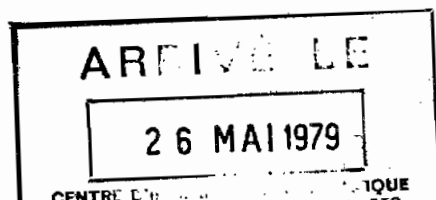


C568

**SYNTHESE SUR LES LANGAGES ET LES SYSTEMES
D'AIDE A LA PROGRAMMATION**

Cette synthèse a été rédigée par **A. Guillon** de la **GIXI**
dans le cadre du contrat **SESORI n° 77.142**



BIBLIOTHEQUE DU CERIST

SOMMAIRE

INTRODUCTION	1
PREMIERE PARTIE : L'ETAT DE L'ART ET SES PROLONGEMENTS	4
CHAPITRE 1 : Le métier de programmation et ses difficultés propres	5
1 - Les limites mathématiques inhérentes à la complexité	6
1.1. Les grandes classifications logiques	6
1.2. La complexité dans les théories décidables	8
1.3. Bilan dans le monde réel.....	10
2 - Les difficultés majeures de la programmation	11
2.1. Prolifération des éléments à gérer.....	11
2.2. Les limites à un contrôle humain centralisé.....	13
2.3. L'évolution des logiciels	14
2.4. L'adéquation au besoin originel.....	15
CHAPITRE 2 : AMENAGEMENTS METHODOLOGIQUES : LES IDEES	20
1 - Les principes très généraux de l'aide à la programmation	20
1.1. L'approche synthétique	21
1.1.1. Le pourquoi de l'approche descendante	21
1.1.2. L'approche synthétique	23
1.2. La réduction de la difficulté	25
1.2.1. Divisions de la difficulté	25
a) Décomposition en phases	
a1) Décomposition de la programmation en étapes	
b) Décomposition modulaire	
c) Décomposition des outils traditionnels	
1.2.2. Restrictions de la variabilité informatique des objets.....	32
a) Réduction de la variabilité dans les langages de programmation	
b) Limitations de variabilité dans les manipulations	

1.3. Les contrôles de cohérence	41
1.3.1. La documentation.....	42
1.3.2. Les vérifications.....	45
1.3.3. Contrôle des restrictions.....	48
2 - Recherche d'une certaine exhaustivité de ces méthodes.....	49
2.1. Exhaustivité de documentation et d'écriture algorithmique...	49
2.2. Quelques règles cherchant une exhaustivité relative à certains points	50
2.3. Cas des modifications internes aux projets.....	50
3 - Réaction face à l'impossibilité de perfection.....	53
3.1. Récupération de programmes.....	53
3.1.1. Paramétrisation.....	54
3.1.2. Bibliothèques d'algorithmes	55
3.1.3. Récupération de fragments de programmes.....	55
3.2. Prise en charge de l'erreur.....	56
CHAPITRE 3 : LES AMENAGEMENTS AUTOMATISES : LES OUTILS	58
1 - Généralités	58
1.1. Sur l'aspect conversationnel	58
1.2. Les maquettes.....	60
2 - Outils relatifs à la réduction de la difficulté	62
2.1. Division de la difficulté.....	62
2.1.1. Décomposition en étapes - transformation de programmes	62
2.1.2. Décomposition modulaire	65
2.1.3. Décomposition d'outils traditionnels.....	66
2.2. Restriction de la variabilité des objets.....	67
2.2.1. Réduction de la variabilité des langages de programmation	68
2.2.2. Réduction de la variabilité dans les manipulations de programmes	69
a) Editeurs de textes	
b) Contrôles inter-modules	
2.2.3. Réduction de la variabilité dans les manipulations générales	71

3 - Outils relatifs aux contrôles de cohérence.....	74
3.1. Support de cohérence des aspects méthodologiques.....	74
3.1.1. Aide à la documentation.....	74
a) Gestion de documentation	
b) Aide à la construction de documentation	
3.1.2. Outils de vérifications.....	79
a) Contrôles des restrictions et limitations	
b) Vérification sur les manipulations	
c) Mise au point	
3.2. Certification des programmes.....	84
3.2.1. Preuve de programmes.....	85
a) Sémantique opérationnelle	
b) Sémantique dénotationnelle	
3.2.2. Preuve partielle des programmes.....	89
3.2.3. Synthèse de programmes.....	90



DEUXIEME PARTIE : VERS LES REPRESENTATIONS MATHÉMATIQUES

CHAPITRE 1 : DES FAITS	94
1 - Les recherches sur les langages.....	95
2 - Les spécifications.....	99
3 - Le cas d'APL	101
3.1. Les critiques.....	102
3.1.1. Les structures de contrôle.....	102
3.1.2. Les types.....	103
3.1.3. Les structures de données.....	103
3.2. Raisons d'un succès.....	104
3.2.1. Conversationnel ?	104
3.2.2. Opérateurs globaux ?.....	105
3.2.3. Conception par objets mathématiques ?.....	105
CHAPITRE 2 : DEUX EXEMPLES	109
1 - Les langages dits de très haut niveau.....	109
1.1. La partie spécification.....	110
1.1.1. Les objets.....	110
1.1.2. Les fonctions.....	111
1.1.3. Les relations.....	111
1.2. Les réalisations	112
2 - Z0-Z1 d'Abrial.....	114
2.1. Le langage Z0.....	114
2.1.1. Le matériau.....	114
2.1.2. Les spécifications.....	115
2.2. Le langage Z1.....	116
3 - Vers le règne de la vision synthétique globale.....	119
3.1. Les difficultés.....	119
3.2. Une perspective vraisemblable.....	120
CONCLUSION	122

INTRODUCTION

S'il est une constatation éculée, c'est bien l'observation des évolutions contraires qu'ont démontré en coût logiciel et coût matériel les projets informatiques, évolutions qui ne cessent de s'accroître et se traduisent par la part prépondérante qui revient désormais au premier dans la plupart des réalisations récentes. Il en résulte bien évidemment une actualité toujours plus pressante du problème d'aide à la programmation. Dans le contexte actuel, où la plus grande part de la facture finale émane de la construction du logiciel, tout aménagement ou amélioration de cette dernière voit un impact immédiatement sensible sur le prix de revient de l'ensemble.

En outre, la charge résultant de la maintenance peut, elle-même dans certains cas, encore outrepasser le coût de la réalisation proprement dite. Ceci alourdit à ce point les opérations que d'aucun considère cette maintenance comme le boulet majeur qui pèse sur tout projet et qui déboutonne bien des rêves informatiques, qui ne demandaient qu'à germer.

C'est dire que le double objectif, réaliser un logiciel de façon efficace d'une part, et le construire robuste d'autre part, constitue un des centres de préoccupation essentielle des responsables techniques dans toute activité où sévit la programmation. Telle est également, pour notre part, l'interrogation originelle qui nous a conduit à aborder le présent travail.

Néanmoins quelques précisions s'imposent quant aux perspectives qui ont présidé à son élaboration.

Selon le titre même, la principale caractéristique de ce travail est un effort de synthèse. Il faut entendre par là que l'on s'est abstenu d'approcher d'une quelconque façon l'esprit d'une "somme", d'une encyclopédie relatant aussi exhaustivement que possible les diverses expériences et techniques ayant trait à la question.

Bien au contraire, la perspective s'est attachée à dégager quelques grandes lignes et s'est traduite par une tentative d'organisation dans une compréhension synthétique globale, qui puisse au moins s'avérer de quelque secours pour le non-spécialiste.

Ceci a plusieurs conséquences immédiates. En ce qui concerne le contenu technique précis des divers points abordés, il n'a jamais paru nécessaire de relater aussi précisément que possible les expériences et recherches correspondantes. Seuls les caractères essentiels, communs à plusieurs, ou bien au contraire divergents, ont été pris en compte. A titre d'exemple, l'aspiration à un logiciel robuste, évoqué quelques lignes plus haut, voit plusieurs tendances aborder différemment la question. Certains, plus portés il est vrai à des considérations sur des logiciels très généraux, tels les systèmes d'exploitation, cherchent à développer des techniques où l'erreur est un événement quotidien, dont on cherche a priori à prendre la mesure et éviter ses trop grands dégâts (confinements d'erreur, récupération, reconfiguration automatique, etc...). D'autres entrevoient la possibilité de certains logiciels certifiés, c'est à dire ayant fait l'objet de vérifications démontrant rigoureusement qu'ils ne comportent aucune faute : la preuve de compilateurs, ou tout au moins de parties de compilateurs, peut par exemple n'apparaître pas tout à fait utopique pour un avenir pas trop lointain.

Or dans aucun des deux cas de figure, la présentation détaillée des techniques correspondantes ne sera abordée. Tout au plus, une ébauche des idées ou thèmes qui y président sera évoquée.

Une autre conséquence de cette approche réside en ce que les recherches, expériences, réalisations, ou systèmes méthodologiques rencontrés sont éclatés et se voient répartis dans les différents points généraux constituant l'ossature de notre tentative de vision synthétique, lorsqu'ils en constituent des images fort représentatives. En outre, la citation dont ils font l'objet est, naturellement, à lire comme une illustration d'un thème ou d'une idée et certes pas comme une référence prétendant qualifier la valeur intrinsèque de la réalisation correspondante.

Ainsi donc, l'ensemble du sujet nous est apparu se stratifier en deux catégories plus homogènes. La première qui se développera dans la première partie contient ce que l'on peut observer de l'état de l'art actuel, ainsi que toutes extensions et recherches qui s'y greffent suffisamment naturellement pour laisser espérer une possibilité de réalisation industrielle dans un avenir pas trop lointain. Le fil de lecture qui nous est apparu le plus simple et concret consiste à rappeler dans un premier chapitre les caractéristiques et difficultés propres aux métiers de la programmation. Cette vision permet plus naturellement d'exposer, en un second chapitre les principes d'aménagements qu'il semble que l'on ait tenté d'apporter à cette réalité. Dès lors, en un troisième chapitre, on pourra envisager ce que l'automatisation informatique elle-même peut ajouter à la mise en oeuvre des moyens résultants.

Le malheur c'est qu'existe dans le monde réel un certain nombre d'anomalies aux recommandations et développements émanant de cette première partie, dont le cas d'APL constitue une illustration assez spectaculaire. Or il se trouve que ces constats d'anomalies s'estompent considérablement lorsqu'ils se trouvent relus au travers des principales idées vers lesquelles semblent curieusement converger les recherches dites de "langage de très haut niveau" et assimilés, langages de spécifications, etc...

C'est pourquoi, même si les résultats que l'on peut attendre de ce mouvement semblent nettement futuristes, néanmoins les perspectives qui les gouvernent nous sont apparues suffisamment fondamentales pour être évoquées ici, et constituer la trame de la seconde partie de ce document.