribut *xml:lang* donne la langue de l'utilisateur ;
- <*languecible*>, dont les attributs *xml:lang* et *rang* donnent respectivement une langue de traduction pour cet utilisateur, et la préférence accordée à cette langue[9] ;
- <*mode*>, dont l'attribue *value* donne le mode d'affichage.

Ces éléments sont optionnels, mais doivent tous être spécifiés au moins une fois avant l'envoi du premier message. Toutefois, ces informations étant transmises par le serveur au moment de la connexion du client, cette contrainte ne pose pas de difficulté.

3.4 Implémentation

3.4.1 Vue d'ensemble

Le système de tchat multilingue implémenté se compose de trois programmes bien distincts : un client, un serveur, et une passerelle venant s'intercaler entre eux.

Le choix du client est laissé à l'entière discrétion de l'utilisateur ; le système fonctionne avec n'importe quel client XMPP standard, sous réserve qu'il soit correctement configuré (*cf.* 3.5.2). Le site de la *Jabber Software Foundation* recense des dizaines de clients pour le réseau Jabber.

Le choix du serveur s'est avéré plus problématique. Aucun serveur gratuit ne possède de fonction de log, et un seul d'entre eux, *Ejabberd*, implémente le tchat en plus de la messagerie instantannée. C'est donc ce dernier que j'ai retenu. Dans ce système, le serveur est configuré pour écouter les clients sur un port non-standard, de manière à ne pas pouvoir communiquer directement avec eux. Par contre, port destiné à la communication avec les autres serveurs n'est pas modifié, si bien que les clients connectés au serveur peuvent toujours dialoguer normalement avec les clients connectés sur d'autres serveurs, sans traduction.

C'est la passerelle qui effectue le liaison entre le serveur et les clients. Elle écoute à la fois sur le port standard des clients du réseau Jabber et sur le port non-standard du serveur, et transfère tout ce qui arrive sur l'autre port, non sans l'avoir préalablement traité.

9 L'utilisateur peut choisir plusieurs langues de traduction, par ordre de préférence (*cf.* 3.5.2).

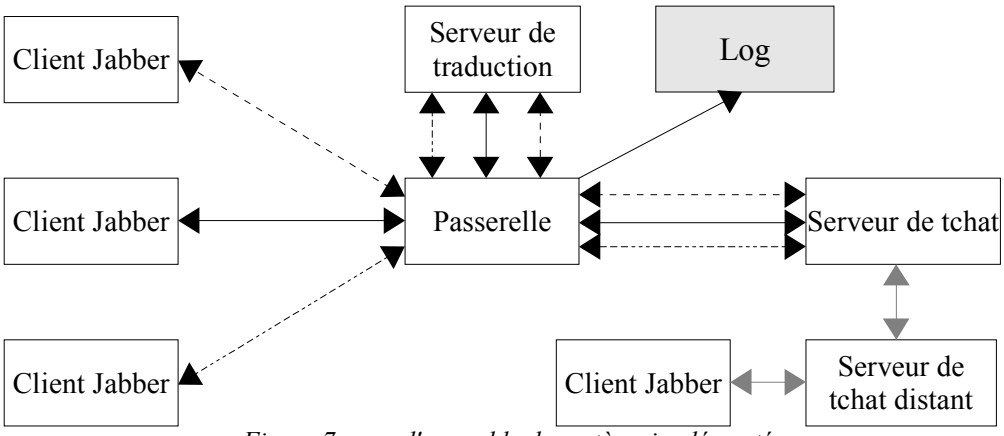


Figure 7 : vue d'ensemble du système implémenté.

3.4.2 La passerelle

La passerelle a été entièrement développée en Java (le code source est disponible en annexe). Comme on vient de le voir, elle sert de relai entre le serveur et les clients. Chaque entrée est parcourue à l'aide d'expressions régulières, afin d'en extraire les éventuels messages, ainsi que d'autres informations utiles, comme le pseudo d'un nouvel utilisateur, le type de client qu'il utilise, ou encore les langues qu'il a sélectionnées. Les traitements effectués ensuite sur les messages sont de deux ordres : traduction et génération de logs.

La passerelle planifie la traduction pour chaque client (choix des langues source et cible, en fonction de la configuration), qui sera effectuée par un serveur spécialisé. Actuellement, il s'agit de celui de Systran, et la communication se fait par requêtes HTTP.

La réalisation du système de génération des logs n'est pas encore achevée. Une classe existe, dont les méthodes sont correctement appelées dans le reste du programme, mais c'est dernières n'implémentent pas pour l'instant l'algorithme de génération de logs.

3.5 Utilisation

3.5.1 Fonctionnalités

Le système propose trois modes de traduction à chaque utilisateur :

- Pas de traduction. Tous les messages sont transférés sans modification.
- Sous-titré. Les messages apparaissent deux fois ; une fois sous leur forme originale, et une fois sous forme traduite. Ce mode peut être utile pour les utilisateurs maîtrisant partiellement la langue de leur interlocuteur : la traduction est alors utilisée comme une aide à la compréhension (lexicale notament).
- Version doublée. Les messages originaux ne sont pas visibles, seules les traductions le sont.

Précisons enfin qu'il est possible de changer de monde à n'importe quel moment, sans se déconnecter.

3.5.2 Configuration

L'utilisation du réseau Jabber nécessite une inscription sur un serveur. Le compte ainsi créé comporte diverses informations, dont certaines sont éditables. C'est notament le cas du champ « description », qui est généralement laissé vierge. Dans notre système de tchat, on configure son compte client en y déclarant les options choisies. Ces déclarations peuvent intervenir n'importe où dans le champ, éventuellement après la véritable description si l'utilisateur souhaite en saisir une. Quatre types de déclarations sont possibles, correspondant à autant d'options de configuration :

- « [lg:*xx*] », où *xx* est le code à deux caractères de la langue source de l'utilisateur ;
- « [so:*x*] », où *x* vaut 1 si les messages originaux (non traduits) doivent être affichés, et 0 sinon ;
- « [st:*x*] », où *x* vaut 1 si les messages traduits doivent être affichés, et 0 sinon ;
- et enfin « [lg*n*:*xx*] », où *xx* est un choix de langue cible (de traduction), et *n* sont ordre de préférence. Si le premier choix n'est pas rendu possible par les paires de langues disponibles, c'est le second choix qui sera essayé, et ainsi de suite jusqu'à épuisement de toutes les possibilités. Si un message ne peut pas être traduit, il est affiché dans sa langue d'origine.

Par exemple, un utilisateur dont le champ « description » contient « [lg:fr] [st:1] [so:0] [lg1:fr] [lg2:en] » est un francophone, qui utilise le mode « version doublée » (pas de message original), et souhaite une traduction de préférence en français, et en anglais si la paire de langue appropriée n'est pas disponible.

Cette méthode est certe peu ergonomique, mais a le mérite de fonctionner avec tous les clients et tous les serveurs. Et ainsi, l'utilisateur peut modifier ces options à n'importe quel moment, sans avoir à se déconnecter. Il est toutefois prévu de créer une interface d'inscription insérant automatiquement les valeurs requises dans le champ approprié du compte client.

3.5.3 Exemple

Dans ce très court exemple, un utilisateur francophone communique avec un utilisateur anglophone grâce à ce système de tchat. L'utilisateur francophone (à gauche) utilise le mode d'affichage « version doublée », tandis que l'anglophone (à droite) se sert du mode « sous-titré ». Bien que de mauvaise qualité, la traduction semble suffisante pour suivre la conversation.

Francophone :	**Anglophone :**
> Salut (1) C'est l'utilisateur francophone qui commence.	[fr] Salut [en] Hello (2) L'utilisateur anglophone lit ceci
[fr] bonjour ! (4) etc...	> hello! (3) L'utilisateur anglophone lui répond
> Que fais-tu ce week-end ?	[fr] Que fais-tu ce week-end ? [en] What do you make this weekend ?
[fr] rien spécial.	> nothing special.
> Une randonnée ça te dirait ?	[fr] Une randonnée ça te dirait ? [en] Would an excursion that say to you ?

Exemple 7 : discussion multilingue

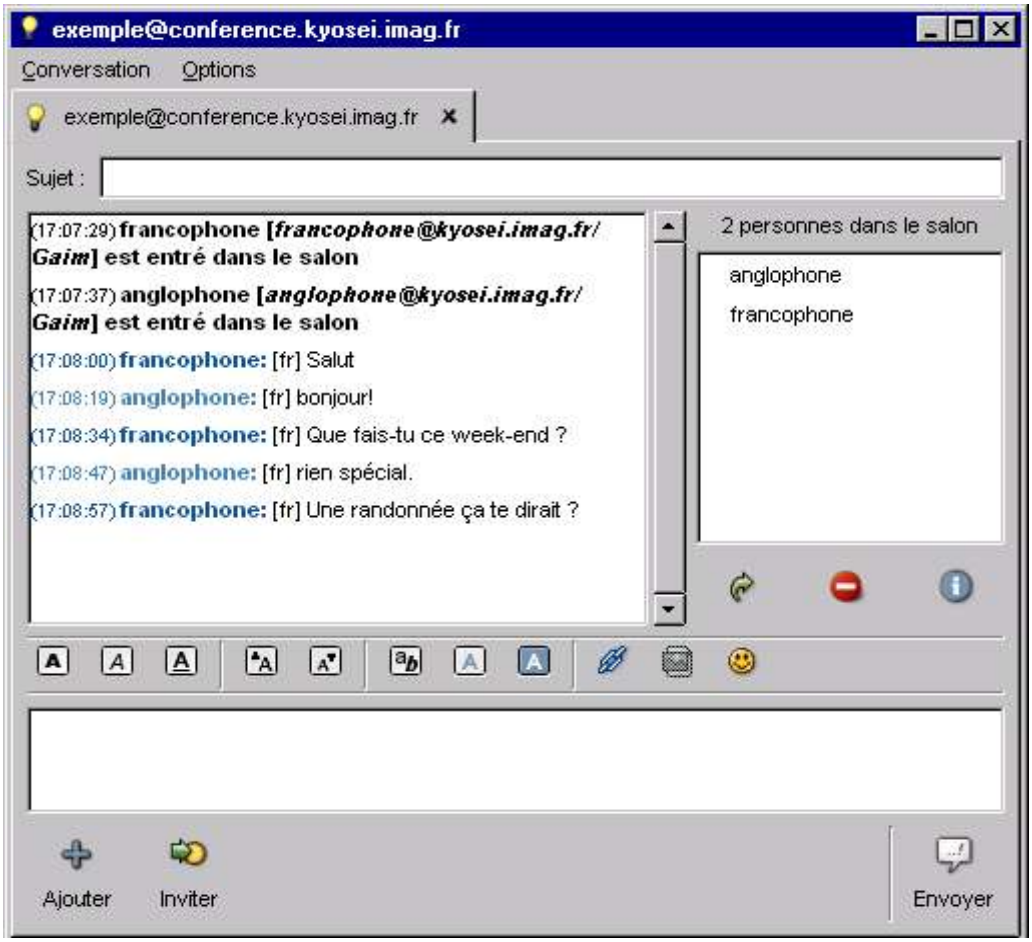


Figure 8 : discussion multilingue de l'exemple 7, sur le client GAIM, du point de vue de l'utilisateur francophone.

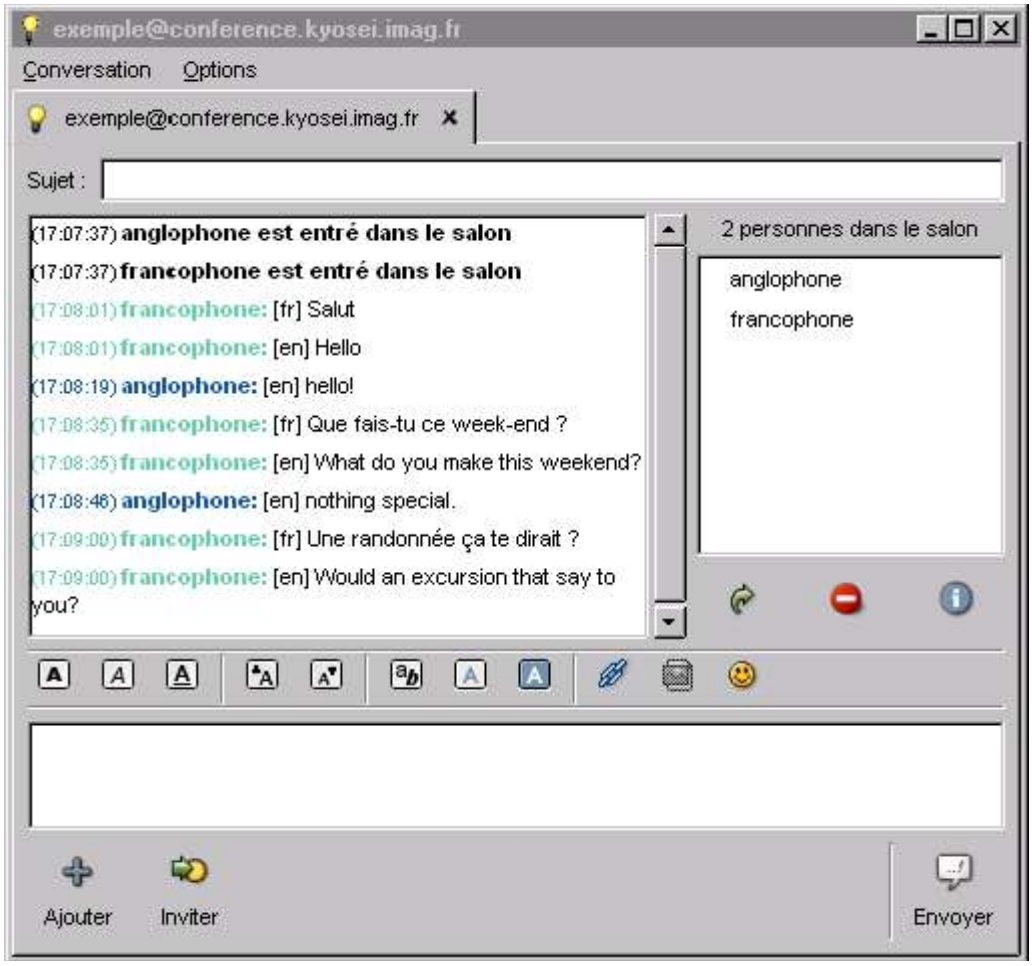


Figure 9 : discussion multilingue de l'exemple 7, sur le client GAIM, du point de vue de l'utilisateur anglophone (l'interface est en français).

IV. VERS DU TCHAT MULTILINGUE MULTIMODAL, AVEC RVAO[10] ET TAFD[11]

A partir de cette étude, on peut envisager divers développements, tant dans le domaine de l'écrit que dans celui de la parole.

4.1 L'écrit : un tchat multilingue avec traduction interactive

Pour compléter cette étude, il faudrait introduire de la traduction interactive dans notre tchat multilingue. Le manière la plus simple de décrire sous quelle forme on compte introduire cette interaction est de s'aider d'un exemple.

On dispose de trois ressources :

1. Un serveur de tchat.

2. Un serveur de traduction interactif. Celui, lorsqu'il est confronté à une ambiguïté (quand plusieurs traductions sont possibles), retourne :
   a. toutes les traductions possibles (si possible ordonnées par probabilité), de manière factorisée ;
   b. une question de désambiguïsation (cf. [BLA94]), c'est à dire une question présentant clairement les différentes alternatives de l'ambiguïté. Par exemple : « *dans l'énoncé "...", le mot "..." est-il un nom ou un verbe ?* ».

3. Un client adapté pour la traduction interactive.

On imagine un utilisateur francophone, connecté sur le tchat en compagnie d'un utilisateur anglophone. La conversation s'engage, et à un moment notre utilisateur francophone envoie le message « *et quoi de neuf avec ton ami richard ?* ». C'est le cheminement de ce message que nous nous proposons de suivre ci-après :

1. Le message part d'abord vers le serveur de tchat.

2. Celui-ci remarque qu'un utilisateur anglophone est connecté. Il transfère donc le message au serveur de traduction, en lui demandant une traduction du français vers l'anglais.

3. Le serveur de traduction constate que deux traductions sont possibles : « *and what's up with your friend richard ?* » et « *and what's up with your rich friend ?* ». Il retourne donc un message factorisé au serveur de tchat, « *and what's up with your* [ⓒ *friend richard* / ⓐ *rich friend*] *?* ». Ce message est accompagné d'une question (en français) pour le client francophone : « *"richard" est-il :* ⓒ *un nom ou* ⓐ *un adjectif ?* ».

10 Reconnaissance Vocale Assistée par Opérateur

11 Traduction Automatisée Fondée sur le Dialogue

4. Le serveur de tchat transmet ce message factorisé au client anglophone. Au client francophone, il envoie la question de désambiguïsation, accompagnée d'une nouvelle traduction en français du message obtenu, ici « et quoi de neuf avec ton ami [♂ richard / ♁ riche] » (rétrotraduction).

5. Chacun des deux clients affiche donc un message factorisé. Le client francophone présente, en plus, la question de désambiguïsation, « *"richard" est-il :* ♂ *un nom ou* ♁ *un adjectif ?* », à son utilisateur. Ce dernier peut choisir de ne rien faire, par exemple s'il estime que l'ambiguïté ne pose pas de problème critique ; auquel cas le traitement du message s'arrête ici. Mais il peut aussi décider, à n'importe quel moment (même après avoir déjà envoyé d'autres messages) de sélectionner l'une des alternatives qui lui sont proposées, soit dans la représentation factorisée du message rétrotraduit, soit en répondant à la question de désambiguïsation. Admettons qu'il réponde ♂ (« *richard* » est un nom). L'affichage est mis à jour sur le client, et la réponse est envoyée au serveur de tchat.

6. Le serveur de tchat transfère cette réponse à l'autre client connecté, sans passer par le serveur de traduction.

Ce dernier « coupe » l'interprétation ♁ dans la représentation factorisée « *and what's up with your* [♂ *friend richard* / ♁ *rich friend*] *?* », qui se simplifie donc en « *and what's up with your friend richard ?* ». L'utilisateur anglophone voit sa représentation factorisée de ce message se simplifier. Cette ambiguïté est maintenant résolue pour tous les clients.

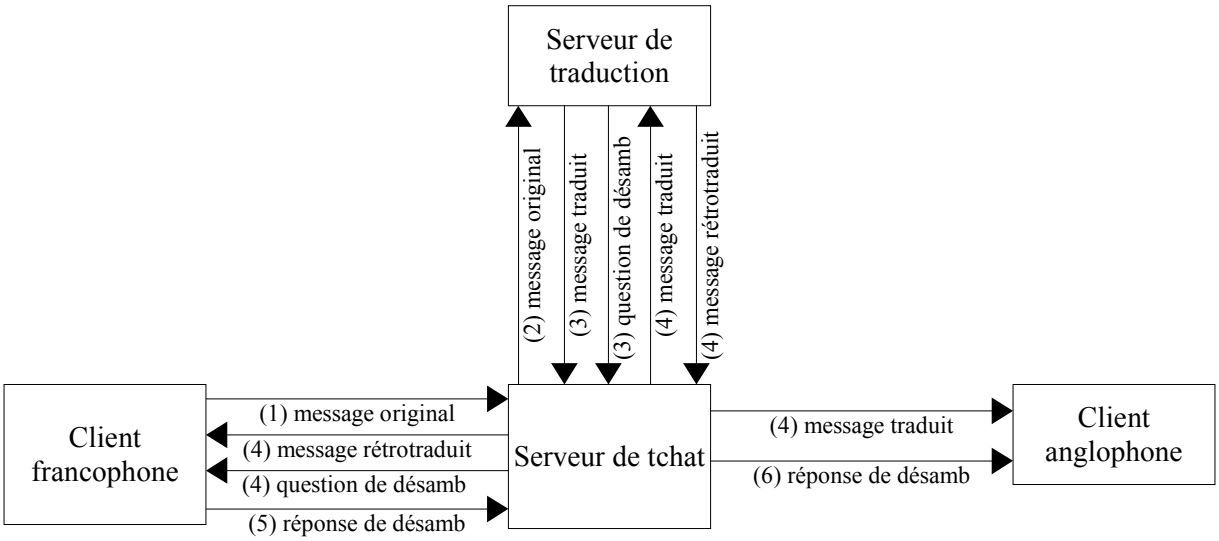


Figure 10 : exemple de traduction interactive en tchat.

4.2 La parole

Proche de la langue parlée, le tchat se prête remarquablement bien à une transposition dans le domaine oral. J'esquisse ci-après quelques uns des scénarios, plus ou moins ambitieux, que l'on peut envisager. Dans le cadre de mon projet de thèse portant sur la traduction interactive de vidéoconférences, je serai sans doute ammené à les approfondir.

4.2.1 Un tchat oral multilingue

Le tchat oral existe déjà, cela s'appelle de la vidéoconférence. Un premier niveau de gestion du multilinguisme dans ce type dialogue consisterait à générer des logs écrits des conversations, par

reconnaissance vocale. Inutile à ce niveau de traduire ces logs. Tout le monde a pu constater, un jour ou l'autre, combien il était comparativement plus simple de comprendre un énoncé en langue étrangère, lorsque celui-ci est sous forme écrite, plutôt que sous forme orale. L'utilisateur d'un tel système pourra ainsi se reporter, s'il éprouve des difficultés à comprendre ce que dit son interlocuteur, à une transcription orthographique de l'énoncé qu'il vient d'entendre. Lorsque l'on a quelques notions d'une langue, mais sans pouvoir prétendre au bilinguisme, ce genre de service peut vraiment être utile.

4.2.2 De la CMAO

Pour aller plus loin, on pourrait maintenant envisager de traduire ces logs. Le recours à de la RVAO[12] peut s'imposer, une bonne fiabilité de la reconnaissance vocale étant alors nécessaire, afin de ne pas cumuler les risques d'erreurs. Ici, notre utilisateur en difficulté peut en plus se reporter à une traduction du log. On pourrait alors parler de CMAO, de Conversation Multilingue Assistée par Opérateur.

4.2.3 CMAO et TAFD

Enfin, on avançant encore dans l'idée d'assistance à la machine, on peut introduire une désambiguïsation interactive au niveau de la traduction automatique. Cette solution, décrite pour l'écrit en 4.1, est actuellement implémentée sur quelques produits commerciaux, comme par exemple la version 5 de *Systran*. Cette désambiguïsation interactive pourrait être, comme pour la RVAO, confiée à des opérateurs, de manière totalement transparente pour l'utilisateur. La qualité de la traduction prometrait alors d'être remarquable.

4.2.4 Exploitation des logs

Pour terminer, une utilisation *a posteriori* des logs ainsi obtenus est aussi envisageable, par exemple dans le cadre d'une génération automatique de documents, tels que comptes rendus et résumés.

12 Reconnaissance Vocale Assistée par Opérateur : identique à de la reconnaissance vocale « classique », mais le système fait appel à des opérateurs humains spécialisés lorsqu'il ne parvient pas à résoudre un problème. Ce type de stratégie est acutellement utilisé par des sociétés telles que *Prosodie*.

CONCLUSION ET PERSPECTIVES

On l'a vu, le tchat constitue finalement, sinon une langue à part entière, du moins un mode de communication tout à fait remarquable, à mis chemin entre la langue écrite et la langue parlée, emprutant ses caractéristiques aussi bien à l'une qu'à l'autre. Le logiciel de tchat développé durant le stage a été testé, et s'avère parfaitement fonctionnel.

Pour la suite du stage, on envisage donc de mettre le système de tchat multilingueen ligne, et dès lors de commencer à collecter les données permettant de construire le corpus. La semaine du 14 juin sera consacrée à la mise en ligne du serveur de tchat et de sa passerelle multilingue. Parallèlement à la mise en ligne du système, une page web devra être réalisée, présentant le projet[13] et ses modalités d'utilisation. Enfin, il faudra faire connaître le projet, et inviter les internautes à venir essayer le système. En l'état, le système est plus avancé que ce que l'on peut trouver gratuitement sur le web, il devrait donc attirer quelques testeurs, voire des utilisateurs réguliers si sa qualité de service s'avère suffisante. Si la semaine du 14 s'est bien passée, cette étape pourrait débuter dès le 21 juin. A plus long terme enfin, j'espère pouvoir approfondir les pistes concernant le développement de services de tchat multimodal et multilingue, évoquées plus haut, dans le cadre d'une thèse.

13 Bien entendu, cette page avertira l'utilisateur que ses conversations seront loggées.

BIBLIOGRAPHIE

[BLA94] Hervé BLANCHON, *LIDIA-1 : une première maquette vers la TA interactive "pour tous"*, Thèse, Université Joseph Fourier - Grenoble 1, janvier 1994.

[BOI94] Christian BOITET, *On the design of MIDDIM-DB, a data base of ambiguities and disambiguation methods*, International conference on recent advances in natural language processing, Bulgarie, septembre 1994.

[BOI95A] Christian BOITET & Mutsuko TOMOKIYO, *Towards ambiguity labelling for the study of interactive disambiguation methods*, ATR Interpreting Telecommunications Research Laboratories, avril 1995.

[BOI95B] Christian BOITET & Mutsuko TOMOKIYO, A*mbiguities & ambiguity: towards ambiguity data bases*, ATR Interpreting Telecommunications Research Laboratories, septembre 1995.

[BOI96] Christian BOITET, *Importance and complementarity of automatic and interactive disambiguation*, *in* MIDDIM-96, Interactive Disambiguation, août 1996.

[BOI02] Christian BOITET & TSAI Wang-ju, *La coédition langue <-> UNL pour partager la révision entre les langues d'un document multilingue: un concept unificateur*, RECITAL, 2002.

[DOL96] William B. DOLAN & Stephen D. RICHARDSON, *Interactive lexical priming for disambiguation*, *in* MIDDIM-96, Interactive Disambiguation, août 1996.

[FAF94] Georges FAFIOTTE & Christian BOITET, *Report on first EMMI Experiments for the MIDDIM project in the context of Interpreting Telecommunication*, TR-IT-0074, septembre1994.

[FAF96] Georges FAFIOTTE & Christian BOITET, *An analysis of the first EMMI-based experiments on interactive disambiguation in the context of automated interpreting telecommunications*, *in* MIDDIM-96, Interactive Disambiguation, août 1996.

[FAI96] Laurel FAIS, Hervé BLANCHON, *Ambiguities in Task-oriented Dialogues*, *in* Proceedings of MIDDIM'96.

[FAR03] Martine FARACO, Marie-Laure BARBIER, Achille FALAISE et Sonia BRANCA ROSOFF, *Codage et traitement automatique de corpus pour l'étude de prises de notes en français langue première et langue seconde*, *in* Arob@se, vol. 1-2, 2003 ; *cf*. http://www.arobase.to.

[FUC96] Catherine FUCHS, *Les ambiguïtés du français*, OPHRYS, 1996.

[GAD03] Fabrice GADAUD, *Jabber/XMPP, une évolution dans l'envoi de messages*, octobre 2003

[HIN] Kajetan HINNER, *IRCNet Statistics*, *in* Kajetan Hinner: IRC Statistics, http://www.hinner.com/ircstat/.

[LAT01] Guillaume LATZKO-TOTH, *L'Internet Relay Chat: un dispositif sociotechnique riche d'enseignements*, *in* Emergences et continuité dans les recherches en information et communication, actes du XIIe Congrès national des sciences de l'information et de la communication, UNESCO, Paris, 2001.

[LAU04] Mikko LAUKKANEN, *Extensible Messaging and Presence Protocol*, Helsinki, 2004

[MAG] Mehdi MAGHSOODNIA, *Instant Messaging, ready for prime time ?*, *in* WSTA Publication, http://www.wsta.org/publications/articles/1202_article04.html.

[NET02] *Les outils de communication de l'Internet*, *in* NetValue, juin 2002.

[PIE03] Isabelle PIEROZAK, *Le "français tchaté" : un objet à géométrie variable ?*, *in* Langage et Société n° 104, juin 2003, pp. 123-144.

[PUJ01] Laurent PUJADE, L'écrit sur internet, mémoire de maîtrise, Toulouse, septembre 2001.

[ROS01] Luigi Canali De Rossi, *Free demo Multilingual chat systems*, 2001, http://www.masternewmedia.org/2001/12/31/free_demo_multilingual_chat_systems.htm

[ZAY96] Laurence ZAYSSER, *Representing morpho-syntactic ambiguity*, *in* MIDDIM-96, Interactive Disambiguation, août 1996.

[ZHA03] Ying ZHANG, *Survey of current speech translation research*, Pittsburgh, 2003.

ANNEXES

Corpus monolingue

Canaux

irc.epiknet.org#--quizz--
irc.epiknet.org#botstats
irc.epiknet.org#-p-u-r-
irc.epiknet.org#18-25ans
irc.epiknet.org#actu
irc.epiknet.org#allsoluces
irc.epiknet.org#anaisgirl
irc.epiknet.org#angel-corp
irc.epiknet.org#blacksky
irc.epiknet.org#blondin
irc.epiknet.org#c++
irc.epiknet.org#caline
irc.epiknet.org#cocktails
irc.epiknet.org#cstrike
irc.epiknet.org#darkcloud
irc.epiknet.org#dbz_legend
irc.epiknet.org#debian
irc.epiknet.org#deejays-world
irc.epiknet.org#deglingo
irc.epiknet.org#dragon-ball-z
irc.epiknet.org#edelweiss
irc.epiknet.org#edensensuelcam
irc.epiknet.org#enjoy
irc.epiknet.org#epik-quizz
irc.epiknet.org#fac
irc.epiknet.org#ffmaniac
irc.epiknet.org#ffparadise
irc.epiknet.org#ffx-2
irc.epiknet.org#fikx
irc.epiknet.org#finaland
irc.epiknet.org#foldingathome
irc.epiknet.org#formypain
irc.epiknet.org#france1
irc.epiknet.org#francophone
irc.epiknet.org#full_download
irc.epiknet.org#funkycops
irc.epiknet.org#g-faction
irc.epiknet.org#games
irc.epiknet.org#gck
irc.epiknet.org#greatnothing
irc.epiknet.org#hikago
irc.epiknet.org#hinatalove
irc.epiknet.org#hokutoteam
irc.epiknet.org#humour
irc.epiknet.org#inuxia
irc.epiknet.org#iquotes
irc.epiknet.org#irpg
irc.epiknet.org#irpg-chat
irc.epiknet.org#ishtar
irc.epiknet.org#japanimation
irc.epiknet.org#jump
irc.epiknet.org#koma
irc.epiknet.org#kyo-music
irc.epiknet.org#le-monde-des-reptiles
irc.epiknet.org#leknaat^_^
irc.epiknet.org#leseigneurdesanneaux
irc.epiknet.org#linux
irc.epiknet.org#lovehina-world
irc.epiknet.org#madness
irc.epiknet.org#magnapoke
irc.epiknet.org#manga
irc.epiknet.org#manga4ever
irc.epiknet.org#manganimation
irc.epiknet.org#mangaparadise
irc.epiknet.org#mew
irc.epiknet.org#mixi
irc.epiknet.org#nemo

irc.epiknet.org#ninou
irc.epiknet.org#nintendojofr
irc.epiknet.org#nokiagame.fr
irc.epiknet.org#php
irc.epiknet.org#planete-gundam
irc.epiknet.org#pokelord
irc.epiknet.org#politique
irc.epiknet.org#princedelu
irc.epiknet.org#programmation
irc.epiknet.org#qcradio
irc.epiknet.org#quebec
irc.epiknet.org#radioabf
irc.epiknet.org#radiofrhub
irc.epiknet.org#ragnarok
irc.epiknet.org#ragot-chan
irc.epiknet.org#raysanctuary
irc.epiknet.org#rhone-alpes
irc.epiknet.org#rien
irc.epiknet.org#sc-team
irc.epiknet.org#scripts
irc.epiknet.org#slackware
irc.epiknet.org#sonia
irc.epiknet.org#squall
irc.epiknet.org#station-fm
irc.epiknet.org#team-realboard
irc.epiknet.org#tg
irc.epiknet.org#thaline
irc.epiknet.org#topsilvermoonusa
irc.epiknet.org#triquet
irc.epiknet.org#venus
irc.epiknet.org#virtuel
irc.epiknet.org#viruscorporation
irc.epiknet.org#wazzo
irc.epiknet.org#webmasters
irc.epiknet.org#wizkids-fr
irc.epiknet.org#wow.fr
irc.epiknet.org#yugioh
irc.epiknet.org#ze-delires
irc.epiknet.org#zeldoot
irc.epiknet.org#zenit
irc.epiknet.org#[dmb]dreamchan

Programme de reformatage du corpus

Programme principal

```
use strict;

package main;

my $racine="d:\\Achille\\Logs de chats";
my $regex = Regex->new;
my $log;
my $nbMots=0;

open(LISTEMOTS, "> $racine\\listeMots.txt");

#*** Corpus BotStats ***
# Index général
open(RACINE, "> $racine\\Botstats\\index.xml");
print RACINE "$_";
print RACINE "<?xml version=\"1.0\" encoding=\"ISO-8859-1\" standalone=\"no\"?>\n";
print RACINE "<?xml-stylesheet type=\"text/xsl\" href=\"$racine/styleracine.xsl\"?>\n";
print RACINE "<listelogindex>\n";
foreach(&getRepFiles("$racine\\Botstats\\Src HTML", "[A-Za-z0-9]+")) {
	print RACINE "\t<logindex src=\"$_\">$_</logindex>\n";
}
print RACINE "</listelogindex>";
close(RACINE);
# Corpus
foreach(&getRepFiles("$racine\\Botstats\\Src HTML", "[A-Za-z0-9]+")) {
	my $rep=$_;
	my @filentities;
	my @date;
	my @heure;
	open(INDEX, "> $racine\\Botstats\\$rep\\index.xml");
	print INDEX "<?xml version=\"1.0\" encoding=\"ISO-8859-1\" standalone=\"no\"?>\n";
	print INDEX "<?xml-stylesheet type=\"text/xsl\" href=\"$racine/styleindex.xsl\"?>\n";
	foreach(&getRepFiles("$racine\\Botstats\\Src HTML\\$rep", "index[A-Za-z0-9]+\.html")) {
		my $file=$_;
		my $texteComplet="";
		print "Traitement du fichier $rep/$file\n";
		$log=TchatLog->new;
		# Chargement
		$log->{theme}="";
		$log->{commentaire}="Log de Botstat";
		$log->{langue}="fr";
		open(SOURCE, "< $racine\\Botstats\\Src HTML\\$rep\\$file") or die("Impossible d'ouvrir le fichier
$racine\\Botstats\\Src HTML\\$rep\\$file : $!");
		my $fichier="";
		while(my $ligne=<SOURCE>) {
			$fichier.=$ligne;
		}
		close(SOURCE);
		$log->{langue}="fr";
		$fichier=~m/<!-- Mirrored from ([^\"<>]+) by HTTrack Website Copier/; $log->{webSrc}=$1;
		$fichier=~m/\t<td width="300" bgcolor=""><div style="height:100px; overflow:auto"><div
align="center"><font face="Verdana" size="2"><b>(#[^<]+)<\/b><\/font><\/div><br><font face="Verdana"
size="1" color="#[0-9A-F]{6}">/; $log->{ircSrc}=$1;
		$log->{webSrc}=~m/\/\?date=([0-9]+)([A-Z][a-z][a-z])(20[0-9][0-9])/; $log->{date}
="$1/".&convDate1($2)."/$3";
		foreach($fichier=~m/<font size=1><font color=#[0-9A-Fa-f]{6}>\[([0-9][0-9]:[0-9][0-9])\]
<\/font> /g) {$log->{heure}.=$_."\n";} chop($log->{heure});
		foreach($fichier=~m/<font size=1><font color=#[0-9A-Fa-f]{6}>\[[0-9][0-9]:[0-9][0-9]\]<\/font>
<font color=[0-9a-fA-F]{6}>&lt;([^&]+)&gt;<\/font> [^<]+<br \/>/g) {$log->{codeScripteur}.=$_."\n";} chop($log-
>{codeScripteur});
		foreach($fichier=~m/<font size=1><font color=#[0-9A-Fa-f]{6}>\[[0-9][0-9]:[0-9][0-9]\]<\/font>
((?:<font color=[0-9a-f]{6}>[^<]+<\/font> )?[^<]+)<br \/>/g) {
			my $contenu=$_;
			#print $contenu."\n";
			if($contenu=~m/^<font color=[0-9a-f]{6}>&lt;[^&]+&gt;<\/font> ((?:!|\/)[^<]+)$/) {
				$log->{texte}.=$1."\n";  $log->{texteType}.="C\n";
			}
```

```
                        elsif($contenu=~m/^<font color=[0-9a-f]{6}>&lt;[^&]+&gt;<\/font> ([^<]+)$/) {
                                $log->{texte}.=$1."\n";  $log->{texteType}.="M\n";
                        }
                        elsif($contenu=~m/^<font color=[0-9a-f]{6}>[^&]+<\/font> ([^<]+)$/) {
                                $log->{texte}.=$1."\n";  $log->{texteType}.="E\n";
                        }
                        elsif($contenu=~m/^([^<]+)$/) {
                                $log->{texte}.=$1."\n";  $log->{texteType}.="E\n";
                        }
                        else {print "N'a pas pu determiner le type>> $contenu\n";}
                        $texteComplet.=$contenu; # Pour le checksum
                }
                chop($log->{texte});
                chop($log->{texteType});
                # Indexation
                push(@date, $log->{date}); my $thisDate=$log->{date};
                $log->{heure}=~m/^([^\n]+)\n/; my $thisHeure=$1;
                push(@heure, $1);
                # Check
                my $goNext=0;
                if($log->{texte} eq "") {$goNext=1;}
                for(my $i=0; $i<@date-1; ++$i) {
                        if($date[$i] eq $thisDate && $heure[$i] eq $thisHeure) {$goNext=1;}
                }
                if($goNext) {
                        pop(@date);
                        pop(@heure);
                        next;
                }
                # Sauvegarde
                push(@filentities, $file);
                $file=~s/\.html$/\.xml/;
                $log->sauver("$racine\\Botstats\\$rep\\$file");
                my @mots=$log->{texte}=~m/\b([A-Za-z0-9éèàçäëüïöâêûîôù]+)\b/g;
                #foreach(@mots) {print LISTEMOTS "$_\n";}
                $nbMots+=@mots;
                print "\t $nbMots mots dans le corpus.\n";
                #print INDEX "\t\t<!ENTITY $filentity SYSTEM \"$file\">\n";
        }
        print INDEX "<logindex ircsrc=\"".$log->{ircSrc}."\">\n";
        for(my $i=0; $i<@filentities; ++$i) {
                my $file=$filentities[$i];
                $file=~s/\.html$/\.xml/;
                print INDEX "<logref src=\"$file\">".$date[$i]." ".$heure[$i]."</logref>";
        }
        print INDEX "</logindex>";
        close(INDEX);
}

die(); # Fin du programme principal
```

Procédures et fonctions

```
sub convDate1 {
	my($s)=@_;
	$s=~s/^Jan$/01/;
	$s=~s/^Feb$/02/;
	$s=~s/^Mar$/03/;
	$s=~s/^Apr$/04/;
	$s=~s/^May$/05/;
	$s=~s/^Jun$/06/;
	$s=~s/^Jui$/07/;
	$s=~s/^Aug$/08/;
	$s=~s/^Sep$/09/;
	$s=~s/^Oct$/10/;
	$s=~s/^Nov$/11/;
	$s=~s/^Dec$/12/;
	return $s;
}

sub getRepFiles {
	my ($rep, $filtre)=@_;
	my $entree;
	opendir(REP, "$rep") or die "Impossible d'ouvrir le répertoire $rep : $!";
	my @fichiers=readdir(REP);
	my @sortie;
	closedir(REP);
	foreach $entree (@fichiers){
		if($entree=~m/$filtre/i) {
			push(@sortie, $entree);
		}
	}
	return @sortie;
}

sub inArray {
	my ($var, @tab)=@_;
	for(my $i=0; $i<@tab; ++$i) {if($_ eq $var) {return $i;}}
	return -1;
}

sub calculeChecksum {
	my($s)=@_;
	my $somme=0;
	foreach(split(//,$s)) {
		$somme+=ord($_);
	}
	return $somme;
}
```

Classe TchatLog

```
# Classe TchatLog
package TchatLog;
sub new {
	my $self={};
	$self->{webSrc}=undef;
	$self->{ircSrc}=undef;
	$self->{theme}=undef;
	$self->{commentaire}=undef;
	$self->{langue}=undef;
	$self->{codeScripteur}=undef;
	$self->{date}=undef;
	$self->{heure}=undef;
	$self->{texte}=undef;
	$self->{texteType}=undef;	#M=Message, C=Commande, E=Evénement
	bless $self;
	return $self;
}

sub sauver {
	# Paramètres
	my ($self, $fichierCibleURL)=@_;
	open(CIBLE, "> ".$fichierCibleURL) or die("Impossible d'ouvrir le fichier $fichierCibleURL : $!");
	print CIBLE "<?xml version=\"1.0\" encoding=\"ISO-8859-1\" ?>\n";
	print CIBLE "<?xml-stylesheet type=\"text/xsl\" href=\"$racine/style.xsl\"?>\n";
	my $webSrc=$self->{webSrc};
	$webSrc=~s/&//g;
	my $ircSrc=$self->{ircSrc};
	$ircSrc=~s/&//g;
	print CIBLE "<log websrc=\"".$webSrc."\" ircsrc=\"".$ircSrc."\" theme=\"".$self->{theme}."\" langue=\"".$self-
>{langue}."\">\n";
	print CIBLE "\t<commentaire>".$self->{commentaire}."</commentaire>\n";
	my @texte=split(/\n/, $self->{texte});
	my @heure=split(/\n/, $self->{heure});
	my @date=split(/\n/, $self->{date});
	my @codeScripteur=split(/\n/, $self->{codeScripteur});
	my @texteType=split(/\n/, $self->{texteType});
	my $indexTexte=0;
	my $indexScripteur=0;
	foreach(@texteType) {
		if($_ eq "M") {
			print CIBLE "\t<message auteur=\"".$codeScripteur[$indexScripteur]."\" date=\"".
(@date==1?$date[0]:$date[$indexTexte])."\" heure=\"".(@heure==1?$heure[0]:$heure[$indexTexte]).
"\">".&formatTexte($texte[$indexTexte])."</message>\n";
			++$indexTexte;
			++$indexScripteur;
		}
		elsif($_ eq "C") {
			print CIBLE "\t<commande auteur=\"".$codeScripteur[$indexScripteur]."\" date=\"".
(@date==1?$date[0]:$date[$indexTexte])."\" heure=\"".(@heure==1?$heure[0]:$heure[$indexTexte]).
"\">".&formatTexte($texte[$indexTexte])."</commande>\n";
			++$indexTexte;
			++$indexScripteur;
		}
		elsif($_ eq "E") {
			print CIBLE "\t<evenement date=\"".(@date==1?$date[0]:$date[$indexTexte])."\" heure=\"".
(@heure==1?$heure[0]:$heure[$indexTexte])."\">".&formatTexte($texte[$indexTexte])."</evenement>\n";
			++$indexTexte;
		}
	}
	print CIBLE "</log>";
	close(CIBLE);
}

sub formatTexte {
	my ($s)=@_;
	$s=~s/&([a-z]*[^;])/&amp;$1/g;
	$s=~s/&([^a-z])/&amp;$1/g;
	$s=~s/&$/&amp;/g;
	return $s;
}
```

Classe Regex

```
# Classe Regex
package Regex;

sub new {
	my $self={};
	$self->{URL}="[^\"<>?]+";
	bless $self;
	return $self;
}

sub getRegex {
	my ($self, $nom)=@_;
	return $self->{$nom};
}
```

Feuille de style

```
  <?xml version="1.0" encoding="ISO-8859-1"?>
  <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns="http://www.w3.org/TR/xhtml1/strict">
  <xsl:output method="html" encoding="ISO-8859-1"/>
  <xsl:template match="log">
  <html>
  <head>
          <title><xsl:value-of select="@ircsrc" /></title>
          <style>body, table {font-family:Verdana,Arial;font-size:10pt;} .evenement {background-
color:skyblue;color:black;font-size:9pt;cursor:hand} .message {background-color:#DDDDDD;color:black;font-size:9pt;}
.commande {background-color:gold;color:black;font-size:9pt;cursor:hand}</style>
  </head>
  <body>
          <p style="font-family:Times new Roman,Times;font-size:16pt;font-weight:bold;" id="menu">Log du corpus
<xsl:value-of select="@ircsrc" /></p>
          <p><a href="index.xml">Retour</a></p>
          <table cellpadding="2">
                  <tr style="background-color:black;color:white;font-weight:bold;text-
align:center"><td>Date</td><td>Heure</td><td>Auteur</td><td>Texte</td></tr>
                  <xsl:apply-templates/>
          </table>
  </body>
  </html>
  </xsl:template>
  <xsl:template match="message">
          <tr><xsl:attribute name="id"><xsl:value-of select="generate-id(.)"/></xsl:attribute><td
class="message"><xsl:value-of select="@date"/></td><td class="message"><xsl:value-of
select="@heure"/></td><td nowrap="" class="message"><xsl:value-of select="@auteur"/></td><td
class="message"><xsl:value-of select="current()"/></td></tr>
  </xsl:template>
  <xsl:template match="commande">
          <tr style="height:3px"><xsl:attribute name="id">OFF<xsl:value-of select="generate-id(.)
"/></xsl:attribute><xsl:attribute name="onclick">this.style.display='none';ON<xsl:value-of select="generate-id(.)
"/>.style.display='';</xsl:attribute><td class="commande" colspan="4"> </td></tr>
          <tr style="display:none"><xsl:attribute name="id">ON<xsl:value-of select="generate-id(.)
"/></xsl:attribute><xsl:attribute name="onclick">this.style.display='none';OFF<xsl:value-of select="generate-id(.)
"/>.style.display='';</xsl:attribute><td class="commande"><xsl:value-of select="@date"/></td><td
class="commande"><xsl:value-of select="@heure"/></td><td nowrap="" class="commande"><xsl:value-of
select="@auteur"/></td><td class="commande"><xsl:value-of select="current()"/></td></tr>
  </xsl:template>
  <xsl:template match="evenement">
          <tr style="height:3px"><xsl:attribute name="id">OFF<xsl:value-of select="generate-id(.)
"/></xsl:attribute><xsl:attribute name="onclick">this.style.display='none';ON<xsl:value-of select="generate-id(.)
"/>.style.display='';</xsl:attribute><td class="evenement" colspan="4"> </td></tr>
          <tr style="display:none"><xsl:attribute name="id">ON<xsl:value-of select="generate-id(.)
"/></xsl:attribute><xsl:attribute name="onclick">this.style.display='none';OFF<xsl:value-of select="generate-id(.)
"/>.style.display='';</xsl:attribute><td class="evenement"><xsl:value-of select="@date"/></td><td
class="evenement"><xsl:value-of select="@heure"/></td><td class="evenement" colspan="2"><xsl:value-of
select="current()"/></td></tr>
  </xsl:template>
  </xsl:stylesheet>
```

Passerelle de tchat multilingue

Classe relaixmpp.CadreInterface

```
package relaixmpp;

import javax.swing.*;
import java.awt.*;
import com.borland.jbcl.layout.*;
import java.awt.event.*;

/**
 * <p>Titre : </p>
 * <p>Description : </p>
 * <p>Copyright : Copyright (c) 2004</p>
 * <p>Société : GETA-CLIPS</p>
 * @author Achille Falaise
 * @version 1.0
 */

public class CadreInterface extends JFrame {
  private Config config;
  GridLayout gridLayout1 = new GridLayout();
  JPanel jPanel1 = new JPanel();
  BorderLayout borderLayout1 = new BorderLayout();
  JLabel jLabel1 = new JLabel();
  JPanel jPanel3 = new JPanel();
  JScrollPane jScrollPane1 = new JScrollPane();
  JPanel connexionsPanel = new JPanel();
  JScrollPane jScrollPane2 = new JScrollPane();
  JTextArea journalRelai = new JTextArea();
  BorderLayout borderLayout2 = new BorderLayout();
  JLabel jLabel2 = new JLabel();
  VerticalFlowLayout verticalFlowLayout1 = new VerticalFlowLayout();
  VerticalFlowLayout verticalFlowLayout2 = new VerticalFlowLayout();
  JLabel nbConnexionsLabel = new JLabel();
  JLabel etatRelaiLabel = new JLabel();
  JPanel jPanel6 = new JPanel();
  JPanel jPanel4 = new JPanel();
  JPanel jPanel2 = new JPanel();
  JTextField nbConnexions = new JTextField();
  JTextField etatRelai = new JTextField();
  JButton jButton1 = new JButton();
  JButton jButton2 = new JButton();
  JButton jButton3 = new JButton();

  public CadreInterface(Config conf) {
    config = conf;
    setTitle(config.titreFenetre);
    setSize(config.largeurFenetre, config.hauteurFenetre);
    try {
      jbInit();
    }
    catch(Exception e) {
      e.printStackTrace();
    }
    //Centrer la fenêtre
    Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
    Dimension frameSize = getSize();
    if (frameSize.height > screenSize.height) {
      frameSize.height = screenSize.height;
    }
    if (frameSize.width > screenSize.width) {
      frameSize.width = screenSize.width;
    }
    setLocation((screenSize.width - frameSize.width) / 2, (screenSize.height - frameSize.height) / 2);
    setVisible(true);
  }

  private void jbInit() throws Exception {
    journalRelai.setBackground(Color.white);
```

```
journalRelai.setEnabled(true);
journalRelai.setFont(new java.awt.Font("Dialog", 0, 10));
journalRelai.setDoubleBuffered(true);
journalRelai.setCaretPosition(0);
journalRelai.setEditable(false);
journalRelai.setText("");
jPanel3.setLayout(borderLayout2);
jLabel2.setEnabled(true);
jLabel2.setFont(new java.awt.Font("SansSerif", 1, 12));
jLabel2.setMaximumSize(new Dimension(43, 24));
jLabel2.setMinimumSize(new Dimension(43, 24));
jLabel2.setPreferredSize(new Dimension(43, 24));
jLabel2.setVerifyInputWhenFocusTarget(true);
jLabel2.setText("Journal");
gridLayout1.setColumns(1);
gridLayout1.setRows(2);
this.getContentPane().setLayout(gridLayout1);
jPanel1.setLayout(borderLayout1);
jLabel1.setFont(new java.awt.Font("SansSerif", 1, 12));
jLabel1.setMaximumSize(new Dimension(102, 24));
jLabel1.setMinimumSize(new Dimension(102, 24));
jLabel1.setPreferredSize(new Dimension(102, 24));
jLabel1.setToolTipText("");
jLabel1.setText("Clients connectés");
connexionsPanel.setBackground(Color.white);
connexionsPanel.setLayout(verticalFlowLayout1);
this.setDefaultCloseOperation(HIDE_ON_CLOSE);
this.setState(Frame.NORMAL);
nbConnexionsLabel.setText("Connexions : ");
etatRelaiLabel.setText("Etat : ");
jPanel2.setLayout(verticalFlowLayout2);
nbConnexions.setDoubleBuffered(true);
nbConnexions.setEditable(false);
nbConnexions.setText("");
nbConnexions.setColumns(5);
etatRelai.setDoubleBuffered(true);
etatRelai.setEditable(false);
etatRelai.setText("");
etatRelai.setColumns(10);
verticalFlowLayout1.setAlignment(VerticalFlowLayout.TOP);
verticalFlowLayout1.setHorizontalFill(false);
jButton1.setText("RàZ du journal");
jButton1.addMouseListener(new CadreInterface_jButton1_mouseAdapter(this));
jScrollPane2.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);
jScrollPane2.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);
jScrollPane2.getViewport().setBackground(Color.lightGray);
jScrollPane2.setDebugGraphicsOptions(0);
jScrollPane2.setDoubleBuffered(true);
jScrollPane2.setToolTipText("");
jScrollPane1.setDoubleBuffered(true);
jButton2.setText("Traiter les logs");
jButton2.addActionListener(new CadreInterface_jButton2_actionAdapter(this));
jButton2.addActionListener(new CadreInterface_jButton2_actionAdapter(this));
jButton3.setText("Déconnecter tous");
jButton3.addActionListener(new CadreInterface_jButton3_actionAdapter(this));
verticalFlowLayout2.setHgap(2);
verticalFlowLayout2.setVgap(2);
this.getContentPane().add(jPanel1, null);
jPanel1.add(jLabel1,  BorderLayout.NORTH);
jPanel1.add(jScrollPane1, BorderLayout.CENTER);
jScrollPane1.getViewport().add(connexionsPanel, null);
jPanel1.add(jPanel2,  BorderLayout.EAST);
jPanel4.add(etatRelaiLabel, null);
jPanel4.add(etatRelai, null);
jPanel2.add(jPanel4, null);
jPanel2.add(jPanel6, null);
jPanel6.add(nbConnexionsLabel, null);
jPanel6.add(nbConnexions, null);
jPanel2.add(jButton3, null);
jPanel2.add(jButton2, null);
jPanel2.add(jButton1, null);
this.getContentPane().add(jPanel3, null);
```

```
    jPanel3.add(jScrollPane2,  BorderLayout.CENTER);
    jScrollPane2.getViewport().add(journalRelai, null);
    jPanel3.add(jLabel2,  BorderLayout.NORTH);
  }

  void jButton1_mouseReleased(MouseEvent e) {
    journalRelai.setText("");
  }

  void jButton2_actionPerformed(ActionEvent e) {

  }

  void jButton3_actionPerformed(ActionEvent e) {

  }


}

class CadreInterface_jButton1_mouseAdapter extends java.awt.event.MouseAdapter {
  CadreInterface adaptee;

  CadreInterface_jButton1_mouseAdapter(CadreInterface adaptee) {
    this.adaptee = adaptee;
  }
  public void mouseReleased(MouseEvent e) {
    adaptee.jButton1_mouseReleased(e);
  }
}

class CadreInterface_jButton2_actionAdapter implements java.awt.event.ActionListener {
  CadreInterface adaptee;

  CadreInterface_jButton2_actionAdapter(CadreInterface adaptee) {
    this.adaptee = adaptee;
  }
  public void actionPerformed(ActionEvent e) {
    adaptee.jButton2_actionPerformed(e);
  }
}

class CadreInterface_jButton3_actionAdapter implements java.awt.event.ActionListener {
  CadreInterface adaptee;

  CadreInterface_jButton3_actionAdapter(CadreInterface adaptee) {
    this.adaptee = adaptee;
  }
  public void actionPerformed(ActionEvent e) {
    adaptee.jButton3_actionPerformed(e);
  }
}
```

Classe relaixmpp.Config

```
package relaixmpp;

/**
 * <p>Titre : Config</p>
 * <p>Description : Cette classe lit le fichier <i>config.xml</i> et charge les paramètres, les rendant accessibles aux
autres classes du programme.</p>
 * <p>Copyright : Copyright (c) 2004</p>
 * <p>Société : GETA-CLIPS</p>
 * @author Achille Falaise
 * @version 1.0
 */

import logxml.LogXML;
import java.util.Iterator;
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileNotFoundException;
import java.io.IOException;
import nanoxml.XMLElement;
import org.apache.regexp.RE;
import java.util.ArrayList;

public class Config {
  private ArrayList listeUsers;
  int delaiEcoute;
  int nbUsersInit;
  int portClient;
  int portServeur;
  boolean debugMode;
  boolean showInterface;
  String titreFenetre;
  int largeurFenetre;
  int hauteurFenetre;
  int delaiRafraichissement;
  ServiceRelai serviceRelai;
  CadreInterface cadreInterface;
  InterfaceManager interfaceManager;
  ArrayList lgDispo;
  String tempLogsRep;
  String archiveLogsRep;
  String tchatLogsRep;
  String imLogsRep;
  LogXML log;

  /**
   * Constructeur
   * @param fichier adresse du fichier XML de configuration
   */
  public Config(String fichier) {
    listeUsers = new ArrayList(nbUsersInit);
    log = new LogXML(this);
    try {
      // Ouverture
      BufferedReader fichierConf = new BufferedReader(new FileReader(fichier));
      // Traitement
      XMLElement racine = new XMLElement();
      racine.parseFromReader(fichierConf);
      parseRacine(racine);
      // Fermeture
      fichierConf.close();
    }
    catch(FileNotFoundException fnfe) {
      System.out.println("Impossible de trouver le fichier de config "+fichier+" ! Arrêt du programme.");
      interfaceManager.addJournalRelai(fnfe.getMessage());
      System.exit(-1);
    }
    catch(IOException ioe) {
      System.out.println("Erreur de lecture du fichier de config "+fichier+" ! Arrêt du programme.");
      interfaceManager.addJournalRelai(ioe.getMessage());
      System.exit(-1);
```

```
    }
  }

  /**
   * Parse le noeud racine du fichier de configuration.
   * @param noeudParent noeud racine du document XML
   */
  private void parseRacine(XMLElement noeudParent) {
    for(Iterator it = noeudParent.getChildren().iterator(); it.hasNext(); ) {
      XMLElement noeudFils = (XMLElement) it.next();
      String tagName = noeudFils.getName();
      String tagContent = noeudFils.getContent();
      if(tagName.equals("delaiecoute")) delaiEcoute = Integer.parseInt(tagContent);
      else if(tagName.equals("portclient")) portClient = Integer.parseInt(tagContent);
      else if(tagName.equals("portserveur")) portServeur = Integer.parseInt(tagContent);
      else if(tagName.equals("nbinterfinit")) nbUsersInit = Integer.parseInt(tagContent);
      else if(tagName.equals("debugmode")) debugMode = tagContent.equals("true")?true:false;
      else if(tagName.equals("interface")) parseInterface(noeudFils);
      else if(tagName.equals("languesdispo")) parseLanguesDispo(noeudFils);
      else if(tagName.equals("templogs")) tempLogsRep = tagContent;
      else if(tagName.equals("archivelogs")) archiveLogsRep = tagContent;
      else if(tagName.equals("tchatlogs")) tchatLogsRep = tagContent;
      else if(tagName.equals("imlogs")) imLogsRep = tagContent;
    }
  }

  /**
   * Parse les paramètres de l'interface fichier de configuration.
   * @param noeudParent noeud racine pour l'interface (balise <interface>)
   */
  private void parseInterface(XMLElement noeudParent) {
    for(Iterator it = noeudParent.getChildren().iterator(); it.hasNext(); ) {
      XMLElement noeudFils = (XMLElement) it.next();
      String tagName = noeudFils.getName();
      String tagContent = noeudFils.getContent();
      if(tagName.equals("titre")) titreFenetre = tagContent;
      else if(tagName.equals("showinterface")) showInterface = tagContent.equals("true")?true:false;
      else if(tagName.equals("largeur")) largeurFenetre = Integer.parseInt(tagContent);
      else if(tagName.equals("hauteur")) hauteurFenetre = Integer.parseInt(tagContent);
      else if(tagName.equals("rafraichir")) delaiRafraichissement = Integer.parseInt(tagContent);
    }
  }

  /**
   * Parse les couples de langues du fichier de configuration.
   * @param noeudParent noeud racine des couples de langues (balise <languesdispo>)
   */
  private void parseLanguesDispo(XMLElement noeudParent) {
    lgDispo = new ArrayList(10);
    for(Iterator it = noeudParent.getChildren().iterator(); it.hasNext(); ) {
      XMLElement noeudFils = (XMLElement) it.next();
      String tagName = noeudFils.getName();
      if(tagName.equals("couple")) {
        lgDispo.add(new String[] {(String)noeudFils.getAttribute("languesource"), (String)noeudFils.getAttribute
("languecible")});
      }
    }
  }

  /**
   * Ajoute un transfert à la table des clients connectés
   * @param t transfert ajouté
   */
  public synchronized void addUser(TransfertClientServeur t) {
    if(!listeUsers.contains(t)) listeUsers.add(t);
  }

  /**
   * Cherche dans la table des clients connectés
   * @param pseudo pseudo du client recherché
   * @return TransfertClientServeur correspondant à ce client
   */
```

```
  public synchronized TransfertClientServeur getUser(String pseudo) {
    for(Iterator it=listeUsers.iterator(); it.hasNext(); ) {
      TransfertClientServeur user = (TransfertClientServeur)it.next();
      if(pseudo.equals(user.pseudo)) return user;
    }
    return null;
  }

  /**
   * Enlève un transfert de la table des clients connectés
   * @param t transfert enlevé
   */
  public synchronized void removeUser(Transfert t) {
    listeUsers.remove(t);
  }

  /**
   * Retourne la liste des clients connectés
   * @return liste des clients connectés
   */
  public synchronized ArrayList getUsers() {
    return listeUsers;
  }

}
```

Classe relaixmpp.Ecoute

```
package relaixmpp;

/**
 * <p>Titre : Ecoute</p>
 * <p>Description : Thread de bas niveau qui récupère tout ce qui arrive sur un socket et le stocke dans un
tampon.</p>
 * <p>Copyright : Copyright (c) 2004</p>
 * <p>Société : GETA-CLIPS</p>
 * @author Achille Falaise
 * @version 1.0
 */

import java.io.BufferedReader;
import java.net.Socket;

public class Ecoute extends Thread {
  private BufferedReader reader;
  private String result = "";
  private Socket socket;

  /**
   * Constructeur
   * @param reader BufferedReader du socket à écouter
   */
  public Ecoute(BufferedReader reader, Socket s) {
    this.reader = reader;
    socket = s;
  }

  /**
   * Lance l'écoute (passer par la méthode start héritée de Thread)
   */
  public void run() {
    try {
      while(socket.isConnected()) {
        int c = reader.read();
        if (c != -1) {
          addChar((char) c);
        }
      }
      //reader.close();
    }
    catch (Exception ioe) {
      //ioe.printStackTrace();
    }
  }

  /**
   * Méthode synchronisée pour accéder au contenu du tampon, qui est ensuite réinitialisé
   * @return contenu du tampon
   */
  public synchronized String read() {
    String res=result;
    result = "";
    return res;
  }

  /**
   * Méthode synchronisée; ajoute un caractère au tampon
   * @param c caractère à ajouter
   */
  private synchronized void addChar(char c) {
    result += (char) c;
  }

}
```

Classe relaixmpp.InterfaceManager

```
package relaixmpp;

/**
 * <p>Titre : </p>
 * <p>Description : </p>
 * <p>Copyright : Copyright (c) 2004</p>
 * <p>Société : GETA-CLIPS</p>
 * @author Achille Falaise
 * @version 1.0
 */

import javax.swing.JPanel;
import javax.swing.JLabel;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import java.awt.Component;
import java.util.Iterator;
import javax.swing.*;
import java.awt.*;

public class InterfaceManager extends Thread {
  Config config;
  CadreInterface cadreInterface;

  public InterfaceManager(Config conf) {
    config = conf;
    cadreInterface = config.cadreInterface;
  }

  public void run() {
    while(true) {
      setPriority(Thread.NORM_PRIORITY);
      long debut = System.currentTimeMillis();
      boolean estModifie = false;
      ServiceRelai sr = config.serviceRelai;
      // Cadre Journal relai
      String etat = "";
      if (sr.isAlive()) {
        if (sr.isInterrupted()) etat = "Interrompu";
        else etat = "Actif";
      }
      else etat = "Inactif";
      cadreInterface.etatRelai.setText(etat);
      cadreInterface.nbConnexions.setText(Integer.toString(config.getUsers().size()));
      // Cadre Connexions
      Component[] listePanels = cadreInterface.connexionsPanel.getComponents();
      for (Iterator it = config.getUsers().iterator(); it.hasNext(); ) { // Pour chaque client
        TransfertClientServeur t = (TransfertClientServeur) it.next();
        if(t.pseudo != null && !t.pseudo.equals("")) {
          boolean sortir = false;
          for(int i = 0; i < listePanels.length && !sortir; ++i) { // Parcours des sous-panels
            if (getPseudoComponent(listePanels[i]).getText().equals(t.pseudo)) { // Oui => màj
              getPseudoComponent(listePanels[i]).setText(t.pseudo);
              getLangueComponent(listePanels[i]).setText('['+t.langue+']');
              getPortComponent(listePanels[i]).setText(Integer.toString(t.entree.getPort()));
              getPrioriteComponent(listePanels[i]).setText('('+Integer.toString(t.getPriority())+')');
              sortir = true;
            }
          }
          if(!sortir) { // Non => ajout
            JPanel nouveauPanel = new JPanel();
            cadreInterface.connexionsPanel.add(nouveauPanel, null);
            nouveauPanel.add(new JLabel(t.pseudo), null);
            nouveauPanel.add(new JLabel('['+t.langue+']'), null);
            nouveauPanel.add(new JLabel(Integer.toString(t.entree.getPort())), null);
            nouveauPanel.add(new JLabel('('+Integer.toString(t.getPriority())+')'), null);
            nouveauPanel.setBackground(Color.WHITE);
            nouveauPanel.setEnabled(true);
            estModifie = true;
          }
```

```
        }
      }
      // Supprimer les clients déconnectés
      for(int i = 0; i < listePanels.length; ++i) {
        boolean trouve = false;
        for(Iterator it = config.getUsers().iterator(); it.hasNext() && !trouve; ) {
          TransfertClientServeur t = (TransfertClientServeur) it.next();
          if(getPseudoComponent(listePanels[i]).getText().equals(t.pseudo)) trouve = true;
        }
        if(!trouve) {
          cadreInterface.connexionsPanel.remove(listePanels[i]);
          estModifie = true;
        }
      }
      if(estModifie) majAffichage();
      // Faire la sieste un moment
      long tps = config.delaiRafraichissement - (System.currentTimeMillis() - debut);
      setPriority(Thread.MIN_PRIORITY);
      if(tps > 0) {
        try {
          sleep(tps);
        }
        catch(InterruptedException ie) {System.out.println(ie.getStackTrace());}
      }
    }
  }

  private JLabel getPseudoComponent(Component p) {
    return (JLabel) ((JPanel)p).getComponent(0);
  }

  private JLabel getLangueComponent(Component p) {
    return (JLabel) ((JPanel)p).getComponent(1);
  }

  private JLabel getPortComponent(Component p) {
    return (JLabel) ((JPanel)p).getComponent(2);
  }

  private JLabel getPrioriteComponent(Component p) {
    return (JLabel) ((JPanel)p).getComponent(3);
  }

  private JTextArea getLogComponent(Component p) {
    return (JTextArea) ((JScrollPane)((JPanel)p).getComponent(4)).getComponent(0);
  }

  /**
   * Met à jour l'interface
   */
  public synchronized void majAffichage() {
    cadreInterface.repaint();
    cadreInterface.show();
  }

  /**
   * Ajoute un message au journal du relai
   * @param message message à ajouter
   */
  public synchronized void addJournalRelai(String message) {
    cadreInterface.journalRelai.setText(cadreInterface.journalRelai.getText()+"\n"+message);
    cadreInterface.journalRelai.setCaretPosition(cadreInterface.journalRelai.getText().length()-message.length()+1);
  }

}
```

Classe logxml.LogXML

```
package logxml;

/**
 * <p>Titre : XMLTools</p>
 * <p>Description : log d'un client de tchat</p>
 * <p>Copyright : Copyright (c) 2004</p>
 * <p>Société : GETA-CLIPS</p>
 * @author Achille Falaise
 * @version 1.0
 */

import relaixmpp.Config;
import relaixmpp.TransfertClientServeur;
import nanoxml.XMLElement;
import java.io.IOException;
import java.io.Writer;
import java.io.FileWriter;
import java.util.Date;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

public class LogXML {
  private Config config;

  /**
   * Constructeur
   * @param conf classe de configuration
   */
  public LogXML(Config conf) {
    config = conf;
  }

  /**
   * Changment de langue source
   * @param lg langue
   */
  public synchronized void setLangue(String langue, String pseudo) {
  }

  /**
   * Changement de langue(s) cible(s), données par ordre de préférence
   * @param liste liste des langues
   */
  public synchronized void setLanguesPreferees(ArrayList listeLangues, String pseudo) {
  }

  /**
   * Montrer le message original ? (non traduit)
   * @param bool
   */
  public synchronized void setShowOriginal(boolean bool, String pseudo) {
  }

  /**
   * Montrer le message traduit ?
   * @param bool
   */
  public synchronized void setShowTranslated(boolean bool, String pseudo) {
  }

  public synchronized void setMsg(String message, String langue, String from, String salle, boolean isIM) {
  }

  public synchronized void setVersion(String message, String langue, String from, String to, String salle, boolean isIM) {
  }

  public synchronized void setClientConnexion(String pseudo, TransfertClientServeur transf) {
  }
```

```
  public synchronized void setClientDeconnexion(String pseudo, TransfertClientServeur transf) {
  }

  private void ouvertureConversation(String salle) {
  }

  private void fermetureConversation(String salle) {
  }

  private synchronized void sauvegardeConversation() {
    Date date = new Date(heureConnexion);
    String nomFichier = config.tempLogsRep+pseudo+"@"+date.getYear()+"-"+date.getMonth()+"-"+date.getDay()
+"-"+date.getHours()+"-"+date.getMinutes()+"-"+date.getSeconds()+"@clientlog.xml";
    try {
      Writer output = new FileWriter(nomFichier);
      output.write(toString());
      output.flush();
    }
    catch(IOException ioe) {
      ioe.printStackTrace();
      config.interfaceManager.addJournalRelai(ioe.getMessage());
    }
  }

 }
```

Classe relaixmpp.ServiceRelai

```
package relaixmpp;

/**
 * <p>Titre : ServiceRelai</p>
 * <p>Description : Classe de base de l'application. Crée un relai entre un serveur et un nombre variable de clients,
utilisant le protocole XMPP.</p>
 * <p>Copyright : Copyright (c) 2004</p>
 * <p>Société : GETA-CLIPS</p>
 * @author Achille Falaise
 * @version 1.0
 */

import java.io.IOException;
import java.net.Socket;
import java.net.ServerSocket;
import logxml.LogXML;

public class ServiceRelai extends Thread implements Runnable {
  private int portEntrant;
  private int portSortant;
  private ServerSocket service;
  private boolean debug;
  private Config config;

  /**
   * Constructeur
   * @param fichierConf adresse du fichier de configuration
   */
  public ServiceRelai(String fichierConf) {
    config = new Config(fichierConf);
    setPriority(Thread.MIN_PRIORITY+1);
    config.serviceRelai = this;
    debug = config.debugMode;
    portEntrant = config.portClient;
    portSortant = config.portServeur;
  }

  /**
   * Lance et gère le service
   */
  public void run() {
    try {
      if(debug) System.out.println("J'essaye d'écouter sur le port "+portEntrant);
      service = new ServerSocket(portEntrant);
      if(debug) System.out.println("J'écoute sur le port "+portEntrant);
      if(config.cadreInterface!=null) config.interfaceManager.addJournalRelai("Service lancé sur le port "+portEntrant);
      while(true) {
        TransfertServeurClient t1;
        TransfertClientServeur t2;
        Socket entree = service.accept(); // On attend qu'un client se connecte, et on récupère le socket
        Socket sortie = new Socket("localhost", portSortant); // On crée un socket vers le serveur
        t1 = new TransfertServeurClient(sortie, entree, config);
        t2 = new TransfertClientServeur(entree, sortie, config);
        t1.setJumeau(t2);
        t2.setJumeau(t1);
        //#LogXML log = new LogXML(config);
        //#t1.setLog(log);
        //#t2.setLog(log);
        t1.start();
        t2.start();
        if(config.cadreInterface!=null) config.interfaceManager.addJournalRelai("Un client vient de se connecter.");
      }
    } catch(IOException ioe) {ioe.printStackTrace();}
  }

  /**
   * Lance l'appli
   * @param args aucun paramètre n'est requis
   */
  public static void main(String[] args) {
```

```
    ServiceRelai sr = new ServiceRelai("config.xml");
    sr.config.cadreInterface = new CadreInterface(sr.config);
    sr.config.interfaceManager = new InterfaceManager(sr.config);
    sr.start();
    sr.config.interfaceManager.start();
  }

}
```

Classe relaixmpp.Transfert

```
package relaixmpp;

/**
 * <p>Titre : Transfert</p>
 * <p>Description : Thread abstrait, pour gérer les connexions clients-serveur et serveur-clients.</p>
 * <p>Copyright : Copyright (c) 2004</p>
 * <p>Société : GETA-CLIPS</p>
 * @author Achille Falaise
 * @version 1.0
 */

import java.io.PrintWriter;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.Socket;
import org.apache.regexp.RE;

public abstract class Transfert extends Thread {
  Ecoute fluxEntree;
  PrintWriter fluxSortie;
  boolean debug;
  Socket entree;
  Socket sortie;
  Config config;
  RE getMsgFromRegex1; // chat (=im)
  RE getMsgFromRegex2; // groupchat (=tchat)
  RE isMsgType1;
  RE isMsgType2;
  RE getMsgToRegex;
  RE getMsgRegex;
  RE getPseudoRegex;
  RE getLangueRegex;
  RE getLanguePrefRegex;
  RE getDefLanguePrefRegex;
  RE getSoRegex;
  RE getStRegex;
  RE finConnexionRegex;
  RE getInitmsgRegex;
  RE getInitmsgFromRegex;
  RE getInitmsgToRegex1; // chat (=im)
  RE getInitmsgToRegex2; // groupchat (=tchat)

  /**
   * Constructeur
   * @param s1 premier socket
   * @param s2 second socket
   * @param conf
   */
  public Transfert(Socket s1, Socket s2, Config conf) {
    setPriority(MIN_PRIORITY+2);
    config = conf;
    debug = config.debugMode;
    entree = s1;
    sortie = s2;
    isMsgType1 = new RE(" type=[\\\"']chat[\\\"']");
    isMsgType2 = new RE(" type=[\\\"']groupchat[\\\"']");
    getMsgRegex = new RE("(<message ([^>]+)>.*<body[^>]*>)(.*)(</body>.*</message>)",
RE.MATCH_CASEINDEPENDENT);
    finConnexionRegex = new RE("</stream:stream>", RE.MATCH_CASEINDEPENDENT);
    getLangueRegex = new RE("<vcard[^>]*>.*<desc[^>]*>.*\\[lg:([a-z][a-z])\\].*</desc>.*</vcard>",
RE.MATCH_CASEINDEPENDENT);
    getLanguePrefRegex = new RE("\\s?\\[lg([0-9]+):([a-z][a-z])\\]\\s?", RE.MATCH_CASEINDEPENDENT);
    getDefLanguePrefRegex = new RE("<vcard[^>]*>.*<desc[^>]*>(.*\\[lg[0-9]+:[a-z][a-z]\\].*)
</desc>.*</vcard>", RE.MATCH_CASEINDEPENDENT);
    getSoRegex = new RE("<vcard[^>]*>.*<desc[^>]*>.*\\[so:([01])\\].*</desc>.*</vcard>",
RE.MATCH_CASEINDEPENDENT);
    getStRegex = new RE("<vcard[^>]*>.*<desc[^>]*>.*\\[st:([01])\\].*</desc>.*</vcard>",
RE.MATCH_CASEINDEPENDENT);
    getMsgFromRegex1 = new RE("from=[\\\"'][^@]+)@[^\\\"']+[\\\"']");
    getMsgFromRegex2 = new RE("from=[\\\"'][^@]+)@[^/]+/([^\\\"']+)[\\\"']");
```

```
    getMsgToRegex = new RE("to=[\\\"']([^@]+)@[^\\\"']+[\\\"']");
    getPseudoRegex = new RE("<iq id='[^']+' type='result'><bind xmlns='urn:ietf:params:xml:ns:xmpp-bind'><jid>
([^@]+)@[^<]+</jid></bind></iq>");
    getInitmsgRegex = new RE("(<message ([^>]+)>.*<body[^>]*>)(.*)(</body>.*</message>)",
RE.MATCH_CASEINDEPENDENT);
    getInitmsgToRegex1 = new RE("to=[\\\"']([^@]+)@[^\\\"']+[\\\"']");
    getInitmsgToRegex2 = new RE("to=[\\\"']([^@]+)@[^/]+/([^\\\"']+)[\\\"']");
    getInitmsgFromRegex = new RE("from=[\\\"']([^@]+)@[^\\\"']+[\\\"']");
  }

  /**
   * Retourne un transfert à partir de son pseudo
   * @param pseudo pseudo à chercher
   * @return transfert correspondant
   */
  public synchronized Transfert getUser(String pseudo) {
    for(Iterator it = jumeau.listeUsers.iterator(); it.hasNext(); ) {
      Transfert t = (Transfert) it.next();
      if(t.pseudo!=null && t.pseudo.equals(pseudo)) return t;
      else if(debugMode) System.out.println(" Config.getUser(): "+pseudo+" != "+t.pseudo);
    }
    return null;
  }

}
```

Classe relaixmpp.TransfertClientServeur

```
package relaixmpp;

/**
 * <p>Titre : TransfertClientServeur</p>
 * <p>Description : Cette classe hérite de Transfert, et gère les connexions client-serveur</p>
 * <p>Copyright : Copyright (c) 2004</p>
 * <p>Société : GETA-CLIPS</p>
 * @author Achille Falaise
 * @version 1.0
 */

import java.io.PrintWriter;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.Socket;
import org.apache.regexp.RE;
import java.util.ArrayList;
import java.io.IOException;

public class TransfertClientServeur extends Transfert {
  TransfertServeurClient jumeau;
  String pseudo;
  String langue;

  /**
   * Constructeur
   * @param s1 Socket client-relai
   * @param s2 Socket relai-serveur
   * @param config
   */
  public TransfertClientServeur(Socket s1, Socket s2, Config config) {
    super(s1, s2, config);
    config.addUser(this);
    //config.log.setClientConnexion(jumeau.pseudo, this); // [LOG] Notifier la connexion du client
  }

  /**
   * Méthode pour le traitement des messages que le client envoie au serveur
   * @param message message que le client tente d'envoyer au serveur
   * @return message effectivement envoyé au serveur
   */
  private String traitementMessage(String message) {
    if(getLangueRegex.match(message)) {
      jumeau.langue = getLangueRegex.getParen(1);
      config.log.setLangue(jumeau.langue, jumeau.pseudo); // [LOG] Notifier le paramétrage de la langue
      if(debug) System.out.println(" Langue: "+jumeau.langue);
    }
    if(getDefLanguePrefRegex.match(message)) {
      jumeau.lgPref.clear();
      int startNextMatch= 0;
      while(getLanguePrefRegex.match(message, startNextMatch)) {
        int index = Integer.parseInt(getLanguePrefRegex.getParen(1));
        String languePref = getLanguePrefRegex.getParen(2);
        synchronized(jumeau.lgPref) {
          jumeau.lgPref.add(languePref);
        }
        if(debug) System.out.println(" Langue préférée: "+languePref);
        startNextMatch = getLanguePrefRegex.getParenEnd(2);
      }
      config.log.setLanguesPreferees(jumeau.lgPref, jumeau.pseudo); // [LOG] Notifier le paramétrage des langues
préférées
    }
    if(getStRegex.match(message)) {
      jumeau.st = getStRegex.getParen(1).equals("1");
      config.log.setShowTranslated(jumeau.st, jumeau.pseudo); // [LOG] Notifier le paramétrage de la visibilité de la
traduction
    }
    if(getSoRegex.match(message)) {
      jumeau.so = getSoRegex.getParen(1).equals("1");
      config.log.setShowOriginal(jumeau.so, jumeau.pseudo); // [LOG] Notifier le paramétrage de la visibilité du message
```

```
original (=non-traduit)
      }
      if(getInitmsgRegex.match(message) && (isMsgType1.match(getInitmsgRegex.getParen(2)) || isMsgType2.match
(getInitmsgRegex.getParen(2)))) { // Capture du msg à l'envoi
        String nomSalleTchat = null;
        String to = null;
        boolean isIM = false;
        if(isMsgType1.match(getInitmsgRegex.getParen(2))) { // IM
          to = getInitmsgToRegex1.getParen(1);
          nomSalleTchat = jumeau.pseudo + "#" + to; // Génération d'un pseudo nom de salle unique pour servir de clef
dans LogXML
          isIM = true;
        }
        else if(isMsgType2.match(getInitmsgRegex.getParen(2))) { // Groupchat
          to = getInitmsgToRegex2.getParen(2);
          nomSalleTchat = getInitmsgToRegex2.getParen(1);
        }
        String msg = getMsgRegex.getParen(3);
        config.log.setMsg(msg, jumeau.langue, jumeau.pseudo, nomSalleTchat, isIM); // [LOG] Notifier l'envoi d'un
message par le client
      }
      return message;
    }

    /**
     * Crée un lien vers le TransfertServeurClient gérant les connexions en sens inverse
     * @param j TransfertServeurClient gérant les connexions serveur-client
     */
    protected synchronized void setJumeau(TransfertServeurClient j) {
      jumeau = j;
      pseudo = jumeau.pseudo;
      langue = jumeau.langue;
    }

    /**
     * Lance le thread (passer par la méthode start() héritée de Thread)
     */
    public void run() {
      // Transfert
      try {
        fluxEntree = new Ecoute(new BufferedReader(new InputStreamReader(entree.getInputStream())), entree);
        fluxSortie = new PrintWriter(sortie.getOutputStream(), true);
        String message = "";
        long lastRead = System.currentTimeMillis();
        fluxEntree.start();
        while(entree.isConnected() && sortie.isConnected() && !jumeau.estFini && fluxEntree.isAlive()) {
          String nouv = fluxEntree.read(); // Ecouter
          if(nouv.length() > 0) {
            message += nouv;
            lastRead = System.currentTimeMillis();
          }
          if((message.endsWith(">")||message.endsWith("\n")) && System.currentTimeMillis() > lastRead +
config.delaiEcoute && message.length() > 0) { // Fin de la requête
            if(debug) System.out.println(entree.toString() + " [" + message + "]");
            config.interfaceManager.addJournalRelai(jumeau.pseudo+"("+entree.getPort()+")->S: "+ message);
            int antePriorite = getPriority();
            setPriority(antePriorite+1);
            if(finConnexionRegex.match(message)) break;
            fluxSortie.write(traitementMessage(message));
            fluxSortie.flush();
            message = "";
            setPriority(antePriorite);
          }
        }
      }
      catch (Exception e) {
        System.out.println(entree.toString());
        e.printStackTrace();
      }
      finally {
        end();
      }
```

```
  }

  /**
   * Méthode appelée à la fin du processus
   */
  private void end() {
    jumeau.estFini = true;
    config.removeUser(this);
    if (debug) System.out.println(entree.toString() + " Vidage du tampon");
    fluxSortie.flush(); // Vider le tampon
    try {
      entree.close();
      sortie.close();
    } catch(IOException ioe) {ioe.printStackTrace();}
    config.log.setClientDeconnexion(jumeau.pseudo, this); // [LOG] Notifier la déconnexion
    config.interfaceManager.addJournalRelai("Un client vient de se déconnecter.");
  }

}
```

Classe relaixmpp.TransfertServeurClient

```
package relaixmpp;

/**
 * <p>Titre : TransfertServeurClient</p>
 * <p>Description : Cette classe hérite de Transfert, et gère les connexions serveur-client</p></p>
 * <p>Copyright : Copyright (c) 2004</p>
 * <p>Société : GETA-CLIPS</p>
 * @author Achille Falaise
 * @version 1.0
 */

import java.io.PrintWriter;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.Socket;
import org.apache.regexp.RE;
import java.io.IOException;
import java.util.ArrayList;

public class TransfertServeurClient extends Transfert {
  TransfertClientServeur jumeau;
  boolean so = true;
  boolean st = false;
  ArrayList lgPref;
  boolean estFini = false;
  String langue;
  String pseudo = new String();

  /**
   * Constructeur
   * @param s1 Socket serveur-relai
   * @param s2 Socket relai-client
   * @param config
   */
  public TransfertServeurClient(Socket s1, Socket s2, Config config) {
    super(s1, s2, config);
    lgPref = new ArrayList(2);
  }

  /**
   * Méthode pour le traitement des messages que le serveur envoie au client
   * @param message message que le serveur tente d'envoyer au client
   * @return message effectivement envoyé au client
   */
  private String traitementMessage(String message) {
    int prevPriority = getPriority();
    setPriority(Thread.NORM_PRIORITY-1);
    if(getPseudoRegex.match(message)) { // Capture du pseudo
      pseudo = getPseudoRegex.getParen(1);
      if(debug) System.out.println(" Pseudo: "+pseudo);
    }
    if(debug && getMsgRegex.match(message)) { // [DEBUG] Si un message est capturé, indique si ce dernier est valide
      boolean contientMsg = getMsgRegex.match(message);
      System.out.println(" getMessage: "+contientMsg);
      if(contientMsg) {
        if(isMsgType1.match(getMsgRegex.getParen(2))) System.out.println(" getFrom(1): "+getMsgFromRegex1.match
(getMsgRegex.getParen(2)));
        else if(isMsgType2.match(getMsgRegex.getParen(2))) System.out.println(" getFrom(2):
"+getMsgFromRegex2.match(getMsgRegex.getParen(2)));
        System.out.println(" getTo: "+getMsgToRegex.match(getMsgRegex.getParen(2)));
      }
    }
    if(getMsgRegex.match(message) && (isMsgType1.match(getMsgRegex.getParen(2)) || isMsgType2.match
(getMsgRegex.getParen(2))) && getMsgToRegex.match(getMsgRegex.getParen(2))) { // Capture du msg et traduction
      String nomSalleTchat = null;
      String from = null;
      boolean isIM = false;
      if(isMsgType1.match(getMsgRegex.getParen(2))) { // IM
        from = getMsgFromRegex1.getParen(1);
        nomSalleTchat = pseudo + "#" + from; // Génération d'un pseudo nom de salle unique pour servir de clef dans
```

```
LogXML
        isIM = true;
       }
       else if(isMsgType2.match(getMsgRegex.getParen(2))) { // Groupchat
        from = getMsgFromRegex2.getParen(2);
        nomSalleTchat = getMsgFromRegex2.getParen(1);
       }
       if(debug) System.out.println(" Interlocuteur: "+from);
       TransfertClientServeur interlocuteur = (TransfertClientServeur)config.getUser(from);
       if(debug) System.out.println(" "+interlocuteur);
       if(interlocuteur==null) return message;
       String msg = getMsgRegex.getParen(3);
       String langueSource = interlocuteur.jumeau.langue;
       String langueCible = getLangueDispo(langueSource);
       if(debug) System.out.println(" "+langueSource+" > "+langueCible);
       String messageOriginal = getMsgRegex.getParen(1) + "["+ langueSource +"] <i>" + msg + "</i>" +
getMsgRegex.getParen(4);
       if(!st || langueCible==null) { // Pas possible ou pas nécessaire de faire la traduction
        config.log.setVersion(msg, langueSource, from, getMsgToRegex.getParen(1), nomSalleTchat, isIM);
        return messageOriginal;
       }
       // Traduction
       if(debug) System.out.println("Source du message: " + from + " " + langueCible);
       String msgTraduit = new Translator(langueSource, langueCible).translate(msg);
       String messageTraduit = getMsgRegex.getParen(1) + "["+ langueCible +"] " + msgTraduit +
getMsgRegex.getParen(4);
       if(debug) System.out.println("Message traduit: " + msgTraduit);
       message = ((so && msg!=msgTraduit)?messageOriginal:"") + (st?messageTraduit:""); // Formatage du message à
retourner
       config.log.setVersion(msgTraduit, langueCible, from, getMsgToRegex.getParen(1), nomSalleTchat, isIM);
      }
      setPriority(prevPriority);
      return message;
    }

    /**
     * Pour savoir si l'on peut traduire cette langue en quelque chose que l'utilisateur pourra comprendre
     * @param langueSource Langue source
     * @return Retourne la première langue préférée que l'on peut obtenir à partir de la langue source
     */
    private String getLangueDispo(String langueSource) {
      synchronized(lgPref) {
        for(int i = 0; i<lgPref.size(); ++i) {
          // Est ce que, pour chaque langue préférée, la langue source peut être traduite vers la langue préférée
          String languePreferee = (String)lgPref.get(i);
          for(int j = 0; j<config.lgDispo.size(); ++j)
            if(((String[])config.lgDispo.get(j))[0].equals(langueSource) && ((String[])config.lgDispo.get(j))[1].equals
(languePreferee))
              return languePreferee;
        }
      }
      return null;
    }

    /**
     * Crée un lien vers le TransfertClientServeur gérant les connexions en sens inverse
     * @param j TransfertClientServeur gérant les connexions client-serveur
     */
    protected synchronized void setJumeau(TransfertClientServeur j) {
      jumeau = j;
    }

    /**
     * Lance le thread (passer par la méthode start() héritée de Thread)
     */
    public void run() {
      // Transfert
      try {
        fluxEntree = new Ecoute(new BufferedReader(new InputStreamReader(entree.getInputStream())), entree);
        fluxSortie = new PrintWriter(sortie.getOutputStream(), true);
        String message = "";
        long lastRead = System.currentTimeMillis();
```

```
      fluxEntree.start();
      while(entree.isConnected() && sortie.isConnected() && !estFini && fluxEntree.isAlive()) {
        String nouv = fluxEntree.read(); // Ecouter
        if(nouv.length() > 0) {
          message += nouv;
          lastRead = System.currentTimeMillis();
        }
        if(message.endsWith(">") && System.currentTimeMillis() > lastRead + config.delaiEcoute && message.length() >
0) { // Fin de la requête
          if(debug) System.out.println(entree.toString() + " [" + message + "]");
          config.interfaceManager.addJournalRelai("S->"+jumeau.pseudo+"("+sortie.getPort()+"): "+ message);
          int antePriorite = getPriority();
          setPriority(antePriorite+1);
          if(finConnexionRegex.match(message)) break;
          fluxSortie.write(traitementMessage(message));
          fluxSortie.flush();
          message = "";
          setPriority(antePriorite);
        }
      }
    }
    catch (Exception e) {
      System.out.println(entree.toString());
      e.printStackTrace();
    }
    estFini = true;
    if(debug) System.out.println(entree.toString() + " Vidage du tampon");
    fluxSortie.flush(); // Vider le tampon
    try {
      entree.close();
      sortie.close();
    } catch(IOException ioe) {ioe.printStackTrace();}
  }

}
```

Classe relaixmpp.Translator

```
package relaixmpp;

/**
 * <p>Titre : Translator</p>
 * <p>Description : Classe pour la traduction, gère la connexion avec les serveurs de traduction et le formatage de la requête et des résultats</p>
 * <p>Copyright : Copyright (c) 2004</p>
 * <p>Société : GETA-CLIPS</p>
 * @author Achille Falaise
 * @version 1.0
 */

import java.net.Socket;
import java.io.PrintWriter;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.UnknownHostException;
import java.net.ConnectException;

public class Translator {
  private String langueSource;
  private String langueCible;

  /**
   * Constructeur
   * @param langueSource code langue source (ex. en, fr...)
   * @param langueCible code langue cible (ex. en, fr...)
   */
  public Translator(String langueSource, String langueCible) {
    this.langueSource = langueSource;
    this.langueCible = langueCible;
  }

  /**
   * Traduction
   * @param source message à traduire
   * @return message traduit
   */
  public synchronized String translate(String source) {
    HTTPRequest httpr = new HTTPRequest("http://translate.google.com/translate_t", "translate.google.com", "text=" + source.replaceAll(" ", "+") + "&langpair=" + langueSource + "|" + langueCible + "&hl=fr&ie=utf8&oe=utf8\r\n");
    httpr.start();
    while (httpr.isAlive()); // Attendre la fin de la traduction
    String res = httpr.getResult();
    String startTag = "<textarea name=q rows=5 cols=45 wrap=PHYSICAL>";
    int db = res.indexOf(startTag);
    int fn = res.indexOf("</textarea>", db);
    if (db == -1 || fn == -1)
      return source;
    return res.substring(db + startTag.length(), fn);
  }

  /**
   * <p>Titre : HTTPRequest</p>
   * <p>Description : Thread pour Gérer les requêtes HTTP.</p>
   * <p>Copyright : Copyright (c) 2004</p>
   * <p>Société : GETA-CLIPS</p>
   * @author Achille Falaise
   * @version 1.0
   */
  public class HTTPRequest extends Thread {
    private String requete;
    private String url;
    private String host;
    private String result = "";

    /**
     * Constructeur
     * @param url URL où envoyer la requête
     * @param host hôte à qui envoyer la requête
```

```
     * @param body corps de la requête
     */
    public HTTPRequest(String url, String host, String body) {
      this.url = url;
      this.host = host;
      String content = body;
      requete = "POST " + url + " HTTP/1.0\r\n";
      requete += "Host: " + host + "\r\n";
      requete += "Content-Type: application/x-www-form-urlencoded\r\n";
      requete += "Content-Length: "+content.length()+"\r\n\r\n";
      requete += content;
    }

    /**
     * Lance la connexion avec le serveur et attend la réponse
     */
    public void run() {
      try {
        // Connexion
        Socket connexion = new Socket(host, 80);
        // Initialisation
        PrintWriter sortie = new PrintWriter(connexion.getOutputStream(), true);
        BufferedReader entree = new BufferedReader(new InputStreamReader(connexion.getInputStream()));
        sortie.println(requete);
        // Ecoute
        String ligne = "";
        while(connexion.isConnected() && !ligne.matches("</body>")) {
          ligne = entree.readLine();
          result+=ligne;
        }
        // Fermeture de la connexion
        connexion.close();
      }
      catch(UnknownHostException uhe) { // Hôte inconnu
        System.out.println("Impossible de trouver l'hôte.");
      }
      catch(ConnectException ce) { // Impossible d'établir la connexion
        System.out.println("Impossible d'établir la connexion. "+ce.getLocalizedMessage());
      }
      catch(Exception e) {
        System.out.println(e.getStackTrace());
      }
    }

    /**
     * Retourne le résultat de la traduction.
     * @return
     */
    public String getResult() {
      return result;
    }

  }

}
```

Fichier de configuration

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<relaixmpp>
 <debugmode>false</debugmode>
 <!-- Couples de langues disponibles -->
 <languesdispo>
  <couple languesource="de" languecible="en"/>
  <couple languesource="de" languecible="fr"/>
  <couple languesource="es" languecible="en"/>
  <couple languesource="en" languecible="de"/>
  <couple languesource="en" languecible="es"/>
  <couple languesource="en" languecible="fr"/>
  <couple languesource="en" languecible="it"/>
  <couple languesource="en" languecible="pt"/>
  <couple languesource="fr" languecible="en"/>
  <couple languesource="fr" languecible="de"/>
  <couple languesource="it" languecible="en"/>
  <couple languesource="pt" languecible="en"/>
 </languesdispo>
 <!-- Configuration de l'interface -->
 <interface>
  <showinterface>true</showinterface>
  <titre>Relai XMPP</titre>
  <rafraichir>500</rafraichir> <!-- Rafraichir l'affichage toutes les X ms -->
  <!-- Dimensions de la fenêtre -->
  <largeur>600</largeur>
  <hauteur>400</hauteur>
 </interface>
 <!-- Répertoires de log -->
 <templogs>./logs/temp/</templogs> <!-- Logs bruts, non-traités -->
 <archivelogs>./logs/archive/</archivelogs> <!-- Logs traités -->
 <tchatlogs>./logs/tchat/</tchatlogs> <!-- Après traitement, logs de tchat -->
 <imlogs>./logs/im/</imlogs> <!-- Après traitement, logs de messagerie instantannée (IM) -->
 <portclient>5222</portclient>
 <portserveur>5225</portserveur>
 <nbtransfinit>10</nbtransfinit> <!-- Nombre de transferts initial pour le vecteur de transferts actifs -->
 <delaiecoute>5</delaiecoute> <!-- Nombre de ms d'attente entre le dernier octet reçu du client ou du serveur, et le
traitement du message -->
</relaixmpp>
```