Vers un méta-EDL complet, puis un EDL universel pour la TAO

Hong-Thai NGUYEN[1], Christian BOITET[2]
GETALP, LIG
385, av. de la Bibliothèque, BP 53 F-38041 Grenoble cedex 9
{Hong-Thai.Nguyen,Christian.Boitet}@imag.fr

Résumé. Un "méta-EDL" (méta-Environnement de Développement Linguiciel) pour la TAO permet de piloter à distance un ou plusieurs EDL pour construire des systèmes de TAO hétérogènes. Partant de CASH, un méta-EDL dédié à Ariane-G5, et de WICALE 1.0, un premier méta-EDL générique mais aux fonctionnalités minimales, nous dégageons les problèmes liés à l'ajout de fonctionnalités riches comme l'édition et la navigation en local, et donnons une solution implémentée dans WICALE 2.0. Nous y intégrons maintenant une base lexicale pour les systèmes à « pivot lexical », comme UNL/U++. Un but à plus long terme est de passer d'un tel méta-EDL générique multifonctionnel à un EDL « universel », ce qui suppose la réingénierie des compilateurs et des moteurs des langages spécialisés pour la programmation linguistique (LSPL) supportés par les divers EDL.

Abstract. A "meta-EDL" (meta-Environment for Developing Lingware) for MT allows to pilot one or more distant EDL in order to build heterogeneous MT systems. Starting from CASH, a meta-EDL dedicated to Ariane-G5, and from WICALE 1.0, a first meta-EDL, generic but offering minimal functionalities, we study the problems arising when adding rich functionalities such as local editing and navigation, and give a solution implemented in WICALE 2.0. We are now integrating to it a lexical database for MT systems relying on a "lexical pivot", such as UNL/U++. A longer-term goal is to evolve from such a multifunctional generic meta-EDL to a "universal" EDL, which would imply the reengineering of the compilers and engines of the specialized languages (SLLPs) supported by the various EDLs.

Mot-clés : génie linguiciel, langages spécialisés pour la programmation linguistique, LSPL, environnement de développement, EDL, TAO, systèmes distribués hétérogènes.

Keywords: lingware engineering, specialized languages for linguistic programming, development environment, EDL, MT, heterogeneous distributed MT systems.

1 Introduction

Il existe des EDL (Environnements de Développement Linguistique) pour systèmes de TAO, plus ou moins complets. Ils sont tous construits autour d'une technologie spécifique. On peut citer Ariane-78 puis Ariane-G5 du GETA (Grenoble), Tapestry du CRDL (Singapour), ETAP-3 de l'IPPI (Moscou), et ceux des fournisseurs de système de TAO commerciaux, non disponibles pour la recherche.

Depuis 10 ans environ, on cherche à réaliser des systèmes de TAO hétérogènes, soit pour combiner plusieurs systèmes pour une nouvelle paire de langues (approche « multimoteur »

de Pangloss (Nirenburg and Frederking, 1994), VerbMobil (Ney H, Och & Vogel, 2000), etc.), soit pour construire un système de TAO fortement multilingue dont les composants peuvent être développés par différents groupes, avec des approches et des EDL différents, comme dans le projet UNL (Projet UNL).

Pour permettre le développement coopératif et distribué de ce type de système, une première étape consiste à développer un « méta-EDL » fonctionnant comme une interface avec plusieurs EDL distants, i.e. permettant d'éditer et de synchroniser les composants linguiciels (dictionnaires, grammaires, automates) et de combiner différents modules distants pour produire des traductions. Un problème intéressant est alors d'intégrer le plus possible de fonctions des EDL (navigation, aide à l'indexage des dictionnaires, etc.), sans devoir effectuer une réingénierie de ces EDL. Enfin, comme cette approche est par nature limitée, un but plus ambitieux est de construire un « EDL générique » pour le développement distribué de systèmes de TAO hétérogènes.

Cet article est organisé en trois parties. Nous détaillons d'abord les fonctionnalités des EDL et des méta-EDL, et en donnons une illustration avec CASH et WICALE 1.0. Dans la deuxième partie, nous montrons les problèmes posés par l'ajout à WICALE des fonctions d'édition et de navigation en local, ainsi que les solutions retenues pour leur implémentation dans WICALE 2.0. Dans la troisième partie, nous montrons la nécessité et la difficulté d'intégrer une base lexicale dans un méta-EDL, et décrivons PIVAX, une base lexicale multilingue organisée autour d'un « pivot lexical », en cours de construction. PIVAX pourra être utilisée non seulement pour la TAO, en particulier pour le projet UNL/U++, mais pour développer d'autres applications comme la recherche d'informations en contexte multilingue (CLIR).

2 EDL et méta-EDL pour la TAO

Un Environnement de Développement Linguistique (EDL) est un environnement de *programmation linguistique* qui connecte ou intègre un ou plusieurs *LSPL* (Langages Spécialisés pour la Programmation Linguistique). Un EDL permet aux développeurs linguistes de réaliser les opérations nécessaires (gestion, manipulation des linguiciels et des données, compilation, test, production) de façon transparente.

Un « méta EDL» permet de piloter à distance un ou des EDL distants. Pour l'instant, nous travaillons à la construction d'un méta-ED pour la TAO le plus puissant possible. Dans le futur, nous voulons développer un EDL « universel » permettant la réingénierie de tout EDL.

Les caractéristiques des EDL sont assez différentes de celles des IDE (environnement de développement de logiciels). Nous l'illustrerons avec deux exemples de méta-EDL existants.

2.1 Fonctionnalités d'un EDL

Table 1: comparaison des IDE et des EDL

	IDE	EDL
Utilisateur	programmeur	linguiste, lexicographe, gestionnaire, utilisateur …
Type de composant	fonction, procédure, objet, module, paquetage…	variables, modèles, grammaire, automate, dictionnaire…
Taille de composant	petite (quelques pages)	grande (100K-1G entrées pour un gros dictionnaire de TA, 100-400 pages pour une grammaire d'analyse)
Type d'évolution	plutôt stable	en perpétuelle évolution

Un EDL de TAO complet doit offrir 4 « classes de fonctionnalités ».

Préparation des composants linguiciels : (1) visualisation, (2) édition, (3) tri, (4) aide à l'indexage.

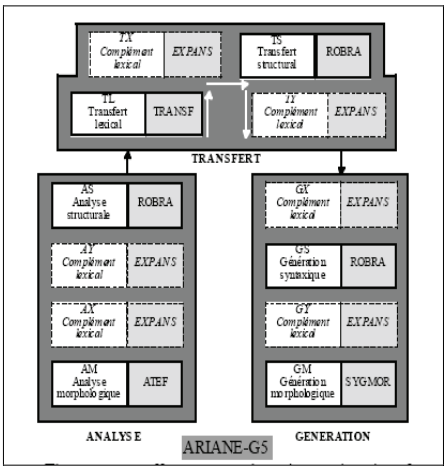
Organisation en étapes et phases : (1) gestion de versions de test pour mise au point, (2) gestion de chaînes d'exécution (totale ou partielle), (3) génération de systèmes de TAO complet ou des parties de tels systèmes.

Gestion de corpus d'essai : (1) création ou modification de corpus ou de textes, (2) passage de tests, (3) traduction de (parties de) corpus, (4) révision humaine.

Actions globales : (1) extraction d'informations, (2) vérification de cohérence, (3) impressions avec filtrage et tri.

2.2 Les EDL de TAO existants

Le plus complet semble toujours être Ariane-G5 (Ariane-Y) du GETA (Figure 1), le seul qui permet de « créer » un nouveau système de TAO en quelques commandes, sans intervention d'informaticiens. Le processus de traduction se compose de trois étapes (analyse, transfert, génération), chaque étape étant composée de phases (obligatoires ou facultatives).



En analyse, on trouve :		
AM	analyse morphologique	**obligatoire**
AX	analyse expansive	facultative
AY	analyse expansive	facultative
AS	analyse structurale	**obligatoire**
En transfert, on trouve :		
TL	transfert lexical	**obligatoire**
TX	transfert expansif	facultative
TS	transfert structural	**obligatoire**
TY	transfert expansif	facultative
En génération, on trouve :		
GX	génération expansive	facultative
GS	génération syntaxique	**obligatoire**
GY	génération expansive	facultative
GM	génération morphologique	**obligatoire**

Figure 1 : étapes, phases et LSPL de l'EDL Ariane-G5

Pour générer un système de TAO, l'utilisateur (un développeur linguiste) écrit un linguiciel pour chaque phase avec le LSPL adéquat, le compile et génère une « chaîne d'exécution ».

Citons quelques EDL de TAO incomplets à notre sens.

ETAP-3 de l'IPPI à Moscou. L'EDL vu en démonstration est partiel, et nous n'avons pas trouvé de référence le décrivant.

Vermobil. Il n'y avait pas d'EDL pour préparer les linguiciels, mais au maximum quelques script sous Linux. Par contre, il y avait un EDL pour l'intégration et la mise au point du système global, construit autour d'une structure commune (« tableau noir ») accédée par chaque module de façon distribuée.

Systran (Systran). Là aussi, les développeurs sont informaticiens-linguistes. Les parties concernant les grammaires et les automates sont définies directement dans le code source. Les « composants » écrits dans les LSPL semblent n'être que les dictionnaires et les

transducteurs finis utilisés pour la morphologie. Le cycle de développement est le même pour le « cœur » de TALN que pour les interfaces et la gestion du flot de travaux, i.e. le cycle de production en génie logiciel. La partie lexicale est séparée et développée avec des outils internes : commandes, scripts pour l'indexage, filtrage d'erreurs, etc. Le code source lexical (sous plusieurs formats : texte, Excel, XML…) est compilé et encrypté par l'outil ACMulti (J. Senellart 2003). La gestion de versions est faite sous le système CVS.

2.3 Exemples de Méta-EDL : CASH et WICALE 1.0

2.3.1 CASH, pour Ariane-G5

Le système Ariane-G5 est normalement utilisable depuis une « machine virtuelle » générée par VM/ESA et accessible via http, smtp ou des sockets. Cette accessibilité ne se limite pas à l'exécution de traductions, mais s'étend à l'ensemble du développement des linguiciels, y compris à la création et à la maintenance de grammaires et de dictionnaires.

Pour faciliter cette exploitation à distance, E.Blanc a réalisé une interface hypertextuelle, CASH (Commande d'Ariane Sous Hypertexte) (E. Blanc 1996). CASH intègre plusieurs fonctions qui en font plus qu'un méta-EDL, comme l'aide à l'indexation dans les dictionnaires et l'édition graphique d'arbres et schémas d'arbres. CASH vient d'être converti de HyperCard (propre à Mac OS 9) vers une plate-forme portable (Revolution).

Dans l'exemple, on a cliqué sur la variable SUBA utilisée dans la définition de la procédure ADJ, d'où l'ouverture d'une autre fenêtre contenant la définition de cette variable.

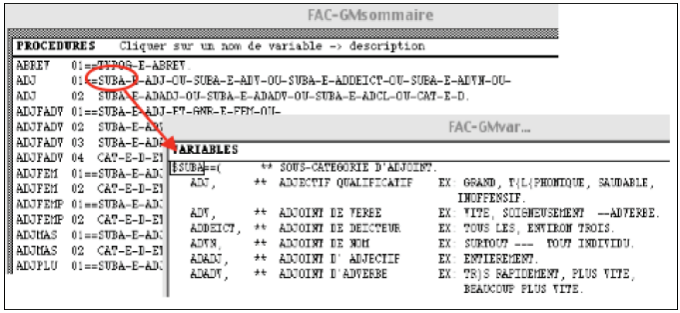


Figure 2: navigation sous CASH

2.3.2 WICALE version 1.0

Un méta-EDL minimal pour la TA, WICALE version 1.0 (V. Carpena 2004) a été construit en 2005. Il offre aux linguistes les mêmes services d'échange de données que CASH, mais aucun autre. En ce sens, il est « minimal ». Par contre, il est « générique » car il permet de travailler avec plusieurs EDL. D'un autre côté, CASH est très riche et très utile pour travailler spécifiquement avec Ariane-G5. WICALE 1.0 a été expérimenté avec les EDL d'Ariane-G5, de PILAF et d'UNL (UNL-deco du GETA).

Table 2 : comparaison entre CASH et WICALE 1.0

	Echange de commandes et données	Edition de données	Navigation dans les données	Généricité	Aide aux linguistes
CASH	oui	oui	oui	non (spécifique à Ariane-G5)	oui
WICALE 1.0	oui	non	non	oui	non

WICALE présente deux avantages principaux, la généricité et la portabilité.

généricité : on peut étendre WICALE à un nouvel EDL sans écrire de code Java, mais simplement en décrivant les commandes et les données de cet EDL

portabilité : elle est simplement due au fait que Java existe dans pratiquement tous les environnements logiciels actuels.

WICALE permet de définir l'architecture et les commandes d'exécution des systèmes connectés. (Des exemples pour Ariane-G5, PILAF et UNL-Deco sont donnés en annexe.)

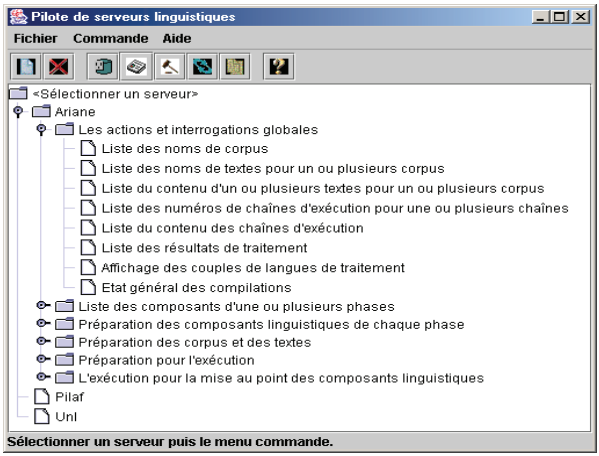


Figure 3: représentation des commandes et sous-commandes d'un EDL sous WICALE

WICALE génère l'interface correspondant aux paramètres définis dans l'architecture de chaque système. Exemple: en Ariane-G5, l'architecture est Machine>Disque>Langue>.

(Le code XML décrivant cette architecture a été supprimé de l'annexe, faute de place.)

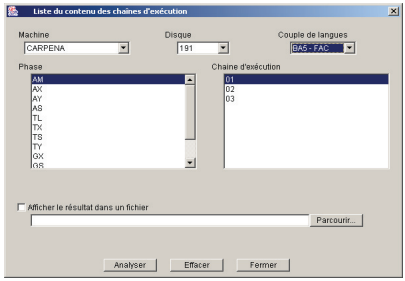


Figure 4: interface générée par WICALE 1.0

3 Enrichissement d'un méta-EDL générique: WICALE 2.0

Nous avons cherché à enrichir WICALE 1.0 en utilisant la même technique générique.

3.1 Édition en local

On utilise tout éditeur disponible (choix paramétrable), alors qu'Ariane-G5 utilise seulement XEDIT, et on s'inspire aussi d'Ariane-G5 au niveau fonctionnel : par sécurité, l'utilisateur édite toujours une copie du composant édité. Il y a deux modes, V (Visualisation) et M (Modification) : dans le premier, les modifications effectuées n'ont aucune conséquence (on avertit cependant l'utilisateur !).

Cette extension de WICALE 1.0 a été très facile à réaliser, grâce à la modularité et à la généricité du code.

3.2 Navigation

Il s'agit ici d'offrir une possibilité similaire à celle de CASH (implémentée par des scripts *ad hoc*). Nous avons proposé et implémenté une solution simple basée sur XML et inspirée de Doxygen, un outil de génération de documentation (Doxygen).

Dans cette approche, un programme analyse et marque la liaison entre les occurrences et la définition de chaque élément. De plus, il prend en compte certains commentaires spéciaux (auteur, date, résumé, …). Finalement, un générateur produit des fichiers HTML où toutes les occurrences dans le source deviennent des liens pointant vers la page contenant la définition. L'utilisateur navigue dans l'ensemble de ces fichiers HTML en utilisant n'importe quel navigateur Web.

Dans WICALE 2.0, la préparation des fichiers de navigation se passe de la même façon, en trois étapes:

transformation en XML du code source des composants linguiciels, réalisée dans notre cas par le compilateur d'Ariane-Y ;

marquage : parcours de la structure intermédiaire XML de chaque composant et insertion de liens entre définitions et occurrences ;

génération d'un fichier html pour chaque composant, avec ajout à chaque occurrence d'un élément d'un lien vers la position de sa définition.

Voici un exemple tiré d'un composant de "définition de variables" (définition d'un jeu de décorations linguistiques) :

```
** Commentaires entre 2 étoiles et point.
** Transformation statique Jeu_1 --> Jeu_2.
-DECVAR- dv .   ** Nom du composant: DV.
-DECO- deco .   ** Nom du jeu : deco.
MT              ** Temps morphologique.
==(IMP,IPR,SPR,IPA,SPA,INF,PPR,PPA,FUT,CDL).
SEXE == (FEMININ, MASCULIN).
DGA == (SYN, ANA, NO).
…
-CVAR-
** Transformation complémentaire (procédure).
CHGMT(C;@S) ==
-SI-    MT(@S)-INC-IPR -ALORS- MT(C):=IPR;
-SNSI- MT(@S)-INC-SPR -ALORS- MT(C):=IPR;
-SNSI- MT(@S)-INC-IPF -ALORS- MT(C):=IPA;
-SNSI- MT(@S)-INC-SPF -ALORS- MT(C):=SPR;
-SNSI- MT(@S)-INC-IPA -ALORS- MT(C):=IPA;
-SNSI- MT(@S)-INC-FUT -ALORS- MT(C):=FUT;
-SNSI- MT(@S)-INC-CDL -ALORS- MT(C):=SPA;
-SNSI- MT(@S)-INC-IMP -ALORS- MT(C):=IMP;
-SNSI- SUBV(@S)-E-INF -ALORS- MT(C):=INF;
-SNSI- SUBV(@S)-E-PPR -ALORS- MT(C):=PPR;
-SNSI- SUBV(@S)-E-PPA -ALORS- MT(C):=PPA;
-FSI-.
…
-FIN-
```

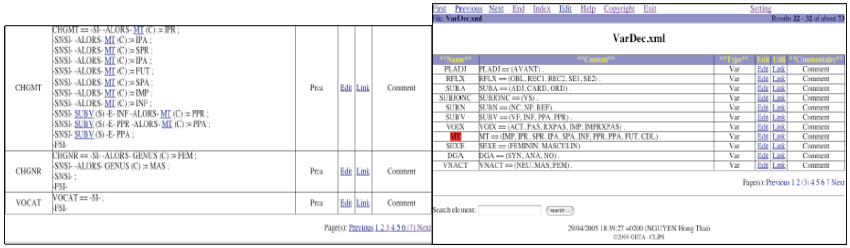


Figure 5: navigation sous WICALE 2.0

Après la génération, on a des fichiers HTML et on peut y naviguer.

4 Intégration d'une base de données lexicales multilingue

4.1 Nécessité & difficulté

4.1.1 Nécessité

Si l'on veut développer un système de TA hétérogène grâce à un EDL, il faut y centraliser de traitement des parties multilingues. En approche transfert, cela implique de traiter les grammaires et les dictionnaires de transfert, la partie lexicale étant la plus importante.

En approche par « pivot interlingue », il faut centraliser le développement des dictionnaires pivot-Li, pour chaque langue Li. G.Sérasset a développé un premier exemple d'une telle base lexicale dans le serveur UNL-deco pour traduire le site B@bel de l'UNESCO en français, espagnol, russe, et chinois (C. Boitet 2005).

4.1.2 Problèmes rencontrés lors de travaux antérieurs

CICC. C'est un projet de l'ODA (Overseas Development Agency) pour la TAO « à pivot » entre japonais, chinois, thaï, malais et indonésien, financé par de grosses sociétés japonaises actives en TAO. On y tenta de développer le vocabulaire IL comme un ensemble de « concepts », identifiés uniquement à l'aide de définitions en anglais. Mais cette contrainte était trop forte (comment distinguer 2 poissons par 2 définitions ?), et il n'y avait pas de base de données partagée en accès direct.

UWGate. Cette base de données lexicales centralisée pour le projet UNL donne l'accès par échange de fichiers (gzip protégé), même pour un seul article de dictionnaire. De plus, le délai d'attente est bien trop long (accusé de réception après 2 ou 3 jours ou jamais…).

UNL-deco. Ce service web de déconversion d'UNL vers le français contient une base de données lexicales accessible par le Web en temps réel, mais actuellement limitée au français et à UNL. Elle n'a en fait pas été utilisée pour développer le système fra-UNL, car elle n'offre aucun outil d'aide aux travaux lexicographiques (tri, filtre, aide à l'indexage…) et elle est inextensible à N langues, au contraire de CASH+PARAX.

4.1.3 Difficulté de principe

Les problèmes décrits plus haut nous semblent provenir d'une difficulté de principe, à savoir que le problème très général de construire une base de données lexicales « universelle » pour la TAO, capable de gérer tous les aspects, de la construction des données jusqu'à l'extraction automatique de dictionnaires des modules des différents systèmes (ex. analyse morphologique de Systran, transfert lexical de Neon), est quasiment insoluble.

Non seulement les difficultés théoriques sont encore plus grandes que dans le cas d'une base lexicale multilingue « d'usage », destinée aussi à la recherche en dictionnairique multilingue, comme la base Papillon (Projet PAPILLON), mais les difficultés pratiques sont quasiment insurmontables (diversité des formats, et pire encore de la nature des informations, problèmes de droits de propriété intellectuelle (IPR)).

4.2 Une première approche, se limiter à des systèmes de TAO à pivot

L'analyse résumée ci-dessus a cependant montré qu'il devrait être possible de simplifier ce problème et d'arriver à un problème soluble en théorie, et à une réalisation utile en pratique. La simplification dont il s'agit a deux aspects :

on considère une architecture lexicale « en étoile », ou « à pivot », ce qui revient à se limiter à des systèmes de TAO à « pivot lexical ».

on renonce à ce que le système contrôle totalement les données lexicales, jusqu'à leur représentation « codée » dans les divers systèmes de TAO considérés.

Cette approche a déjà été réalisée et validée par la base de données lexicales PARAX (Blanc 1999). Mais cet environnement de développement, très adapté pour les manipulations lexicales, ne permet pas le travail coopératif à distance. PIVAX s'inspire donc de PARAX, mais l'étend au niveau structurel, et vise un fonctionnement distribué de type Wiki.

4.3 Vers PIVAX, une base de données lexicales contributive pour systèmes à pivot lexical

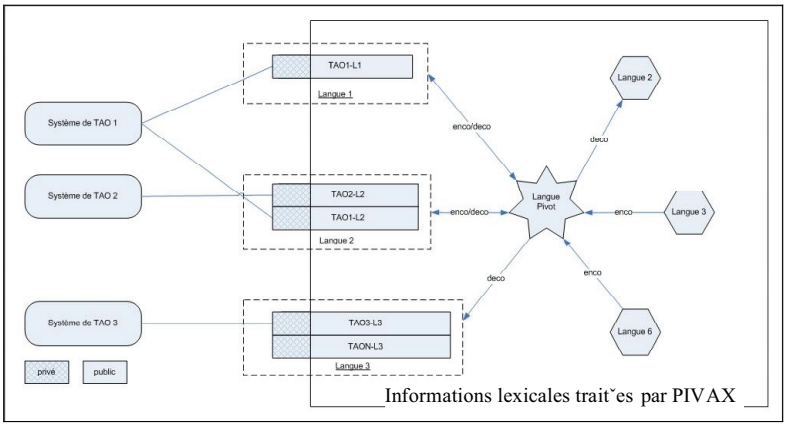


Figure 6: schéma de l'information présentée dans PIVAX

Dans PIVAX, on considère que chaque système possède ses propres linguiciels privés non gérés par PIVAX, et partage via PIVAX la partie « pivot » ainsi que sa partie « publique ». Dans la partie privée, on mettra par exemple des informations spécifiques réservées pour ce système comme les codes (morphologiques, syntaxiques et sémantiques) liés aux LSPL, les familles dérivationnelles (UL du GETA), et les formules sémantiques. La partie « pivot » contient des unités conceptuelles (IF, CATALYST) ou des acceptions interlingues (ATLAS-II, ULTRA, PIVOT, UNL), et la partie « publique » contient des lemmes ou des lexies (lemme avec indication de sens), entités par essence non propriétaires.

PIVAX sera accessible pour le travail lexicographique (humain) par une interface du type de celle de PARAX. D'autre part, PIVAX offrira une API pour la synchronisation avec divers systèmes de TAO, réalisée avec les modules existants de WICALE. Notons enfin que PIVAX est développé sur la plate-forme Jibiki de G. Sérasset, déjà utilisée pour développer la base lexicale PAPILLON, la terminologie multilingue de la Convention Alpine (Projet LexAlp), et le Grand Dictionnaire Estonien-Français GDEF (Projet GDEF).

Conclusion et perspectives

Nous avons exposé la conception d'un méta-EDL générique et d'un EDL « intégrateur », qui ne seraient pas nécessairement limités à la TAO comme WICALE. Dans un premier temps, nous avons cherché à incorporer au méta-EDL WICALE 1.0 toutes les fonctions de CASH. Concrètement, nous y avons ajouté une possibilité d'édition des composants linguiciels, puis une possibilité de navigation utilisant une compilation « légère » vers un format XML,

obtenant WICALE 1.1. D'autre part, nous avons décrit la construction en cours de PIVAX, une base de données lexicales contributive à pivot interlingue destinée à la TAO et aux autres applications multilingues (RI, traitement de contenu…). Nous espérons pouvoir présenter lors de TALN une première version de PIVAX, appliquée au développement de la base lexicale du projet UNL/U++ (français, anglais, espagnol, russe au moins).

Références

BLANC, E. (1996). Une maquette de base lexicale multilingue à pivot lexical ("acceptions multilingues"): PARAX. *Actes des quatrièmes Journées scientifiques du Réseau "Lexicologie, Terminologie, Traduction" de l'AUF, Lyon (France)*, 43-58.

BLANC, E. (1999). An interactive hypertextual environment for MT development. *MT Summit VIII, Santiago de Compostela, Galicia, Spain*, EAMT, 67-81.

BOITET, C. (1988). Dictionnaires intégrés multiusage et multicible, une première exprérience. *Colloque sur l'histoire de la terminologie, Inistitut Libre Marie Haps, Bruxelles*, 6 p.

BOITET, C. (1989). Software and lingware engineering in modern M(A)T systems. *Computational Linguistics, an International Handbook on Computer-Oriented Language Research and Applications*, Niemeyer, 670-682.

BOITET, C. and NÉDOBEJKINE, N. (1986). Towards integrated dictionary for M(a)T: motivations and linguistic organization. *Proc. COLING-86, Bonn,* 1/1, 423-428.

CARPENA, V. (2004). Interface cliente générique pour le pilotage de serveurs linguistiques. *Mémoire CNAM GETA, CLIPS, Grenoble*, 86 p.

DOXYGEN. http://www.stack.nl/~dimitri/doxygen/, accédé en 2007.

LAFOURCADE, M. (1994). Génie Logiciel pour le Génie Linguistique. *Thèse, UJF, Grenoble*.

MANGEOT-LEREBOURS, M. (1999). Accès unique à des dictionnaires hétérogènes. *VIe journées scientifiques du réseau thématique LTT de l'AUF (Lexicologie, Terminologie, Traduction)*, 311-316.

MANGEOT-LEREBOURS, M. (2001). Environnements centralisés et distribués pour lexicographes et lexicologues en contexte multilingue. *Thèse, UJF, Grenoble,* 279 p.

MANGEOT-LEREBOURS, M., SÉRASSET, G. (2001). Projet Papillon: architecture du serveur Web. *JST'2001 Journées Science et Technologie, National Olympic Memorial Youth Center, Tokyo, Japon* 1/1, 149-150.

NEY, H., OCH, F. J. AND VOGEL, S. (2000). Statistical Translation of Spoken Dialogues in the Vermobil System. *Proc. MSC2000*, 69-74.

NGUYEN, H.-T. (2005). Vers un "méta-EDL", puis un "EDL générique" pour la TAO. Mémoire de master recherche (M2R), UJF, Grenoble, 85 p.

NIRENBURG, S. AND FREDERKING, R. (1994). Toward multi-engine machine translation. *Proc. of the workshop on Human Language Technology, Plainsboro, New Jersey, USA*, 147-151.

PROJET PAPILLON (2003). http://www.papillon-dictionary.org/, accédé en 2007.

PROJET ARIANE-Y (2004). http://www-clips.imag.fr/geta/User/jean-philippe.guilbaud/DOCUMENTS/ARIANE-Y/ARIANE-Y-Index.html, accédé en 2007.

PROJET C-STAR (2004). http://www.c-star.org/, accédé en 2007.

PROJET UNL (1997). http://www.unl.ias.unu.edu/, accédé en 2007.

PROJET GDEF (2005). http://estfra.ee/Home.po, accédé en 2007.

PROJET LEXALP (2004). http://217.199.4.152:8080/general/lexalp/index.php, accédé en 2007.

SENELLART, J., YANG, J. AND REBOLLO, A. (2003). Technologie "Intuitive Coding" de SYSTRAN. *MT Summit IX*, 8p.

SÉRASSET, G. (1994). SUBLIM: un système universel de bases lexicales multilingues et NADIA: sa spécialisation aux bases lexicales interlingues par acceptions. *Thèse,* UJF*, Grenoble*, 194 p.

SYSTRAN (2006). https://systran.fr, accédé en 2007.

Annexe

Exemple de déclaration des commandes d'Ariane-G5, de PILAF et d'UNL-deco :

```
< !-–Description des commandes Ariane-G5 -->
<LST_SERVEUR>
 <SERVEUR>
  <nom_serveur>ARIANE-G5</nom_serveur>                        < !--Nom de serveur -->
  <classe>ServerAriane</classe>                              < !--Type de serveur -->
  <Communication>Socket</Communication>            < !--Protocole de communication-->
  <adresse>tupai.imag.fr</adresse>                        < !--Adresse du serveur -->
  <port>5768</port>                                          < !--Port du serveur -->
  <codage>iso-8859-1</codage>                          < !--Encodage des données -->
  <entete_ligne> &lt;&lt;&lt; 19283 &gt;&gt;&gt; --- premier enregistrement ---
ARIANET --- LIDIA20 ---.                                                </entete_ligne>
  <fin_ligne> &lt;&lt;&lt; 19283 &gt;&gt;&gt; --- premier enregistrement --- ARIANET
--- LIDIA20 ---.                                                          </fin_ligne>
 </SERVEUR>
```

```
 <SERVEUR>
  <nom_serveur>PILAF</nom_serveur>
  <classe>ServerHttp</classe>
    <communication>Http</communication>
    <adresse>http://clips.imag.fr/cgi-
bin/pilaf/</adresse>
    <port></port>
    <codage>iso-8859-1</codage>
    <entete_ligne></entete_ligne>
    <fin_ligne></fin_ligne>
 </SERVEUR>
```

```
<SERVEUR>
  <nom_serveur>UNL</nom_serveur>
  <classe>ServerUnl</classe>
    <communication>Socket</communication>
    <adresse>tupai.imag.fr</adresse>
    <port>5768</port>
    <codage>iso-8859-1</codage>
    <entete_ligne></entete_ligne>
    <fin_ligne></fin_ligne>
</SERVEUR>
</LST_SERVEUR>
```

Description de la commande d'Ariane-G5 qui demande la liste des noms des corpus. En natif, sa forme est **LISNOMCORP** (Terminal | Imprimante | TI).

```
<COMMANDE num_cde="1">
  <num_cde>1</num_cde>
  <nom_cde>LISNOMCORP</nom_cde>
  <intitule_cde>Liste des noms de
corpus</intitule_cde>
  <PARAMETRE_SAISIE> </PARAMETRE_SAISIE>
  <SYNTAXE>
    <ETAPE>
    <mot_cle>TRAIT = LISNOMCORP (*)
</mot_cle>
    <num_param></num_param>
    <expression></expression>

  <separateur>&retour_chariot;</separateur
> ...

  <saisie_obligatoire>false</saisie_obliga
toire>
    </ETAPE>
  </SYNTAXE>
```

```
 <RESULTAT>
  <resultat_OK>-> Tout est
O.K.</resultat_OK>
  <resultat_type>-> Tout est
O.K.</resultat_type>
    <ETAPE>
    <nom_methode>find</nom_methode>
    <expr_deb>Liste des noms de
corpus</expr_deb>
    <expr_corps> [A-Za-z0-9]*
</expr_corps>
    <expr_fin>-LISTE TERMINEE-
</expr_fin>

  <expr_concat>&retour_chariot;</expr_c
oncat>

  <expr_remplacement></expr_remplacemen
t>
    </ETAPE>
  </RESULTAT>
</COMMANDE>
```