# La saga du Lse et de sa famille (LSD-LSG-LST)

Yves Noyelle

# La Saga du LSE et de sa famille (LSD/LSG/LST)

**Yves NOYELLE**

Les LSx (LSD, LSE, LSG, LST) sont des langages de programmation développés durant les années 1968-1976 à l'École Supérieure d'Électricité, dans le but de mettre l'ordinateur au service du programmeur. La philosophie qui a guidé leur conception est qu'un outil doit faciliter au maximum la vie de celui qui l'emploie et non lui imposer des contraintes, surtout si celles-ci sont évitables facilement.

Cette philosophie gagne du terrain actuellement, mais à l'époque où fut conçu LSD (1968), elle n'était pas très courante. Elle est née de l'exaspération du néophyte que j'étais devant les complexités (apparemment) inutiles de la majorité des langages et systèmes de l'époque, et aussi du contact avec le langage PAF (Programmation Automatique de Formules) de la CAB 500, qui au contraire s'efforçait déjà (avant 1960 !) de mettre en oeuvre une philosophie similaire.

## ENVIRONNEMENT HISTORIQUE

Fin 1967 se posait le problème suivant au Service Informatique de l'École Supérieure d'Électricité : il fallait donner aux élèves de l'école (environ 600 à l'époque) accès à "l'informatique", c'est-à-dire en pratique à un ordinateur. Le service avait depuis longtemps (1962) une "calculatrice" SEA CAB 500, mais son usage était réservé aux quelques privilégiés qui avaient choisi la section Calcul Automatique, et sa puissance de toute façon très limitée. Pour résoudre ce problème des crédits avaient permis d'acheter un ordinateur IBM 1130, dont la puissance de calcul semblait suffisante pour le mettre à la disposition de tout le monde.

Malheureusement, le système d'exploitation n'était pas à la hauteur : il nécessitait un personnel qualifié non négligeable ("perfo-vérifs", opérateur, assistants pour aider à la programmation et à la mise au point), et ne rendait pas pour autant un service extraordinaire ; en

particulier, le temps de service était long, l'imprimante à 80 lignes/minute n'améliorant pas les choses...

Il avait donc été décidé de réaliser un système en temps partagé, au moins pour éviter les perfo-vérifs et l'opérateur, et un Hewlett-Packard 2116A (un des premiers mini-ordinateurs, bien adapté à la gestion de périphériques) était venu rejoindre le 1130. Mais le langage prévu restait FORTRAN, dont évidemment le 1130 fournissait un compilateur et le support d'exécution.

Or, il se trouve qu'à l'époque je revenais de l'université de Stanford (Etats-Unis), un diplôme en poche mais surtout plein d'idées en tête, idées enrichies par de nombreux contacts avec les gens de l'équipe de Patrick Suppes qui avaient réalisé, à des fins d'enseignement assisté par ordinateur (déjà !), un système en temps partagé convivial sur une petite machine (PDP-1). Et donc on me chargea de ce projet.

En y réfléchissant, je me suis vite aperçu que l'optique temps partagé influait sur le choix du langage : vu la faible puissance (malgré tout) des ordinateurs dont nous disposions, il était notamment souhaitable que ce langage soit compilable ligne à ligne, pour ne pas avoir à reprendre toute la traduction d'un programme à chaque modification (et celles-ci sont nombreuses chez un débutant!).

D'autre part, FORTRAN n'est pas un modèle de langage permettant une détection aisée des erreurs à l'exécution, et celles-ci sur la 1130 se traduisaient souvent par la mort du système ; or nous voulions un système fiable, et nous cherchions également à nous passer d'assistants pour la mise au point. Ceci conduisait directement à un système interprété, dans lequel la détection complète d'erreurs sémantiques est beaucoup plus facile que dans un système compilé, et qui permet de réaliser toute protection nécessaire pour assurer la fiabilité.

Il restait encore trois objectifs :

- obtenir un système vraiment utilisable, c'est-à-dire avec des temps de réponse corrects et des tailles de programme permettant de traiter des problèmes réels,
- faire l'économie du cours de programmation ; le langage défini devait donc être suffisamment "naturel" pour permettre l'auto-apprentissage, mais sans donner de mauvaises habitudes,
- faire en sorte que le produit obtenu soit facile d'emploi et heurte le moins possible le bon sens de l'utilisateur (application de la

philosophie présentée plus haut) ; d'autre part, comme l'erreur est humaine, il fallait également que ce produit résiste à n'importe quelle fausse manipulation.

**LSD**

Toutes ces contraintes ont mené entre Mars 1968 et Mars 1969 à la conception de LSD (Langage Symbolique Didactique, selon son appellation officielle), et à sa réalisation sur le couple 1130 (16 ko de mémoire, disque amovible de 1Mo), 2116 (16 Ko de mémoire, 13 terminaux Télétypes ASR33), reliés via leur accès direct mémoire.

Pour la petite histoire, le nom LSD est en réalité le second nom du langage, le premier étant LPS, pour Langage de Programmation Scientifique, nom sérieux s'il en fut, mais dont il s'avéra assez vite qu'il était déjà utilisé. Un facétieux de l'équipe, Paul Gavarini, a alors proposé LSD, en apportant, à l'appui de sa proposition, un lot de cartes postales "LSD, j'aime", que nous nous empressâmes de coller sur les ordinateurs. Il restait à trouver une signification pour cet acronyme ; le premier fut Langage Symbolique Didactique, alors que Langage Sans Difficulté, tellement mieux adapté, ne vint que plus tard.

LSD satisfaisait les objectifs assignés :

- langage compilable ligne à ligne, dont n'importe quelle ligne, même une ligne de déclaration, pouvait être ajoutée, modifiée, ou supprimée, sans recompilation du reste,
- identificateurs multilettres pour permettre une bonne lisibilité des programmes ; à vrai dire, les spécifications initiales du langage prévoyaient uniquement des identificateurs monolettres, ceci permettant de se passer de table de symboles ; mais la réalisation ayant montré qu'il restait de l'espace mémoire disponible dans la 2116, des tables de symboles furent introduites,
- procédures fonctions et sous-programmes, avec objets locaux pour permettre une structuration des programmes ; modes de passage de paramètres par valeur et par adresse (pour éviter les problèmes de FORTRAN avec le passage par adresse seul ; le passage par valeur venait d'ALGOL) ; récursion possible (à vrai dire parce qu'elle arrivait tout naturellement avec le mode de réalisation choisi pour l'interpréteur) ; appel des procédures facilement repérable dans le texte source (caractère '&' en tête de nom de procédure),

- un seul type de données : le type réel, le langage étant prévu pour satisfaire les besoins supposés des élèves-ingénieurs de l'école et de leurs enseignants, donc uniquement numériques. L'optique du langage étant d'être naturel à utiliser (par un non-informaticien), je ne voyais pas l'intérêt de demander au programmeur s'il voulait utiliser des entiers ou des réels. Plus généralement, je voulais le moins de contraintes possible, en particulier pas de déclarations, sauf quand elles sont manifestement nécessaires (objets composés), ou quand elles peuvent aider à la modularité (variables locales dans les procédures). En fait, la nécessité d'un autre type, le type caractère, est très vite apparue, et donc il fut rapidement ajouté,
- un seul genre d'objets composés : le tableau (à une ou deux dimensions). L'approche "facilité d'emploi" rendait impératif que la déclaration en soit dynamique, car cela paraît normal à l'utilisateur (au moins débutant). De plus, cela permet de moduler au mieux l'encombrement mémoire d'une tâche, dont dépendent les performances d'un système avec "swap" (voir ci-après). La réalisation ne m'en paraissait pas impossible puisque les compilateurs ALGOL y arrivaientÉ,
- instructions conditionnelle et de branchement (je ne voyais pas bien à l'époque l'intérêt de l'instruction de boucle, qui pouvait parfaitement être simulée par une instruction de branchement, et aurait pris un peu de précieux espace dans le compilateur et l'interpréteur) ; également instruction composée, pour augmenter la puissance des instructions conditionnelles (qui devaient néanmoins tenir sur une ligne de télétype, c'est-à-dire 72 caractères...),
- instructions de lecture et d'écriture cherchant à faciliter la vie de l'utilisateur ; en particulier, pour l'instruction de lecture, redemande automatique d'un autre nombre si la syntaxe du précédent était incorrecte, et désactivation du clavier (sauf en ce qui concerne la touche d'interruption, voir plus loin) quand le programme n'était pas en lecture ; pour l'instruction d'écriture, sortie non tronquée d'un nombre même si le format était trop court.

De plus, la structure de LSD était telle qu'un programme pouvait facilement être représenté en notation polonaise inverse ("chaîne post-fixée"), dont les avantages sont les suivants :

. facile à interpréter efficacement,

. peu encombrante (environ deux fois moins que le texte source) ; il faut se souvenir que la mémoire centrale ne faisait que 16 Ko, et qu'il devait y tenir, en plus de la zone de travail de l'utilisateur (programme et données), l'interpréteur et le système ; pour des raisons de performance, ces derniers devaient être résidents. Une technique d'échange ("swap") était employée, permettant à chaque utilisateur d'occuper jusqu'à environ 10 Ko,

. décompilable ; ainsi, il n'y avait pas besoin de sauvegarder le texte source (qui aurait encombré le disque, et augmenté le temps d'accès) : il était recréé, en cas de listage, à partir de la chaîne post-fixée.

Divers autres points étaient incorporés pour faciliter la tâche du programmeur :

- utilisation de mots-clés français (c'était bien la moindre des choses !, mais des mots anglais auraient peut-être été mieux venus pour l'avenir du langage...),
- langage de commande intégré, à la fois concis et explicite : il suffisait de frapper les deux premières lettres de chaque commande, et le système complétait le libellé de la commande avant de demander ses arguments. Un mode "abrégé" permettait d'éviter aux utilisateurs avertis la perte de temps causée par la frappe (à 10 caractères/seconde !) de ce complément,
- peu de modes : j'étais déjà de l'avis de Larry Tesler, dans son article Don't Mode Me In (The Smalltalk Environment, BYTE Août 81) ; il n'y avait donc que deux modes dans le système, le mode "Exécution" et le mode "Moniteur". Dans ce dernier mode, l'unité d'information était la ligne, et le système savait s'il avait affaire à une ligne de commande ou une ligne de programme grâce au numéro qui commençait chaque ligne de programme (et servait à indiquer leur ordre d'exécution),
- outil de mise au point intégré : il était possible d'arrêter un programme à tout moment par appui sur la touche ESC (touche d'interruption), de savoir où il avait été interrompu, et de demander ou de modifier la valeur de n'importe quel objet ; on pouvait ensuite faire continuer l'exécution sans aucune perte d'information (pas même un caractère si l'arrêt avait eu lieu en cours d'impression). La même chose était possible après arrêt sur une instruction PAUSE ou sur une erreur (mais dans ce dernier

cas, on ne pouvait évidemment faire continuer). Cet aspect me paraissait (et me paraît toujours) très important, mais il implique une conception adaptée et de l'interpréteur, et du système d'exploitation,

- gestion de fichiers, limitée à vrai dire aux fichiers programme et ajoutée après coup, vers 1970, car nous commencions à épuiser les charmes du ruban perforé, seul moyen de sauvegarde auparavant. Elle était conçue selon les mêmes principes que la gestion mémoire, à savoir rapidité et allocation dynamique ; pour ne pas prendre sur l'espace mémoire des utilisateurs (et voir arriver des gens se plaindre que "bien qu'ils n'aient rien changé", leur programme n'entre plus en mémoire), elle était réalisée en recouvrement sur l'interpréteur,
- détection d'erreurs rapide (grâce à la compilation ligne à ligne) et complète (en particulier, détection de l'emploi d'objets non initialisés).

Au niveau du système d'exploitation, toujours pour essayer de satisfaire au mieux l'utilisateur, on s'était fait un point d'honneur à lui faire croire que l'ordinateur était rapide, en particulier grâce au passage en priorité des travaux courts (un algorithme fort simple avait été trouvé pour détecter les "travaux courts"), grâce à la simultanéité entre les traitements et l'impression, etc. Par exemple, le temps de compilation (analyse syntaxique et génération de la chaîne post-fixée) était en général invisible, car recouvert par le temps d'impression des caractères faisant passer à la ligne le terminal.

De même, on s'était efforcé de tout mettre en oeuvre pour qu'un programme ne soit jamais perdu par la faute du système, même en cas de "plantage". D'ailleurs, ceux-ci ne devaient pas exister (et cessèrent assez vite d'exister, grâce à une traque opiniâtre des anomalies).

Les résultats ont été tout à fait à la hauteur des efforts, les utilisateurs se sont déclarés ravis des temps de réponse (ou, plus exactement, ils ont trouvé cela normal...), et pourtant le système gérait treize terminaux ; les critiques venaient par contre du bruit des télétypes, de leur lenteur et du manque de temps disponible (le système ne fonctionnait qu'entre 12 h et 14 h, à cause de l'organisation des études à Sup-Élec et de l'utilisation du 1130 l'après-midi par les élèves de la section Informatique).

A titre indicatif, les tâches étaient réparties entre les deux machines de la façon suivante :

1130 : interprétation, gestion de fichiers, gestion du "swap", gestion de l'accès aux objets en mode "mise au point",

2116 : compilation, décompilation, analyse des commandes, gestion des terminaux, gestion de l'arrêt du système et des coupures/retours secteurs, gestion du temps partagé.

Il est bien évident que tout le logiciel de ces deux machines avait dû être réécrit, soit pour économiser de la place, soit pour satisfaire aux fonctionnalités du système (en particulier, la spécification qu'un programme en cours d'exécution puisse être interrompu immédiatement à tout moment, même en cours d'impression, ce qui avait mené à l'écriture d'un pilote de terminal tirant astucieusement parti d'une caractéristique du coupleur 2116, caractéristique probablement liée à l'économie d'une porte ET de la part de son concepteur...) ; de même, une méthode originale de compilation avait été développée pour obtenir un compilateur à la fois rapide, peu encombrant et fiable (ne laissant en particulier passer aucune erreur, toute chaîne post-fixée incorrecte risquant de créer une catastrophe au moment de l'exécution ou de la décompilation).

Ont participé à ce projet :

- S. Berche (utilitaires variés, en particulier conversion astucieuse en virgule flottante)
- E. Fizzarotti (décompilateur)
- P. Gavarini (compilateur)
- Y. Noyelle (conception et définition du langage, interpréteur, gestion de fichier, système de la 1130)
- J-Ph. Szylowicz (réalisation de la liaison matérielle 1130-2116, système de la 2116, gestion des terminaux et idée initiale du décompilateur)

**LSE**

La saga du LSE suit celle du LSD, et son origine est le colloque CERI/OCDE de Sèvres (Mars 1970) où il avait été débattu de ce que l'informatique pourrait apporter à l'enseignement. La décision ayant été prise d'installer des ordinateurs, à titre expérimental, dans quelques lycées, restait à choisir le langage qui serait employé pour les

programmer. L'un des participants du colloque de Sèvres était Jacques Hebenstreit, Chef du service Informatique de l'ESE, qui proposa le LSD pour les raisons suivantes :

- il comportait un noyau initial facile à apprendre, tout en étant relativement puissant (notamment à cause des procédures),
- il était manifestement réalisable de manière efficace sur des petits ordinateurs (les crédits alloués pour cette expérience n'étaient pas énormes),
- il "parlait" français (il ne fallait pas donner aux utilisateurs l'impression que informatique était synonyme d'anglophonie),
- il ne nécessitait pas de personnel spécialisé (opérateur, etc.).

Après une démonstration sur un programme de calcul d'amplificateur à transistor, LSD fut choisi comme langage support de l'introduction de l'informatique dans l'enseignement, à deux conditions :

- que son nom soit changé...
- qu'on le rende exploitable, non seulement par des scientifiques, mais également par les autres enseignants d'un lycée.

Ainsi naquit LSE, successeur de LSD (E est la lettre qui suit D) ; il y avait deux significations initiales à cet acronyme :

- Langage Symbolique d'Enseignement,
- Langage de Sup-Élec,

et d'autres sont venus après : Langage Simple à Enseigner, Langage Sans Espoir (ce dernier nom donné évidemment par les inconditionnels de BASIC !).

Après une réflexion, guidée essentiellement par J. Hebenstreit, pour rendre LSE le plus universel possible, il fut décidé d'ajouter le type "chaîne de caractères", le texte paraissant le dénominateur commun de nombreuses activités d'enseignement.

Il restait à définir des opérations pratiques portant sur ce type de données, ce que je fis de Juin à Septembre 1971, en m'appuyant sur les sources suivantes :

- une connaissance de LISP, qui me faisait envisager sans inquiétude la gestion d'objets de taille variable,
- les fonctionnalités similaires de PL/1,

- une remarque de J-Ph. Szylowicz, me conseillant de chercher les opérations les plus courantes effectuées par un programme manipulant du texte (en l'occurrence le compilateur LSD).

A cette occasion, S. Berche donna une solution élégante au problème de la définition d'une sous-chaîne, non par sa longueur, mais par un caractère d'arrêt (ou un ensemble de caractères d'arrêt).

J'en profitais pour améliorer d'autres points de LSD :

- introduire une instruction de boucle, dérivée de celle d'ALGOL,
- améliorer l'instruction d'écriture (facteur de répétition variable, sortie d'un nombre sous la forme la plus agréable possible, élimination du format dans les cas simples),
- ajouter des fonctions diverses donnant la date, un nombre pseudo ou réellement aléatoire, etc.,
- introduire des fichiers données permanents et temporaires, dotés de plusieurs caractéristiques originales, toujours pour faciliter la vie du programmeur :
  . un enregistrement contenait un (et un seul) objet complet, et le type de celui-ci ; ainsi l'exploitation d'un fichier était extrêmement simple,
  . chaque enregistrement était muni d'une clé numérique arbitraire, l'accès se faisant selon cette clé et non séquentiellement,
  . un espace temporaire de taille bien définie était garanti à chaque utilisateur ; ainsi, un programme utilisant des fichiers temporaires de taille fixe, s'il était déjà passé une fois, était sûr de ne pas se faire avorter ultérieurement par manque d'espace fichier, quel que soit le degré de remplissage à ce moment de la mémoire auxiliaire,
  . pas de notion d'ouverture ou de fermeture infligée à l'utilisateur, celles-ci étant implicites et dynamiques.
- introduire une instruction de liaison, permettant, pour un programme décomposé en plusieurs modules, de faire charger un module à la place du précédent (la communication entre deux modules se faisant via les fichiers).

Au niveau du langage de commande, les fonctionnalités suivantes étaient ajoutées :

- outils de manipulation de programme : plutôt que d'introduire des outils variés, comme éditeur de texte ou éditeur de liens, la solution retenue était de donner la possibilité de "décoder" ou "d'encoder" un programme, c'est-à-dire de passer d'un programme à un fichier de texte représentant son source, et réciproquement ; la manipulation du source se faisait alors par un autre programme écrit en LSE, utilisant ses caractéristiques de gestion de texte et de fichier,
- système de protection des fichiers, à base "d'identifications" (numéro + clé secrète) ; il y avait aussi un système de protection des programmes chargés en mémoire, les rendant non listables et non modifiables, afin d'éviter qu'un utilisateur de didacticiel aille consulter le texte du programme pour savoir la réponse attendue, au lieu de la trouver tout seul...,
- mode d'exécution immédiate, permettant de voir facilement l'effet d'une instruction (avec certaines restrictions sur les instructions exécutables dans ce mode) ; ceci donnait en particulier la possibilité d'effectuer des calculs comme avec une calculette à mémoire, et de visualiser ou de modifier les objets d'un programme interrompu,
- extension des outils de mise au point par ajout d'une possibilité de pas à pas et de point d'arrêt.

Le rapport de définition de LSE fut remis au Ministère de l'Éducation Nationale (Mission à l'Informatique) en Octobre 1971 ; il restait encore à réaliser le système de programmation et le système d'exploitation en temps partagé pour au moins 9 terminaux (selon les spécifications du Ministère) sur les deux machines retenues : les mini-ordinateurs T1600 de Télémécanique et MITRA-15 de SEMS, avec comme date limite Septembre 1972 (rentrée scolaire).

Les gens de Télémécanique mirent le paquet sur ce projet, nous demandant également de leur servir de conseil ; ils tinrent (presque) les délais, et livrèrent deux systèmes fin Octobre 1972 . Ces systèmes tournaient sur des machines de 16 Ko de mémoire centrale, avec un disque rapide de 256 Ko.

Quant à la SEMS, le projet semblait l'intéresser beaucoup moins ; elle nous confia le soin de le réaliser, ce que nous fîmes sur une machine dotée d'un disque rapide de 384 Ko et de 8 Ko de mémoire centrale. Inutile de dire que ce fut du sport de faire tenir tout ce qui était

nécessaire dans cette petite mémoire, d'autant que nous voulions qu'il y ait assez de place en zone utilisateur pour y faire tenir un module de fonctionnalités équivalentes à celles d'un programme d'environ 300 lignes de FORTRAN.

De nombreuses techniques furent mises en oeuvre, en particulier des techniques de compactage et de recouvrement dynamique, permettant de configurer en mémoire le sous-ensemble de l'interpréteur tel que le programme courant puisse s'exécuter avec le minimum d'accès disque. De même, seule une toute petite partie des tampons associés aux terminaux était résidente, le reste étant placé sur disque, mais néanmoins disposé de manière à garantir un débit minimal de 600 bauds sur chacun des terminaux (nous avions prévu le système pour 16 terminaux). Ceci nous permit de donner la moitié de la zone mémoire (4 Ko) à chaque utilisateur (avec encore une technique d'échange, bien entendu).

Pour obtenir un bon temps de réponse (bien inférieur à la demi-seconde en général, c'est-à-dire encore meilleur qu'en LSD), nous disposâmes les différentes branches des structures de recouvrement statiques (compilateur, décompilateur, etc.) de façon à ce que le délai rotationnel du disque lors de leur accès soit minimal, et nous nous efforçâmes de les charger le moins souvent possible.

Deux raisons nous avaient incitées à entreprendre ces exercices de haute voltige :

- faire en sorte que les utilisateurs puissent disposer d'un disque de 384 Ko (la différence de prix avec celui de 192 Ko était quasiment le coût des 8 Ko de mémoire économisés),
- quelques remarques (de personnes extérieures à l'ESE) insinuant que nous n'y arriverions pas...

Mais ce ciselage prit du temps, et nous ne livrâmes qu'une version très primitive du système en Décembre 1972, avec deux bons mois de retard, le système complet se faisant encore attendre jusqu'en Juin 1973. Ceci valu des pénalités à la SEMS, à laquelle le Ministère commanda moins de machines qu'à Télémécanique, mais nous nous consolâmes en constatant que les performances de notre système étaient entre trois et quatre fois meilleures que celles du système concurrent.

Au point de vue fiabilité, les produits étaient très bons tout les deux ; mais je me souviens encore d'une version de notre système livrée avec une grosse erreur (une virgule mal placée dans un programme tuait

tout le système), erreur détectée et corrigée très rapidement ; mais entre-temps une copie de cette version était partie vers les machines utilisées par le DEUG de PARIS VI, sans que la correction de l'erreur suive, ce qui fait que pendant au moins deux ans (avant que nous n'en ayons l'écho) les utilisateurs de ce système ont dû pester contre la qualité du logiciel Sup-Élec !

Ont participé au projet LSE :

- S. Berche (interpréteur)
- J. Hebenstreit (idée que le texte était le dénominateur commun des activités d'enseignement, et que donc le langage devait pouvoir le manipuler facilement ; réflexions plus générales sur l'apport de l'informatique à l'enseignement)
- C. Lhermitte (décompilateur)
- Y. Noyelle (définition de LSE et rédaction du rapport correspondant, analyseur de commande, système d'exploitation, système de gestion de fichier)
- P. Pressigout (compilateur)

**NORME LSE**

Ceci n'était pas la fin de la saga du LSE ; après l'interruption de l'équipement des lycées en mini-ordinateurs (1976), l'opération des "10 000 micro" vint la réveiller en 1979. Une commission fut alors établie au sein de l'AFNOR, dans le but de rédiger une norme définissant sans ambiguïté le langage et l'environnement de programmation associé ; ainsi, la portabilité des programmes pourrait être garantie d'une réalisation à l'autre, et une cohérence d'emploi respectée.

A cette occasion, et après consultation des utilisateurs, des extensions furent apportées au langage, en particulier la possibilité d'appeler des procédures compilées séparément et chargées dynamiquement (procédures externes), la possibilité de compiler dynamiquement une procédure (pour résoudre le problème de l'utilisation dans un programme de formules entrées au clavier), la notion de périphérique logique, la possibilité de verrouiller un fichier ou un enregistrement pour éviter des mises à jour simultanées (utilisation multi-poste ou en réseau), les types booléen et numérique-étendu, ce dernier permettant de faire des calculs de précision arbitrairement grande, la possibilité d'appeler des fonctions écrites dans un autre langage, la possibilité de comparer

des chaînes de caractères selon un ordre lexicographique quelconque, ainsi que d'autres améliorations ponctuelles.

Les principaux participants à la rédaction de cette norme ont été :

- B. Commiot (ENS St Cloud)
- J.P. Lamoitier (ingénieur-conseil)
- B. Leroy (CII-HB)
- P. Muller (INRP)
- Y. Noyelle (ESE)
- J. Robert (CAP SOGETI)

Grâce à (ou malgré) de nombreuses pressions et contre-pressions d'ordre non technique, la rédaction de cette norme a été terminée en Juillet 1982.

Une autre extension importante de LSE fut l'introduction (postérieure à la norme) de fonctionnalités graphiques (tracé de vecteurs, notion de "motif", etc.), fonctionnalités récupérées du langage LSG (voir plus loin).

Il restait à réaliser le système correspondant sur les diverses machines choisies par l'Éducation Nationale ; comme nous avons refusé de le faire (arguant du fait que réaliser une fois un système LSE pouvait être qualifié de recherche, vu ses aspects novateurs, mais que le réaliser en série n'était plus de notre ressort), trois sociétés se sont montées pour satisfaire à la demande, les sociétés MICRODUR, EDL et, plus récemment, LOGICIA.

## ÉTAT ACTUEL DU LSE

Finalement, malgré son peu de diffusion, LSE existe toujours, non seulement sur les matériels Éducation Nationale (où il devait par contrat être implanté , ce que tous les fournisseurs sauf un, le plus important, ont fait), mais encore sur tous les compatibles PC, le Macintosh, des matériels COMMODORE, etc.

On peut se demander pourquoi il n'a pas eu plus de succès, car il semblait répondre de manière tout à fait satisfaisante aux critères spécifiés ; en particulier il était, sur de nombreux points, bien supérieur à BASIC. Peut-être déroutait-il les réalisateurs potentiels par son aspect dynamique ? Nous aurions dû publier les méthodes permettant de le mettre en œuvre, méthodes pourtant simples et systématiques comme le

prouve le faible encombrement du logiciel nécessaire (environ 37 Ko au total sur MITRA, y compris le système d'exploitation en temps partagé).

Il avait aussi quelques défauts... Au moins une tentative extérieure à l'ESE a été faite pour les corriger, celle de J. Arsac avec LSE83 (cf. bibliographie), mais elle sacrifie la compatibilité.

**LSG**

LSG (Langage Symbolique Graphique) a été défini en 1972 par Jean Le Bourdon (également du service informatique de l'ESE) ; son but était de donner le moyen de tirer le meilleur parti possible d'outils graphiques économiques (en particulier consoles), en vue d'application à la CAO et à l'EAO.

Tout comme LSE, le langage devait être conversationnel, donner des outils puissants, mais rester d'utilisation simple et donc manipuler des concepts faciles à maîtriser.

Les entités introduites étaient principalement :

- la notion d'item, composé d'une description de forme graphique (formes élémentaires : point, vecteur, cercle, etc., et dérivés de ces formes obtenus par rotation, symétrie, concaténation, superposition, etc.), et d'une liste de valeurs associées, permettant de donner une sémantique à l'item : ainsi, on pouvait définir un item "condensateur 10nF 20%" en associant une forme graphique (suite de vecteurs représentant le dessin d'un condensateur) et les valeurs 10nF et 20%,
- la notion de motif, qui représentait une sous-image graphique ; ainsi, dans un schéma avec beaucoup de composants identiques, le dessin d'un composant correspondait à un motif, qui pouvait ensuite être répété un grand nombre de fois de manière très facile pour l'utilisateur et très compacte pour l'ordinateur,
- la notion de cible, permettant d'indiquer de manière interactive un point d'un dessin sur l'écran et de remonter jusqu'à l'item correspondant.

LSG a été conçu comme une extension de LSE, utilisant ce dernier pour effectuer les traitements sur les valeurs associées aux items ; il avait été ajouté pour cette application un mode de passage par nom, mode nécessaire si l'on veut pouvoir passer en paramètre les expressions décrivant des courbes.

Ont participé à la réalisation de LSG :

- D. Clar (développement d'une version portable de LSE, écrite en FORTRAN, et adaptations diverses)
- J. Le Bourdon (primitives graphiques et interface multi-périphérique)
- B. Vivinis (maquettes initiales)

Ce travail a été réalisé sur contrat avec la Délégation à l'Informatique du Ministère du Développement Industriel et Scientifique (ancêtre de l'ADI).

**LST**

Un autre rejeton de LSE, implanté dans l'industrie et toujours bien vivant, est le LST (Langage Symbolique de Test), qui a été développé dans les années 1973-1976 à la demande de la société Thomson-CSF.

Son historique est le suivant : plusieurs divisions de cette société font du test automatique de composants et de sous-systèmes produits en petites séries, et recherchaient pour ce faire un langage d'apprentissage aisé pour des non-informaticiens, permettant d'écrire des programmes faciles à modifier et à (re-)mettre au point, et fonctionnant sous système en temps partagé de manière à limiter le coût unitaire du poste de test. Il leur fallait également un système fiable ; de plus, une fausse manipulation sur un terminal ou un équipement de test devait être récupérable, et ne pas avoir de conséquence pour les autres utilisateurs.

Initialement, un système Hewlett-Packard avait été employé, avec BASIC comme langage ; mais ensuite avait été prise la décision d'utiliser du matériel CII, et donc le MITRA-15. Comme un BASIC fiable et multiposte sur cette machine tardait à venir, il leur fut proposé LSE qui, après essais, fut adopté à l'unanimité de la commission "Automatisation des Essais", moyennant deux extensions principales :

- augmenter la taille de la zone de travail (il suffisait d'étendre la mémoire),
- pouvoir gérer n'importe quel périphérique, et mettre au repos une tâche en attente d'un événement.

Cette seconde extension posait de sérieux problèmes ; ils furent résolus en donnant la possibilité d'écrire des procédures externes en langage d'assemblage, dotées des mêmes modes d'appel et de

communication que les procédures LSE, et chargeables dynamiquement, soit dans la zone utilisateur, soit dans une zone résidente commune ; là, on pouvait les déclarer "clouées", et donc gérer des interruptions. Ainsi, moyennant l'ajout de quelques appels systèmes (communication avec l'appelant, mise au repos d'une tâche, accès à des fonctions de calcul) et grâce à l'architecture astucieuse du MITRA-15 (existence d'un vrai registre de base en particulier), on obtenait un système pouvant se configurer dynamiquement, et en particulier gérer n'importe quel nouveau périphérique sans modification (ceci à comparer avec les générations de système fréquemment nécessaires pour ajouter un "driver" à un système classique).

On avait également donné la possibilité d'ajouter à l'infini des commandes (pour un poste particulier, ou pour plusieurs postes), toujours dynamiquement, et de rendre partageables les procédures externes résidentes (qu'elles soient écrites en assembleur ou en LSE).

Un système de protection logiciel faisait que seuls les utilisateurs autorisés pouvaient passer en mode maître, ou utiliser les appels systèmes dangereux. Ceci, joint au mécanisme de protection mémoire du MITRA, faisait que la fiabilité restait excellente.

Pour avoir une chaîne complète de développement, un assembleur (assez lent à vrai dire) et les utilitaires associés (éditeur de texte, etc.) avaient été écrits en LSE ; les outils de mise au point avaient été adaptés en conséquence.

Ce système fonctionne toujours chez Thomson-CSF ; il semble donner satisfaction puisque, quand les MITRA-15 n'ont plus été fabriqués, un compatible MITRA, baptisé CMP-16, a été développé pour continuer à l'utiliser ; environ 600 exemplaires du CMP-16 ont été produits. Actuellement, LST est en cours de portage sous UNIX.

Les extensions menant à LST ont été développées conjointement par Y. Noyelle et S. Berche, mais c'est à ce dernier qu'il faut donner le crédit de ce développement ; en effet, je trouvais à l'origine que les modifications demandées étaient peu systématiques (elles étaient beaucoup plus disparates que ce qui précède peut laisser croire) ; mais nous avons réussi à les rendre cohérentes et homogènes, ce qui à permis de transformer LSE en un système ouvert (moyennant un minimum de 8 Ko en plus).

## CONCLUSION

Le développement des LSx s'est étendu sur neuf années (1968-1976), avec quelques interruptions. Il nous a permis de tester les idées que nous avions sur les langages de programmation et sur les systèmes d'exploitation (en particulier au niveau de la "convivialité"), et a été très instructif pour notre tâche principale, l'enseignement. En effet, bien que le travail fourni soit important (de l'ordre de deux homme-année pour chacun des quatre projets), il a été effectué globalement à mi-temps, l'autre mi-temps étant consacrée en gros à enseigner.

Les LSx nous ont permis d'explorer des concepts novateurs (types chaîne de caractères et numérique-étendu, fichiers d'utilisation simple, primitives de synchronisation pour LST, items et motifs pour LSG, etc.), et de les mettre en oeuvre, non seulement pour les maîtriser, mais également pour en assurer une réalisation industrielle. Ceci nous a notamment permis d'affiner une démarche "qualité du logiciel", basée en particulier sur des batteries de tests enrichies à chaque anomalie, et sur la pratique de la relecture du code écrit par un collègue.

On peut regretter que nous n'ayons pas plus publié sur le sujet (cf bibliographie) ; ce n'est pas faute d'y avoir été invité, mais entre écrire du code et écrire du papier, nous n'hésitions guère...

Il me faut signaler pour terminer que cette saga n'aurait pu être écrite sans la souplesse des structures de l'ESE, et la direction avisée du Chef du Service Informatique, J. Hebenstreit. Qu'il en soit remercié ici, ainsi que tous mes collègues non cités, mais qui forment la trame de ce service et avec lesquels les discussions ont souvent été fructueuses.

mai 1988

Yves NOYELLE
Service Informatique,
École Supérieure d'Électricité
Plateau de Moulon,
91192 GIF/YVETTE CEDEX

**Bibliographie**

- "*Utilisation du Système en Temps Partagé ESE (langage LSD)*", Polycopié n° 2162, ESE, 1969,
- "*Un Langage Symbolique destiné à l'Enseignement : LSE*", J. HEBENSTREIT, Y. NOYELLE, Congrès AFCET, Grenoble, Novembre 1972,
- "*LST : A Highly Adaptative Real-Time Time-Sharing System*", S. BERCHE, J. HEBENSTREIT, Y. NOYELLE, Symposium IFAC/IFIP, Tallinn, Mai 1976,
- "*Le LSE, son histoire, son développement*", Y. Noyelle, Éducation 2000 n° 16, Institut Supérieur de Pédagogie, Paris, 1980,
- "*The Smalltalk Environment*", LARRY TESLER, BYTE, Août 1981,
- Norme expérimentale Z65-020 : "*LSE : Langage Symbolique d'Enseignement*", AFNOR, Paris, Décembre 1983,
- "*String Handling Facilities of the LSE language*", Y. NOYELLE, SIGPLAN, Août 1984,
- "*LSE83*", J. ARSAC, bulletin de l'EPI n° 38, p. 116-139, Juin 1985.
- *Le système LSE, mieux connaître son fonctionnement sur micro-ordinateur*, Dossier EPI n°3, juin 83 (épuisé)
- *LSE POUR TOUS*, CNDP-EPI cinq éditions.

**NDLR :** cet article est extrait des actes du colloque sur **l'histoire de l'informatique en France** (Grenoble 3-4-5 mai 1988) volume II. Nous remercions M. Philippe CHATELIN, éditeur des actes et M. Yves NOYELLE d'avoir autorisé sa publication.