COMMISSARIAT A L'ENERGIE ATOMIQUE

CENTRE D'ETUDES NUCLEAIRES DE SACLAY
Service de Documentation
F91191 GIF SUR YVETTE CEDEX

CEA-CONF -9780

L1

LANGAGES POUR LE CALCUL DES STRUCTURES

THOMAS J.B.- CHAMBON M.R.
CEA Centre d'Etudes Nucleaires de Saclay, 91 - Gif-sur-Yvette (FR).
Dept. d'Etudes Mecaniques et Thermiques

Communication présentée à : 2. Symposium GRECO-GIS on structures calculation

Giens (FR)
24-27 May 1988

Langages pour le calcul des structures

1 Introduction

2 Langage humain, langages informatiques

Le problème essentiel du langage pour le calcul des structures c'est qu'il faut faire communiquer un homme et une machine sur le sujet particulier qu'est le calcul des structures. Nous examinerons donc d'abord succintement le problème du langage humain puis celui des langages informatiques. Et nous verrons que d'une certaine façon ils sont structurés de façons antagonistes. En rappelant le problème du calcul des structures, nous donnerons quelques points et quelques niveaux d'attaque, où le problème du langage est crucial.

2.1. Langage humain

Le langage pose des questions centrales dans le problème général de la connaissances. L'après guerre a vu parmi les sciences humaines, la linguistique prendre une part prépondérente (la renommée de N. Chomsky en fut un signe), en même temps que se développait l'informatique dont un des thèmes rémanant est celui du "bon langage". Ce rôle central du langage s'explique assez bien par l'existence de liens irréfutables et irréductibles entre le langage et ce qu'il véhicule: la connaissance. Relisons les philosophes contemporains:

"La pensée, à partir d'un certain seuil est inséparable du langage surement en pratique et probablement en principe" W.V.O. Quine "Le mot et la chose" [Quine1]

"A l'époque où l'enfant est capable de soutenir une conversation rudimentaire dans sa petite communauté, son savoir de la langue et son savoir du monde forment une masse indissociable" W.V.O. Quine "Le domaine et le langage de la science" [Quine2]

"Une notation parfaite serait un substitut à la pensée" B. Russell "préface du tractatus logico-philosophicus de L. Wittgenstein" [Russell1]

Enfin citons un linguiste peu suspect de sympathie pour les idées plutôt formalistes de Russell et Quine. C. Hagege dans son livre l'homme de parôle nous dit:

"Les langues sont des modèles d'articulation du pensable grace auquels se déploie une réflexion capable d'ordonner le monde." [Hagege1]

"Le langage est à la fois, en contexte de groupe, méthode de pensée et produit de la pensée au sens général ..." [Hagege1]

Si le langage véhicule le plus souvent la connaissance, on peut se poser la question de savoir s'il y a connaissance sans langage. Les opinions sont à cet égard plus variées.

Mais cependant ne peut-on pas dire alors qu'il s'agit d'un langage d'un autre type. Les mécaniciens connaissent bien le langage graphique dont la puissance d'expression est considérable.

Toutefois si l'on retourne à une analyse fonctionnelle de l'utilisation locale d'un langage dans une communauté donnée, à une époque donnée, sans s'intéresser aux effets à long terme, on peut admettre qu'un langage sert à décrire un environnement, des problèmes et des conduites attachées à la résolution de ces problèmes. Tout ceci peut être mémorisé décrit et comuniqué.

D'une façon générale nous préférons pouvoir nous exprimer (au sens large) dans notre langage, car il correspond à notre structuration du monde. Même si parfois des discutions semblent être des querelles de mots, il n'en demeure pas moins vrai que souvent elles recouvrent des divergences quant au fond.

Le langage sert naturellement à transmettre de l'information. Pour le langage ordinaire l'émetteur et le récepteur de cette information sont des hommes. Déjà les difficultées se posent: par exemple comment faire dialoguer des spécialistes de disciplines différentes. L'utilisation d'un jargon de spécialiste sert à compacter (compiler?) l'information. Les concepts sont construit en général sur des concepts plus simples mais bien plus nombreux. Ainsi un jargon de spécialiste peut apparaitre comme construit sur des langages de plus en plus larges mais de plus en plus vastes. on peut voir cela comme une hiérarchie (en simplifiant beaucoup) de langages de moins en moins spécialisés, avec à la base le langage ordinaire et ses concepts flous. La grande force du langage naturel est la gestion collective de l'à peu près.

"Bref nous réussissons l'exploit d'utiliser un langage imprécis pour introduire un langage plus précis. Toute utilisation d'instruments se fait de cette manière. On utilise des instruments moins raffinés pour en fabriquer de plus raffinés" nous dit H. Putnam "Ce que les théories ne sont pas" [Putnam1]

2.2. Langages informatiques, vers les systèmes à base de connaissances

Ainsi que nous venons de le voir, au niveau humain langage et connaissances sont indissociables. C'est pourquoi nous adopterons la même idées pour les langages informatiques. Nous aurons ainsi une vue très large de la notion de langage informatique. Depuis l'assembleur jusqu'aux systèmes à base de connaissances, on peut ainsi hiérarchiser les langages informatiques (en s'inspirant partiellement de la classification des représentations de connaissances donnée par J.L. Laurière [Laurière1]).

L'assembleur est très proche de l'automate. Les premiers langages de plus haut niveau (Fortran par exemple) ont permis d'oublier le niveau précédent. Au dessus de ces langages ont été écrit des langages plus souples, moins

procéduraux (Gosseyn par exemple écrit en Fortran [Fouet1] ou Spiral écrit en langage C [Souchet1]). On peut considérer enfin qu'un système expert batit sur Gosseyn (par exemple) est un langage d'encore plus haut niveau donnant lieu à un langage restreint quant à son expressivité (à son domaine d'expertise) mais ayant atteint un niveau tel que la communication avec l'homme (non informaticien) est possible sinon facile.
D'une certaine façon cette hiérarchie de langage peut aussi être mise en perspective historique ainsi que l'a fait A. Kay [Kay1] dont la figure 1 est directement inspirée.

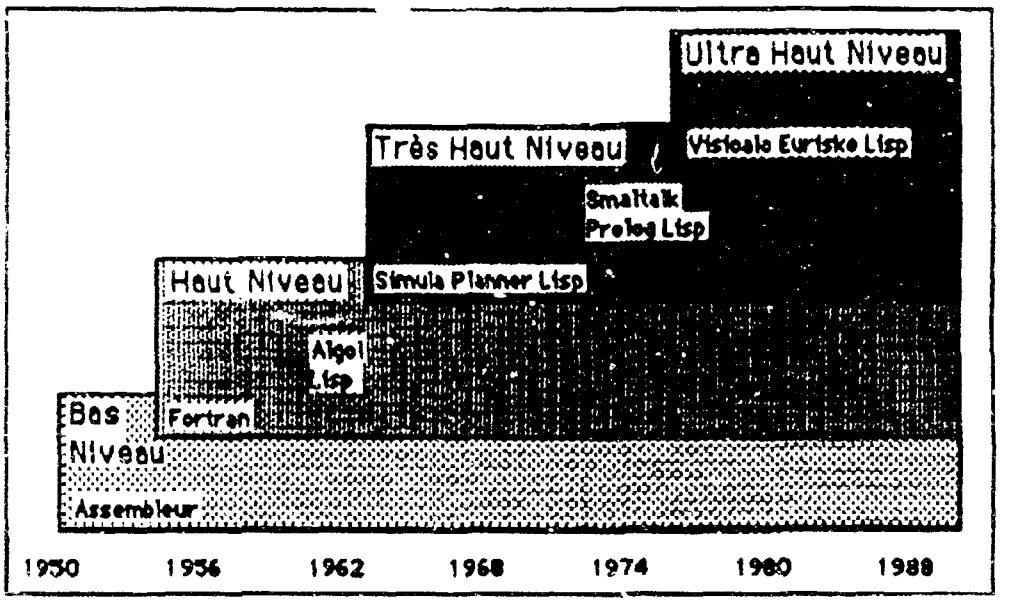

Figure 1 Hiérarchie et historique des langages (d'apres A. Kay)

Cette classification peut surprendre. D'abord par la présence de logiciels tels que Visicalc que d'aucuns n'apelleraient pas des langages, mais dans la discution précédente nous avons admis de considérer une acception large du mot langage. Ce diagramme d'autre part "mélange" des travaux issus de l'I.A. comme par exemple Eurisko, Planner et des langages classiques. Le langage Lisp a par ailleurs une place particulière puisqu'il apparait à tous les niveaux. C'est d'une certaine façon justice, car les potentialités d'évolutions du langage sont réellement unique. Celui-ci peut ainsi s'enrichir d'une version à l'autre et des fonctions que chacun se fabriquait à une période deviennent des fonctions standard à une période ultérieure. Un exemple concret et important dans l'ensemble de notre sujet est donné par les fonctionnalités liées aux objets désormais présente dans la version 15.2 de Le_Lisp [Le_Lisp1] et qui initialement ont été développées comme des aplications du langage [Hullot1]. Dans ce diagramme on notera un absent de marque: ADA qu'il faudrait sans doute mettre ainsi que le langage C au plus haut niveau des langages de haut

niveau. Enfin le dernier point discutable est la position (en ce qui concerne sa complexité) de Smalltalk historiquement le premier langage orienté objet.

Finalement en gros l'ensemble des langages informatiques apparait comme une pyramide dont la base est constituée de langages très généraux mais d'expressivité nulle et le sommet des langages particuliers des jargons d'experts (en d'autres termes des systèmes à base de connaissances) de grande expressivité mais limité quant à leur objet.

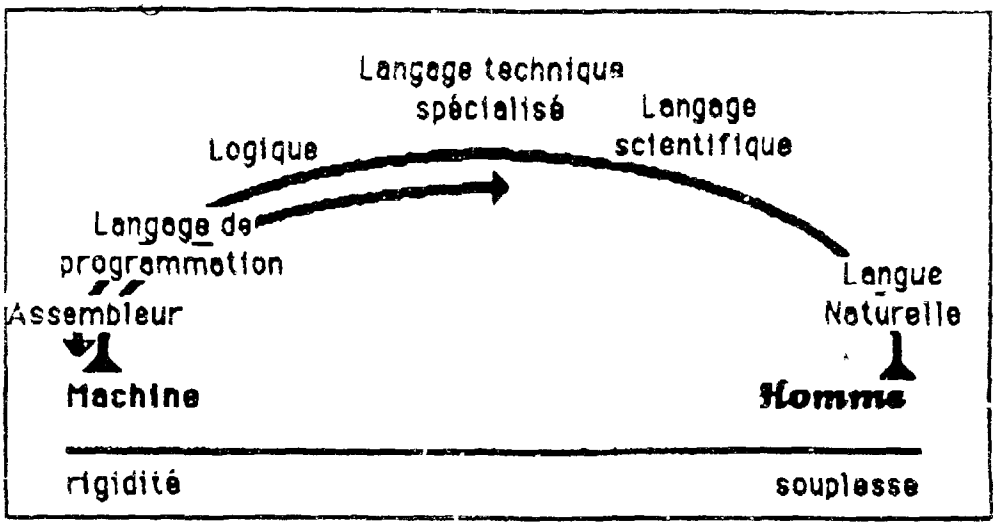

Figure 2 Communication homme machine

D'une certaine façon nous avons une situation analogue ainsi que nous l'avons vu sur le plan humain. Les jargons de spécialistes sont bâtit sur la langue naturelle générale, mais il apparait tout de suite une différence fondamentale: le langage de base informatique est le plus rigide qui soit, le langage naturel est le plus flou qui soit, acceptant polysémie et connotation sémantique. Actuellement ces langages ne se rejoignent qu'au niveau des systèmes experts ainsi que veut le suggérer la figure 2. Jusqu'à maintenant c'est l'homme qui a fait le plus gros du chemin puisqu'il a accepté de "parler" jusqu'au langages les plus inexpressifs comme le langage machine!

3 Le calcul des structures

Dans un premier temps nous allons tenter de replacer le problème du calcul des structures dans le cadre plus vaste du travail de l'ingénieur. Ensuite nous rappellerons succintement la démarche générale de résolution de problème. Comme nous avons mis à jour différents niveaux de résolution de problèmes et donc la nécéssitée de disposer de différents niveaux de langages, nous conclurons enfin sur cette question: la question de savoir quel est le meilleur niveau d'attaque.

3.1. Le problème ou les problèmes de calcul des structures

Nous allons rapidement examiner le problème du calcul des structures.
Nous partirons de la donnée au plan physique du problème. Ce problème au niveau physique est en général posé comme devant fournir des données dans des raisonnements visant à répondre à des questions globales. D'une façon générique le problème global en question peut être vu comme la recherche d'un optimum sous contrainte. Empressons nous d'ajouter que la fonction coût est en général fort mal connue, sinon inconnue. Sans ignorer leurs présences, nous n'examinerons pas les "méta-niveaux" du problème de calcul des structures. Mais il ne faut pas oublier pour l'avenir que le calcul est une action ponctuelle à l'intérieur d'un raisonnement d'ensemble.
Détaillons donc "le problème" de la résolution d'un problème de calcul des structures à partir du niveau physique seulement. Ce qui suit s'appuie sur la figure 3. Il s'agit ainsi que nous venons de le voir, en général et pour diverses raisons de pré[illegible] le comportement d'une structure mécanique dans certaines conditions. Ce problème peut être considéré comme s'exprimant naturellement en langage physique.
Le premier niveau de traduction est donc celui de la modélisation mathématique du problème. En général on transforme le problème physique en un problème de mécanique des milieux continus. Ainsi on choisi le "système mécanique" à isoler. Son comportement élémentaire est choisi en fonction d'une adéquation entre celui-ci et les effets physiques que l'on veut modéliser (par exemple tel matériau sera choisi à comportement purement élastique dans un problème, mais le même matériau sera supposé à comportement élastoplastique dans un autre cas et à comportement visco-élastique dans un troisième cas ...). Les actions extérieures à ce système sont modélisées par des conditions aux limites. Le type de connaissances nécessaires dans cette modélisation est donc (relativement) clair. Il s'agit d'une part des connaissances de la réalité "concrète" et d'autre part des connaissances de base du mécaniciens. Les premières ne sont cependant pas très faciles à formaliser. En résumé il s'agit de la traduction du problème d'un langage "physique" (plus ou moins qualitatif) en le langage mathématique de la mécanique des milieux continus.
Le deuxième niveau modélisation est la traduction numérique du problème de physique des milieux continus obtenus précédemment sur le plan de l'analyse numérique. Ainsi on devra répondre à la question de savoir quelle méthode numérique choisir. Supposons pour fixer les idées que l'on choisisse une méthode d'éléments finis spatiaux et de différences finies temporelle. Alors vont se poser tous les problèmes "techniques".
Pour les éléments finis: quel nature de maillage prendre (éventuellement quel mailleur automatique utiliser et quelles options lui fournir pour travailler); quels types d'éléments utiliser; comment traduire le mieux posible les conditions aux limites ...

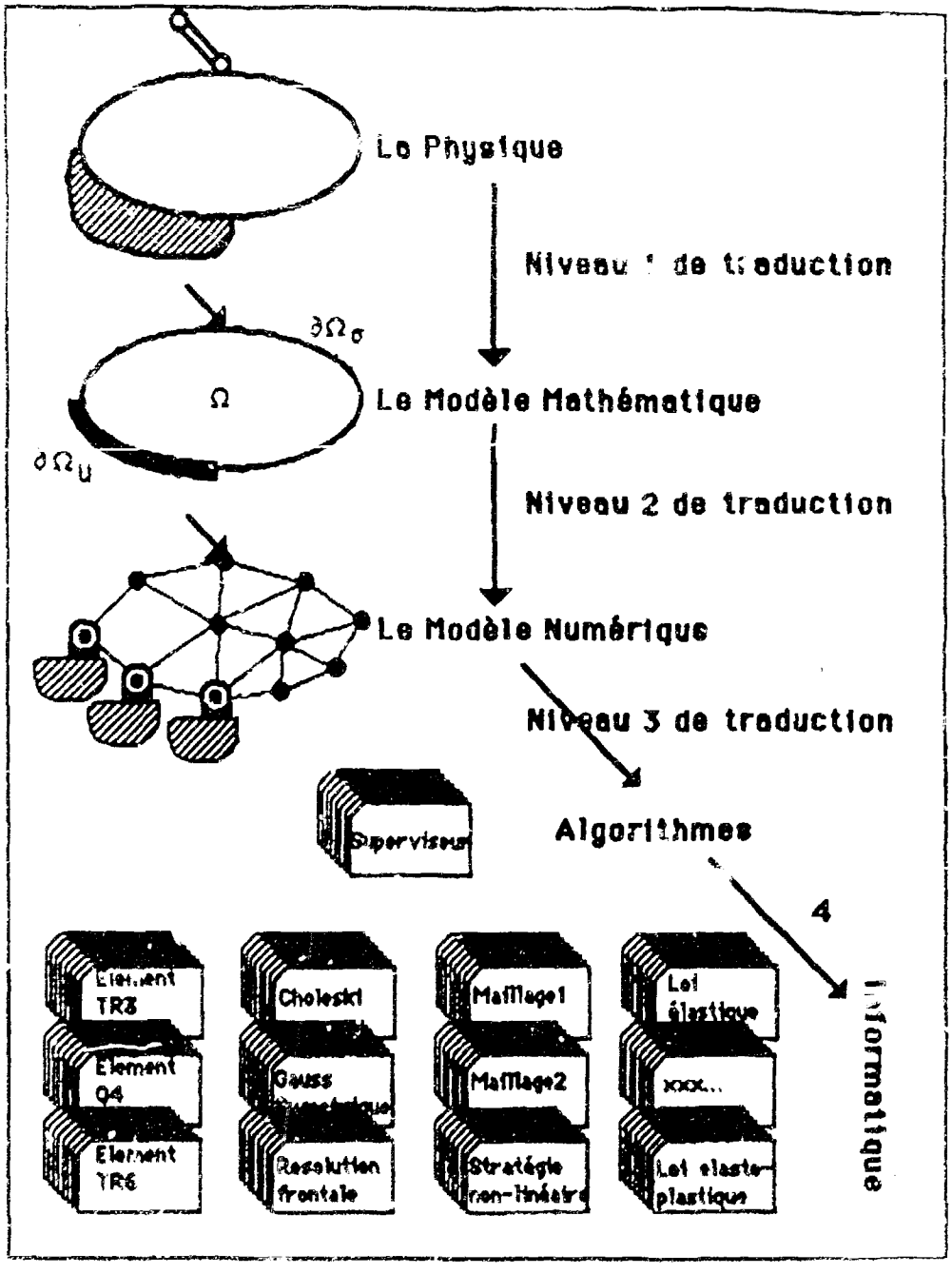

Figure 3 Le problème du calcul des structures

Sur le plan de la discrétisation temporelle: quel schéma employer, quels pas de temps utiliser ... Si le problème est en plus non-linéaire: quelle stratégie utiliser... Bref chaque utilisateur habituel d'un code de calcul sait bien d'expérience la masse de connaissance nécessaire à une telle utilisation.

Enfin le troisième niveau de modélisation est la traduction informatique du problème numérique. Les problèmes qui se posent alors sont ceux de la

disponibilité sur le code utilisé des modules correspondant aux choix numériques faits précédemment (existence des éléments, des algorithmes, des lois de comportement ...) et leurs compatibilités.
Eventuellement, il peut se passer qu'un algorithme n'existe pas dans le code utilisé et donc que l'utilisateur soit amené à le traduire (quatrième niveau de traduction), cette fois dans le langage substrat du code faisant apparaitre un quatrième niveau de langage au bas de la hiérarchie.
La présentation hiérarchique précédente est en fait quelque peu schématique car les quatre niveaux de traduction se trouvent souvent étroitement mêlés, il n'empêche qu'elle correspond à un certaine réalité du point de vue qui nous préocupe, celui des langages (On trouvera dans [Laug1] une présentation assez semblable bien qu'il distingue les choses encore plus finement vers le bas notemment entre le niveau algorithmique et le niveau macro-commande). On peut donc distinguer pour traiter l'ensemble des problèmes: le langage du physicien qui parlera de réacteur, de bride, de bielle, ... le langage du mécanicien qui parlera de sollicitations, de réponses, de conditions aux limites, ... le langage du numéricien qui parlera de fonction de forme, (sinon de fonction de base) de précision, de vitesse de convergence, de stabilité d'algorithme, ... le langage du spécialiste éléments finis qui parlera d'éléments, de maillage, de matrices de rigidité, de seconds membres, d'assemblage, ... sans parler du langage (au sens informatique cette fois) substrat dans lequel est écrit le code et dans lequel ainsi que nous l'avons vu il peut être parfois nécessaire de "descendre" (actuellement naturellement essentiellement Fortran).
Ainsi selon le niveau auquel on pose le problème de départ le langage pour le formuler et donc le résoudre sera différent.
Globalement on voit que pour réaliser des calculs il est indispensable de raisonner. Un bon calcul sous-entends en général la résolution d'un problème complexe engageant de multiples connaissances même si l'on dispose d'une bonne boite à outil puissante et complète, comme on en trouve dans les systèmes de codes de calcul modernes. Et ces connaissances ne seront jamais urtilisées deux fois de la même façon. Il serait vain de vouloir les traduire de façon purement procédurale. Ce qui importe ce sont les connaissances et leur utilisation oportuniste selon le problème rencontré. Dans le cas où certaines de ces connaissances sont absentes, on peut aboutir soit au bon résultat après tatonnements soit à ... une catastrophe, si le sens physique a d'une certaine façon disparu.

3.2. Résolution de problèmes

Ainsi donc à plusieurs niveaux, nous retrouvons un problème à résoudre c'est pourquoi nous rapellons ci-dessous les idées sous jacentes au concept de résolution de problème.

3.3. Quel niveau privilégier

Désormais la place des systèmes à base de connaissances pour le calcul des structures parait établi et donc l'utilisation de langages adaptés à ces systèmes nécessaires. Quelles conséquence cela a-t-il vis à vis d'un existant considérable en Fortran?

Si l'on veut faire cohabiter plusieurs niveaux de problèmes et donc de langages, une des réponses possibles est de privilégier un niveau et de l'imposer aux autres. En fait il serait bien plus souhaitable d'avoir un langage qui possède en quelque sorte virtuellement tous les niveaux décelés précédemment. Ceci a été tenté dans le passé récent. Ainsi peut-on interpréter les travaux fait sur Gosseyn (lui même étant écrit en Fortran) comme une première tentative. De même Spiral (lui même écrit en langage C) représente un autre essai de même nature. Autre exemple le développement fait sur ADA de ADLOG [Pitette1]. Enfin il convient de signaler dans le cadre du GRECO le projet de Griffiths [Griffiths1] dont nous reparlerons dans la suite.

Dans les trois premiers cas cités, il s'agit en fait de la construction de langage de plus haut niveau sur un langage substrat de bas niveau. Peut-on faire autrement en essayant d'ajouter des niveaux "plus bas" à un langage de plus haut niveau comme Prolog ?

4. Historique

Nous allons ici replacer ce problème de langage dans un historique du calcul des structures.

Dans les débuts du calcul des structures (il y a environ une vingtaine d'années), l'utilisateur était obligé de faire lui même toutes les traductions jusqu'à exprimer son problème dans un langage informatique (à l'époque d'ailleurs seul Fortran existait). C'est à dire que l'utilisateur étant, au moins partiellement le concepteur du code, celui-ci était d'un abord très difficile pour qui ne connaissait pas la langue ultra rigide des données. Les codes se constituaient en fonctions des problèmes par la concaténation de ... bacs de cartes !

L'ambition de généralité donna naissance aux premiers vrais codes. Ils étaient des entités fermées pour lesquelles la moindre modification était une affaire considérable. La seule façon de les utiliser était de rentrer dans le moule qu'ils fournissait. En caricaturant il s'agissait de programme ayant des fonctions quasi-uniques à déroulement séquentiel, ceci étant renforcé par le fait que ces programmes était important et donc qu'ils "tournaient" sur les machines en batch. Nous dirons qu'il s'agissait de la première génération de codes de calcul des structures.

Pour rentabiliser le travail de programmation (par exemple ne pas dupliquer une procédure de résolution de système linéaire dans un programme fluide et dans un programme solide), d'une part et pour permettre plus facilement d'insérer diverses options on a alors abouti à

la deuxième génération des codes celle des codes modulaires. Reste alors à gérer les modules, non pas du point de vue du développeur, mais du point de vue de l'utilisateur. Sur des codes de deuxième génération se sont alors naturellement développé des langages de commandes permettant alors à l'utilisateur un langage de plus haut niveau pour exprimer son problème. Ce langage de commande permet de piloter le superviseur de code ainsi qu'on peut le voir dans [Rousset1] ou dans [Dumas1]

D'une certaine façon, nous voudrions introduire dans ce chapitre l'aboutissement de cette deuxième génération de codes de calcul ce que l'on pourrait appeler la deuxième génération et demi, en ce sens qu'au dessus du superviseur de code agit une couche incorporant certaines connaissances du code et des problèmes à résoudre.

Comme dans l'informatique en général, l'histoire du calcul des structures représente donc une montée de la part des possibilités de la machine dont le langage devient de plus en plus souple l'homme n'est plus nécessairement obligé de faire tout le chemin.

En plus de cet historique synthétique envisageant une certaine montée du calcul des structures dans la hiérarchie des langages, il convient d'examiner le problème du langage substrat. Dans les codes existant Fortran domine comme chacun sait. Ce n'est pas sans poser certains problèmes. La reconnaissance comme "discipline" de l'informatique: du génie logiciel au travers en particulier de la problématique de la preuve de programme et de la programmation structurée, interpelle le monde du calcul des structures dont la masse d'existant en Fortran a freiné inertiellement les changements. En particulier la crainte de l'absence de portabilité et les difficutés bien connues de communication entre les différents langages ont perennisé Fortran dans son rôle de langage universel de calcul des structures. Pascal n'a pas changé cet état de fait, ADA le fera-t-il? Certains le pensent et argumentent leur opinion [Gendre1], mais l'inertie n'est-elle pas trop grande?

Il faut cependant être attentif au fait que la micro-informatique a fait naitre des codes destinés en particulier aux P.M.E. libérés des pesanteurs du passé [colloque1]. On a pu ainsi voir naitre dans les débuts de la micro-informatique des codes écrits en Basic, ce qui n'est pas vraiment un progrès, mais aussi actuellement des codes écrits en Pascal, le langage historique de la programmation structuré.

Y aura-t-il convergence de la notion de génie logicielle de méthode de programmation orienté objets et de l'émergence en I.A. des langages orienté objets pour fournir au calcul des structures les base d'une éventuelle (et souhaitable) troisième génération ainsi que nous le verrons dans le paragraphe 5.

5 Que peut-on faire?

5.1. Le mariage à la mode I.A. Calcul

Un des grands apports de l'intelligence artificielle aura été de montrer que le traitement des connaissances était mal adapté aux langages informatiques existant. Le travail des pionniers de l'I.A. a souvent consisté pour une grande part à élaborer des langages, mais en utilisant un langage substrat mieux adapté que les langages "scientifiques" usuels. Dans les débuts Lisp (ou plutôt ses nombreux dialectes) ont été utilisés exclusivement. L'incompatibilité de ces dialectes et la lenteur des interpréteurs n'ont pas été pour peu dans les réticences de l'utilisation industrielle de l'I.A.. La situation semble se stabiliser au travers de Common Lisp et de Le_Lisp. Par ailleurs de plus en plus de développeurs en I.A. utilisent aussi Prolog.

Cependant que ce soit Prolog ou Lisp ces langages à dominantes symboliques ne sont pas utilisables pour des programmes de calcul, leur performances numériques sont inacceptables. Si on les utilise en calcul des structures il faut donc qu'il puissent communiquer avec les "vrais" langages de calcul des langages totalement procéduraux, or d'une part ils ne sont pas complètement standardisés et d'autre part ils communiquent assez difficilement avec les autres langages.

La cohabitation langage dédié à l'I.A., langage de calcul bien que difficile a l'avantage de pouvoir réutiliser l'existant (en particulier en Fortran).

C'est la voie choisie par les chercheurs de l'INRIA qui ont écrit un système en Le_Lisp permettant d'engendrer et de piloter l'algorithme de résolution d'un problème, les modules de calcul étant ceux de MODULEF donc écrit en Fortran [Laug1].

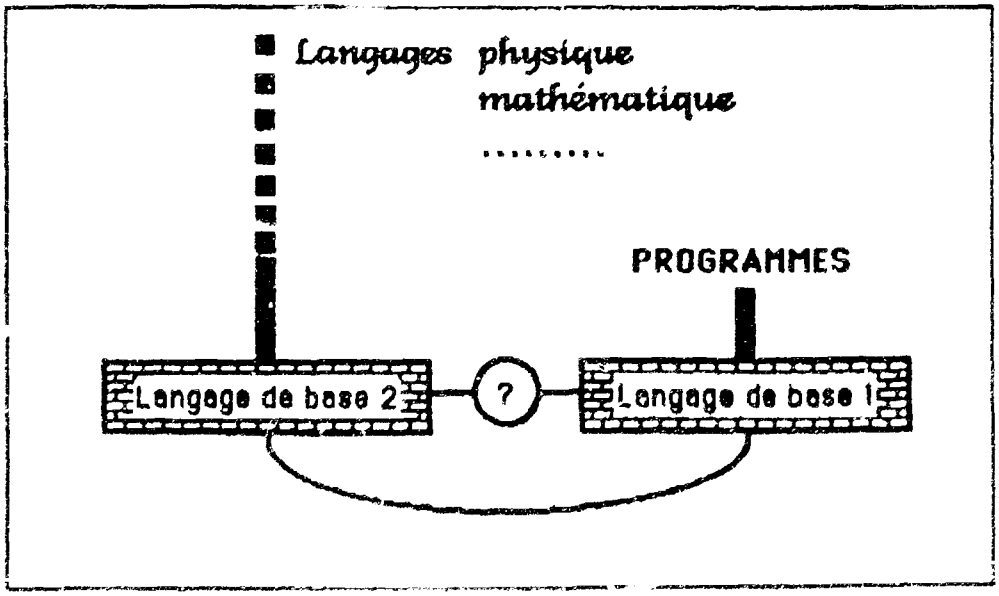

Figure 4 Le mariage symbolique procédural

C'est également bien qu'avec variante la voie choisie au C.E.A. [Dumas1]. La

variante consiste en ce que le langage d'I.A. utilisé est un langage reconstruit sur un substrat de Langage C [Souchet1].
Nous avons schématisé cette démarche sur la figure 4 qui pose le problème de la communication. En attendant des futures communications simples entre langages de natures différentes, celles-ci ne peuvent se faire qu'à plus bas niveau.
Les modules de codes actuels sont écrits par des équipes diverses même au sein d'un organisme unique. Il est nécessairte d'en séparer le contenu interne des liaisons externes pour les intégrer dans la bibilothèque des procédures utilisables. Les travaux présentés dans la suite vont plus loin, ils consistent à partir d'un langage de commande du code:
- à construire des bases de connaissances nécessaire à son exploitation
- à élaborer des logiciels d'assistance à la mise en œuvre des modules au sein de calculs complexes et non standard pour une utilisation en batch ou interactive. Ainsi que nous l'avons déjà dit nous verrons diverses illustrations de cette démarche sur les superviseurs de codes dans la suite.

5.2. Utilisation d'un langage d'I.A. basé sur un langage dédié au calcul scientifique

En poussant plus loin la structure évoquée précédemment en ce qui concerne le travail réalisé au C.E.A., une idée est d'utiliser le langage de calcul des structures lui-même pour construire un langage d'I.A.. C'est d'une certaine façon ce qui a été réalisé au travers de Gosseyn qui comme chacun sait était écrit en Fortran, dans les thèses passées à l'école normale de Cachan [Reynier1] [Trau1] [Tichklewitch1] par exemples.

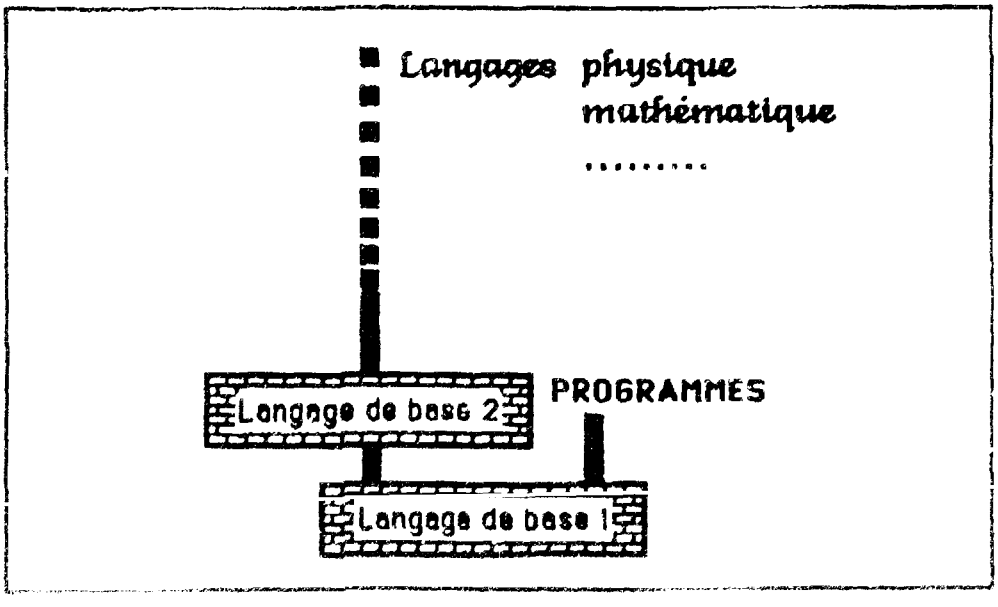

Figure 5 Un langage symbolique fils d'un langage procédural

Nous avons représenté ce principe sur la figure 5 dans laquelle le langage de base 1 est celui du calcul des structures (Fortran dans notre exemple) et le langage de base 2 est le langage d'I.A. basé sur le précédent (Gosseyn dans notre exemple.

Comme précédemment, la réutilisation de l'existant en calcul des structures est évidemment possible.

5.3. Un nouveau langage intégré

Un autre solution envisageable est l'élaboration d'un nouveau langage qui serait à la fois efficace en calcul et adapté à des écritures symboliques. En imaginant ce que pourrait être un code de calcul des structures écrit dans ce langage avec de plus des implémentations de langages symboliques de plus haut niveau, on obtient la structure représentée sur la figure 6.

C'est dans cette perspective que se situe la travail fait par Griffiths et son équipe [Griffiths1] dont on pourra trouver la description plus loin.

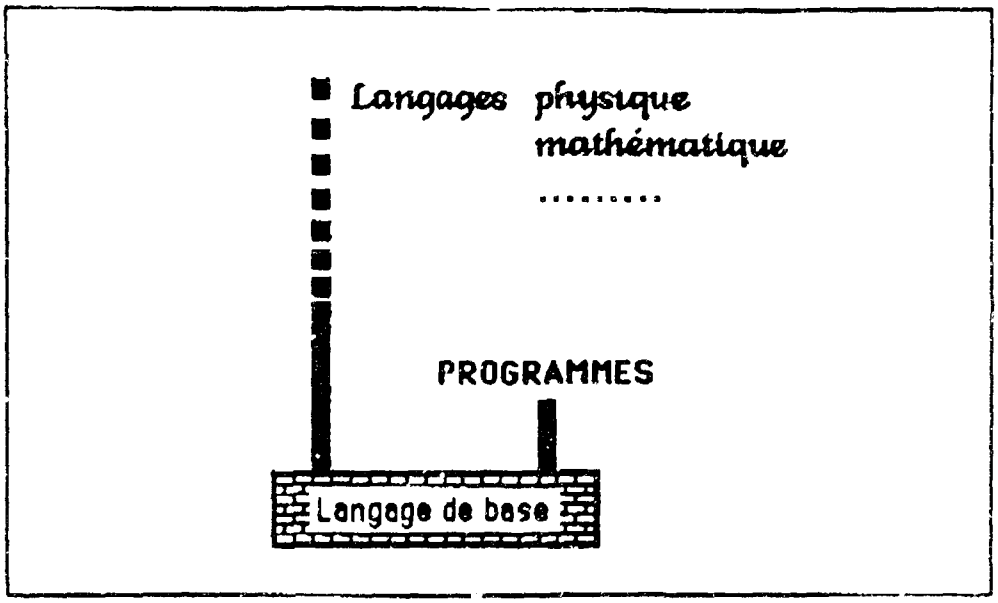

Figure 6 un langage de base à la fois symbolique et procédural

5.4. Perspective objets

Dans l'optique de cette intégration, il convient de remarquer une convergence actuelle en informatique autour de l'orientation objet.

D'une part du coté des langages "classiques" il aparait en génie logicel une tendance à la remise en cause de la programmation structurée pure. Bien qu'étant naturellement un grand progrè par rapport à l'absence de méthode préexistant, ses inconvénient sont patents. Une analyse d'un problème de programmation dans l'optique de la programmation structurée serait parfaite si l'analyse descendante préconisée faisait apparaitre une arborescence dans laquelle toutes les braches, les sous branches, et les feuilles seraient indépendantes. En pratique, et les spécialistes

d'intelligence artificielle le savent depuis longtemps, une décomposition de problèmes en sous problèmes conduit rarement à un indépendances des sous problèmes de "base". C'est pourquoi se développe actuellement une méthode semble-t-il plus efficace que la structuration: la méthode de conception orientée objets [Ros ..]. Il est a noté que cette école de conception de logiciels est très liée au langage ADA.
Il convient de rappeller par ailleurs qu'en inteiligence artificielle, après avoir élaboré des systèmes "plats" où la connaissance était la plus déstructurée possible et présente seulement sous forme de grains (chunk) de connaissances, les systèmes actuels sont organisés avec une structuration plus grande. Cette structuration se fait autour du concept d'objet dont le premier langage batit sur cette notion est Smalltalk.
La plupart des sytème d'I.A. actuels et en particulier la quasi totalité des générateurs de systèem du marché utilise de façon non exhaustive une structuration en objets. Et il faut bien dire qu'il est difficile d'imaginer de s'en passer lorsque comme c'est notre cas nous voulons travailler symboliquement sur des objets (au sens physique cette fois) mécaniques.
S'il ne faut pas confondre les langages orientés objets et la méthode de conception orienté objet, il n'en reste pas moins que l'analogie va au dela de l'utilisation commune du mot objet. Et s'il est vrai que les langages orientés objets traitent essentiellement de la connaissance symbolique alors que la méthode orientée objet s'applique à des programmes procéduraux, il n'empêche qu'ils partage des concepts en commun. On voit bien alors l'avantage que le calcul des structures tirerait de cette nouveauté si un langage à la fois procédural et symbolique comme le projet présenté dans la suite par [Griffiths1] possédait les caractéristique d'un langage orienté objets.
Une présentation de ce que pourrait être ce calcul des structures de "troisième génération" est présenté dans la suite par [Fouet2].
Il peut sembler remarquable de voir s'opérer une convergence aussi forte sur certains thèmes, comme l'approche objets, qui risque par contrecoup d'être perçue comme une "tarte à la crème" et d'être victime d'une défiance justifiée vis à vis de certains effets de mode.
Aussi convient-il de dégager en quoi l'approche objet constitue une évolution naturelle. Avec l'accroissement du nombre de concepts à gérer et le retour en force au sens large des données sur le devant de la scène, il est inévitable que la gestion de l'ensemble des entités et des masses de connaissances nécessire à leur utilisation intelligente occupent une place centrale. Ce qui change c'est le niveau d'abstraction. On peut considérer en citant Rosen [Rosen1]: "qu'un programme n'est plus une suite d'instruction à faire exécuter par une machine, mais une certaine description du monde réel". On ne va plus se préoccuper de la seule description des actions, mais de la représentation cohérente de l'ensemble des entités concernées.

On retrouve différents niveaux qu'on pourrait qualifier d'"ontologique":
le niveau statique correspondant aux classes, propriétés, relations ...
le niveau dynamique correspondant aux actions opérations ...
le niveau "épistémique" correpondant aux contraintes, aux règles de choix vis à vis des diverses utilisations de l'objet.
Cette conception de l'objet incluant toutes les valeurs et opérations nécessaires et suffisantes pour le caractériser en fait "une unité à forte cohésion et faible couplage" ([Rosen1]). il est enfin plus facilede définir des objets parallèles et d'utiliser une méthodologie unique pour les systèmes séquenciels ou parallèles. Les relations de dépendances deviennent plus faciles à gérer. Les capacités d'autonomie et de copération des objets sont accrus, ce qui conduit à une vraie modularité statique et dynamique.

6 Conclusion

La méthode de conception objet se trouve tout naturellement à mi chemin entre d'une part la programmation structurée et sa structure arborescent stricte et d'autre part l'absence de méthode qui prévalait auparavent et sa logique monolithique.
De même les langages orientés objets se trouvent à mi chemin entre une connaissance écrite totalement en vrac et de ce fait difficile à gérer, et des connaissances totalement structurées dans une structure figée.
Bref la notion d'objet et ses applications tant méthodologiques que langagières représentent peut-être le point d'équilibre actuel et pour les quelques années qui vont suivre.

7 Bibliographie

[Quine1] W.V.O. Quine "Le mot et la chose"
[Quine2] W.V.O. Quine "Le domaine et le langage de la science"
[Russell1] B. Russell "préface du tractatus logico-philosophicus de L. Wittgenstein"
[Hagege1] C. Hagege "L'homme de parôle"
[Putnam1] H. Putnam "Ce que les théories ne sont pas"
[Lauriere1] J.L. Lauriere "Représentation et utilisation des connaissances" T.S.I. Vol 1 n°2 pp 109-131 1982
[Fouet1]
[Souchet1]
[Kay1] A. Kay "Les logiciels" Pour la Science n° 85 pp 14-25 1984
[Le_Lisp 15.2] Manuel de référence
[Hullot1] J.M. Hullot "Alcyone la boite à outil Objets" Rapport technique n°60 INRIA 1985
[Pitette1] G. Pitette "ADLOG: comment introduire des composants deductifs dans une application ADA" Genie Logiciel n°9 Novembre 1987 p. 58-61
[Griffiths1] M. Griffiths "xx" ce volume
[Laug1] P. Laug A. Perronet "Langages pour le calcul des structures:

expérience INRIA-MODULEF" ce volume
[Rousset1] M. Rousset "Un interpréteur procédural pour SYSTUS" ce volume
[Dumas1] M. Dumas "Prototype de superviseur Système expert d'assistance à l'exploitation de système de codes modulaires" ce volume
[Gendre1] P. Gendre M. Hittinger "ADA pour laprogrammation de la méthode des éléments finis: une alternative possible?" Genie Logiciel n°9 Novembre 1987 pp 38-40
[colloque1] colloque à retrouver sur le calcul des structures sur micro-ordinateurs.
[Rosen1] J._P. Rosen "Méthode de conception orientée objets" Genie Logiciel n°9 Novembre 1987 pp 16-21
[Reynier1]
[Trau1]
[Tichkiewitch1]
[Fouet2] E. Collain J.-M. Fouet G. Regnier "Pour uncalcul des structures orienté objets" ce volume