# Apport de la Traduction Automatique au développement d'une Interface Homme-Machine

**Pierre-André Buvet**
LDI-CNRS, Université Paris 13
**Laurent Tromeur**
LDI-CNRS, Université Paris 13 - Société Ontomantics

**Synergies** *Tunisie* n° 2 - 2010 pp. 125-135

**Résumé :** *L'article porte sur une interface homme-machine conçue pour une plateforme de génération automatique d'applications informatiques. La spécificité de cette interface est de fonctionner selon les principes combinés de la traduction automatique et la génération automatique compte tenu du fait que ce ne sont pas deux langues naturelles qui sont mises en regard mais une langue naturelle et un langage machine. Après avoir rappelé les principales caractéristiques des IHM existantes, nous précisons quelles particularités des systèmes de TA et de GA sont prises en compte dans l'IHM que nous avons conçue puis nous présentons les spécificités du module qui permet le transfert d'instructions formulées en langue naturelle en instructions spécifiées en langage machine et, ensuite nous présentons un cas concret qui illustre le fonctionnement du module.*

**Mots-clés :** *interface homme machine, traduction automatique, dictionnaire électronique, grammaire locale.*

**Abstract :** *The article is about a human-machine interface conceived for a platform of automatic generation of IT applications. The specificity of this interface is to work according to the principles of the machine translation and the automatic generation; it is not two natural languages which are translated but a natural language and a machine language. We remind the main characteristics of the existing human-machine interfaces, then we specify what are the peculiarities of our human-machine interface and we present the specificities of the module which allows the transfer of instructions formulated in natural language in instructions specified in machine language. We present also a concrete case which illustrates the functioning of the module.*

**Keywords :** *human-machine interface, machine translation, electronic dictionary, local grammar.*

## 0. Introduction

Dans le cadre du Traitement Automatique des Langues (TAL), une Interface Homme-Machine (IHM) se conçoit comme un outil qui permet, d'une part, de paramétrer une application à partir des instructions de l'utilisateur exprimées en langue naturelle et, d'autre part, de fournir des messages de la machine également formulés en langue naturelle. Sur le plan théorique, une IHM simule la communication humaine, l'un des deux interlocuteurs étant un système informatique. C'est pourquoi sa conception relève

souvent du champ de l'Intelligence Artificielle et consiste notamment à modéliser des situations de dialogue. La modélisation linguistique est généralement limitée à sa composante pragmatique, les autres composantes étant prises en charge par des analyseurs syntaxiques à large couverture et par des ontologies. La plupart des IHM dépendent de leur contexte applicatif et autorisent un nombre restreint d'interactions entre l'utilisateur et l'application. La complexité langagière explique les performances limitées des IHM ; elle est largement sous-estimée dans les modèles linguistiques pris en compte par les prototypes ou les systèmes développés.

Nous discutons ici du développement d'une IHM dont le contexte d'utilisation est également restreint mais qui se caractérise par son architecture. Les interactions entre l'utilisateur et le système informatique sont prises en charge de deux façons différentes selon que les messages émanent de l'homme ou de la machine. Dans le premier cas, les messages sont traités par un module qui fonctionne selon les principes de la traduction automatique (TA) et, dans le second cas, par un module qui fonctionne selon les principes de la génération automatique (GA). Le premier module est un système de transfert entre la langue naturelle et la langue machine alors que le second est un système créateur de messages en langue naturelle qui prend appui sur des éléments du langage machine produit par le processus de transfert.

Dans un premier temps, nous rappelons les principales caractéristiques des IHM existantes et nous précisons quelles particularités des systèmes de TA et de GA sont prises en compte dans l'IHM étudiée ici. Dans un deuxième temps, nous présentons les spécificités du module qui permet le transfert d'instructions formulées en langue naturelle en instructions spécifiées en langage machine. Dans un troisième temps, nous présentons un cas concret qui illustre le fonctionnement du module.

## 1. IHM, TA et GA

La modélisation du dialogue est généralement une problématique majeure pour la conception des IHM car il faut impérativement préciser la façon dont est structuré l'échange d'informations entre un usager et un système informatique (Pierrel : 1991). Nous évoquons ici les principaux modèles élaborés jusqu'à présent. Nous discutons ensuite des spécificités de la TA et de la GA qui caractérisent le traitement de l'échange d'informations dans l'IHM développée pour la plateforme Ontomantics.

### 1.1. La modélisation du dialogue dans les IHM

Idéalement, l'utilisateur d'une application informatique intégrant une IHM n'a pas à se soucier de la façon dont il s'exprime car le système informatique est censé analyser ses besoins quelle que soit leur formulation. Le système est alors caractérisé par sa flexibilité et sa gestion de toutes sortes de situations ; il est notamment capable de traiter les fautes de frappe, d'orthographe ou de diction, d'identifier les chaînes de référence dans le texte ou de traiter les informations implicites. Ces conditions sont indépendantes des situations de communication car le fait d'être limité à un domaine spécifique ne réduit pas le nombre de phénomènes linguistiques à traiter (Lerat : 1995).

Pratiquement, la plupart des systèmes fonctionnent de la manière suivante : dans un premier temps, l'utilisateur formule une instruction qui est analysée puis représentée

symboliquement ; dans un deuxième temps, la représentation de la demande est associée à une réponse également représentée symboliquement ; dans un troisième temps, la représentation de la réponse est transformée en texte que l'utilisateur peut lire ou entendre. Les réponses produites par les systèmes sont généralement des textes prélevés dans un corpus de phrases préenregistrées intégrant des variables qui sont spécifiées en fonction de chaque situation de communication (Weizenbaum : 1966).

Récemment, les recherches ont porté sur la façon de modéliser la composante lexico-syntaxique des échanges d'informations. Nous discutons de deux systèmes qui reflètent les travaux en cours sur les IHM.

Le système G-TAG est fondé sur un modèle grammatical lexicalisé exploitant une grammaire d'arbres adjoints, dite TAG (Tree-Adjoining Grammar) (Danlos : 1998). Les informations échangées sont associées à des représentations conceptuelles, annotées d'informations pragmatiques qui sont traitées par une interface sémantique telle que les concepts sont associés à une base de données lexicales correspondant à une grammaire TAG. Pour chaque concept, les lexèmes correspondant sont enregistrés avec leur structure argumentale ; on enregistre également les correspondances entre les arguments conceptuels et les arguments sémantiques.

Le système COALA (Co Adaptation Langagière pour l'Apprentissage) est basé sur un modèle calculatoire et dynamique ; il est indépendant du champ d'application (Coala : 1997). Il permet l'acquisition de connaissances en rapport avec la compétence langagière de la machine. Cette approche est centrée sur les méta-connaissances qui permettent d'acquérir des connaissances langagières. À l'inverse des méthodes qui exploitent un lexique et un répertoire de formulations pour les injecter dans le système, il s'agit de doter le système d'une capacité à construire son propre lexique et sa propre syntaxe tout en dialoguant. L'apprentissage consiste à exploiter les expressions et les termes réellement employés pendant les échanges d'information afin de les structurer *in situ* et non *a posteriori*.

La plupart des IHM ont des performances limitées qui ne permettent de réaliser qu'un nombre restreint d'actions. Malgré les nombreuses avancées réalisées dans le domaine, les contraintes qui subsistent, essentiellement de nature linguistique, ne permettent pas à l'utilisateur d'échanger des informations formulées en langue naturelle avec un système. Pour intégrer, une IHM à l'application Ontomantics, nous avons opté pour une approche différente de celles qui cherchent à modéliser le dialogue. Nous avons décomposé l'échange d'informations selon que les informations sont émises par l'utilisateur ou qu'elles proviennent du système informatique. Dans le premier cas, leur traitement procède de la TA alors que dans le second il relève de la GA.

**1.2. TA et IHM**

Selon Boitet (2007), il y a quatre types de méthodes en traduction automatique : la traduction automatique par pivot interlingue, la traduction automatique procédurale, la traduction automatique experte et la traduction automatique empirique.

La traduction automatique par pivot interlingue concerne les systèmes fondés sur des représentations symboliques qui permettent le transfert entre la langue source et la langue cible, par exemple les systèmes ATLAS-II de Fujitsu (Uchida : 1987), KANT/

CATALYST de CMU pour Caterpillar (Nyberg : 1997), PIVOT de NEC (Miura : 1992), UNL, CSTAR-II ou MASTOR-1.

La traduction automatique procédurale concerne des systèmes fondés sur des algorithmes, d'une part, des dictionnaires, d'autre part (Polguère 1990).
La traduction automatique experte concerne des systèmes écrits en LSPL (Langage Spécialisé pour la Programmation Linguistique). Ce langage permet d'implémenter un modèle de calcul statique ou dynamique ; le premier type de modèle utilise une grammaire formelle combinée à un algorithme universel alors que le second type de modèle est fondé sur modèle de calcul (Boitet Guillaume et Quezel-Ambrunaz 1985).

La traduction automatique empirique est fondée sur le traitement statistique de corpus de telle sorte que le modèle linguistique de la langue cible soit obtenu à partir de celui de la langue source.

La méthode de traduction automatique prise en compte dans l'IHM Ontomantics est du premier type. Elle s'inspire de travaux sur la traduction automatique qui s'appuient sur la catégorisation des unités linguistiques selon qu'elles relèvent de la fonction prédicative, de la fonction argumentale ou de la fonction actualisatrice (Blanco et Buvet : 2004 ; Mejri : 2009). Les représentations symboliques sont, d'une part, de nature métalinguistique (Blanco et Buvet : 2004) et d'autre part, s'inspirent du langage SQL (Van Der Lans : 2006), ou bien correspondent à des scénarios-types (*cf. infra*). Elles sont mises en correspondance pour que s'effectue le transfert d'instructions rédigées en français en instructions formulées dans le langage Ontomantics.

Le processus de transfert s'effectue en trois étapes. La première étape consiste à analyser les instructions produites par l'utilisateur afin de les rapporter à autant de représentations métalinguistiques, correspondant à des structures prédicat-argument. Le recours à des outils linguistiques développés pour le TAL permet de procéder à l'analyse des instructions. La deuxième étape concerne le transfert proprement dit : chaque représentation métalinguistique est associée, selon le contexte d'utilisation de la plateforme Ontomantics, soit à une commande SQL soit à un scénario type. La troisième étape consiste à générer automatiquement des instructions dans le langage Ontomantics à partir d'une commande SQL ou d'un scénario. Lorsque des éléments d'information, généralement en rapport avec une variable non spécifiée, sont manquants, la machine s'adresse à l'utilisateur par le biais d'un autre système qui relève de la seule problématique de la génération automatique de textes.

### 1.3. GA et IHM

Les demandes de la machine adressées à l'utilisateur constituent la seconde composante de l'IHM d'Ontomantics. Elles sont prises en charge par un module de GA afin de créer un texte à partir de données fournies par le 'Designer' d'Ontomantics durant la conception d'une application.

D'une façon générale, la conception d'un générateur implique trois phases de traitement (Bateman & Zock : 2003). La première phase concerne le traitement du « *Quoi dire ?* ». Il s'agit de déterminer la finalité de la communication, la nature du contenu informatif et l'organisation de ce contenu. L'application qui intègre le générateur n'est généralement

pas sans influence sur ces trois tâches. Elle donne lieu à une structure conceptuelle qui constitue l'entrée du générateur. Lorsque les formalismes de l'application et du générateur ne sont pas compatibles, la transition entre les deux formalismes s'effectue *via* une interface, généralement fondée sur l'exploitation de graphes conceptuels (Sowa : 1984). La deuxième phase concerne le traitement du « *Comment le dire ?* ». Il s'agit d'effectuer la transition entre le niveau conceptuel et le niveau linguistique en associant des éléments lexico-syntaxiques aux concepts stipulés en entrée. Cette phase implique que le générateur exploite des connaissances linguistiques préalablement intégrées ou bien résultant d'une méthode d'apprentissage (Smets, Gamon, Corston-Oliver, E. Ringger 2003). La troisième phrase concerne le traitement du « *A qui le dire ?* ». Il s'agit de produire un texte correctement structuré en tenant compte de facteurs d'ordre discursif et rhétorique. De ce point de vue, il est fondamental de prendre en compte les connecteurs et les chaînes de référence (Danlos : 1994).

La plupart des générateurs sont dédiés à une seule application. Les générateurs qui ne le sont pas correspondent à des modules portables conçus en dehors de tout cadre applicatif. Ils possèdent généralement leurs propres caractéristiques qu'ils imposent à l'application, mais ne tiennent pas compte des spécificités de l'application ou de l'utilisateur. Antoniadis (1996) a développé un générateur automatique de textes que l'on peut intégrer dans toutes sortes de plateformes. Son mode de fonctionnement est indépendant de l'application qui l'héberge et il n'impose aucune contrainte spécifique à l'application et à l'utilisateur.

Les générateurs automatiques de textes les plus récents ont des architectures élaborées à partir du même modèle. Ils impliquent également trois phases, dites « *Détermination de contenu* », « *Planification de phrases* » et « *Réalisation de surface* » (Danlos : 2000). Ces générateurs fonctionnent selon différents niveaux, dits « sélection du contenu profond », « structuration rhétorique », «planification syntaxique », « lexicalisation », « ajustement morphologique », « formatage typographique », « génération d'expressions référentielles ». Chaque niveau correspond à une tâche et l'agrégation des tâches permet la création automatique d'un texte à partir de la représentation symbolique d'une information.

Les principaux formalismes linguistiques exploités dans les systèmes de GA sont la Théorie Sens-Texte (Mel'čuk : 1998), les Grammaires d'Arbres Adjoints (Joshi : 1975), la Théorie des Structures Rhétoriques (Mann : 1988), la Grammaire systémique (Halliday : 1973), et la Grammaire d'unification fonctionnelle (Abeillé : 1993). L'ajout de composants statistiques aux systèmes permet souvent d'améliorer leurs performances.

Les textes générés par l'IHM d'Ontomantics ont comme particularité d'être relativement courts et de porter essentiellement sur des demandes de spécification de variable. Ces textes ont comme source des séquences d'éléments de code du langage Ontomantics identifiés comme défectueux. A partir de l'analyse de ces séquences, on établit quelles variables sont non spécifiées et quels sont leurs contextes d'utilisation. Les deux éléments d'information sont associés à une structure prédicat-argument qui est ensuite actualisée pour produire un énoncé correspondant à une demande d'information adressée à l'utilisateur par la machine (Buvet : 1999)[1].

## 2. L'IHM d'Ontomantics

Après une rapide présentation de la plateforme Ontomantics, nous discutons de la façon de transférer les instructions formulées par l'utilisateur en instructions rédigées en langage Ontomantics ; nous distinguons deux méthodes de transfert selon qu'il s'agit de renseigner une base de données ou de construire une interface utilisateur.

### 2.1. Ontomantics

Ontomantics est une plateforme qui génère automatiquement des applications Web[2]. Ses utilisateurs bénéficient d'un environnement technique intégré ne nécessitant aucune connaissance en programmation. La création des applications s'effectue au moyen d'une interface. Elle permet de créer un modèle de données et un modèle de traitement des données. Dans la mesure où l'interface nécessite néanmoins des connaissances approfondies en informatique, le rôle de l'IHM en cours de développement sera de rendre accessible l'utilisation de la plateforme aux personnes qui n'ont pas de compétence en informatique.

L'interaction entre l'utilisateur et le système Ontomantics est géré par trois modules : (i) un module orienté vers l'utilisateur qui prend en charge la langue naturelle ; (ii) un module dédié au transfert entre la langue naturelle et le langage Ontomantics ; (iii) un module orienté vers la machine qui prend en charge le langage Ontomantics.

Le premier module est constitué de deux sous-modules : un analyseur de textes et un générateur de textes, les textes étant rédigés en français. Le premier sous-module associe des représentations métalinguistiques aux phrases simples, le second sous-module produit des phrases simples à partir de représentations métalinguistiques.

Le second module sert d'interface entre une langue naturelle, en l'occurrence le français, et le langage Ontomantics. Il permet d'associer des représentations métalinguistiques d'énoncés et des représentations symboliques d'éléments du langage Ontomantics. Ces dernières sont de trois sortes selon qu'elles concernent le modèle des données, le modèle de l'interface ou du code défectueux. Seules les deux premières catégories sont prises en charge par la composante de l'interface fondée sur les principes de la TA. La dernière est prise en charge par la composante de l'interface fondée sur les principes de la GA (*cf. supra*).

Le troisième module est également constitué de deux sous-modules mais, au lieu du langage naturel, c'est le langage Ontomantics qui est traité : le premier module est un générateur et le second module un analyseur. L'entrée du générateur est constituée de représentations du langage machine correspondant soit à des commandes en langage SQL (Van Der Lans : 2006), soit à des scénarii (*cf. infra*), et la sortie d'instructions en langage Ontomantics. L'analyseur a comme entrée des codes défectueux et en sortie une représentation symbolique relative à une variable non spécifiée et son contexte d'utilisation.

Les méthodes de transfert, qu'elles soient en rapport avec le modèle de données ou avec le modèle de traitement des données, sont telles que la chaîne de traitement de l'information se décompose en trois phases : une phase d'analyse qui transforme un

énoncé en sa représentation métalinguistique ; une phase de transfert proprement dite qui associe la représentation métalinguistique en une représentation symbolique ; une phase de génération qui produit du code écrit en langage Ontomantics à partir de la représentation symbolique.

La plateforme Ontomantics a deux composantes fondamentales, le Designer et le Player. La première permet de modéliser son application, la seconde de la créer. Data Model, Applications et Librairies constituent le Designer. La fonction du premier est de définir un modèle de données indépendamment du modèle de traitement des données de telle sorte qu'un modèle de données peut être utilisé par plus d'une application. Le second permet de créer un modèle de traitement des données afin d'élaborer une application qui exploite les données enregistrées à partir du modèle de la base de données. Le troisième correspond à l'ensemble des outils informatiques utilisés pour faire fonctionner la plateforme. Le processus de transfert des instructions n'est pas de même nature selon qu'elles concernent Data Model ou Applications.

**2.2. Transfert des instructions relatives à une base de données**

Le modèle de données donnant lieu à l'élaboration d'une base de données, le processus de transfert des instructions relatives à Data Model utilise le langage SQL[3] comme langage pivot entre la langue naturelle et le langage Ontomantics (Buvet & Tromeur 2010).

Les instructions de l'utilisateur rédigées en français sont analysées de telle sorte que chaque énoncé est associé à sa représentation métalinguistique. Celle-ci est ensuite rapportée à une instruction SQL à partir de laquelle est généré du code écrit en langage Ontomantics.

Les procédures d'analyse reposent sur l'exploitation de dictionnaires morphosyntaxiques (Mathieu-Colas : 2009) et de dictionnaires syntactico-sémantiques (Buvet : 2009) d'une part, et sur l'utilisation de grammaires locales (Gross : 1995) d'autre part.

Les dictionnaires syntactico-sémantiques mentionnés sont conçus pour le TAL. Autrement dit, ils ont une microstructure normalisée qui permet l'exploitation informatique des descripteurs associés à chaque vedette. Il y a trois types de dictionnaires selon que la macrostructure concerne des prédicats, des arguments élémentaires ou des actualisateurs.

Une grammaire locale décrit le cotexte d'une unité lexicale donnée en tant qu'ensemble de configurations de mots. Elle est représentée par un graphe comportant : un nœud initial, un nœud final et un ensemble de nœuds intermédiaires ; des arcs qui relient les nœuds en fonction des configurations de mots de la grammaire locale. Les nœuds intermédiaires sont associés à des mots, à des lemmes, à des catégories grammaticales ou à d'autres sortes d'informations métalinguistiques.

Un automate à états finis est un outil informatique qui permet de définir une grammaire locale et d'analyser une séquence de mots. Un automate à états finis permet de décider si la séquence de mots analysée correspond à l'une des configurations de la grammaire locale. Un transducteur à états finis est un automate à états finis qui permet d'associer une nouvelle information à de l'information reconnue. Qu'il s'agisse d'automates ou

de transducteurs, les informations métalinguistiques enregistrées dans les graphes au niveau des nœuds sont celles qui sont encodées dans les dictionnaires électroniques associés au graphe.

Un script écrit en langage Perl[4] permet de créer automatiquement dans la plateforme Ontomantics le code informatique de l'action correspondant à une instruction de l'utilisateur. Le script exécute l'intégralité de la chaîne de traitement : il lance l'invite de commande permettant à l'utilisateur d'entrer sa requête, puis les exécutables Unitex[5] qui génèrent les représentations métalinguistiques à partir de transducteurs à états finis et, enfin produisent le code Ontomantics.

**2.3. Transfert des instructions non relatives à une base de données**

Dans la version actuelle d'Ontomantics, la modélisation de l'interface utilisateur, dit 'Interface Model' se fait par l'intermédiaire d'un écran ; il s'agit d'effectuer des 'glisser-déposer' afin de définir les fonctionnalités du designer en termes de contrôle, d'une part, d'action, d'autre part, au moins une action étant rattachée à un type de contrôle (*cf*. 2.1). La modélisation de l'interface utilisateur permet de spécifier quels sont les différents écrans créés et quel est leur mode d'agencement. Elle implique une seconde modélisation, dite 'Operationnal Model' afin de faire état du mode de structuration de l'interface.

La conception des deux modèles nécessite des compétences spécifiques qui rendent également problématique l'exploitation de l'application Ontomantics par un utilisateur non averti. Il s'ensuit le recours à l'IHM pour pallier les difficultés que pourrait rencontrer l'utilisateur. À la différence du processus de transfert relatif à la conception du modèle de données, celui qui porte sur la conception de ces deux modèles n'est pas fondé sur le langage SQL car les informations traitées ne sont pas stockées dans des bases de données. La notion de scénario est une seconde stratégie qui permet le transfert entre la langue naturelle et le langage machine.

L'utilisation de l'application Ontomantics concerne un nombre limité de situations qui ont été rapportées à cinq cas de figure représentés sous forme scénarisée. Les scénarii sont élaborés en fonction des principales caractéristiques des modèles que l'on peut concevoir. Ces caractéristiques portent sur les deux types de fonctionnalités et les modes de structuration d'une interface compte tenu des fonctionnalités spécifiées. Le traitement de l'information a comme point de départ des indications préalables afin d'aider le locuteur à formuler ses instructions. Ces dernières sont rapportées à un scénario compte tenu de la combinatoire des éléments constants qui sont identifiés.

La phase d'analyse du traitement consiste à repérer puis extraire les informations qui correspondent à des configurations d'éléments constants. Elle est fondée sur l'utilisation d'outils linguistiques qui étiquettent ces configurations. Pendant la phase de transfert, chaque configuration est transformée en scénario de telle sorte que, pendant la phase de génération, le scénario soit associé à une règle propre à la syntaxe d'Ontomantics pour produire le code nécessaire à la création de l'interface utilisateur.

## 3. Cas d'école

Nous présentons une situation correspondant à une utilisation concrète de la plateforme Ontomantics. Il s'agit de la réalisation d'une application de gestion du matériel d'un laboratoire. Nous indiquons comment l'IHM contribue à la mise en œuvre de l'application.

Les besoins de l'utilisateur sont de développer une application qui lui permettra de s'informer sur le matériel informatique du laboratoire aussi bien en termes de hardware que de software et sur la façon dont le matériel est utilisé par les membres du laboratoire selon qu'il s'agit de membres permanents ou de membres non permanents. Cette dernière distinction est importante pour la façon d'allouer le matériel aux membres. Pour ce qui est du matériel, différents paramètres entrent en ligne de compte : date de livraison, date de fin de garantie, date de renouvellement du contrat de mise à jour, etc. L'utilisateur de la plateforme Ontomantics qui a recours à l'IHM est informé préalablement qu'il y a deux opérations fondamentales à effectuer au niveau du designer pour que la plateforme génère automatiquement une application qu'il utilisera au niveau du player (*cf. supra*).

La première opération fondamentale a trait au modèle de données. Il s'agit de générer une base de données comportant des tables faisant état des membres du laboratoire, des ordinateurs, des logiciels et les propriétés qui les caractérisent. L'utilisateur est informé qu'il doit spécifier la nature des informations traitées par l'application créée à partir de la plateforme Ontomantics.

La deuxième opération fondamentale a trait à l'interface utilisateur. Il s'agit de spécifier au moins une application qui traitera les informations enregistrées dans la base de données. Par exemple, une interface qui permet d'assigner un ordinateur à un membre du laboratoire et de spécifier la période d'attribution de l'ordinateur à cet utilisateur ou bien de préciser la date de renouvellement du contrat de mise à jour du logiciel antivirus installé sur les ordinateurs.

Quelle que soit l'opération fondamentale, les instructions fournies par l'utilisateur sont prises en charge par l'IHM afin qu'elles produisent du code écrit en langage Ontomantics. Le processus de transfert utilise un langage pivot fondé sur le langage SQL pour ce qui est de la première opération et sur un ensemble de scenarii en ce qui concerne la deuxième opération. La conception du modèle de données implique de créer une base de données, de créer puis de structurer des tables et de les associer. La conception de l'interface utilisateur doit permettre de définir la structure de l'application envisagée et de stipuler ses différentes fonctionnalités. Lorsque les informations transmises par l'utilisateur sont insuffisantes, le système lui retourne des demandes d'instruction en rapport avec les manques identifiés.

En l'état actuel de nos travaux, la chaîne de traitement est opérationnelle du début jusqu'à la fin. Notre objectif à moyen terme est d'augmenter, dans le contexte applicatif envisagé, le nombre d'instructions analysées afin de les transposer en langage machine. Parallèlement, nous travaillons sur la mise en place d'un système d'acquisition semi-automatique de vocabulaire spécialisé à partir de corpus. L'objectif est d'accroître les ressources linguistiques exploitées par le système car ces performances sont subordonnées à la qualité des analyses des instructions formulées par les utilisateurs.

## Bibliographie

Abeillé A., 1993, *Les Nouvelles Syntaxes: grammaires d'unification et analyse du français* (Linguistique). Paris: Armand Colin, 1993, 326 pp.

Antoniadis G. & Ponton C., 1996, *Un système noyau de génération automatique*, in *Informatique et Langue Naturelle*, *Actes du colloque de Nantes*, *9-10 octobre 1996*, pp. 37-48.

Bateman J. et Zock M., 2003, « Natural Language Generation », *The Oxford Handbook of Computational Linguistics*. R. Mitkov (éd.), Oxford University Press, New York, p. 284-304.

Boitet C., 2007, « Corpus pour la TA : Types, tailles et problèmes associés, selon leur usage et le type de système », *Revue française de linguistique appliquée*, vol. XII, 2007/1. p. 25-38.

Boitet C., Guillaume P. et Quezel-Ambrunaz M., 1985, "A case study in software evolution: from ARIANE-78 to ARIANE-85", *Proc. of the Conf. on theoretical and methodological issues in Machine Translation of natural languages* p. 27-58, ColgateUniv., Hamilton, N.Y.

Buvet P.-A., 1999, « Détermination et génération automatique », *Actes du colloque GAT99*, Grenoble.

Blanco X. & Buvet P.-A., 2004, « Verbes supports et significations grammaticales. Implications pour la traduction espagnol-français », *Lingvisticae Investigationes* 27 (2), John Benjamins B.V., Amsterdam.

Buvet P.-A., 2009, « Dictionnaires sémantiques du laboratoire LDI (lexiques, dictionnaires, informatique) : modèle, outil de gestion informatique, outil d'exploitation », *Ela* 4/2009 (n° 156), p. 475-489.

Buvet P.-A. & Tromeur L., 2010, « Le dialogue homme-machine : un système de traduction automatique spécifique », *Meta*, Vol. 55-1, *Le parcours du sens : d'une langue à l'autre,* Mejri S. et Gross G. (dirs.), Mélanges offerts à André Clas, p. 58-70.

COALA 1997 : http://www-lium.univ-lemans.fr/~lehuen/recherche/these/index.html

CSTAR-II : http://www.clips.imag.fr/geta/herve.blanchon/Recherche/Projets/CSTAR2/CSTAR2.html

Danlos L., 1985, *Génération automatique de textes en langue naturelle*, MASSON, Paris

Danlos L., 1994, « Génération de textes dans le formalisme G-TAG inspiré des grammaires d'arbres adjoints », *Rapport technique TALANA*, 1.

Danlos L., 1998, *G-TAG : un formalisme lexicalisé pour la génération de textes inspirés de TAG*, *TAL*. Traitement automatique des langues, Association pour le traitement automatique des langues, Paris, FRANCE (éd.), 1998, vol. 39, n°2, p. 7-33 (2 p.1/2).

Danlos L. & Roussarie L., 2000, « La génération automatique de textes », *Ingénierie de la langue*, J.M. Pierrel (dir.), Hermès.

Gross M., 1995, « Une grammaire locale de l'expression des sentiments », *Langue française*, Vol. 105 N°1. Grammaire des sentiments. p. 70-87.

Halliday M.A.K., 1973, *Explorations dans les fonctions de la langue*, Londres : Edouard Arnold, 1973.

Joshi A. K., Levy L. S., Takahashi M., 1975, *Tree Adjunct Grammars*, Journal of Comput Syst. Sci., Vol. 10-1, 1975

Lerat P., 1995, *Les langues de spécialité*, Paris, PUF.

Mann W.C., & Thompson S.A., 1988, « Rhetorical Structure Theory : Toward a functional theory of text organization », *Text* 8(3), p. 243-281.

Mathieu-Colas M., 2009, « *Morfetik* : une ressource lexicale pour le TAL », *Cahiers de lexicologie*, 2009-1, n° 94, p. 137-146.

Mejri Salah 2009, « Le mot : problématique théorique », *Le Français Moderne* 77 (1), CILF, Paris, p. 68-82.

Mel'čuk I., 1998, « The Meaning-Text Approach to the Study of Natural Language and Linguistic Functional Models », [Invited lecture.], S. Embleton (ed.): *LACUS Forum* 24, Chapel Hill: LACUS, p. 3-20.

Mitsuru M., Mikito H., Nawi H., 1992, *Learning mechanism in machine translation system "PIVOT"*, *Proceeding COLING '92 Proceedings of the 14th conference on Computational linguistics - Volume 2*, 1992.

Nyberg E., Kamprath Ch., Mitamura T., 1997, « The KANT Machine Translation System: From R&D to Initial Deployment », *Presentation at LISA Workshop on Integrating Advanced Translation Technology,* Washington, D.C.

Pierrel J.-M. & Sabah G., 1991, « Dialogue en langage naturel écrit ou oral : bilan des approches du CRIN et du LIMSI », *Communication Homme-Machine*, p. 91-111, EC2 Éditeur.

Polguère A, 1990, *Structuration et mise en jeu procédurale d'un modèle linguistique déclaratif dans un cadre génération de texte*, Thèse de doctorat, Université de Montréal.

Smets M., Gamon M., S. Corston-Oliver et E. Ringger, 2003, "French Amalgam : A machine-learned sentence realization system", *Actes de la conférence Traitement Automatique du Langage Naturel (TALN'2003)*, Batz-sur-mer.

Sowa J. F., 1984, *Conceptual Structures: Information Processing in Mind and Machine*, Addison-Wesley, (ISBN 0-20114-472-7), 1984.

Hiroshi U., 1987, *ATLAS : Fujitsu Machine Translation System*, *Machine Translation Summit*, 17-19 septembre 1987, Hakone Prince Hotel, Japan.

UNL : http://en.wikipedia.org/wiki/Universal_Networking_Language

Van Der Lans R.-L., 2006, *Introduction to SQL: Mastering the Relational Database Language, éd.* Addison-Wesley.

Weizenbaum J., 1966, « Eliza : a computer program for the study of natural language communication between man and machine », CACM, 9, p. 26-45.

## Notes

[1] Dans cet article, la composante de l'interaction entre l'homme et la machine fondée sur les principes de la GA n'est pas détaillée.

[2] http://www.ontomantics.com

[3] SQL signifie Structured Query Langage ; c'est un langage informatique dédié à la constitution et l'interrogation de bases de données.

[4] http://www.perl.org

[5] http://www-igm.univ-mlv.fr/~unitex/