Titre: Title:	Analyse statistique des métriques du logiciel
Auteur: Author:	Daniel Coupal
Date:	1990
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Coupal, D. (1990). Analyse statistique des métriques du logiciel [Mémoire de maîtrise, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/59253/

UNIVERSITÉ DE MONTRÉAL

ANALYSE STATISTIQUE DES MÉTRIQUES DU LOGICIEL

par

Daniel COUPAL
DÉPARTEMENT DE GÉNIE ÉLECTRIQUE
ÉCOLE POLYTECHNIQUE

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU GRADE DE MAÎTRE ÈS SCIENCES APPLIQUÉES (M.Sc.A.)

Mai 1990

ISBN 0-315-58943-4

Canada

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE

Ce mémoire intitulé:

ANALYSE STATISTIQUE DES MÉTRIQUES DU LOGICIEL

présenté par Daniel COUPAL

en vue de l'obtention du grade de Maître ès Sciences appliquées

a été dûment accepté par le Jury d'examen constitué de:

M. Louis Granger, Ph. D. , président

M. Bernard Clément, Ph. D.

M. Pierre N. Robillard, Ph. D.

"There are very few things which are not capable of being reduc't to a mathematical reasoning; and when they cannot, it's a sign our knowledge of them is very small and confus'd"

John Arbuthnot (1692) [MERT84]

SOMMAIRE

Ce mémoire présente une analyse détaillée du domaine des métriques statiques du code source jusqu'à ce jour. Des hypothèses quant aux comportements des métriques sont vérifiées avec des données provenant d'expériences, de projets commerciaux ainsi que de la littérature.

La recherche présente une revue exhaustive des métriques de la littérature. Celles-ci sont répertoriées selon leurs auteurs et une formalisation mathématique du calcul de la métrique en fonction de la théorie des graphes est proposée.

L'analyse discriminante démontre la dépendance interne des routines à l'intérieur d'un projet envers les programmeurs ou le projet lui-même.

L'analyse factorielle démontre le faible nombre de dimensions mesurées par des ensembles de métriques sur des projets. Un nouveau modèle en est déduit pour l'analyse de larges projets avec des ensembles de métriques.

Les résultats d'une expérience sur la complexité intuitive qu'ont les individus face au code source sont rapportés.

Finalement, des modèles de représentations de données et de leurs relations sont proposés.

ABSTRACT

This document is a comprehensive analysis of static metrics of source code. Hypothesis on the behaviour of metrics are verified with data from experiments, commercial projects and published papers.

This research presents an exhaustive list of metrics from papers. These metrics are classified under authors and mathematical formalism based on graph theory is used to describe each of these.

Discriminant analysis shows dependant relationships within a project for programmers or project itself.

Factor analysis shows that few dimensions are mesured by sets of metrics. A new model is proposed for analyzing large projects with sets of metrics.

Results from an experiment on intuitive complexity related to source code are reported.

Finally, models for data representation and their relationship are proposed.

REMERCIEMENTS

J'aimerais remercier en tout premier lieu le Dr. Pierre N. Robillard, mon directeur de recherches, pour ses encouragements et le temps qu'il a consacré à mon égard. Je remercie également M. François Coallier, directeur du secteur d'Ingénierie de qualité chez Bell Canada, pour sa co-direction dans le cadre du projet DATRIX.

Je suis reconnaissant au conseil de recherches en sciences naturelles et en génie du Canada (CRSNG) ainsi que Bell Canada Inc. pour les bourses qu'ils m'ont octroyées.

Les nombreuses discussions avec mes collègues de travail du laboratoire de Recherche en Génie Logiciel (RGL): André Beaucage, Alain Grenier, Mario Simoneau et Jean-Bernard Trouvé m'ont aidé à faire la lumière sur les points les plus obscurs de ma recherche.

Je ne voudrais pas oublier ma famille et mes amis qui ont toujours cru en moi. L'étincelle de fierté qu'ils m'ont donnée a contribué énormément à mon cheminement intellectuel.

Finalement je remercie les autres membres de l'équipe RGL ainsi que les professeurs Yves Lepage et Bernard Clément pour leurs conseils sur la partie statistique.

Table des matières

Liste des figures

Liste des tableaux

Table des symboles

Ensembles

$\mathbb{R}$	Ensemble des nombres réels.
$\mathbb{N}$	Ensemble des entiers positifs ou nuls.
$\mathbb{Z}$	Ensemble des entiers positifs, négatifs ou nuls.
$\|A\|$	Cardinalité de l'ensemble A.
$a \in A$	L'élément a fait partie de l'ensemble A.
$a \notin A$	L'élément a ne fait pas partie de l'ensemble A.
$A \times B$	Produit cartésien de A par B.
$\forall x$	Pour tout x.
$\exists x$	Il existe un x.
$\mid$	Tel que.

Fonctions mathématiques

$\mathrm{Ln}(x)$	Logarithme népérien de x.
$\mathrm{Log}(x)$	Logarithme en base 10 de x.
$\mathrm{Abs}(x)$	Valeur absolue de x.

Matrices

$\underset{\sim}{X}$	Vecteur des éléments X_1 à X_p
X'	Matrice transposée de la matrice X
X^{-1}	Matrice inverse de la matrice X

Pour un graphe G

V	Ensemble des sommets dans le graphe.
E	Symbolise l'application qui définit un graphe et du même coup l'ensemble des arcs ou des arêtes du graphe.
$E_G(v_x)$	Ensemble des arcs incidents à v_x.
$E_G^+(v_x)$	Ensemble des arcs incidents extérieurement à v_x.
$E_G^-(v_x)$	Ensemble des arcs incidents intérieurement à v_x.
$\Gamma_G(v_x)$	Ensemble des voisins du sommet v_x.
$\Gamma_G^+(v_x)$	Ensemble des successeurs du sommet v_x.
$\Gamma_G^-(v_x)$	Ensemble des prédécesseurs du sommet v_x.
$C_G(v_x)$	Ensemble des sommets reliés au sommet v_x par une chaîne.
$d_G(e_x)$	Degré du sommet e_x.
$d_G^+(e_x)$	Demi-degré extérieur du sommet e_x.
$d_G^-(e_x)$	Demi-degré intérieur du sommet e_x.

I. Le mémoire

A. Historique du projet

Le projet de maîtrise en Génie Logiciel fait suite à une expérience acquise dans ce domaine au cours des huit mois qui ont précédé la fin de mon baccalauréat.

Le projet d'un outil logiciel évaluant la mesure d'interconnectivité du code source (POLEMICS) comme travail d'été, l'évaluation des résultats de la métrique comme projet de fin d'études et l'étude de différentes métriques dans le cadre d'un projet UPIR ont été une très bonne introduction au domaine du génie logiciel.

Je tiens à souligner la collaboration de l'équipe de travail du Laboratoire de Recherche en Génie Logiciel qui a permis la réalisation efficace de mes recherches.

B. Organisation du mémoire

Le chapitre III décrit les différents aspects de la mesure statique du logiciel. On y décrit les graphes de contrôles, les métriques du logiciel ainsi que les outils de mesure du logiciel.

Le chapitre IV présente les notions statistiques élémentaires nécessaires à la compréhension de ce mémoire.

Le chapitre V présente un prototype d'interfaçage des données pour l'outil DATRIX avec d'autres logiciels.

Le chapitre VI décrit les données expérimentales aux analyses statistiques, leur collectes, etc.

Les chapitres VII et VIII rapportent respectivement des résultats d'analyses statistiques sur les métriques et les résultats d'expériences sur la notion de complexité intuitive.

Finalement la conclusion rappelle les points importants du mémoire et donne l'occasion de proposer une orientation aux futures recherches dans la mesure statique du code source.

II. Introduction

De nos jours, la majeure partie des coûts d'un système informatique est accaparée par la portion du logiciel qui fonctionne sur l'ordinateur (figure 2.1).

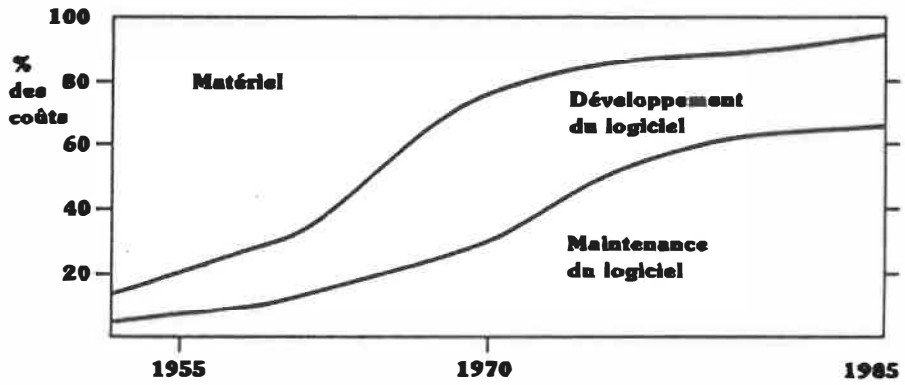

Figure 2.1: Évolution de la répartition des coûts matériel-logiciel [ROBI85]

De ces coûts, une forte proportion va à la maintenance des systèmes dans le but de les adapter, les améliorer ou les corriger (figure 2.2).

Cette répartition des coûts a amené la communauté scientifique à se poser des questions sur la façon de réduire les coûts de maintenance des logiciels. Beaucoup ont attribué les coûts élevés de la phase de maintenance à la complexité que représente un programme sous forme de code source.

Répartition des coûts de maintenance d'un logiciel

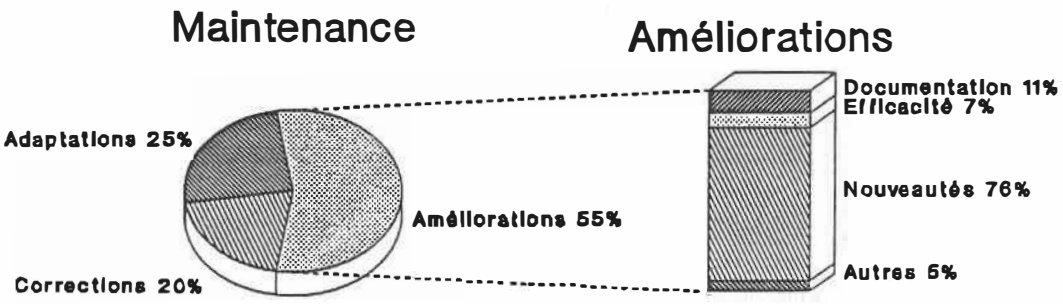

Figure 2.2: Répartition des coûts de maintenance d'un logiciel [MART83]

Au milieu des années 70, Halstead [HALS77] et McCabe [MCCA76] proposent les premières mesures destinées à quantifier cette complexité du code source. Ces mesures sont alors appelées métriques. Halstead intitule son système de mesure "la science du logiciel" tandis que McCabe appelle sa métrique "compexité cyclomatique".

Ces deux métriques vont générer une quantité importante d'articles dans les revues reliées au domaine de l'informatique [COOK82] [COTE88]. Certains auteurs créent de nouvelles métriques semblables [MYER77] ou totalement différentes [BOLO88]. D'autres auteurs effectuent des régressions avec le nombre d'erreurs essayant d'appuyer ou d'infirmer les dires d'Halstead [LAUR82] et de McCabe [HANS78]. Malgré cette effervescence, très peu de scientifiques s'entendent pour désigner une métrique comme la meilleure. Pendant ce temps, la majorité des utilisateurs se contentent du nombre de lignes pour quantifier la complexité d'un projet développé.

Vers 1982, on voit apparaître les premiers programmes calculant la complexité de code source [SCHN82]. Ces programmes sont alors rudimentaires et fonctionnent sur des environnements très rigides. Depuis la venue d'outils comme LEX et YACC servant à construire des compilateurs [AHO88], les analyseurs de code source sont devenus beaucoup plus souples et plus performants.

Le laboratoire de Recherche en Génie Logiciel de l'École Polytechnique a développé un analyseur de code source appelé DATRIX [COAL89]. L'outil produit des résultats à partir de code source écrit en FORTRAN, Pascal, C et plusieurs dialectes de ces langages de programmation.

Aujourd'hui, les outils pour générer des métriques permettent d'obtenir facilement des données. C'est dans cette mer de données que l'on patauge. De l'ordre doit être mis dans toutes ces mesures afin d'orienter la recherche subséquente dans des voies plus fructueuses. Le tableau 2.1 présente un résumé des phases de la recherche de solutions aux problèmes de maintenance à l'aide des métriques du logiciel.

Année	Phase
1968	Règles de bonne programmation
1976	Définition des métriques
1983	Automatisation de la prise de mesure
1990	Interprétation de l'information mesurée

Tableau 2.1: Recherche de solutions au problème de la maintenance des logiciels

Les données recueillies sur des projets, des expériences et au travers de publications scientifiques doivent aider à éclaircir certains points par rapport aux métriques.

- Les métriques suivent-elles des distributions connues?
- La valeur d'une métrique diffère-t-elle d'un projet à un autre, d'un langage à un autre?
- Les routines d'un projet possèdent-t-elles une signature qui permet de reconnaître à quel projet elles appartiennent?
- Quelles métriques sont dépendantes des programmeurs et/ou de l'environnement et/ou de la complexité de la tâche?
- Existe-t-il des facteurs communs à des métriques?
- Peut-on définir des nouvelles métriques hybrides qui ont un contenu d'information plus élevé que les métriques traditionnelles?
- Existe-t-il un sous-espace de métriques qui permette de mieux représenter la variabilité de l'ensemble utilisé?
- Quelle est la meilleure façon de représenter l'information sur une routine?

Tableau 2.2: Exemples de problèmes à résoudre

III. La mesure statique du logiciel

A. Définitions

Cette section définit des mots-clés du vocabulaire du génie logiciel employé dans ce mémoire. Comme le projet utilise à la fois des mots-clés propres aux statistiques et au génie logiciel, il semble approprié de définir un vocabulaire concis pour le mémoire.

1. complexité

Le terme "complexité" est utilisé dans le domaine du génie logiciel pour définir:

"la difficulté à maintenir, à changer et à comprendre un logiciel [ZUSE90]."

2. métrique

L'utilisation du terme "métriques" dans le domaine du génie logiciel provient du terme anglais "metrics". Alors que l'ensemble des dictionnaires anglais/français s'accordent pour traduire d'un terme à l'autre, les ouvrages anglophones utilisent des définitions telles que perçues par les chercheurs en génie logiciel. Le "Webster Third New International Dictionnary (1971)" définit entre autre le mot "metric" par:

"...a standard of measurement."

La définition applicable à ce mémoire est tirée du génie logiciel. Aucune distinction n'est faite entre le terme mesure et le terme métrique.

Métrique: Mesure qualitative ou quantitative d'un trait d'un logiciel.

Trait: Caractéristique ou attribut d'un logiciel.

B. Les analyseurs de code source

Cette section présente différents outils commerciaux d'analyse de code source.

1. PAPA

L'acronyme PAPA tient pour "Product Assurance Program Analyser" [SCHN82]. Il s'agit d'un projet de la firme IBM section Boeblingen en Allemagne. L'outil développé dans le cadre du projet devait produire les informations suivantes pour une routine:

- un graphe de contrôle;
- le nombre cyclomatique [MCCA76];
- le nombre de conditions [MYER77];
- l'expression de la complexité;
- l'imbrication maximale des boucles et des conditionnelles;
- une liste des sommets et des arcs;

- une liste de la fréquence des énoncés utilisés;
- une liste des étiquettes, des variables de communication et des noms des modules appelés.

De plus l'outil devait détecter le code mort (Dead Code), les routines avec plusieurs points d'entrée ou plusieurs points de sortie en plus de produire un graphe d'appel entre les routines analysées. Finalement les routines devaient être classées dans un ordre croissant de complexité et une interface homme-machine permettait de sélectionner des routines à l'aide de certains critères.

L'article était le résumé d'une session tenue lors de la conférence annuelle de l'ACM SIGMetrics. Cet article de 2 pages annonçait que les résultats obtenus par l'outil devraient être publiés à la fin de l'année en cours. Hélas, une recherche bibliographique a été effectuée en juin 1988 sur plusieurs banques de données et aucun autre article ne fut publié par Karl Ernst Schnurer ou avait comme référence cet article.

2. The Analysis of Complexity Tool (ACT).

The Analysis of Complexity Tool de McCabe & Associates Inc. est présenté comme un outil qui permet de maintenir les logiciels [MCCA89]. L'outil analyse du code source écrit en C, COBOL, FORTRAN, Pascal ASM86 et Ada. L'analyse lexicale dans un langage donné est basée sur le dernier standard ou la version la plus utilisée du langage.

Les métriques produites sont la complexité cyclomatique [MCCA76] et la complexité essentielle [MYER77]. Les graphes de contrôle (section IIIc du mémoire) pour chaque routine et le graphe d'appel pour le projet sont produits. Un autre outil permet aussi d'obtenir des jeux de tests. Ceux-ci sont composés de Vg cycles indépendants du graphe de contrôle [BERG73].

3. Logiscope

Logiscope de Verilog fut commercialisé vers 1986 [VERI87]. L'outil fonctionne sur le système d'exploitation UNIX. Le logiciel permet d'analyser du code source écrit en C, Pascal, Modula-2, FORTRAN et COBOL. 20 différentes métriques sont extraites du code, incluant la complexité cyclomatique et 9 métriques de la science du logiciel.

Le logiciel n'a pas été utilisé, cependant des rapports présentant des résultats nous ont été distribués. Ces rapports sont d'ailleurs une source importante de données comme le lecteur pourra le vérifier au chapitre VI.

4. PC-Metric

Le logiciel est disponible depuis 1988. Les résultat produits peuvent être visualisés à l'aide de macros fournies avec le logiciel. L'utilisation est simple et son bas prix vise à introduire l'usager au domaine des métriques [PERR88].

Les métriques calculées sont celles de Halstead, le complexité cyclomatique de McCabe et quelques métriques traditionnelles. L'outil permet d'analyser du code écrit en FORTRAN, C, Pascal et Modula-2. L'analyse de dialectes particuliers de formes non-standardisées de langages de programmation est attendue sous peu.

5. Datrix

Datrix [COAL89] est un logiciel d'analyse de code source en développement au Laboratoire de Recherche en Génie Logiciel depuis 1986. L'outil génère une trentaine de métriques basées pour la plupart sur le graphe de contrôle du code source. Ce graphe est défini d'une façon formelle qui le rend indépendant des particularités des langages. L'outil sert aussi à documenter les projets. Il produit la liste des appels entre les routines, leurs occurences. Les métriques de Halstead ne sont pas calculées.

Les analyseurs syntaxiques sont complets pour chaque dialecte d'un langage donné. Ils permettent d'analyser du code source en Pascal, FORTRAN et C.

C. Les graphes de contrôle

Le graphe de contrôle d'une routine représente la façon dont le flux de contrôle est cheminé à l'intérieur de celle-ci. On peut tirer une analogie entre le graphe de contrôle et le "flow chart" utilisé pour décrire les spécifications d'une routine.

Le "flow chart" est réalisé avant le code et comprend l'ensemble des spécifications qui doivent être implantées dans la routine. Pour sa part le graphe de contrôle est construit à partir du code source, donc après la réalisation.

La façon traditionnelle de modéliser le contrôle à l'aide d'un graphe est de créer des noeuds pour chaque décisionnelle et de placer le code séquentiel dans les arcs ou dans les sommets attachés aux sommets décisionnels. Lors de la représentation, on omet le contenu des arcs et des sommets. La figure 3.1 montre un exemple de graphe traditionnel.

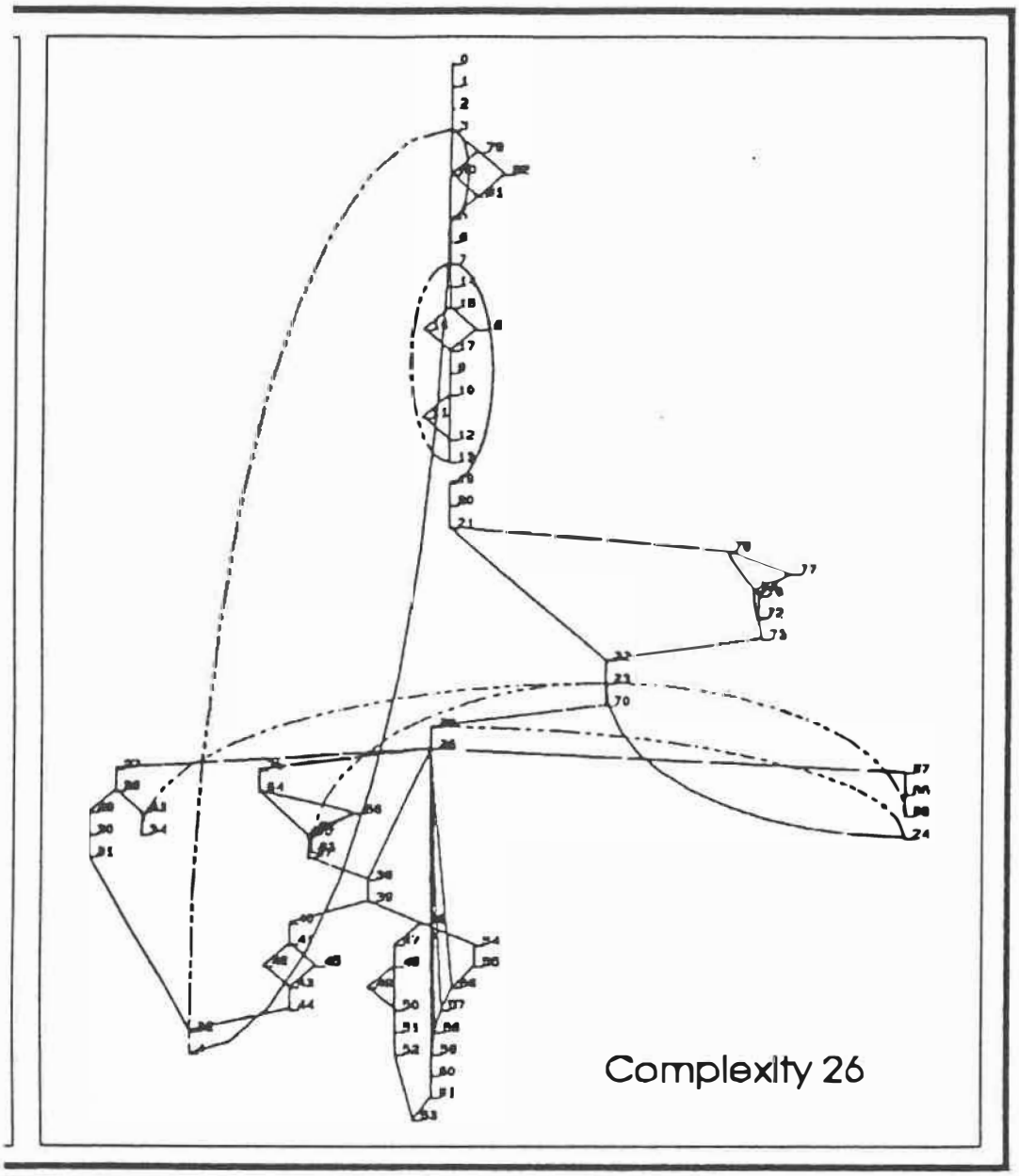

Figure 3.1: Exemple de graphe de contrôle généré par ACT de McCabe

La figure 3.2 montre une autre façon de représenter un graphe de contrôle. Cette représentation consiste à le visualiser dans un demi-plan. Cette approche présente l'avantage de montrer les croisements et les bris de structures présents dans les structures de contrôle.

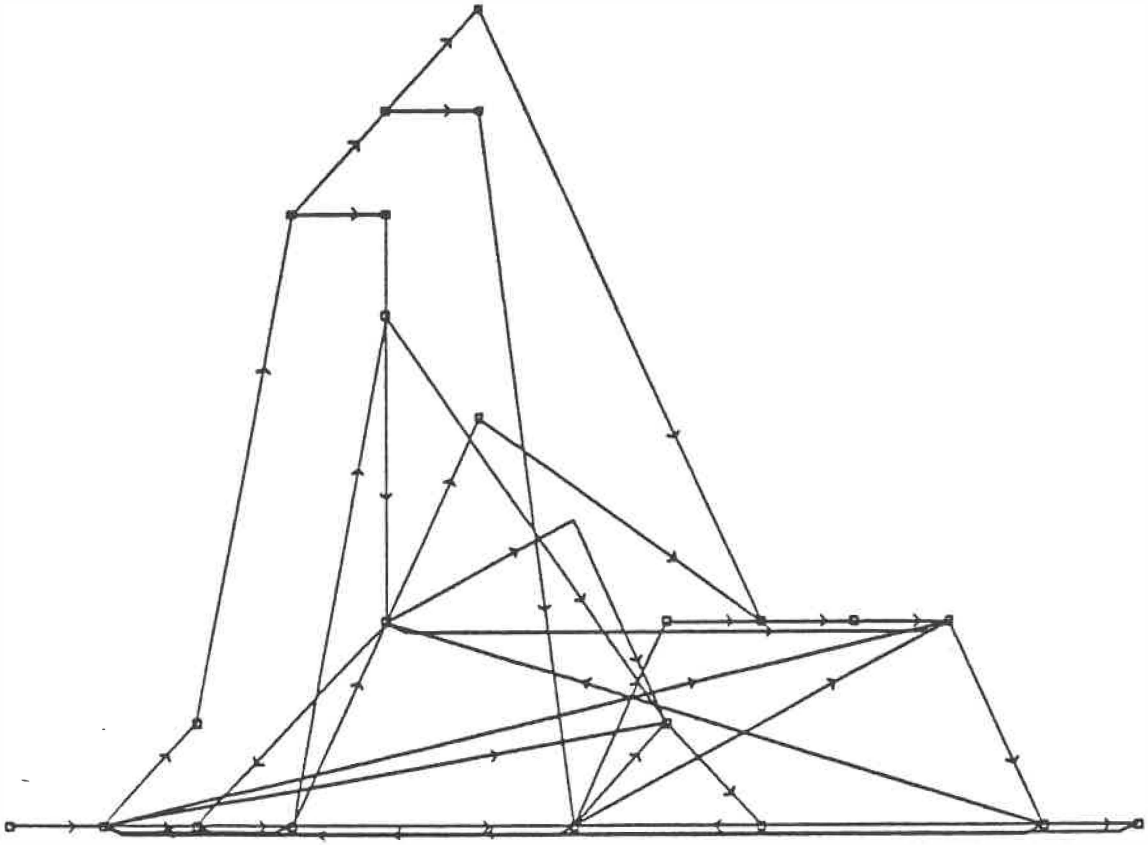

Figure 3.2: Exemple de graphe de contrôle généré par Logiscope de Verilog

Un autre approche consiste à utiliser les avantages de la représentation dans le demi-plan en plus de conserver l'ordonnancement des énoncés dans le source. Cette représentation permet d'établir un lien plus direct entre le code source et le graphe de contrôle. La figure 3.3 montre un exemple de cette représentation.

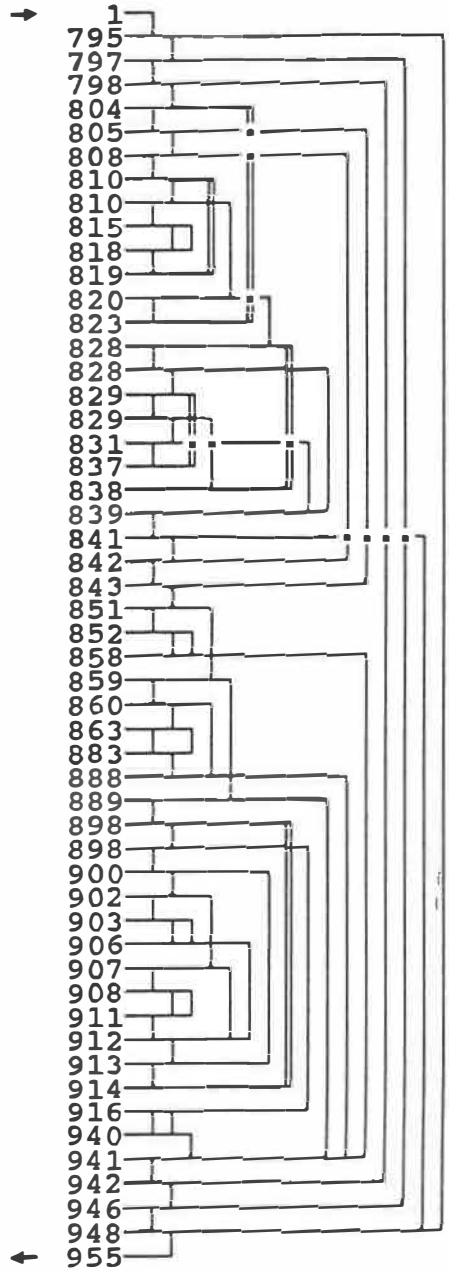

Figure 3.3: Exemple de graphe de contrôle généré par DATRIX de Bell

Il existe bien sûr d'autres représentations possibles. Celles présentées ici sont les plus courantes. De plus, elles sont respectivement implantées dans les logiciels commerciaux ACT de McCabe, Logiscope de Verilog et Datrix de Bell Canada.

D. Les métriques de complexité du logiciel

Dans cette section, nous faisons une liste des métriques du logiciel référencées par les publications scientifiques. La plupart de ces métriques sont utilisées dans ce mémoire. Il serait trop fastidieux de rapporter toutes les métriques introduites dans la littérature. De plus, certaines métriques définies par des auteurs ont soulevé très peu d'intérêt.

Les tableaux 3.1 et 3.2 rapportent l'ensemble des métriques définies dans ce mémoire. Celles-ci sont classées par ordre alphabétique du sigle de la métrique. Par la suite, vous trouvez ces métriques décrites formellement à l'aide d'un symbolisme mathématique. Dans cette dernière section, les métriques sont classées par ordre alphabétique de références bibliographiques. Les références bibliographiques sont composées des quatre premières lettres du nom de famille de l'auteur et de l'année de publication.

Il est possible en connaissant soit le sigle d'une métrique ou son auteur de la retrouver directement dans cette section. Connaissant l'auteur de la métrique, le lecteur trouve la section qui s'y rapporte directement. Si par contre on connait le sigle, les tableaux 3.1 et 3.2 donneront le nom de l'auteur sous forme de référence.

Les métriques sans référence sont classées sous la section "métriques classiques" que vous trouvez en tout premier lieu du classement par références.

Sigle	Référence	Description
AC	[MAGE81]	Complexité algébrique
Band	[BELA80]	Niveau moyen d'imbrication
BLOCKS	[ELSH84]	Nombre d'énoncés de blocs
Bmax	[BELA80]	Niveau maximal d'imbrication
Bs	[COAL87]	Nombre de bris de structures
Bsp	[COAL87]	Nombre de bris de structures pondéré
C		Nombre de commentaires
CALL	[ELSH84]	Nombre d'appels à des sous-routines
CCHAR		Nombre de caractères des commentaires
CHAR		Nombre de caractères du source
CL	[GILB77]	Complexité logique absolue
cL	[GILB77]	Complexité logique relative
CM	[MCCL76]	Complexité globale
CP	[HARR81]	Complexité psychologique
D	[HALS77]	Difficulté observée du programme
D^	[HALS77]	Difficulté calculée du programme
DD	[HALS77]	Difficulté des données
DE	[CONT86]	Nombre de décisions
DpT	[SCHR84]	Profondeur de l'arbre de syntaxe
E	[HALS77]	Effort de programmation
e	[MCCA76]	Nombre d'arcs
E^	[HALS77]	Effort de programmation calculé 1
E^^	[HALS77]	Effort de programmation calculé 2
ESC		Nombre d'énoncés exécutables à compiler
I^	[HALS77]	Contenu d'intelligence
IF	[HENR81]	Flux d'information
Ipgm	[BOLO88]	Interconnectivité des énoncés
KNOT1	[WOOD79]	Le nombre de bris mesurés
KNOT2	[WOOD79]	Le nombre de bris potentiels
λ	[HALS77]	Niveau du langage
L		Nombre de lignes
L	[HALS77]	Niveau observé du programme
L^	[HALS77]	Niveau calculé du programme
LC		Nombre de lignes à compiler
LE		Nombre de lignes exécutables
LNE		Nombre de lignes non-vides
LOC		Nombre de lignes de code
Ls	[COAL87]	Longueur moyenne des noms des variables utilisées

Tableau 3.1: Les métriques de complexité du logiciel (partie 1)

Sigle	Référence	Description
η	[HALS77]	Vocabulaire du programme
N	[HALS77]	Longueur observée du programme
N^	[HALS77]	Longueur calculée du programme
η1	[HALS77]	Nombre d'opérateurs uniques
η1*	[HALS77]	Nombre d'opérateurs minimaux
N1	[HALS77]	Nombre total d'opérateurs
η2	[HALS77]	Nombre d'opérandes uniques
η2*	[HALS77]	Nombre d'opérandes minimaux
N2	[HALS77]	Nombre total d'opérandes
Nc	[COAL87]	Complexité moyenne des sommets conditionnels
Ncmax	[COAL87]	Complexité maximale des sommets conditionnels
Ncn		Nombre de sommets conditionnels
Ne	[COAL87]	Nombre de sommets de sortie
Nel	[COAL87]	Niveau d'imbrication pondéré
NF	[JENS82]	Longueur estimée du programme
Ni	[COAL87]	Nombre de sommets d'entrée
Nl	[COAL87]	Nombre de boucles
Np	[SCHN79]	Nombre de chemins indépendants
Nr	[COAL87]	Nombre de sommets récursifs
NV		Nombre de variables
Psc	[COAL87]	Portée moyenne des sommets conditionnels
Pscmax	[COAL87]	Portée maximale des sommets conditionnels
PVG	[SUNO81]	Processus
Rac	[COAL87]	Ratio des arcs commentés
Rls	[COAL87]	Ratio structurel de boucle
Rnc	[COAL87]	Ratio des noeuds commentés
Rlw	[COAL87]	Ratio structurel de pondération de boucles
S		Nombre d'énoncés
SC		Nombre d'énoncés à compiler
SCHAR		Nombre de caractères du code
SCOPE	[LI87]	Portées
SCORT	[LI87]	Ratio des portées
SD10	[ELSH84]	Nombre d'énoncés imbriqués à 10 niveaux ou plus
T^	[HALS77]	Temps de programmation estimé
TT	[SCHR84]	Grosseur de l'arbre de syntaxe
V	[HALS77]	Volume du programme
v	[MCCA76]	Nombre de sommets
V*	[HALS77]	Volume potentiel du programme
V**	[HALS77]	Volume limite
Vcd	[COAL87]	Volume de commentaires des déclarations
Vcs	[COAL87]	Volume de commentaires des structures
Vg	[MCCA76]	Complexité cyclomatique
Vge	[MYER77]	Complexité cyclomatique étendue
Vp	[COAL89]	Nombre de sommets pendants
Vs	[COAL87]	Volume structurel
Vw	[COAL87]	Somme des pondérations
WSC	[SUNO81]	Nombre d'énoncés pondéré

Tableau 3.2: *Les métriques de complexité du logiciel (partie 2)*

1. Métriques classiques

Sont référencées sous cette rubrique les mesures communément utilisées qui ne sont pas attribuées à aucun auteur.

a. Nombre de lignes (L)

L — Somme des lignes de la routine du fichier source conservé sur support magnétique.

b. Nombre de lignes non-vides (LNE)

LNE — Somme des lignes de la routine du fichier source conservé sur support magnétique duquel on soustrait les lignes vides.

c. Nombre de lignes de code (LOC)

LOC — Somme des lignes de la routine du fichier source conservé sur support magnétique. On soustrait les lignes vides et les lignes de commentaires.

d. Nombre de lignes exécutables (LE)

LE — Somme des lignes de la routine du fichier source conservé sur support magnétique. On soustrait les lignes vides, les lignes de commentaires et les lignes de déclarations.

e. Nombre de lignes à compiler (LC)

LC Somme des lignes de la routine du fichier destiné au compilateur. Les fichiers inclus, macro-opérations et autres ont alors déjà été traités par les préprocesseurs.

f. Nombre d'énoncés exécutables à compiler (ESC)

ESC Somme des énoncés de la routine du fichier destiné au compilateur qui produisent des actions de contrôle ou de transfert de données dans le programme.

g. Nombre d'énoncés à compiler (SC)

SC Somme des énoncés exécutables et des énoncés de déclarations de la routine du fichier destiné au compilateur.

h. Nombre d'énoncés (S)

S Somme des énoncés de la routine du fichier, incluant les commentaires.

i. Nombre de sommets conditionnels (Ncn)

Définition sommet conditionnel:
Un sommet v_i est dit conditionnel si son demi-degré extérieur est supérieur à 1.

La métrique mesure la quantité des sommets permettant de brancher à plus d'un endroit dans la routine.

$Ncn = |V'|$ où $V' = \{ v_i \mid d_G^-(v_i) > 1 \}$

j. Nombre de commentaires (C)

Définition — commentaire physique:
texte compris entre un début physique de commentaire et une fin physique de commentaire tels que définis par la syntaxe du langage.

Définition — commentaire logique:
suite de commentaires physiques consécutifs compris entre 2 énoncés. Il ne peut y avoir plus d'un commentaire logique entre 2 énoncés exécutables.

C — Somme des commentaires logiques de la routine du fichier source.

k. Nombre de caractères du source (CHAR)

CHAR — Somme en octets de l'espace requis pour conserver la version textuelle de la routine du fichier source sur support magnétique. Le format est prédéfini selon un standard (ASCII, EBCDIC,...).

l. Nombre de caractères des commentaires (CCHAR)

CCHAR — Somme des longueurs des chaînes de caractères constituant les commentaires de la routine.

m. Nombre de caractères du code (SCHAR)

SCHAR — Différence entre le nombre de caractères du source et le nombre de caractères des commentaires.

n. Nombre de variables (NV)

NV — Nombre d' identificateurs locaux et globaux, déclarés ou utilisés.

2. Belady [BELA80]

a. Niveau moyen d'imbrication (Band)

Belady a défini une métrique appellée Band basée sur l'affirmation qu'il est plus difficile de développer et maintenir des programmes ayant des structures de contrôle très imbriquées. La métrique Band d'un programme est un indicateur du niveau moyen d'imbrication du graphe de contrôle.

$$Band = \frac{\sum_{i=1}^{|V|} D(v_i)}{|V|}$$

$D(v_i)$ est le niveau d'imbrication du sommet v_i. On considère que les noeuds non compris dans une structure de contrôle ont un niveau d'imbrication égal à 1. Pour chaque structure conditionnelle ou répétitive, on ajoute 1 au niveau d'imbrication.

b. Niveau maximal d'imbrication (Bmax)

A l'aide de la définition de la métrique Band , on peut aussi déterminer la structure de contrôle ayant l'imbrication la plus profonde. On définit alors le niveau maximal d'imbrication comme étant cette valeur.

$$Bmax = \max_{i=1}^{|V|} D(v_i)$$

3. Boloix et Robillard [BOLO88]

a. Interconnectivité des énoncés (Ipgm)

Cette métrique est basée sur les relations qu'un énoncé possède avec les autres énoncés au point de vue des variables et des structures. Le terme structure désigne l'ensemble des conditionnelles, boucles, sélections multiples,... L'analyse d'un code source permet de déterminer une matrice dite d'interconnexions qui conserve toutes ces relations. La matrice est bidimensionnelle. Les lignes de la matrice représentent les énoncés du programme dans l'ordre rencontré. Les colonnes représentent les variables logiques et les structures. Ainsi une colonne ne correspond qu'à une seule variable logique ou une seule structure. Dans cette matrice, on peut rencontrer 4 types de symboles:

M	définition d'une nouvelle variable logique par une assignation ou un changement de valeur;
m	utilisation d'une variable logique;

P énoncé conditionnel de la structure (test d'une conditionnelle, condition de sortie d'une boucle,...);

p énoncé imbriqué dans la structure.

La matrice reçue est transformée en une matrice binaire où apparaît un 1 s'il y a présence d'un code M, m, P ou p et un 0 s'il y a absence de code. On conserve quand même l'information à savoir si la colonne contenait des M (M ou m) ou des P (P ou p).

De la matrice binaire complète, on calcule l'excès d'entropie. On répète le même calcul sur les matrices où l'on ne considère que les colonnes associées aux variables (M et m) et les colonnes associées aux structures (P et p).

L'entropie de la matrice $H_{matrice}$ s'obtient en conservant toute les colonnes de la matrice. Pour calculer l'entropie d'une ligne H_i, on crée une sous-matrice où les seules colonnes conservées sont celles où des 1 apparaissent sur la ligne en question. Le nombre de lignes est toujours le même.

Le calcul de l'entropie s'effectue en calculant l'occurence de chacun des patrons présents dans la matrice. Chaque ligne de la matrice binaire détermine un patron de bits 0 et 1. On détermine tous les patrons différents K de la matrice et l'on compte l'occurence k_i de chaque patron i de l'ensemble K. L'entropie est ensuite déterminée par la relation suivante:

$$H = \ln Q - \sum_{i=1}^{|K|} \frac{k_i * \ln (k_i)}{Q}$$

Q représente le nombre de lignes

Le calcul de l'excès d'entropie I_{PGM} s'obtient en soustrayant la somme de l'entropie de chacune des lignes H_i de l'entropie totale de la matrice $H_{matrice}$.

$$I_{PGM} = \sum_{i=1}^{nb\ lignes} H_i - H_{matrice}$$

4. Coallier [COAL87]

a. Nombre de bris de structures (Bs)

Nombre d'arcs se croisant dans le graphe de contrôle qui sont le résultat de la violation des principes de la programmation structurée. C'est-à-dire que chaque structure de contrôle doit avoir une seule entrée et une seule sortie.

b. Nombre de bris de structures pondéré (Bsp)

Somme des poids de chaque paire d'arcs impliqués dans des bris de structures.

c. Longueur moyenne des noms des variables utilisées (Ls)

$$Ls = \frac{\sum_{j=1}^{NV} length(\ I_j\ )}{NV}$$

I_j est la $j^{ème}$ variable utilisée.

d. Complexité moyenne des sommets conditionnels (Nc)

La complexité du sommet v_i est définie comme la somme du nombre d'opérateurs et d'opérandes du prédicat moins 1 . La complexité moyenne est donc la somme des complexités des sommets conditionnels divisée par le nombre de ces sommets.

$$Nc = \frac{\sum_{i=1}^{Ncn} (\eta_{1i} + \eta_{2i} - 1)}{Ncn}$$

η_{1i} est le nombre d'opérateurs du sommet v_i

η_{2i} est le nombre d'opérandes du sommet v_i

e. Complexité maximale des sommets conditionnels (Ncmax)

$$Ncmax = \max_{i=1}^{Ncn} Nc_i$$

Nc_i est la complexité du sommet v_i

f. Nombre de sommets d'entrée (Ni)

$Ni = |VI|$

VI Ensemble des sommets du graphe de contrôle référencés par un énoncé tel qu'il est possible d'entrer dans le graphe à ce sommet.

g. Nombre de sommets de sortie (Ne)

$Ne = |VO|$

VO Ensemble des sommets du graphe de contrôle référencés par un énoncé tel qu'il est possible de sortir du graphe à ce sommet.

h. Nombre de boucles (Nl)

Somme des arcs du graphe de contrôle branchant vers le haut.

$Nl = |E'|$ où $E' = \{ e_{ij} \mid i > j \}$

i. Nombre de sommets récursifs (Nr)

$Nr = |VR|$

VR Ensemble des sommets du graphe de contrôle dont l'un des arcs sortants possède un énoncé rappelant la routine dans la portée de l'arc.

j. Portée moyenne des sommets conditionnels (Psc)

$$Psc = \frac{\sum_{i=1}^{Ncn} Psc_i}{Ncn}$$

où la portée Psc_i d'un sommet conditionnel v_i est définie comme la longueur moyenne des arcs sortant de ce sommet.

$$Psc_i = \frac{\sum_{e_{ij} \in E^+(v_i)} Abs(i - j)}{d_G^+(e_i)}$$

k. Portée maximale des sommets conditionnels (Pscmax)

$$Pscmax = \underset{i=1}{\overset{Ncn}{MAX}} Psc_i$$

l. Ratio des arcs commentés (Rac)

$$Rac = \frac{|E'|}{|E|}$$

E' Ensemble des arcs du graphe qui ont au moins un commentaire logique à l'intérieur de la portée de l'arc.

m. Ratio des noeuds commentés (Rnc)

$$Rnc = \frac{|V'|}{|V|}$$

V' Ensemble des sommets du graphe qui sont immédiatement précédés d'un commentaire logique.

n. Volume des commentaires des déclarations (Vcd)

Vcd Somme des longueurs des chaînes de caractères constituant les commentaires de la routine avant le premier énoncé exécutable.

o. Volume des commentaires des structures (Vcs)

Vcs Somme des longueurs des chaînes de caractères constituant les commentaires de la routine après le premier énoncé exécutable. On obtient la relation CCHAR = Vcs + Vcd.

p. Volume structurel (Vs)

$$Vs = \frac{|E| + |V| + 1}{6}$$

q. Somme des pondérations (Vw)

Somme des pondérations de tous les arcs, soit le nombre d'instructions exécutables dans le programme moins les instructions de contrôle.

$$Vw = \sum_{i=1}^{|E|} e_i$$

e_i prend la valeur de l'arc, soit le nombre d'énoncés dans la portée de l'arc.

r. Ratio structurel de boucle (Rls)

On définit dans un premier temps le ratio structurel $Rls(e_{ij})$ d'une boucle e_{ij}.

$$Rls(e_{ij}) = Vs(G') \quad \text{où} \quad G' = (V',E')$$

$$E' = \{ e_{kl} \mid j \leq k \leq i \text{ et } j \leq l \leq i \}$$

$$V' = \{ v_k \mid \exists e_{km} \in E' \}$$

Le ratio structurel d'une boucle est alors défini par

$$Rls = \frac{\sum_{k=1}^{Nl} Rls(e_k)}{Vs}$$

s. Ratio structurel de pondération de boucles (Rlw)

On définit dans un premier temps le ratio structurel de pondération $Rls(e_{ij})$ d'une boucle e_{ij}.

$$Rlw(e_{ij}) = Vw(G') \quad \text{où} \quad G' = (V',E')$$

$$E' = \{ e_{kl} \mid j \leq k \leq i \quad \text{et} \quad j \leq l \leq i \}$$

$$V' = \{ v_k \mid \exists e_{km} \in E' \}$$

Le ratio structurel d'une boucle est alors défini par

$$Rlw = \frac{\sum_{k=1}^{Nl} Rlw(e_k)}{Vw}$$

t. Ratio du volume de commentaires (Rvc)

$$Rvc = \frac{CCHAR}{Vs}$$

u. Niveau d'imbrication pondéré (Nelw)

C'est le niveau d'imbrication tel que défini par Belady que l'on pondère en fonction du nombre d'énoncés dans la portée du noeud.

$$Nel = \frac{\sum_{i=1}^{|V|} \left(D(v_i) \sum_{e_{ij} \in E^+(v_i)} e_{ij} \right)}{Vw}$$

$D(v_i)$ est le niveau d'imbrication du sommet v_i.

5. Coallier, Robillard et Beaucage [COAL89]

a. Nombre de sommets pendants (Vp)

Somme des sommets qui ne peuvent être accédés par le programme.

$Vp = |V'|$ où $V' = \{ v_i \mid v_i \notin VI \text{ et } d_G^-(v_i) = 0 \}$

VI Ensemble des sommets du graphe de contrôle référencés par un énoncé tel qu'il est possible d'entrer dans le graphe à ce sommet.

6. Conte [CONT86]

a. Nombre de décisions (DE)

$DE = |V'|$ où $V' = \{ v_i \mid d_G^-(v_i) > 1 \}$

Cette définition est la même que celle du nombre de noeuds conditionnels. Les auteurs tendent de plus en plus à référer au terme "nombre de décisions" depuis la publication de l'ouvrage de CONTE sur les métriques [CONT86].

7. Elshoff [ELSH84]

a. Nombre d'énoncés de blocs (BLOCKS)

Somme des occurences des énoncés PROCEDURE , BEGIN , DO , IF , ON et SELECT dans un programme écrit en PL/1.

b. Nombre d'énoncés imbriqués à 10 niveaux ou plus (SD10)

$$SD10 = \sum_{e_{ik} \in E} e_{ik} \quad \text{où} \quad D(v_i) > 9$$

$D(v_i)$ est le niveau d'imbrication du sommet v_i;

c. Nombre d'appels à des sous-routines (CALL)

Somme des occurences des énoncés permettant de faire des appels à des routines.

8. Gilb [GILB77]

a. Complexité logique absolue (CL)

$$CL = |V'| \quad \text{où} \quad V' = \{ v_i \mid d_G^-(v_i) > 1 \}$$

b. Complexité logique relative (cL)

$$cL = \frac{CL}{ESC}$$

9. Halstead [HALS77]

La science du logiciel offre une gamme d'estimateurs basés sur l'aspect textuel des métriques. On calcule les opérateurs et les opérandes d'un programme à partir desquels on en déduit des formules estimant toutes sortes de propriétés des routines. Les métriques d'Halstead sont les pionnières dans le domaine.

a. Nombre d'opérateurs uniques (η_1)

η_1 Somme des opérateurs uniques ou distincts apparaissant dans la routine (exemple: IF, =, DO, +, ...).

b. Nombre d'opérandes uniques (η_2)

η_2 Somme des opérandes uniques ou distincts (exemple: noms de variables , constantes,...).

c. Nombre d'opérateurs minimaux (η_1^*)

Somme des opérateurs essentiels pour n'importe quel algorithme. L'ensemble des opérateurs minimaux consiste en un nom pour la routine et un autre qui sert à assigner ou grouper des symboles.

$\eta_1^* = 2$

d. Nombre d'opérandes minimaux (η_2^*)

η_2^* Somme des opérandes essentielles pour n'importe quel algorithme. Pour de petits algorithmes, l'ensemble des opérandes minimales est constitué des différences entre les paramètres d'entrées/sorties.

e. Vocabulaire du programme (η)

Somme des opérandes et opérateurs uniques ou distincts apparaissant dans la routine.

$\eta = \eta_1 + \eta_2$

f. Nombre total d'opérateurs (N_1)

Somme des opérateurs apparaissant dans le programme.

$$N_1 = \sum_{j=1}^{\eta_1} f_{1,j}$$

$f_{1,j}$ est le nombre d'occurence du jième plus utilisé opérateur.

g. Nombre total d'opérandes (N_2)

Somme des opérandes apparaissant dans le programme.

$$N_2 = \sum_{j=1}^{\eta_2} f_{2,j}$$

$f_{2,j}$ est le nombre d'occurence de la jième plus utilisée opérande.

h. Longueur observée du programme (N)

Somme des opérandes et opérateurs apparaissant dans le programme.

$$N = N_1 + N_2$$

i. Longueur calculée du programme (N^)

$$N^{\wedge} = \eta_1 \log_2 \eta_1 + \eta_2 \log_2 \eta_2$$

j. Volume du programme (V)

$$V = N \log_2 \eta$$

k. Difficulté des données (DD)

$$DD = \frac{N_2}{\eta_2}$$

l. Volume potentiel du programme (V*)

$$V^* = \eta^* \log_2 \eta^*$$

m. Volume limite (V)**

$$V^{**} = (2 + \eta_2^* \log_2 \eta_2^*) \log_2 (2 + \eta_2^*)$$

n. Niveau observé du programme (L)

$$L = \frac{V^*}{V}$$

o. Niveau calculé du programme (L^)

$$L^\wedge = \frac{\eta_1^* \eta_2}{\eta_1 N_2}$$

p. Difficulté observée du programme (D)

$$D = \frac{1}{L}$$

q. Difficulté calculée du programme (D^)

$$D^\wedge = \frac{1}{L^\wedge}$$

r. Contenu d'intelligence (I^)

$$I^{\wedge} = L\ V$$

s. Effort de programmation (E)

$$E = \frac{V}{L}$$

t. Effort de programmation calculé 1 (E^)

$$E^{\wedge} = \frac{V}{L^{\wedge}}$$

u. Effort de programmation calculé 2 (E^^)

$$E^{\wedge\wedge} = \frac{N^{\wedge}\ \mathrm{Log}_2\eta}{L^{\wedge}}$$

v. Niveau du langage (λ)

$$\lambda = L\ V^{*}$$

w. Temps de programmation estimé (T^)

$$T^{\wedge} = \frac{E}{S} = \frac{\eta_1\ N_2\ (\eta_1\ \log_2\eta_1 + \eta_2\ \log_2\eta_2)\ \log_2\eta}{2\ \eta_2\ S}$$

S Nombre de Stroud. $5 \leq S \leq 20$ par seconde.

10. Harrison et Magel [HARR81]

a. Complexité psychologique (CP)

La métrique est calculée en assignant un index de complexité à chaque sommet du graphe de contrôle non-réduit. Lorsqu'un sommet est conditionnel, l'index de complexité associé à ce sommet est la somme des index de complexité de tous les sommets se trouvant dans le sous-graphe défini par ce sommet. Pour les autres sommets, la complexité associée est le volume de Halstead de ce sommet.

11. Henry and Kafura [HENR81]

a. Flux d'information (IF)

$$IF = ESC \times (fan\_in \times fan\_out)^2$$

`fan_in` le nombre de liens possibles directs, indirects ou globaux où la routine reçoit de l'information par les autres routines.

`fan_out` le nombre de liens possibles directs, indirects ou globaux où la routine transmet de l'information aux autres routines.

12. Jensen [JENS82]

a. Longueur estimée du programme (NF)

Pendant l'étude des métriques de Halstead, Jensen trouve une approximation qu'il juge meilleure pour la longueur d'un programme.

$$NF = \log_2 \eta_1! + \log_2 \eta_2!$$

13. Li [LI87]

a. Portées (SCOPE)

Pour chaque sommet conditionnel x_i , on peut trouver une borne inférieure x_j qui succède tous les successeurs du sommet décisionnel x_i . Le nombre de sommets entre la borne inférieure minimale x_j qui précède toutes les autres bornes inférieures et le sommet x_i représente la portée de sommet décisionnel. Cette portée pour le sommet x_i est appelée GLB_i .

$$GLB_i = j - i$$

avec x_j la borne inférieure minimale de x_i .

La métrique SCOPE est donc

$$SCOPE = \frac{\sum_{i=1}^{Ncn} GLB_i}{Ncn}$$

b. Ratio des portées (SCORT)

La portée des conditionnelles est pondérée en fonction du nombre de sommets dans le graphe. On parle alors du ratio des portées.

$$SCORT = 1 - \left(\frac{|V|}{SCOPE} \right)$$

14. Magel [MAGE81]

a. Complexité algébrique (AC)

Le graphe de contrôle est représenté sous forme d'expression régulière où on utilise le symbole | pour les choix (sommets conditionnels) et * pour les répétitions (boucles). AC représente la somme des opérateurs et des opérandes de l'expression.

15. McCabe [MCCA76]

a. Nombre de sommets (v)

v = |V|

b. Nombre d'arcs (e)

e = |E|

c. Complexité cyclomatique (Vg)

Le nombre cyclomatique ou complexité cyclomatique de McCabe est tiré de la théorie des graphes [BERG70][PELL66]. McCabe applique ce théorème aux graphes de contrôle représentant le code source. Cette métrique est l'une des plus utilisées actuellement.

Vg = e - v + 2p

p le nombre de composantes connexes.

ou

$$Vg = e - v + Ni + Ne$$

si l'on exprime la métrique en fonction du nombre de points d'entrée et de sortie de la routine.

16. McClure [MCCL76]

a. Complexité globale (CM)

La métrique de McClure mesure la complexité des programmes COBOL selon la formule:

$$CM = C + VR$$

C est le nombre de comparaisons;

VR le nombre de variables de contrôle référencées dans le module.

17. Myers [MYER77]

a. Complexité cyclomatique étendue (Vge)

$$Vge = \sum_{i=1}^{|V|} boole(v_i) + 1$$

$boole(v_i)$ retourne le nombre d'opérateurs booléens du sommet v_i.

18. Schroeder [SCHR84]

Un code source peut être représenté sous forme d'arbre opérandes-opérateurs. Cette représentation est similaire à la façon de représenter une expression mathématique à l'aide d'un arbre. Un noeud est un opérateur que l'on applique sur les opérandes qui sont ces enfants. Cette représentation du code source est appelée `arbre de syntaxe` et peut être modifiée à l'aide du langage de manipulation d'arbre. Deux métriques sont déduites par l'auteur.

a. Grosseur de l'arbre de syntaxe (`TT`)

Nombre de sommets dans l'arbre de syntaxe. Cette métrique donne le même résultat que le volume de Halstead. Le traitement pour y arriver est par contre très différent.

b. Profondeur de l'arbre de syntaxe (`DpT`)

Nombre de sommets de la branche la plus profonde de l'arbre.

19. Schneidewind et Hoffman [SCHN79]

a. Nombre de chemins indépendants (Np)

Np est défini à l'aide du graphe de contrôle d'une routine. On calcule le nombre de chemins en sommant l'ensemble des chemins possibles partant des points d'entrée en allant aux points de sortie. Ce nombre de chemins est infini si le graphe admet un cycle; pour pallier à cet inconvénient, chaque chemin ne peut emprunter le retour d'une boucle donnée qu'une seule fois au maximum.

20. Sunohara, Takano, Uehara et Ohkawa [SUNO81]

a. Nombre d'énoncés pondéré (WSC)

$$WSC = \sum_{i=1}^{ESC} (DR_i + NS_i)$$

DR_i est le nombre de données référencées dans le ième énoncé (interconnectivité de l'énoncé);

NS_i est le nombre d'énoncés successifs à être exécutés après l'exécution du ième énoncé (portée de l'énoncé).

b. Processus (PVG)

Cette métrique est dérivée de la mesure P2 de Sullivan [SULL75] effectuée sur des réseaux. Pour une routine, on cherche à mesurer l'interconnexion de l'ensemble des variables. La métrique est adaptée aux logiciels en remplaçant l'indice C2 par la complexité cyclomatique Vg [MCCA76] des sous-graphes pour chaque donnée.

$$PVG = \sum_{i=1}^{|D|} V(G_{di})$$

D	Ensemble des variables de la routine.
G_{di}	(V_{di},E_{di}), le graphe de données pour la variable d_i.
V_{di}	Ensemble des sommets referant à d_i.
E_{di}	Ensemble des arcs qui permettent de rejoindre les éléments de V_{di} par le flux de contrôle.

21. Woodward, Hennel and Hedley [WOOD79]

L'identification des bris de structures dans un graphe n'est pas chose facile. Pour estimer le désordre dans le graphe de contrôle, ces auteurs ont proposé de mesurer les croisements lors de la représentation du graphe de contrôle. Ces croisements sont calculés en fonction des portées de chaque paire d'arcs.

a. Le nombre de bris potentiels (KNOT2)

KNOT2 Sommation du nombre d'intersections d'arcs de branchements lorsque le graphe de contrôle d'un programme source est sous la forme d'une représentation planaire.

b. Le nombre de bris mesurés (KNOT1)

KNOT1 Sommation du nombre d'intersections d'arcs lorsque les deux arcs de l'intersection contiennent des énoncés.

E. Discussion

De la section des métriques, on peut relever deux problèmes. Le premier révèle le nombre de façons différentes que l'on peut référer au volume d'énoncés ou de lignes de code d'une routine. La section 1 sur les métriques classiques en énumère 8 qui sont utilisées dans la littérature utile à ce mémoire. Il est facile d'en trouver beaucoup d'autres si l'on veut allonger la liste. Il y a un besoin de normalisation pour ces termes que sont une ligne et un énoncé, et aussi sur la façon de les mesurer.

Le deuxième problème évident est l'attribution d'un nom et d'un sigle à une métrique alors que la métrique est déjà définie par un autre auteur. Ainsi le nombre de noeuds conditionnels reçoit le sigle Ncn [COAL86] d'un auteur alors que la métrique est définie exactement de la même façon sous le nom de nombre de décisions [CONT86] et complexité logique absolue [GILB77]. Un autre exemple peut aussi être donné pour Band et Nelw ou pour SCOPE et d'autres métriques non-utilisées dans ce mémoire mais référées dans la littérature.

IV. L'analyse statistique

Le chapitre précédent démontre le nombre important de métriques définies. Il est difficile de mesurer les inter-relations entre ces diverses métriques avec seulement leurs définitions. On doit utiliser l'analyse statistique pour déterminer les paramètres et les modèles contenus dans ces données. Les deux citations suivantes expriment très bien l'importance que l'on a accordé aux statistiques au travers des âges.

> "There are very few things which are not capable of being reduc't to a mathematical reasoning; and when they cannot, it's a sign our knowledge of them is very small and confus'd - John Arbuthnot 1692 [MERT84]"

> "...when you can measure what you are speaking about, and express it in numbers, you know something about it; but when you cannot measure it, when you cannot express it in numbers, your knowledge is of a meagre and unsatisfactory kind - Lord Kelvin [MERT84]"

A. Définitions

1. Observation

Unité d'une population ou d'un échantillon sur laquelle diverses mesures sont effectuées.

2. Variable

Valeurs d'une mesure effectuée pour chacune des observations d'une population ou d'un échantillon.

3. Paramètre

Mesure utilisée pour décrire une population, un échantillon ou une distribution.

Par exemple, une distribution gaussienne est paramétrisée par la moyenne et l'écart-type de la fonction de densité.

4. Comportement

Caractéristique associée à un groupe d'observations ou un ensemble de variables.

5. Population

On appelle population l'ensemble total des observations sur lesquelles on veut calculer ou prédire un comportement ou un paramètre.

Si l'on veut prédire le comportement de l'ensemble des routines écrites en langage C, on refère à ce groupe comme une population. Toute affirmation faite pour ce groupe ne peut être alors généralisée au groupe des routines écrites en langage COBOL. On doit alors refaire l'analyse sur le groupe des routines COBOL ou bien sur l'ensemble des routines de tous les langages.

6. Échantillon

On appelle échantillon un sous-ensemble aléatoire et représentatif des observations de la population totale. On estime des paramètres ou on mesure un comportement sur un échantillon dans le but de vérifier des hypothèses concernant le comportement de la population.

Par exemple, pour vérifier un comportement sur la population constituée par l'ensemble des routines écrites en Pascal, on sélectionne aléatoirement un sous-ensemble de routines parmi la population. On mesure des paramètres et comportements que l'on généralise à l'aide de l'inférence statistique. Cette généralisation est soutenue par un coefficient de confiance.

7. Description des données et inférence

Les practiciens des sciences statistiques reconnaissent généralement qu'il existe principalement deux sous-divisions dans le domaine de l'analyse statistique:

- les statistiques descriptives et
- l'inférence statistique.

Les statistiques descriptives sont concernées par l'observation et le résumé des données sous forme quantitative. On y retrouve les représentations graphiques, les mesures de dispersion, les régressions, analyses factorielles, etc.

L'inférence statistique permet de répondre aux questions que l'on se pose par rapport aux données au moyen de tests. Cette partie des statistiques est la seule qui nous permet de conclure ou d'infirmer des faits sur les données. On y retrouve principalement les tests paramétriques et non-paramétriques.

Les tests paramétriques sont construits à l'aide de distributions hypothétiques des données. Les données doivent donc distribuées selon ces courbes pour vérifier les hypothèses du modèle en question. Certains tests vont assumer d'autres hypothèses comme l'égalité des moyennes et/ou de la variance des sous-groupes, etc. L'ensemble des hypothèses relatives au test à effectuer doivent être a priori vérifiées.

Les tests non-paramétriques n'exigent pas des données quelles soient distribuées selon un modèle connu. Comme les tests non-paramétriques sont plus larges, l'incertitude sur ces tests est habituellement plus grande. Il est donc fortement recommandé d'utiliser un test paramétrique si on le peut.

8. Indépendance

Quelque soit le type d'inférence statistique utilisé, paramétrique ou non-paramétrique, l'hypothèse n'est valide que si il y a indépendance des données pour l'échantillon mesuré. Si cette condition n'est pas vérifiée, aucune conclusion tirée de l'analyse d'un échantillon ne peut être généralisée à l'ensemble de la population.

Par exemple, considérons l'ensemble des individus de la terre comme une population et un ensemble constitué essentiellement d'hommes comme l'échantillon. On ne peut par inférence généraliser la grandeur de la population. Le paramètre grandeur n'a pas le même comportement dans le sous-groupe des femmes. Par contre, les résultats obtenus permettent d'inférer sur la grandeur des hommes de la population.

9. Erreur de première espèce

On définit comme erreur de première espèce la probabilité d'accepter l'hypothèse posée alors qu'elle est fausse. On utilise la lettre α pour indiquer cette erreur. C'est la probabilité de rejeter une hypothèse VRAIE.

B. Types de données

Une variable est classée selon l'un des 4 types suivants. Les types sont ordonnés selon une augmentation du contenu d'information des variables. Les types nominal et ordinal sont regroupés sous le thème de catégorie alors que les types intervalle et ratio sont regroupés sous le thème numérique.

1. Nominale (Type I)

Une variable est nominale lorsqu'elle n'admet que des valeurs qualitatives non ordinales.

- ensemble des pays, races de chiens, sexes...

2. Ordinale (Type II)

Une variable est ordinale lorsqu'elle admet des valeurs ordinales mais dont la notion de distance entre les valeurs n'a pas réellement de sens mathématique.

- rangs, évaluations intuitives, couleurs...

3. Intervalle (Type III)

Une variable numérique est de type intervalle si elle peut être exprimée par un nombre auquel on attache un comptage ou une mesure. Une variable numérique est discrète si elle admet un nombre fini ou infini dénombrable[1] de valeurs dans un intervalle fini. Une variable est continue si les valeurs qu'elle peut prendre consiste en tous les nombres réels dans un intervalle donné.

- (discrètes) montants d'argent, cardinalités, entiers ...
- (continues) temps, températures, distances ...

---

[1] Ensemble dont les éléments peuvent être numérotés ou prédits. L'ensemble des nombre entiers a une cardinalité infinie mais dénombrable.

4. Ratio (Type IV)

Une variable numérique est dite de type ratio si l'information qu'elle présente est constitué d'un rapport de deux variable de type III. Le type IV représente le plus haut niveau de mesure.

- densités, pressions, pourcentages d'erreurs ...

5. Hiérarchie des types

Les contenus d'information des variables sont croissants du type I au type IV. Sous cette affirmation il existe certaines règles qui permettent la création de nouvelles variables contenant un degré d'information supérieur ou inférieur.

Corollaire 5.1 Une variable de type II peut être modifiée pour être interprétée comme étant de type I.

Corollaire 5.2 Une variable de type III ou IV peut être modifiée pour être interprétée comme étant de type II.

Corollaire 5.3 Une variable du type IV peut être obtenue à l'aide de une ou plusieurs variables de type III.

A titre d'exemple pour le corollaire 5.1, il suffit de ne pas considérer l'effet d'ordre de la variable de type II pour la considérer de type I. Ainsi différentes couleurs ne seront plus classées par ordre d'intensité, mais tout simplement par les couleurs qu'elles représentent.

Pour le corollaire 5.2, il suffit de définir des intervalles précis sur l'étendue des variables de type III et IV, et de numéroter les intervalles pour obtenir une nouvelle variable de type II. L'âge d'une personne, étant classé selon les dizaines d'années qu'a l'individu, permet de conserver moins d'information et du même coup de réduire l'ensemble des valeurs possibles que prend la variable.

Le corollaire 5.3 n'est qu'une application de la définition donnée pour une variable de type IV.

C. Les distributions continues

Le but de cette section est d'illustrer les différents types de distributions qui peuvent être utilisés dans l'analyse statistique des métriques. Pour pouvoir utiliser certains tests paramétriques, les observations doivent être distribuées selon certaines distributions.

1. La distribution normale

La distribution normale est sans aucun doute la plus utilisée au niveau de la théorie et des applications en analyse statistique des données [CLEM86].

La fonction de densité de la distribution est:

$$f_X(\, x;\mu,\sigma\,) \;=\; \frac{1}{\sigma\sqrt{(2\pi)}}\exp\left[\,-\frac{1}{2}\left[\,\frac{x-\mu}{\sigma}\,\right]^2\,\right]$$

avec moyenne μ et écart-type σ

Nous utilisons la notation:

$$X \sim N(\mu, \sigma^2)$$

pour indiquer qu'une variable x est distribuée selon une courbe normale N avec paramètres μ et σ^2 où μ est la moyenne et σ^2 la variance.

La figure 4.1 [CONT86] montre la fonction de densité f_X d'une normale $N(0,1)$. L'axe des x représente le nombre d'écarts-type entre la variable et la moyenne.

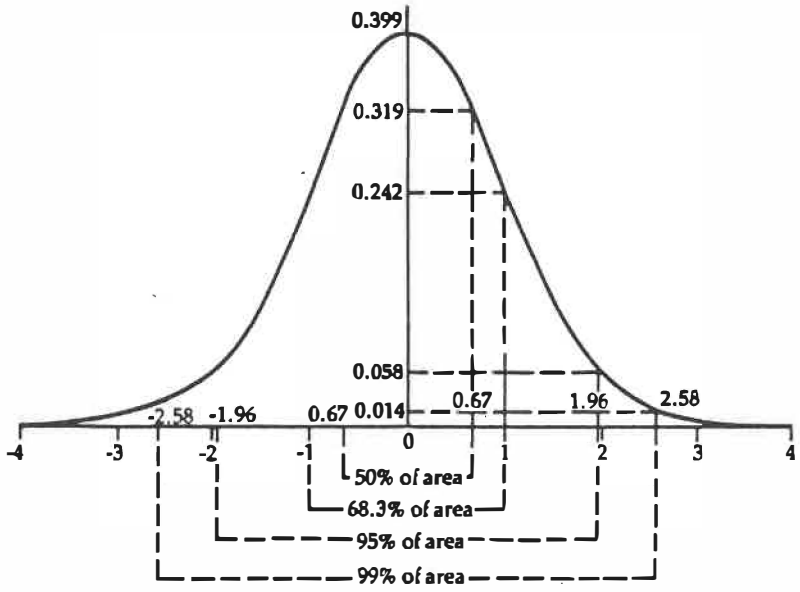

Figure 4.1: Distribution d'une normale

La distribution de nombreux phénomènes physiques ou sociaux semble suivre cette courbe. C'est d'ailleurs à l'aide de cette distribution que de nombreux modèles statistiques d'inférence sont bâtis.

Si l'on réussit à vérifier la normalité dans la distribution de métriques, les conclusions tirées sur nos échantillons pourront être inférées avec de meilleurs coefficients de confiance.

2. La distribution exponentielle

La fonction de densité d'une distribution exponentielle est:

$$f_x(x;\alpha) = \alpha \exp (- \alpha x)$$

où α est le facteur d'amortissement. Les moments de la distribution sont:

$$\mu = \frac{1}{\alpha}$$

$$\sigma^2 = \frac{1}{\alpha^2}$$

Cette distribution est la seconde plus courante lors de l'analyse de données. Elle se prête très bien aux transformations qui permettent de la ramener à une normale.

3. Transformations de Tukey et de Box-Cox

Les transformations de Tukey [TUKE57] et de Box-Cox [BOXC64] ont pour objectif de transformer une distribution quelconque à une distribution normale. Le but évident de ces transformations est de pouvoir appliquer des procédures supposant la normalité des observations.

Une variable x est de la λ-tranformée lorsque l'on lui applique la transformée $x^{(\lambda)}$ de Tukey.

$$x^{(\lambda)} = \begin{cases} x^{\lambda}, & \lambda \neq 0, \\ \log x, & \lambda = 0 \quad \text{et} \quad x > 0. \end{cases}$$

Les transformations de Box-Cox plus connues ne sont qu'une translation linéaire des transformées de Tukey. Les équations les régissant sont les suivantes:

$$x^{(\lambda)} = \begin{cases} \dfrac{x^{\lambda} - 1}{\lambda} & \lambda \neq 0, \\ \log x, & \lambda = 0 \quad \text{et} \quad x > 0. \end{cases}$$

Lorsque l'on applique une transformation, on se réfère aux tests d'ajustement pour vérifier l'efficacité de la transformation appliquée. Il est recommandé d'utiliser des valeurs entières ou des fractions simples pour conserver une perception de la variable.

4. La distribution multinormale

Lorsque les observations sont multidimensionnelles, on peut vouloir tester la multinormalité des données pour appliquer certains tests statistiques. Puisque l'on mesure plusieurs variables sur les routines, nos observations sont multivariées.

Le vecteur d'observations

$$\underline{X} = (X_1, X_2, \ldots, X_p)'$$

suit une distribution multinormale de p dimensions avec vecteur moyenne

$$\underline{\mu}$$

et matrice de covariance Σ si la distribution des observations suit la fonction de densité suivante.

$$f_{\underset{\sim}{X}}(\underset{\sim}{x}) = (2\pi)^{-p/2} \mid \Sigma \mid^{-1/2} \exp \left[- \frac{1}{2} (\underset{\sim}{x}-\underset{\sim}{\mu})' \Sigma^{-1} (\underset{\sim}{x}-\underset{\sim}{\mu}) \right]$$

Théoriquement, certaines procédures multidimensionnelles supposent la multinormalité des observations, alors que pratiquement, on se contente de la normalité des observations dans chacune des dimensions.

D. Mesures de dispersion

Les mesures de dispersions sont souvent utilisées pour caractériser les distributions connues. Dans le cas où l'on mesure la dispersion d'échantillon, on utilise le terme estimateur.

Les formules qui suivent sont les estimateurs pour la variable X_1.

1. Estimateur de la moyenne

$$\bar{x}_1 = \sum_{i=1}^{n} x_{1i}$$

2. Estimateur de la variance

$$s^2_1 = s_{11} = \frac{1}{n-1} \sum_{i=1}^{n} (x_{1i}-\bar{x}_1)^2$$

3. Estimateur de l'écart-type

$$s_1 = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (x_{1i} - \bar{x}_1)^2}$$

4. Estimateur de la covariance entre les variables X_1 et X_2

$$s_{12} = \frac{1}{n-1} \sum_{i=1}^{n} (x_{1i} - \bar{x}_1)\ (x_{2i} - \bar{x}_2)$$

5. Matrice des covariances

$$\Sigma = \begin{pmatrix} s_{11} & s_{12} & \cdots & s_{1p} \\ s_{21} & s_{22} & \cdots & s_{2p} \\ \cdots & \cdots & \cdots & \cdots \\ s_{p1} & s_{p2} & \cdots & s_{pp} \end{pmatrix}$$

6. Régression simple par les moindres carrés

$$\min_{\alpha,\beta} \left[\sum_{i=1}^{n} (X_{2i} - \alpha X_{1i} - \beta)^2 \right]$$

Ainsi en supposant un modèle linéaire,

$$X_2 = \alpha X_1 + \beta$$

on obtient une estimation des coefficients α et β par

$$\hat{\beta} = \frac{S_{12}}{S_{11}}$$

$$\hat{\alpha} = \bar{X}_2 - \beta \bar{X}_1$$

La mesure d'exactitude la plus utilisée est le coefficient de corrélation de Pearson, noté r. C'est un nombre sans dimension qui mesure l'intensité de la liaison linéaire entre deux variables X_1 et X_2 de type III ou de type IV. Cet indice s'obtient en calculant le rapport suivant:

$$r_{12} = \frac{\sum_{i=1}^{n} (x_{1i} - \bar{x_1}) (x_{2i} - \bar{x_2})}{\sqrt{\sum_{i=1}^{n} (x_{1i} - \bar{x}_1)^2} \sqrt{\sum_{i=1}^{n} (x_{2i} - \bar{x}_2)^2}}$$

n est le nombre d'observations

Sous forme matricielle, le coefficient de corrélation s'écrit tout simplement:

$$r_{12} = \frac{S_{12}}{S_1 S_2}$$

Bien que cette méthode d'analyse de dépendance entre deux variables soit la plus répandue, elle présente d'énormes désavantages. Ainsi, on conclut souvent à des dépendances entre des variables alors qu'il n'en existe pas réellement.

Par exemple, dans les pays scandinaves, une relation entre le nombre de cigognes vivant dans une région et le nombre de bébés dans cette même région fut notée. Est-ce que les cigognes apportent les bébés? Une simple observation du coefficient de corrélation pourrait nous faire croire que oui! D'après les notions que nous ont apprises nos parents, on peut douter de la relation. En observant un peu mieux les données, on remarque que les grosses villes ont un nombre élevé de naissances et aussi beaucoup d'édifices dans lesquels les cigognes peuvent faire leurs nids (figure 4.2). Il faut donc utiliser des méthodes permettant de mesurer l'impact de chaque variable dans le modèle. On utilisera la corrélation partielle ou multiple entre les observations pour vérifier les impacts de chaque variable dans un ensemble multidimensionnel.

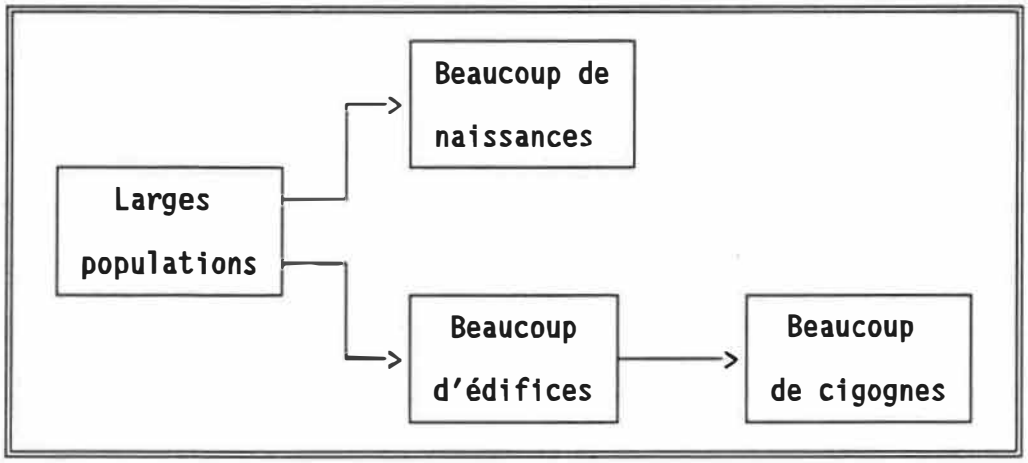

Figure 4.2: Exemple de l'effet d'une troisième variable

7. Le test d'ajustement de Shapiro-Wilk

Les tests d'ajustement sont une constituante essentielle de l'analyse des données. Beaucoup de méthodes statistiques supposent à priori que les observations sont distribuées selon certaines distributions. On doit donc s'assurer que les hypothèses de distributions sur lesquelles reposent ces tests ne sont pas violées.

On peut vérifier l'ajustement à l'aide des méthodes suivantes [CLEM88]:

- les coefficients d'asymétrie et d'aplatissement;
- l'usage de graphique percentiles;
- les tests statistiques du Khi-deux de Pearson, Kolmogorov-Smirnov, Shapiro-Wilk...

Les études comparatives entre les divers tests ont montré que le test de Shapiro-Wilk est le meilleur pour détecter des écarts à la distribution normale. Dans le cas où l'on a plus de 2000 observations, le test de Kolmogorov-Smirnov semble préférable [SASP85]. Puisque les échantillons sur lesquels nous allons tester la normalité sont tous inférieurs à 2000 observations dans ce mémoire, seul le test de Shapiro-Wilk sera utilisé.

Le paramètre W de Shapiro-Wilk est le ratio du meilleur estimateur de la variance sur la somme corrigée des estimateurs de la variance [SHAP65]. W est compris entre 0 et 1, où des petites valeurs de W mènent au rejet de l'hypothèse nulle.

Les bornes significatives pour W sont fonction du nombre d'observations n . Pour ne pas rejeter l'hypothèse de normalité, la valeur de W devra être beaucoup plus grande si l'échantillon est composé de beaucoup d'observations.

E. Analyses factorielles

Lorsqu'il tire des mesures sur des personnes ou des objets, le chercheur inclut fréquemment le plus grand nombre de variables possible pour éviter d'avoir à recommencer la prise des données. Malheureusement, quand la dimension d est large, la banque des données peut-être difficile à gérer et difficile à étudier sans être condensée d'une quelconque façon. Dans ce cas, plusieurs options sont offertes au chercheur.

La première consiste à sélectionner un sous-ensemble de k variables qui réflètent le mieux les proprités géométriques et statistiques des d variables originales. La façon dont cette procédure est effectuée dépend des propriétés considérées. Par exemple, certaines analyses factorielles permettent de sélectionner un sous-ensemble qui possède un pouvoir de discrimination semblable aux variables originales. D'autres méthodes permettent de sélectionner des variables à l'aide de nuages de points tels que les variables choisies sont bien représentatives de ces nuages.

Le terme "analyse factorielle" dénote une famille de méthodes (tableau 4.1) dont l'objectif principal est de définir un espace factoriel (espace mathématique formé de vecteurs orthogonaux) permettant de représenter les données initiales dans un espace de faible dimension en minimisant la perte d'information.

- composantes principales
- factorielle classique
- factorielle de tableau de distances
- correspondance binaire
- correspondance multiple
- canonique
- discriminante

Tableau 4.1: Principales méthodes d'analyses factorielles

1. Analyse factorielle classique

Le but de l'analyse factorielle classique est de tranformer les observations d'un espace de dimension N dans un espace orthogonal. Le premier axe a la caractéristique de représenter le maximum de variabilité des observations, le deuxième axe représente le maximum de la variabilité restante parmi les observations et ainsi de suite. Les axes trouvés sont appelés facteurs, ils sont linéairement indépendants et orthogonaux entre eux. La figure 4.3 montre un exemple de l'orientation des axes des facteurs pour un nuage de points en 2 dimensions. Lorsqu'il y a beaucoup de dépendance linéaire entre les données, l'espace des facteurs est de dimension réduite par rapport à l'espace initial.

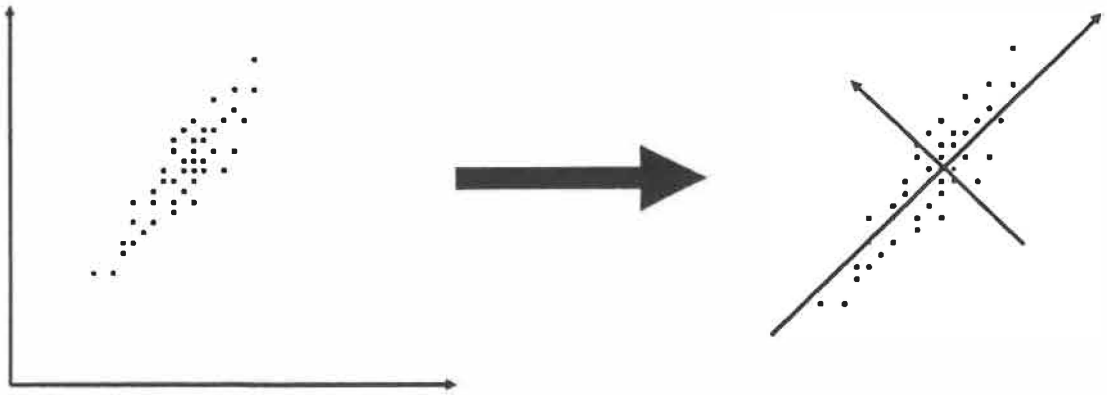

Figure 4.3: Recherche de l'orientation des facteurs dans un espace bidimensionnel

La figure 4.4 donne un exemple où 10 variables sont réduites en un espace tridimensionnel. Le facteur 1 est alors orienté de façon à représenter le maximum de variabilité des axes initiaux. Le deuxième facteur s'oriente dans la direction où il reste le plus de variabilité à expliquer et ainsi de suite.

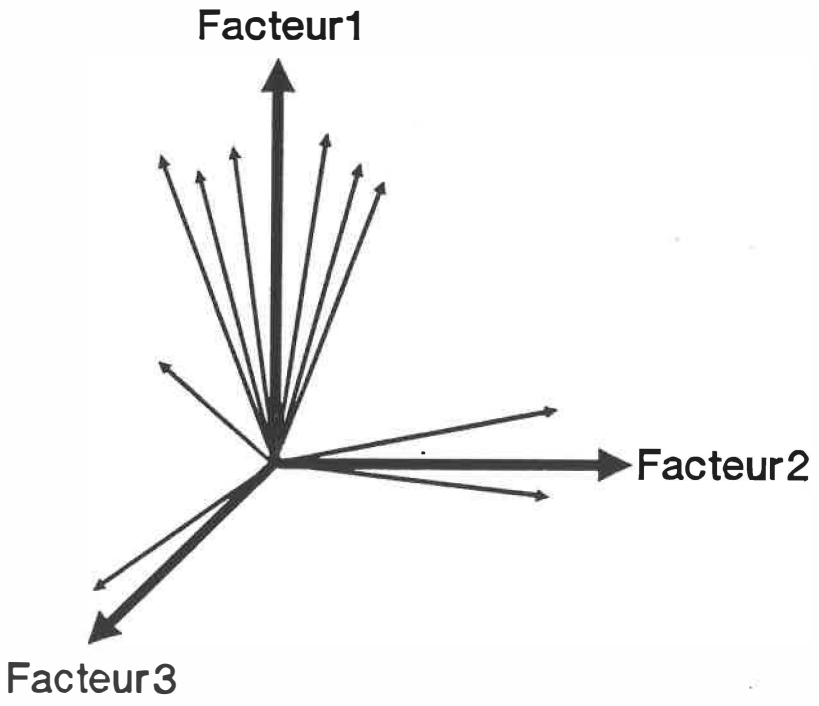

Figure 4.4: Espace des facteurs

Le modèle d'analyse factorielle où l'on a P variables observées avec moyennes nulles et Q facteurs communs peut s'écrire:

$$x_{ij} = \sum_{k=1}^{Q} y_{ik}\, b_{kj} + e_{ij}$$

x_{ij} est la valeur de la ième observation pour la jème variable;

y_{ik} est la valeur de la ième observation pour le kième facteur;

b_{kj} est le coefficient de régression du kième facteur prédisant la jème variable;

e_{ij} est le facteur spécifique pour la ième observation et la jème variable.

Sous forme matricielle l'équation précédente se réduit à:

$$X = Y B' + E$$

X est la matrice des valeurs des observations normalisée à une moyenne nulle (DATA);

Y est la matrice des valeurs des facteurs (PROJECTIONS);

B est la matrice des régresseurs (PATTERN);

E est la matrice des facteurs spécifiques (ERROR).

Dans un cas idéal la matrice E , qui représente aussi les résidus, est nulle. Il y a deux hypothèses critiques:

- un facteur spécifique est orthogonal avec les autres facteurs spécifiques;
- un facteur spécifique est orthogonal avec les facteurs communs.

Sous les contraintes d'indépendance entre les facteurs Y_i, la covariance entre 2 facteurs doit aussi être nulle, ce qui s'exprime par:

$$COV(Y_i, Y_j) = 0 \quad \text{pour} \quad i \neq j$$

La technique de l'analyse factorielle consiste à estimer les régresseurs B dans le modèle. Comme le nombre de solutions est infini, plusieurs méthodes permettent de réaliser cette tâche et de trouver ainsi une solution initiale (tableau 4.2). Une fois les régresseurs trouvés, des rotations orthonormales permettent d'interpréter les facteurs plus facilement en ajustant ceux-ci plus près des éléments perceptibles que sont les axes originaux. Ces rotations distribuent la variance sur l'ensemble des facteurs retenus pour augmenter l'importance de certains de ces facteurs.

- Méthode des composantes principales
- Méthode de vraisemblance maximale
- Moindres carrés
- Principale
- Kaiser-Jiffy
- Harris
- Pattern
- Alpha
- Score

Tableau 4.2: Différentes méthodes pour estimer les régresseurs

La transformation inverse qui consiste à transposer les données dans l'espace des facteurs s'écrit:

$$Y = X C$$

Comme la matrice B' n'est pas carrée, la transformation inverse n'est pas B'^{-1}, la matrice résultante sera référée par C (SCORE). On en déduit naturellement l'équation suivante où I est la matrice identité.

$$I = B' C$$

On peut obtenir la matrice des régresseurs à l'aide des observations, de la matrice de covariance ou de la matrice de corrélation entre les variables.

Si l'on considére l'importance des échelles des variables, la méthode correspond à utiliser la matrice de covariance. Par contre, lorsque les échelles des covariances diffèrent énormément, il est conseillé d'utiliser la technique de la matrice de corrélation pour éviter que la variabilité due à une variable noie la variabilité des autres.

L'extraction des facteurs se fait à l'aide de l'une de ces deux matrices. L'avantage de posséder l'ensemble des observations permet de vérifier la validité du modèle sur chaque observation à l'aide de la matrice des facteurs spécifiques (ERROR) [COUP90b].

Si la dimension de la matrice des covariances ou des corrélations est N et que son rang est M. Alors M facteurs suffiront à expliquer la totalité de la variabilité des données observées puisque le rang d'une matrice est défini comme le maximum de variables linéairement indépendantes. De plus, les derniers facteurs expriment souvent une variabilité non significative au terme statistique et on peut donc les éliminer. Cette variabilité perdue est considérée comme du bruit. Il en résulte que le nombre de facteurs utiles est souvent inférieur à M. L'espace résultant est donc très souvent de dimension considérablement réduite.

La plupart des progiciels statistiques permettent d'appliquer les différentes procédures d'analyses factorielles. La partie la plus critique et la seule non-automatisable est l'identification des facteurs. On interprète la signification des facteurs à l'aide des projections de chacun des axes sur ceux-ci. Ces projections permettent de déterminer les variables dont les axes sont les plus proches des facteurs. Les rotations sont utiles pour interpréter la signification des facteurs. Par contre, il faut être conscient que ces rotations distribuent les variances des axes les plus significatifs sur les axes les moins significatifs [COUP90a].

2. Analyse discriminante

Le modèle d'analyse factorielle discriminante permet de juger de la variabilité des données pour classer celles-ci en différents groupes. La méthode utilise à priori les données fournies pour en déduire des critères de discrimination. Une fois les critères établis, la méthode essaie de reclasser les données à l'aide de ces critères. Il existe différentes méthodes paramétriques et non-paramétriques pour établir les critères que ce soit à l'aide de probabilités de nuages de points, de densités ou de distances. Un fort pourcentage de classification indiquera que les données présentent une signature qui permet de les reclasser selon la variable d'identification désignée.

Pour un ensemble d'observations, les variables de type III seront les variables explicatives et la variable de type I celle à expliquer. L'analyse discriminante développe un critère qui permet de classifier chaque observation dans l'un des groupes de la variable à expliquer.

Lorsque les observations à l'intérieur de chaque groupe sont distribuées selon une loi multinormale, une méthode paramétrique peut être utilisée pour développer une fonction discriminante. La fonction discriminante, aussi connue sous le nom de critère de classification, est déterminée par une méthode générale des distances au carré. Le critère de classification est basé sur les matrices de covariances à l'intérieur des groupes (fonction quadratique) ou bien sur la matrice de covariance de toutes les observations (fonction linéaire); il tient aussi compte des probabilités initiales des groupes ou de toute autre répartition des probabilités.

Lorsque les distributions à l'intérieur des groupes ne sont pas multinormales, des méthodes non-paramétriques peuvent être utilisées pour estimer les densités spécifiques des différents groupes. Les méthodes des noyaux et des voisins les plus proches sont les plus connues. La méthode du noyau utilise différentes techniques uniformes, normales, Epanechnikov, etc. pour estimer la densité. La méthode des voisins les plus proches classifie les observations en fonction des observations adjacentes (figure 4.5).

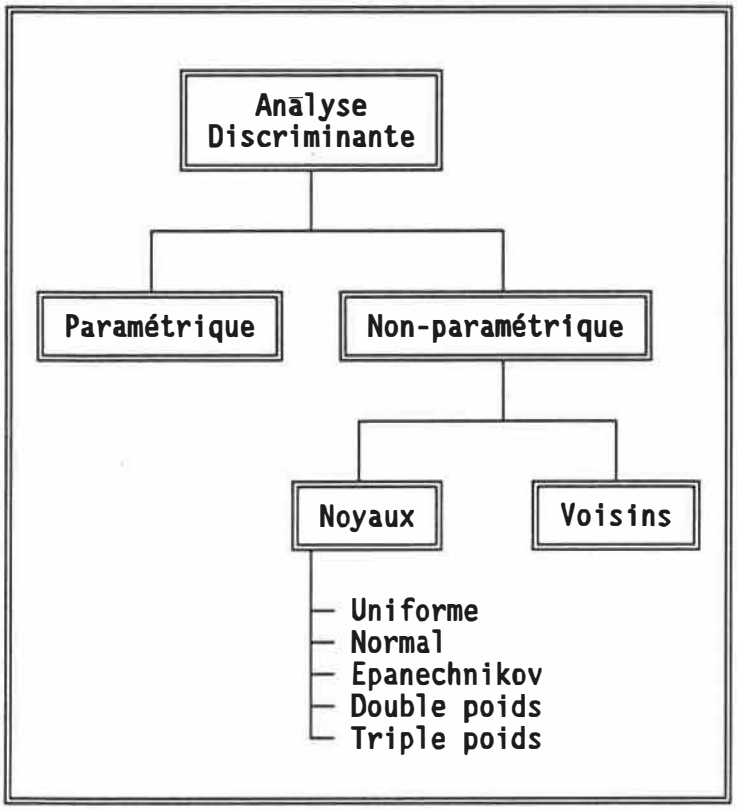

Figure 4.5: L'analyse discriminante et ses variantes

Pour ce mémoire, si les observations pour lesquelles on effectue une analyse discriminante ne sont pas distribuées selon des modèles connus. Seule une méthode non-paramétrique peut être utilisée.

La méthode des voisins consiste à classifier une observation d'après le caractère le plus commun parmi un nombre fixe de voisins. Les méthodes basées sur les noyaux consiste à regarder le caractère parmi tous les voisins compris dans un espace autour de l'observation. Pour la méthode des voisins, on en fixe le nombre. Pour les noyaux, on fixe les paramètres qui permettront de déterminer la forme et la grosseurs de ceux-ci. Puisque le modèle des voisins est moins dépendant des paramètres que l'on doit fixer, il est utilisé pour ce mémoire.

Les figures 4.6 et 4.7 démontrent l'effet du nombre de voisins sur les classements pour un échantillon donné. La figure 4.6 représente un classement où le système connait à priori les proportions de chaque classe. En cas d'incertitude, le classement se fait dans la classe la plus nombreuse. La figure 4.7 représente un classement où chaque classe est considérée comme ayant une proportion égale d'observations.

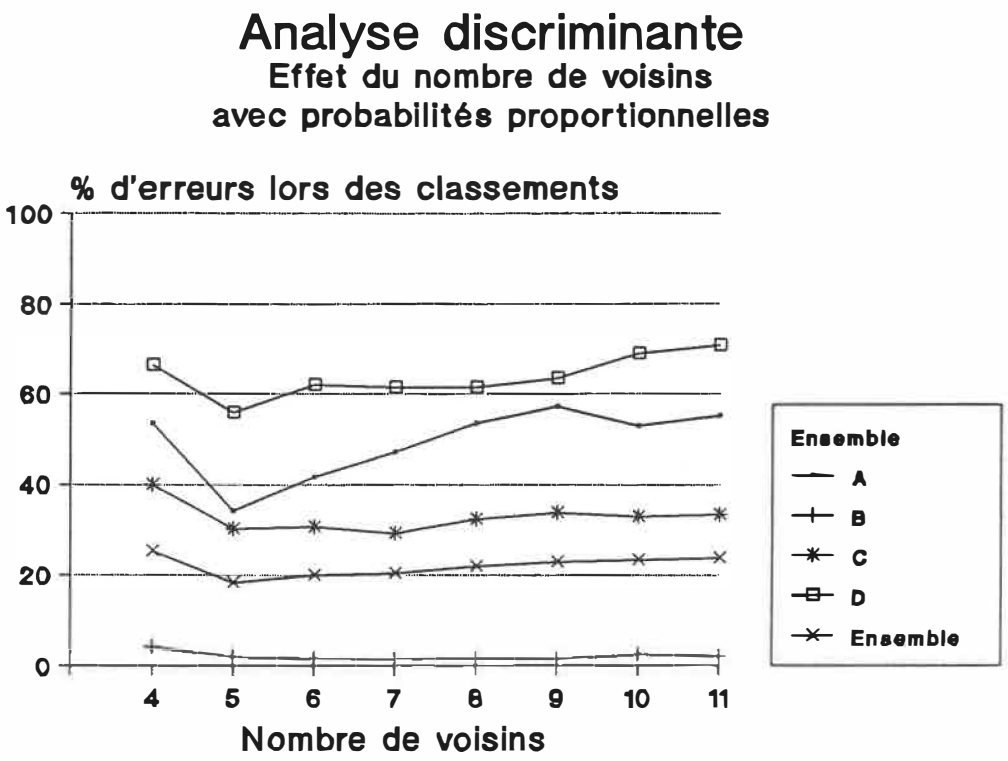

Figure 4.6: Effet du nombre de voisins dans le cas de probabilités proportionnelles

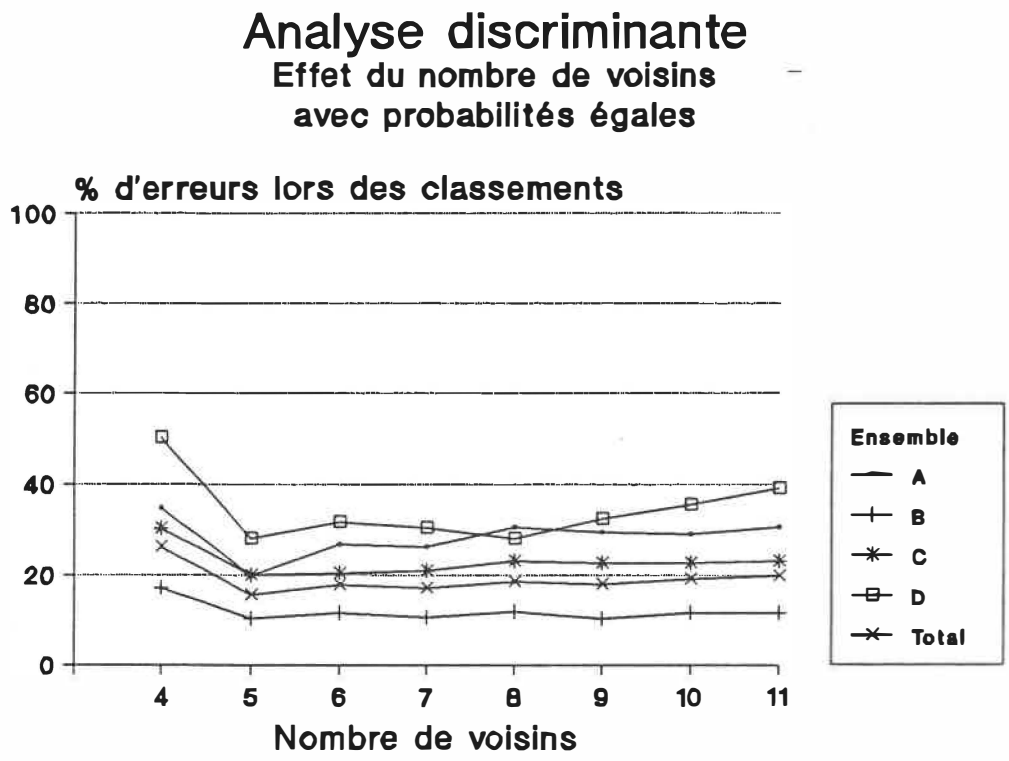

Figure 4.7: Effet du nombre de voisins dans le cas de probabilités supposées égales

Les 5 courbes de chaque figure représentent les pourcentages d'erreurs pour les quatre valeurs de la classe ainsi que l'ensemble des quatre valeurs. Pour la figure 4.6, les proportion d'erreurs sont fonction du nombre d'observations dans les groupes. La figure 4.7 présente une atténuation de cet effet. Le modèle avec probabilités égales avec 5 voisins donne les résultats les plus satisfaisants. Ce modèle est donc retenu pour les analyses.

Cette technique d'analyse factorielle devrait nous aider à déterminer si certains facteurs comme le langage de programmation utilisé, les programmeurs, etc. laissent des signatures dans les données.

V. Interface pour les données

Ce chapitre présente les relations possibles entre différents logiciels commerciaux connus et DATRIX. Si l'analyse du code source est la spécialité de DATRIX, il est utopique de croire que l'on peut incorporer des facilités de représentation et d'analyse aussi élaborées que possèdent des outils comme SAS, 123, etc. DATRIX possède donc un mécanisme de transfert d'information et les logiciels spécialisés sont accompagnés d'utilitaires qui permettent de traiter les résultats plus facilement.

A. Exportation des résultats à partir de DATRIX

DATRIX produit 3 types de fichiers pour conserver les résulats de ses analyses. Les extensions de ces fichiers sont CMT, CRT et APP.

Le fichier CMT contient les valeurs des métriques pour chaque routine.

Le fichier CRT contient l'information pour construire un graphe de contrôle. . On y retrouve les noeuds, les arcs, le poids des arcs, les noeuds d'entrées, les noeuds de sorties, les noeuds récursifs, etc. Cette information est stockée sous un format binaire de façon à réduire l'espace occupé sur le disque. L'approche utilisée pour ce type de fichier n'est pas organisée de façon relationnelle. L'image de l'information contenue dans le fichier est semblable à celle de la mémoire.

Le fichier APP contient le graphe d'appel d'un projet et par le fait même les informations physiques de chaque routine: fichiers sources, positions dans les fichiers, listes des routines qui appellent et appelées par la dite fonction.

Il n'existe aucun mode permettant à un usager de récupérer cette information. DATRIX doit donc permettre un mode de transmission de ses résultats vers les autres logiciels.

A - Incorporer des modules de représentation graphique et de calcul statistique.
B - Attacher un SGBD à Datrix et ainsi permettre cette translation via les utilitaires du SGBD.
C - Générer un ensemble de fichiers qui seraient destinés spécifiquement aux progiciels courants.

Tableau 5.1: Solutions possibles

Chacune des solutions proposées dans le tableau 5.1 présente ses avantages et ses inconvénients.

Pour la solution A, il semble évident que l'outil devient beaucoup plus puissant. Le fait de ne pas retenir cette solution n'implique pas que l'on rejette totalement l'idée d'incorporer un jour à l'outil des facilités statistiques et/ou graphiques. Par contre, on ne pourra jamais offrir à l'usager la souplesse et la puissance d'analyse qu'offrent des progiciels comme BMDP, SAS ou encore SPSS. Il est donc préférable d'avoir un mode de transfert des données vers le monde extérieur. Les solutions B et C offrent ce transfert vers le monde extérieur.

La solution B est beaucoup moins liée à la structure interne et permet de restructurer les données de travail dans une forme relationnelle. Par contre, il se crée un pertes d'efficacité dû au SGBD. Cette solution permet d'obtenir des données beaucoup plus manipulables. La contrainte majeure est que l'utilisateur de DATRIX doit aussi posséder le SGBD augmentant donc considérablement le coût d'utilisation de l'outil. Par exemple, on parle de quelques milliers de dollars pour obtenir une licence de ORACLE.

La solution C présente l'avantage d'être la moins coûteuse à la fois pour le développement et pour l'utilisation. Elle permet d'offrir plus d'interaction avec les logiciels déjà primés par l'opinion publique. Le principal inconvénient de cette solution est qu'elle ne résout pas le problème de la représentation de type "fichier" des données et ne simplifie pas la complexité de ces données.

La solution C a été retenue. Les figures 5.1, 5.2 et 5.3 montrent les choix et les fonctions disponibles à partir des écrans présentés à l'usager.

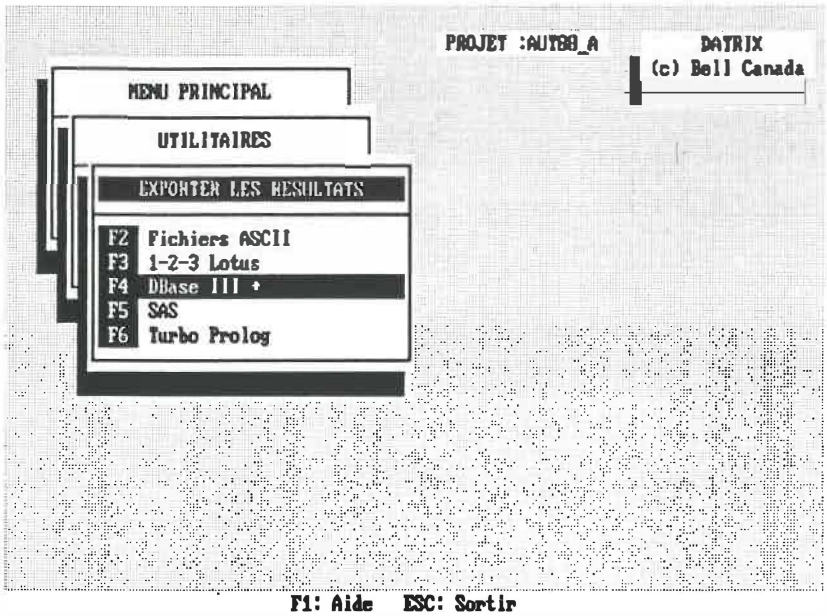

Figure 5.1: Choix du logiciel pour lequel on exporte les résultats

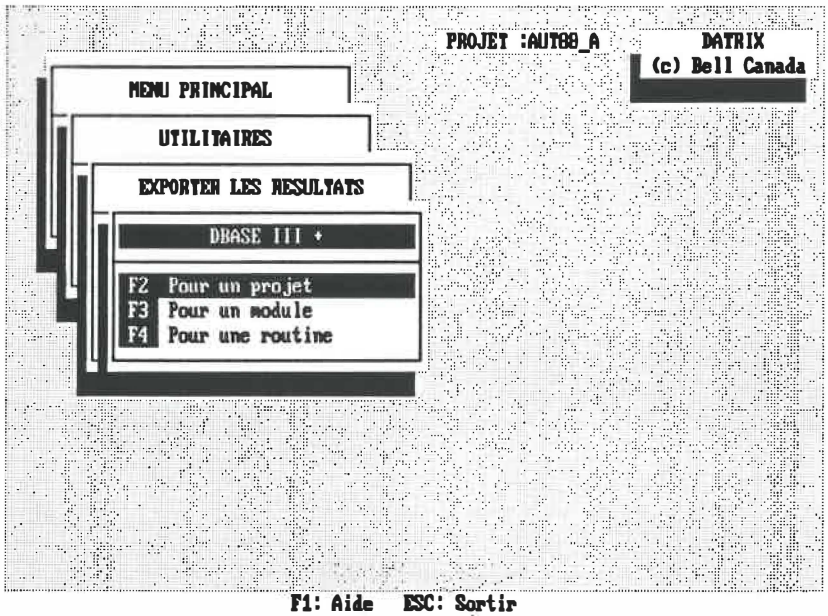

Figure 5.2: Sélection de l'ensemble des données

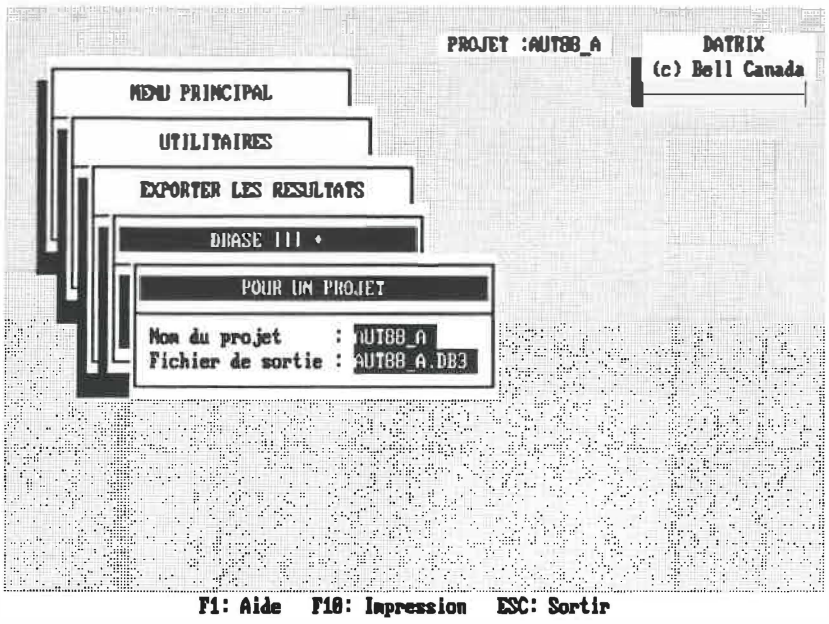

Figure 5.3: Description du fichier de transfert

B. Représentation avec 123 de Lotus

L'exportation pour une feuille de travail comme LOTUS permet de manipuler interactivement les données. Ainsi à l'aide de macro-fonctions, on peut voir les graphes des distributions, classer selon une métrique, incorporer des données à l'intérieur de rapports.

Les figures 5.4, 5.5 et 5.6 présentent des exemples de graphiques que l'on peut produire. Dans la figure 5.4, chaque routine est représentée par une barre verticale où sa hauteur est directement proportionnelle à la valeur de la métrique. Dans cet exemple, les différentes valeurs du nombre cyclomatique (Vg) sont rapportées.

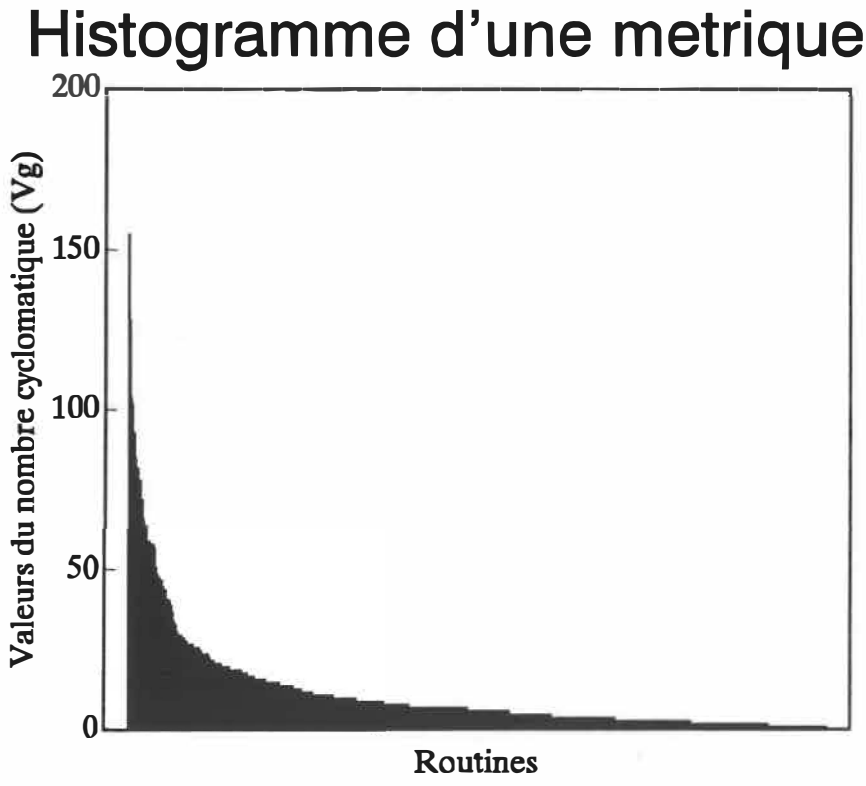

Figure 5.4: Distribution d'une métrique par 123 de Lotus

La figure 5.5 présente un graphique de percentiles. Dans ce type de graphe, une barre verticale est tracée pour chaque métrique. La région blanche montre la proportion des routines à l'intérieur des normes. Les régions noires au bas représentent les proportions des routines pour lesquelles les valeurs des métriques sont inférieures à la plage des normes tandis que les régions noires du haut sont les valeurs des métriques qui sont supérieures à l'écart acceptable.

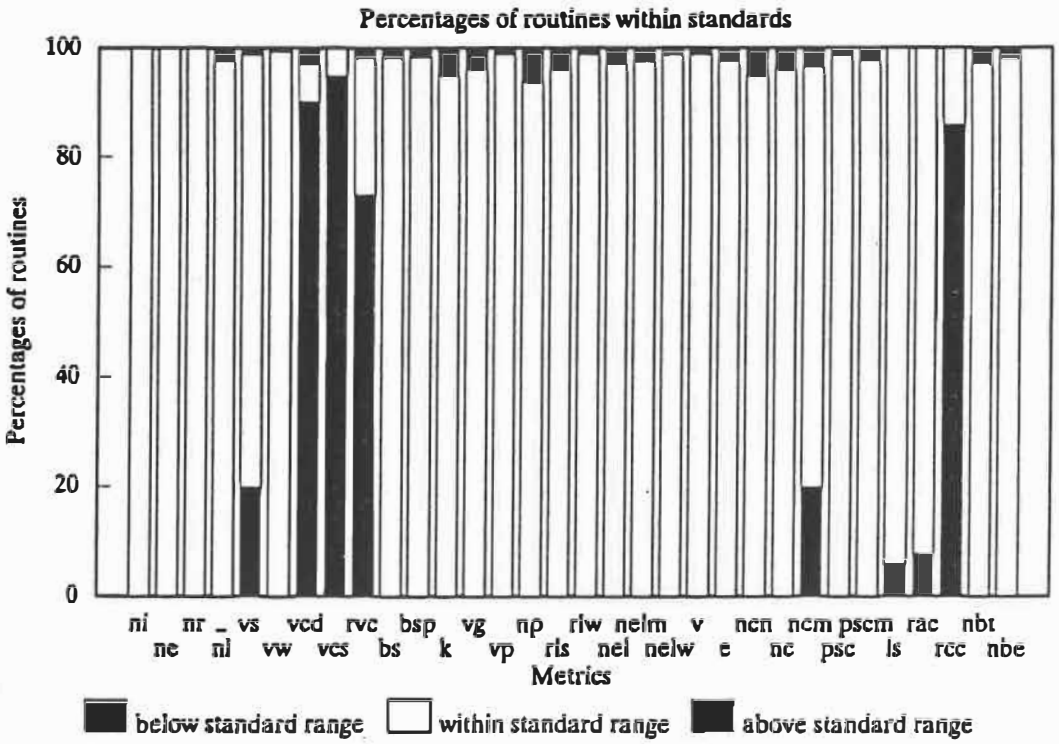

Figure 5.5: Graphique de percentiles par 123 de Lotus

En identifiant les attributs du logiciel qui sont mesurés par chaque métrique, on produit un profil quantitatif de la qualité (figure 5.6). Sept attributs sont définis comme l'intégrité, la modularité, la lisibilité, la concision, la testabilité, la simplicité et la structuration. Ces attributs sont les moyennes des pourcentages des routines respectant l'étendue des normes pour chacune des métriques appliquées à la norme. Par exemple, l'attribut de testabilité peut être composé des métriques Np, Nel et Nc.

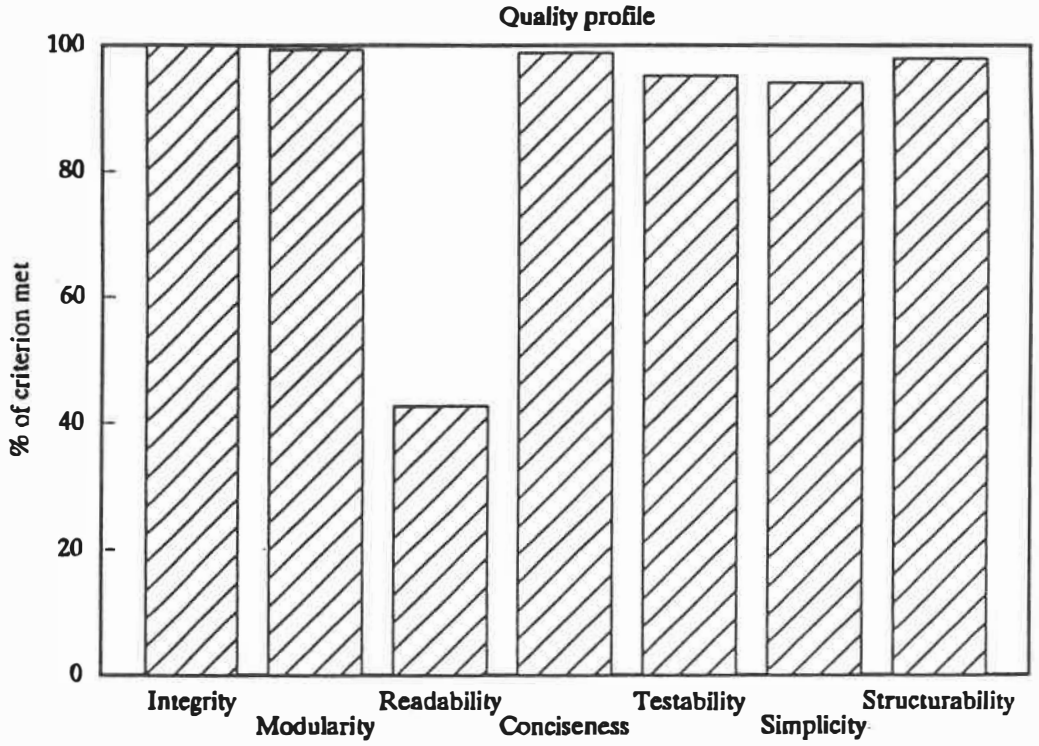

Figure 5.6: Profil qualitatif de la qualité

C. Forme relationnelle

L'exportation des résultats pour DBase III+ permet d'obtenir les données sous forme de relations. Une seule relation est définie. Les champs sont présentés par le tableau 5.2.

■ le langage de programmation ■ le nom du projet ■ le nom de la routine ■ la valeur de la métrique 1 ■ la valeur de la métrique 2 ... ■ la valeur de la métrique n

Tableau 5.2: Définition de la relation du format Dbase III +

Une définition de la structure de la relation et de la dimension de chacun des champs est fournie. L'approche n'est pas relationnelle car la dimension de la relation est fonction du nombre de métriques.

Pour obtenir une forme relationnelle, on propose la relation présentée par le tableau 5.3. Le champ valeur prend un nombre qui peut être entier, réel ou encore sous forme exponentielle. Cette approche n'est par contre pas implantée dans le module d'exportation de DATRIX.

■ le langage de programmation ■ le nom du projet ■ le nom de la routine ■ le nom de la métrique ■ la valeur de la métrique

Tableau 5.3: Définition de la relation sous la quatrième forme normale

D. Prédicats

L'option d'exportation pour Turbo-Prolog permet de générer un fichier qui sera chargé comme une base de faits. En respectant la terminologie de Turbo-Prolog, la définition du prédicat est la suivante:

database

metrics(langage,projet,routine,liste_valeurs_metriques)

où les champs sont définis par les types suivants:

domains

langage = symbol
projet = symbol
routine = symbol
metrique = symbol
valeur_metrique = value(metrique,real)
liste_valeurs_metriques = valeur_metrique*

Cette représentation sous forme de listes est indépendante du nombre de métriques. Il est alors facile de définir un système de règles qui sélectionneront les métriques respectant certains critères.

Par exemple, on peut définir les prédicats HARD_TEST (tableau 5.4), HIGH_VOLUME (tableau 5.5) et LOW_DOCUMENTATION (tableau 5.6) . Il est évident que ces 3 règles ne constituent pas un système complet ou correct pour tous les projets.

```
/* Print units hard to test */
/* Units with a large number of paths are printed ( Np >10000 ) */

   hard_test :-
      write("Units hard to test"), nl, nl,
      metrics(LANGUAGE,PROJECT,UNIT,METRIC_LIST),
      get_value("NP",METRIC_LIST,NP),
      NP > 10000,
      write("Language = ",LANGUAGE,"  Project = ",PROJECT,"  Routine = ",UNIT,
      "  Np = ",NP), nl,
      fail.

   hard_test :-
      nl.
```

Tableau 5.4: Exemple de clause pour trouver les routines difficiles à tester

```
/* Print routines with high volume */
/* Units with a large McCabe's cyclomatic number are unacceptable */

   high_volume :-
      write("Units with high volume") ,nl, nl,
      metrics(LANGUAGE,PROJECT,UNIT,METRIC_LIST),
      get_value("VG",METRIC_LIST,VG),
      VG > 15,
      write("Language = ",LANGUAGE,"  Project = ",PROJECT,"  Routine = ",UNIT,
      "  Vg = ",VG), nl,
      fail.

   high_volume :-
      nl.
```

Tableau 5.5: Exemple de clause pour trouver les routines avec un gros volume

```
/* Print routines having low levels of documentation                    */
/* A unit is considered not enough documented if:                       */
/*                                                                      */
/*   ■ the amount of comments in declaration are not sufficient;        */
/*   ■ the amount of comments in structures are not sufficient;         */
/*   ■ the amount of comments are not sufficient for the volume         */
/*     of the unit.                                                     */

   low_documentation :-
      write("Units with low levels of documentation"), nl, nl,
      metrics(LANGUAGE,PROJECT,UNIT,METRIC_LIST),
      get_value("VCD",METRIC_LIST,VCD),
      get_value("VCS",METRIC_LIST,VCS),
      get_value("RVC",METRIC_LIST,RVC),
      VCD < 100,
      VCS < 1000,
      RVC < 5.0,
      write("Language = ",LANGUAGE,"  Project = ",PROJECT,"  Routine = ",UNIT,
      "  Vcd = ",VCD,"  Vcs = ",VCS,"  Rvc = ",RVC), nl,
      fail.

   low_documentation :-
      nl.
```

Tableau 5.6: Exemple de clause pour trouver les routines peu documentées

E. Analyses statistiques avec SAS

Le transfert d'information le plus important pour cette recherche a été l'exportation des résultats vers SAS. Les données sont générées sous forme de relations identiques à celles présentées par le tableau 5.2. Les relations sont divisées en plusieurs lignes pour ne pas excéder 80 caractères par ligne. Cette contrainte imposée par SAS, BMDP et SPSS est évidemment une conséquence historique du développement de ces progiciels statistiques. Ils ont été développés pour des ordinateurs centraux qui utilisaient des cartes perforées.

L'analyse statistique avec SAS comporte beaucoup d'avantages. On peut citer les nombreuses procédures statistiques et le traitement des valeurs manquantes pour chacune de ces procédures.

L'analyse des données comporte souvent une partie interactive où à l'aide des résultats observés, on trace la direction des recherches subséquentes. Par contre, il est toujours nécessaire de vérifier les distributions des données, les statistiques élémentaires, etc. Pour ce faire, un prototype de générateur de rapport a été écrit en langage SAS. Il serait fastidieux d'inclure ce programme composé de plus de 1000 lignes. Le programme est exécuté en "batch" par SAS à l'aide de macro-opérations. Les tableaux 5.7 et 5.8 présentent respectivement les divers paramètres d'environnement à déterminer et la liste des résultats qui sont générés selon le choix de l'usager.

F. Autres

Il est évident que l'utilisation des résultats produits par DATRIX n'est pas limitée aux quatres logiciels précédents. Pour la plupart des autres logiciels un autre format devrait convenir. De plus, la plupart des logiciels ont un utilitaire qui permet de convertir les données provenant d'autres logiciels concurrents plus connus, comme ceux énumérés ci-haut. Dans un cas extrême, l'usager peut utiliser l'option ASCII. Ce format est identique à celui de DBase III+ auquel on a ajouté le nom des champs comme première ligne du fichier.

```
* Nom du projet et du repertoire où se trouve les sources;
%LET projet=datr89_f;

* Liste des métriques à considérer;
* Choix = { TOUTES ou la liste suivit de FIN};
%LET met_util=TOUTES;

* Type des données dans le DATASET;
* Choix = { DAT COR };
%LET type_don=DAT;

* Type de DEVICE pour les graphiques;
* Choix { EGAL (écran EGA), HPLJ0, HPLJ5P2, HPLJS2;
*                        (Imp HP 75, 150, 300 dpi)};
GOPTIONS DEVICE=EGAL;

* Critère pour la sélection des facteurs de l analyse factorielle;
* utilisant la méthode des composantes principales;
%LET crit_fac=0.5;

* Nombre de régions (Clusters) dans les analyses de clustering;
%LET nb_clust=10;

* Langue de travail pour la génération des rapports;
* Choix = { FRANCAIS ANGLAIS };
%LET langue=FRANCAIS;
```

Tableau 5.7: Options du générateur de rapports avec SAS

```
* Tâches à effectuer;

* Choix = { ON OFF } *;

%LET initial =ON;  * Crée les initialisations, macros, ... ;
%LET formdata=ON;  * Création du DATASET;
%LET imp_stat=ON;  * Statistiques élémentaires;
%LET grafdist=OFF;  * Graphiques (HBAR) des distributions des métriques;
%LET ana_fact=ON;  * Analyse factorielle, méthode des comp. princ.;
%LET clus_met=ON;  * Clustering sur les métriques;
%LET clus_fac=ON;  * Clustering sur les facteurs;

%LET sauvgraf=ON;  * Sauvegarde des graphes dans une librairie;
%LET sauv_log=ON;  * Sauvegarde du log dans un fichier;
%LET sauv_out=ON;  * Sauvegarde des résultats dans un fichier;
%LET quit_sas=ON;  * Quitter ou non SAS à la fin de la session de travail;
```

Tableau 5.8: Résultats pouvant être générés par le générateur de rapports avec SAS

VI. Description des données expérimentales

A. Les observations

Lors de la cueillette d'observations pour obtenir des métriques, certains facteurs sont difficilement contrôlables. Ce sont les sujets, les langages de programmation et les environnements de programmation. Deux alternatives s'offrent alors à l'expérimentateur:

A - effectuer une expérience contrôlée sur des sujets précis en choisissant les programmeurs, le langage et l'environnement ou

B - utiliser des données provenant de projets déjà réalisés.

L' alternative A comporte l'avantage d'obtenir des données plus précises. Cette optique s'effectue dans le cadre d'un cours sinon elle risque d'entraîner des coûts considérables pour l'entreprise qui veut la supporter puisque le logiciel produit est souvent jetable avant usage.

L'alternative B consiste à recueillir des données de projets déjà réalisés. Les données représentent mieux les projets de plus grande envergure. Par contre, certains paramètres sont inconnus comme l'expérience des programmeurs, le temps de développement, les coûts, etc.

Les données présentées dans ce mémoire proviennent des deux sources. L'alternative A est utilisée pour mesurer la perception intuitive de la complexité. La description de l'expérience et de ses résultats est effectuée au chapitre VIII. Pour l'ensemble des résultats des analyses statistiques du prochain chapitre, l'alternative B est utilisée.

Les données sur lesquelles des analyses statistiques sont effectuées sont présentées dans le tableau 6.1. Les observations recueillies sont des unités de logiciels, procédures ou fonctions, que l'on appelle routines. Les observations sont regroupées en projets recueillis auprès des entreprises ou dans la littérature. Le champ NOM, referant à la publication source identifie l'ensemble de données pour les analyses subséquentes. Les champs ROUTINES et LANGAGE identifient respectivement le nombre de routines et le langage de programmation utilisé. Le champ DONNEES indique si celles-ci sont des observations complètes, une matrice de covariance ou une matrice de corrélation. Le champ METRIQUES réfère aux tableaux de la section suivante sur les variables. Ils décrivent la liste des métriques pour chacun des ensembles de données suivants. Le champ NB donne le nombre de métriques dans l'ensemble de données. Finalement, la source des données est identifiée par une référence à l'article utilisé.

Notre étude porte sur 19 projets pour un total de 14348 routines. 6 différents languages ont été utilisés pour écrire ces routines.

NOM	ROUTINES	LANGAGE	DONNEES	METRIQUES	NB	SOURCE
DATR89-a	187	FORTRAN	Observations	DATR89-1	32	Projet commercial
DATR89-b	718	FORTRAN	Observations	DATR89-1	32	Projet universitaire
DATR89-c	216	FORTRAN	Observations	DATR89-1	32	Projet commercial
DATR89-d	161	FORTRAN	Observations	DATR89-1	32	Projet commercial
DATR89-e	255	C	Observations	DATR89-1	32	Projet commercial
DATR89-f	503	C	Observations	DATR89-1	32	Projet commercial
DATR89-g	2112	C	Observations	DATR89-1	32	Projet commercial
ELSH84	585	PL/1	Corrélations	ELSH84	20	[ELSH84]
HENR84	136	C	Corrélations	HENR84	6	[HENR84]
LI87	255	FORTRAN	Corrélations	LI87	18	[LI87]
LIND89-a	3442	Pascal	Corrélations	LIND89	11	[LIND89]
LIND89-b	1123	FORTRAN	Corrélations	LIND89	11	[LIND89]
SCHR84	921	Pascal	Corrélations	SCHR84	9	[SCHR84]
SUNO81	200	FORTRAN	Corrélations	SUNO81	8	[SUNO81]
VERI88-a	1020	FORTRAN	Corrélations	VERI88	19	[VERI88]
VERI88-b	927	Pascal	Corrélations	VERI88	19	[VERI88]
VERI88-c	914	C	Corrélations	VERI88	19	[VERI88]
VERI88-d	417	Modula-2	Corrélations	VERI88	19	[VERI88]
VERI88-e	256	COBOL	Corrélations	VERI88	19	[VERI88]

Tableau 6.1: Description des projets

Pour les données rassemblées sous les étiquettes DATR89-x, les métriques sont tirées directement du code source. Tous les projets recueillis sont complets. Dû à la confidentialité de l'information, les projets sont associés à des lettres. On dénote 4 projets réalisés avec le langage FORTRAN et 3 autres réalisés en langage C. Un projet est conçu par une équipe de programmeurs. Tous ces projets sont réalisés par des équipes différentes dans des lieux différents et conduisirent tous à des logiciels commerciaux.

Une routine est écrite par un programmeur et parfois modifiée par plusieurs autres programmeurs. Un programmeur touche donc à plusieurs routines, mais pas à l'ensemble du projet.

B. Les variables

Cette section ne s'applique qu'aux ensembles de données pour lesquels on a les observations complètes.

Pour chaque observation on retrouve les variables suivantes:

- LANGAGE (language de programmation)
- PROJET (le nom du projet auquel appartient la routine)
- ROUTINE (le nom de la routine, fonction, procédure...)

La variable LANGAGE identifie à la fois le type de langage (FORTRAN, Pascal, C) et le type de machine et/ou compilateur si la version du langage ne respecte pas les standards ANSI. Par exemple, pour le langage FORTRAN, la variable peut prendre les valeurs ANSI FORTRAN, VAX FORTRAN ou HP FORTRAN.

En plus de ces variables descriptives, on retrouve les valeurs des métriques. Le tableau 6.2 présente les métriques calculées par DATRIX. Elles constituent les variables pour les observations. L'ensemble complet de ces métriques est référé par DATR89-1.

Une analyse factorielle classique réalisée pour une publication a utilisé un sous-ensemble des métriques de DATR89-1. Ce sous-ensemble est référencé par DATR89-2 et est présenté dans le tableau 6.3.

Les tableaux 6.4 à 6.10 présentent les variables des matrices de corrélations tirées de la littérature. Chaque tableau rapporte le sigle de la métrique tel qu'utilisé par l'auteur sous le champ IDENTIFICATEUR, sa DESCRIPTION et le nom formel de la METRIQUE tel qu'il a été défini dans la section IIIa de ce mémoire.

Identificateur	Description	Métrique
Bs	Nombre de bris de structures	(Bs)
Bsp	Nb de bris de str. pondéré	(Bsp)
e	Nombre d'arcs	(e)
K	Nombre de croisements des arcs	(KNOT1)
Ls	Longueur moy. des noms des var.	(Ls)
Nbe	Nombre d'énoncés exécutables	(ESC)
Nbt	Nombre total de lignes	(LC)
Nc	Complexité moyenne des cond.	(Nc)
Ncmax	Complexité maximale des cond.	(Ncmax)
Ncn	Nb de sommets conditionnels	(Ncn)
Ne	Nombre de sommets de sortie	(Ne)
Nel	Niveau d'imbrication moyen	(Band)
Nelmax	Niveau d'imbrication maximal	(Bmax)
Nelw	Niveau d'imbrication pondéré	(Nelw)
Ni	Nombre de sommets d'entrée	(Ne)
Nl	Nombre de boucles	(Nl)
Np	Nombre de chemins indépendants	(Np)
Nr	Nombre de sommets récursifs	(Nr)
Psc	Portée moyenne des cond.	(Psc)
Pscmax	Portée maximum des cond.	(Pscmax)
Rac	Ratio des arcs commentés	(Rac)
Rls	Ratio structurel de boucle	(Rls)
Rlw	Ratio pondéré de boucle	(Rlw)
Rnc	Ratio des noeuds commentés	(Rnc)
RVc	Ratio du vol. de commentaires	(Rvc)
v	Nombre de sommets	(v)
Vcd	Vol. de com. des déclarations	(Vcd)
Vcs	Vol. de com. des structures	(Vcs)
Vg	Nombre cyclomatique	(Vg)
Vp	Nombre de sommets pendants	(Vp)
Vs	Volume structurel	(Vs)
Vw	Somme des pondérations	(Vw)

Tableau 6.2: Description de l'ensemble DATR89-1 (32 métriques)

Identificateur	Description	Métrique
e	Nombre d'arcs	(e)
K	Nombre de croisements des arcs	(KNOT1)
Nbe	Nombre d'énoncés exécutables	(ESC)
Nbt	Nombre total de lignes	(LC)
Ncn	Nb de sommets conditionnels	(Ncn)
Nel	Niveau d'imbrication moyen	(Band)
Nelmax	Niveau d'imbrication maximal	(Bmax)
Nelw	Niveau d'imbrication pondéré	(Nelw)
Nl	Nombre de boucles	(Nl)
Psc	Portée moyenne des cond.	(Psc)
Pscmax	Portée maximum des cond.	(Pscmax)
v	Nombre de sommets	(v)
Vg	Nombre cyclomatique	(Vg)
Vs	Volume structurel	(Vs)

Tableau 6.3: Description de l'ensemble DATR89-2 (14 métriques)

Identificateur	Description	Métrique
STATEMENTS	Nombre d'énoncés exécutables et non-exécutables	(SC)
IDENTIFIERS	Nombre de variables déclarées	(NV)
SOURCE LINES	Nombre de lignes incluant commentaires et lignes vides	(LOC)
INPUT LINES	Nombre de lignes après l'expension par le préprocesseur	(LC)
PREDICATES	Complexité cyclomatique	(Vg)
CONDITIONS	Nombre de conditions	(Ncn)
BLOCKS	Nombre d'énoncés définissant des blocks	(BLOCKS)
STATEMENTS AT LEVEL 10	Nombre d'énoncés imbriqués à plus de 10 niveaux	(SD10)
CALL STATEMENTS	Nombre d'énoncés CALL	(CALL)
UNIQUE OPERATORS	Nombre d'opérateurs uniques	(η1)
UNIQUE OPERANDS	Nombre d'opérandes uniques	(η2)
TOTAL OPERATORS	Nombre total d'opérateurs	(N1)
TOTAL OPERANDS	Nombre total d'opérandes	(N2)
VOCABULARY	Vocabulaire	(η)
LENGTH	Longueur du programme	(N)
VOLUME	Volume du programme ond. des cond.	(V)
DATA DIFFICULTY	Moyenne d'apparition de chaque opérande	(DD)
DIFFICULTY	Difficulté	(D)
UNDERSTANDING EFFORT	Effort de compréhension	(E)
CONSTRUCTION EFFORT	Effort de construction	(E^)

Tableau 6.4: Description de l'ensemble ELSH84 (20 métriques)

Identificateur	Description	Métrique
N	Longueur du programme	(N)
NHAT	Longueur estimée du programme	(N^)
V	Volume du programme	(V)
E	Effort de programmation	(E)
McCabe	Complexité cyclomatique	(Vg)
Information flow	Flow d'information	(IF)

Tableau 6.5: Description de l'ensemble HENR84 (6 métriques)

Identificateur	Description	Métrique
STMTS	Nombre d'énoncés	(SC)
LN-CM	Lignes de code sans les commentaires	(LE)
NODES	Nombre de sommets du graphe de contrôle	(v)
EDGES	Nombre d'arcs du graphe de contrôle	(e)
McCBE	Complexité cyclomatique	(Vg)
SCOPE	Métrique scope	(SCOPE)
n2	Nombre d'opérandes uniques	(g2)
N1	Nombre total d'opérateurs	(N1)
N2	Nombre total d'opérandes	(N2)
n	Vocabulaire	(ξ)
N	Longueur du programme	(N)
N^	Longueur du programme calculée	(N^)
V	Volume du programme	(V)
IC	Contenu d'intelligence	(I^)
E^	Effort calculé 1	(E^)
E^^	Effort calculé 2	(E^^)
CL	Complexité logique absolue de Gilb	(CL)
KNOT2	Nombre de noeuds possibles de Knot	(KNOT2)

Tableau 6.6: Description de l'ensemble LI87 (18 métriques)

Identificateur	Description	Métrique
Total Lines	Nombre total de lignes	(L)
Code Lines	Nombre de lignes de code	(LOC)
Total Chars	Nombre de caractères du source	(CHAR)
Comments	Nombre de commentaires	(C)
Comments Chars	Nombre de caractères des commentaires	(CCHAR)
Code Chars	Nombre de caractères de code	(SCHAR)
Halstead's N	Longueur observée du programme	(N)
Halstead's NH	Longueur calculée du programme	(N^)
Jensen's NH	Longueur estimée du programme	(NF)
McCabe's MC	Complexité cyclomatique	(Vg)
Bandwidth BW	Niveau moyen d'imbrication du programme	(Band)

Tableau 6.7: Description de l'ensemble LIND89 (11 métriques)

Identificateur	Description	Métrique
Nst	Nombre d'énoncés	(SC)
TT	Grosseur de l'arbre syntaxique	(TT)
Voc	Vocabulaire	(η)
Ond	Nombre d'opérateurs distincts	(η1)
Otd	Nombre d'opérandes distinctes	(η2)
cmd	Complexité cyclomatique	(Vg)
Dpt	Profondeur de l'arbre syntaxique	(DpT)
Mxd	Niveau d'imbrication maximum	(Bmax)
Mnd	Niveau d'imbrication moyen	(Band)

Tableau 6.8: Description de l'ensemble SCHR84 (9 métriques)

Identificateur	Description	Métrique
STEPmax	Nombre d'énoncés incluant commentaires	(S)
STEPmid	Nombre d'énoncés excluant commentaires	(SC)
STEPmin	Nombre d'énoncés excluant commentaires et déclarations	(ESC)
V(G)max	Complexité cyclomatique en considérant les conditions	(Vg)
V(G)min	Complexité cyclomatique en considérant les prédicats	(Vge)
E	Effort de programmation	(E)
WSC	Nombre d'énoncés pondéré	(WSC)
PVG	Processus	(PVG)

Tableau 6.9: Description de l'ensemble SUNO81 (8 métriques)

Identificateur	Description	Métrique
Nins	Nombre d'instructions	(SC)
Ncom	Nombre de commentaires	(C)
Tcom	Ratio de commentaires	
Oopa	Nombre d'occurences des opérandes	(N2)
Nopa	Nombre d'opérandes distinctes	(η2)
Narc	Nombre d'arcs	(e)
Nnoe	Nombre de sommets	(v)
Noep	Nombre de sommets pendants	(Vp)
V(G)	Nombre cyclomatique	(Vg)
D-C	Densité de contrôle	
Nmnv	Nombre maximum de niveaux	
Nmdg	Nombre maximum de degrés	
Nvpg	Niveau du programme	(L)
Cont	Contenu d'intelligence	(I^)
Diff	Difficulté du programme	(D)
Effo	Effort mental	(E)
Nerr	Nombre estimé d'erreurs	
Tpro	Temps de programmation	(T^)
Nvlg	Niveau du langage	(λ)

Tableau 6.10: Description de l'ensemble VERI88 (20 métriques)

Quelques identificateurs tirés de l'ensemble VERI88 n'ont pas de références à des métriques prédéfinies au chapitre IV. Malgré l'accès à la documentation du produit, il a été impossible de savoir ce qui est réellement mésuré par ces métriques.

VII. Résultats des analyses statistiques sur les données

A. Mesures de dispersion

1. Moyennes

Le tableau 7.1 présente les valeurs moyennes des 32 métriques de l'ensemble DATR89-1. Les résultats sont fournis pour chacun des sept projets DATR89_a à DATR89_g.

On remarque les différences notables dans les moyennes entre chaque projet. Si l'on suppose que l'ensemble des projets existants est tiré d'une population caractérisée par divers paramètres, une analyse de variance (ANOVA) confirmera ou rejettera les hypothèses d'égalité de moyennes pour tous les projets.

On peut utiliser ce tableau pour comparer les projets entre eux. Les données présentées décrivent les projets en ce sens:

- Le projet DATR89_a a des routines qui sont très imbriquées (Nel). Beaucoup de bris de structures (Bs et Bsp) et une testabilité difficile (Np) caractérisent aussi le projet.
- Le projet DATR89_b effectue beaucoup d'opérations itératives (Nl) et ne respecte pas les règles de la programmation structurée.
- Le projet DATR89_c n'est pas documenté (Vcd, Rac et Rnc) et est composé de beaucoup d'opérations itératives (Nel) et de non-structure (Bs et Bsp).

- Le projet DATR89_d décrit la moyenne des projets dans tous les sens à part le fait qu'il est bien documenté (Rnc et Rac) et bien structuré (Bs).
- Le projet DATR89_e a des routines (Nbt) et des déclarations (Vcd) très longues. Très bien documenté, par contre les expressions booléennes des conditionnelles semblent un peu complexes (Nc). On remarque aussi le nombre astronomique de lignes et d'énoncés par routine (Nbt et Nbe).

Métrique	FORTRAN				C		
	DATR89_a	DATR89_b	DATR89_c	DATR89_d	DATR89_e	DATR89_f	DATR89_g
Bs	28.08	14.34	13.92	0.00	0.00	1.14	1.81
Bsp	269.44	319.05	251.76	0.00	0.00	47.60	126.68
e	39.89	44.61	44.30	37.60	29.90	34.29	21.35
K	33.10	25.02	107.21	3.98	3.20	7.42	5.32
Ls	3.00	4.20	3.27	4.68	5.39	6.68	4.89
Nbe	64.49	137.26	51.54	80.00	941.48	1031.78	99.77
Nbt	205.50	463.44	71.31	148.87	2551.21	2753.91	1148.46
Nc	1.39	2.29	2.58	1.77	3.76	3.70	4.06
Ncmax	3.23	5.74	6.25	4.04	8.06	8.00	7.22
Ncn	13.26	12.98	12.00	11.33	9.47	10.43	6.06
Ne	0.83	1.19	1.78	0.99	1.00	1.39	2.45
Nel	5.33	3.70	2.96	2.69	2.76	2.95	2.31
Nelmax	9.07	6.25	4.66	4.72	4.46	4.69	3.49
Nelw	3.88	2.34	2.23	2.18	2.86	2.24	2.20
Ni	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Nl	1.75	2.78	2.65	1.52	1.11	1.69	0.03
Np	2.02	1.46	1.10	1.72	1.59	1.51	1.07
Nr	0.00	0.00	0.00	0.00	0.01	0.09	1.23
Psc	7.10	4.92	4.03	3.12	3.64	4.97	2.42
Pscmax	19.47	21.00	18.48	14.57	14.55	17.50	7.96
Rac	31.81	27.79	0.22	49.47	47.32	65.53	44.37
Rls	0.48	0.45	0.50	0.38	0.30	0.50	0.39
Rlw	0.36	0.23	0.43	0.22	0.02	0.05	0.24
Rnc	10.93	15.56	0.00	19.04	11.92	13.19	7.72
Rvc	23.04	32.49	3.39	30.92	33.91	87.04	129.69
v	27.64	31.29	31.27	27.27	21.44	24.84	16.74
Vcd	1196.79	6402.25	2.99	1173.39	13436.41	7991.58	4296.67
Vcs	168.74	345.29	107.36	489.78	433.83	829.33	320.69
Vg	14.25	15.32	15.03	12.33	10.47	11.84	8.06
Vp	0.10	0.07	0.01	0.00	0.00	1.12	0.35
Vs	11.09	12.48	12.43	10.64	12.74	14.68	9.49
Vw	48.87	116.01	37.29	56.53	977.55	1158.73	161.36

Tableau 7.1: Moyennes pour différents projets

- Le projet DATR89_f ressemble énormément au projet DATR89_e en beaucoup de points. Le projet possède par contre de la non-structure (Bs et Bsp) et un volume plus petit (e, v et Psc).
- Le projet DATR89_g ressemble aux 2 précédents sauf que le niveau de documentation est inférieur. On note aussi un volume moins gros (Nbe, Vs, Psc, ...) et plus de non-structure (Bs et Bsp).

2. Écarts-type

Pour pouvoir tester si différents échantillons proviennent d'une même population à l'aide des moyennes, on doit savoir si les écarts-type sont égaux ou non. Dans un cas ou dans un autre, on n'utilisera pas la même procédure pour tester l'égalité des moyennes.

Le tableau 7.2 présente les valeurs des écarts-type des 32 métriques de l'ensemble DATR89-1. Les résultats sont fournis pour chacun des sept projets DATR89_a à DATR89_g. Les écarts-type n'ont aucun sens s'il ne sont pas accompagnés des moyennes et si l'on a pas une idée générale de la distribution.

Du tableau 7.2, on remarque des différences prononcées entre les écarts-type. Cette différence est beaucoup moins marquée si l'on regarde les projets réalisés dans un même langage, soit DATR89_a à DATR89_d pour le FORTRAN et DATR89_e à DATR89_g pour le langage C.

Métrique	FORTRAN				C		
	DATR89_a	DATR89_b	DATR89_c	DATR89_d	DATR89_e	DATR89_f	DATR89_g
Bs	45.22	113.57	56.31	0.00	0.00	7.36	15.37
Bsp	591.80	3284.14	2159.26	0.00	0.00	527.03	1820.82
e	35.28	81.09	128.41	57.29	52.27	52.86	37.44
K	50.52	135.77	980.74	6.58	5.63	23.17	24.31
Ls	0.67	1.08	1.53	1.12	1.13	1.63	2.16
Nbe	47.86	139.05	124.57	112.73	80.30	353.10	173.72
Nbt	95.63	423.79	138.41	175.87	257.60	835.06	2315.68
Nc	0.90	2.81	2.64	1.15	2.75	3.31	3.59
Ncmax	2.42	7.66	9.67	4.55	8.88	10.28	6.59
Ncn	12.91	24.07	38.90	21.45	17.26	17.30	11.13
Ne	0.41	0.80	3.42	0.34	0.06	2.11	4.30
Nel	4.06	3.95	5.35	1.89	1.82	1.86	1.41
Nelmax	7.43	6.95	9.48	4.14	3.89	3.71	2.89
Nelw	2.95	2.51	4.22	1.61	1.78	1.66	1.32
Ni	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Nl	2.85	6.00	10.70	2.28	1.74	2.80	0.28
Np	2.12	2.11	1.70	2.48	2.02	1.70	1.45
Nr	0.00	0.00	0.00	0.00	0.11	1.79	2.46
Psc	6.16	5.12	13.89	2.39	3.37	4.34	2.24
Pscmax	20.13	44.67	69.55	27.60	29.79	30.20	17.38
Rac	24.76	24.31	3.33	24.13	28.43	25.20	33.43
Rls	1.13	0.96	2.94	0.45	0.40	0.61	0.59
Rlw	0.82	0.69	2.69	0.30	0.07	0.14	0.48
Rnc	17.40	24.07	0.00	23.51	11.45	11.28	14.28
Rvc	38.12	80.55	11.08	33.76	32.73	412.23	884.33
v	22.57	51.35	83.07	36.72	35.02	35.79	26.58
Vcd	546.50	6656.79	44.90	1008.47	980.48	2140.48	8813.02
Vcs	148.99	769.47	526.97	934.93	716.03	1286.80	874.62
Vg	12.89	30.53	45.52	21.45	17.26	17.43	14.04
Vp	0.32	0.36	0.09	0.00	0.00	3.08	1.42
Vs	9.63	22.04	35.24	15.64	21.82	22.14	15.94
Vw	37.14	115.89	84.97	79.21	134.87	433.46	251.24

Tableau 7.2: Écarts-type pour différents projets

B. Distribution des données

Le test de Shapiro-Wilk est effectué sur chacune des variables de chacun des ensembles de données de DATR89_a à DATR89_g. La normalité est loin d'être atteinte comme le démontre le tableau 7.3. Pour éviter d'énumérer toutes les valeurs de W du test, seule la valeur la plus élevée parmi tous les projets pour chacune des métriques est rapportée. La colonne suivante rapporte la probabilité de rejeter l'hypothèse de normalité alors qu'elle est vraie. La dernière colonne indique quel projet a fourni la distribution la plus près de la normalité.

Les résultats de ce tableau permettent de conclure qu'il n'existe aucune normalité dans les données recueillies. Aucune des 224 (32*7) distributions n'est normale.

Métrique	W	Prob<W	Projet
Bs	0.67	0.0	DATR89_a
Bsp	0.53	0.0	DATR89_a
e	0.87	0.0	DATR89_a
K	0.69	0.0	DATR89_a
Ls	0.94	0.0	DATR89_f
Nbe	0.89	0.0	DATR89_a
Nbt	0.97	0.0146	DATR89_f
Nc	0.93	0.0001	DATR89_d
Ncmax	0.88	0.0	DATR89_a
Ncn	0.86	0.0	DATR89_a
Ne	0.51	0.0	DATR89_d
Nel	0.88	0.0	DATR89_a
Nelmax	0.89	0.0	DATR89_a
Nelw	0.85	0.0	DATR89_a
Ni	-	-	DATR89_a
Nl	0.68	0.0	DATR89_e
Np	0.82	0.0	DATR89_a
Nr	0.01	0.0	DATR89_a
Psc	0.90	0.0	DATR89_a
Pscmax	0.85	0.0	DATR89_a
Rac	0.93	0.0001	DATR89_d
Rls	0.81	0.0	DATR89_d
Rlw	0.77	0.0	DATR89_d
Rnc	0.90	0.0	DATR89_f
Rvc	0.82	0.0	DATR89_d
v	0.88	0.0	DATR89_a
Vcd	0.96	0.0001	DATR89_a
Vcs	0.89	0.0	DATR89_a
Vg	0.86	0.0	DATR89_a
Vp	0.36	0.0	DATR89_f
Vs	0.88	0.0	DATR89_a
Vw	0.89	0.0	DATR89_f

Tableau 7.3: Test de Shapiro-Wilk pour chacune des métriques.

On peut voir dans le tableau 7.4 les valeurs de Prob<W les plus près de la normalité.

Si on veut utiliser les tests basés sur la normalité des observations, on doit utiliser des transformations permettant d'approcher la normalité.

Projet	Vcd	Nc	Rac	Nbt
DATR89_a	0.96	0.86	0.78	0.97
DATR89_b	0.81	0.52	0.85	0.84
DATR89_c	0.07	0.84	0.07	0.46
DATR89_d	0.78	0.93	0.93	0.70
DATR89_e	0.32	0.84	0.84	0.49
DATR89_f	0.83	0.64	0.92	0.80
DATR89_g	0.81	0.52	0.85	0.84

Tableau 7.4: Coefficient W pour quelques métriques pour tous les projets

C. Transformations

Il serait fastidieux de rapporter toutes les transformations expérimentées pour approcher la normalité. Il est important de souligner qu'une transformation est intéressante si elle demeure simple. On veut être capable de comprendre la variable transformée. Dans des cas extrêmes, une transformation plus compliquée est acceptée si le comportement de chacun des ensembles de données est bien représenté par cette transformation.

Comme il a été vu précédemment dans les tableaux des moyennes et des écarts-type, les variables sont très peu homogènes. Les transformations de TUKEY sont effectuées sur les variables. Elles permettent d'approcher une courbe normale.

Les résultats des transformations pour les variables Vcd, Vg, Nbt et Nel sont représentées par le graphique suivant. Les données sont tirées du projet DATR89_a. L'axe des Y représente la valeur W du test de Shapiro-Wilk tandis que l'axe des X représente du même coup l'indice (λ) de la transformation et l'exposant de la variable. La figure 7.1 montre la divergence du résultat pour des valeurs s'éloignant de 1.

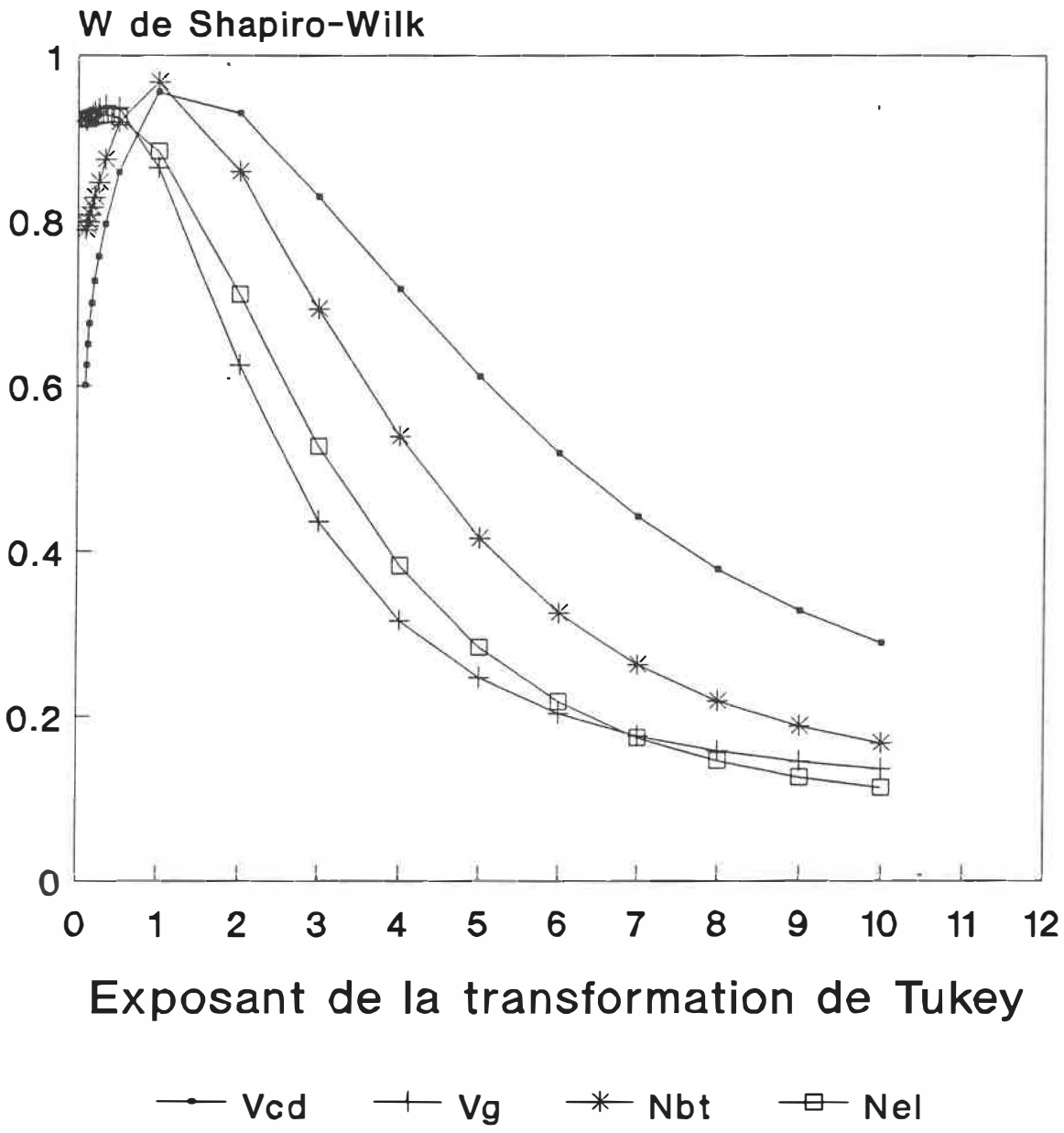

Figure 7.1: Recherche de normalité à l'aide des transformations de Tukey

Pour la transformation impliquant le logarithme de la valeur de la variable (transformation d'indice (0)), les résultats (tableau 7.5) ne sont pas plus satisfaisants.

Variable	W	Prob<W
Nbt	0.75	0.0146
Vcd	0.65	0.0
Vg	0.90	0.0
Nel	0.92	0.0

Tableau 7.5: Transformation d'indice (0) de Tukey.

Les résultats vont dans le même sens que ceux de la section précédente. Il n'y a aucune normalité dans la distribution des données et il ne semble pas évident d'y arriver à l'aide de transformations. On ne peut donc pas utiliser les tests paramétriques sur les données.

De plus, l'utilisation des paramètres tels que la moyenne et l'écart-type sont beaucoup moins significatifs si l'on ne sait pas à quel genre de distribution s'attendre. Lors de la représentation des données, on décrira mieux la distribution à l'aide de divers percentiles que la moyenne et l'écart-type.

D. Analyse de variance

L'analyse de variance permet de vérifier si plusieurs moyennes d'échantillons peuvent être considérées égales. Le non-rejet de l'hypothèse de l'égalité des moyennes entraîne le non-rejet de l'hypothèse que tous les échantillons proviennent de la même population. Ainsi, si l'on ne peut rejeter le fait que toutes les moyennes pour une métrique donnée sont identiques, on pourrait considérer qu'il existe une population des routines et que chaque projet peut être vu comme un échantillon. Cette considération n'est valable que sous l'hypothèse d'indépendance des données.

Les tests d'analyse de variances sont bâtis à l'aide de distributions normales. Par contre, les statisticiens considèrent ces tests assez robustes et assument généralement que l'hypothèse de normalité n'est pas essentielle.

Dans l'analyse rapportée par le tableau 7.6, chaque projet est considéré comme un échantillon. La population totale est l'ensemble de toutes les routines réalisées. L'hypothèse est que chaque projet est un échantillon de la population avec lequel on peut estimer la moyenne des métriques de la population.

Le tableau 7.6 présente les résultats pour les 7 projets de DATR89. Les résultats démontrent que très peu de métriques (Ni, Rvc, Rls et Bsp) peuvent être considérés comme ayant des moyennes égales pour tous les projets.

On marque par M les distributions faisant partie du groupe moyen, par H le groupe supérieur à la moyenne, par HH le groupe supérieur au groupe H, par B le groupe inférieur au groupe moyen M et par BB le groupe inférieur au groupe B. Les projets présentant la même lettre pour une métrique donnée sont considérés comme ayant la même moyenne avec une erreur de première espèce de 0.05.

On utilise le test des multiples intervalles de Ryan-Eliot-Gabriel-Welsh avec la procédure ANOVA de SAS/STAT. Ce test vise à trouver le nombre optimal de groupes qui possèdent la même moyenne.

Métrique	FORTRAN				C		
	DATR89_a	DATR89_b	DATR89_c	DATR89_d	DATR89_e	DATR89_f	DATR89_g
Bs	H	M	M	B	B	B	B
Bsp	M	M	M	M	M	M	M
e	M	H	H	M	M	M	B
K	M	M	H	M	M	M	M
Ls	BB	B	BB	M	H	HH	M
Nbe	B	M	B	B	H	HH	M
Nbt	M	M	B	M	HH	HH	H
Nc	B	M	M	B	H	H	H
Ncmax	B	M	M	B	H	H	M
Ncn	M	M	M	M	M	M	B
Ne	M	M	H	M	M	M	HH
Nel	HH	H	M	M	M	M	B
Nelmax	HH	H	M	M	M	M	B
Nelw	HH	M	M	M	H	M	M
Ni	M	M	M	M	M	M	M
Nl	M	H	H	M	M	M	B
Np	H	M	B	M	M	M	B
Nr	M	M	M	M	M	M	H
Psc	HH	H	M	M	M	H	B
Pscmax	M	M	M	M	M	M	B
Rac	B	B	BB	M	M	H	M
Rls	M	M	M	M	M	M	M
Rlw	M	M	H	M	B	B	M
Rnc	M	H	BB	HH	M	M	B
Rvc	M	M	M	M	M	M	M
v	M	H	H	M	M	M	B
Vcd	BB	M	BB	BB	HH	H	B
Vcs	B	M	B	M	M	H	M
Vg	M	M	M	M	M	M	B
Vp	M	M	B	B	B	H	M
Vs	M	M	M	M	M	M	B
Vw	B	M	B	B	H	HH	M

Tableau 7.6: Analyse de variance pour tous les projets

Les tableaux 7.7 et 7.8 qui suivent rapportent les résultats pour les projets réalisés dans un même language. Les populations cibles pour ces tableaux sont respectivement l'ensemble des routines écrites en FORTRAN et l'ensemble des routines en C.

Le tableau 7.7 décrit les résultats pour le langage FORTRAN, tandis que le tableau 7.8 rapporte les résultats des projets réalisés en language C.

Métrique	DATR89_a	DATR89_b	DATR89_c	DATR89_d
Bs	H	M	M	M
Bsp	M	M	M	M
e	M	M	M	M
K	M	M	H	M
Ls	BB	H	B	HH
Nbe	M	H	B	M
Nbt	M	H	B	M
Nc	B	H	H	B
Ncmax	B	H	H	B
Ncn	M	M	M	M
Ne	B	M	H	M
Nel	H	M	M	B
Nelmax	H	M	M	M
Nelw	H	M	M	M
Ni	M	M	M	M
Nl	M	M	M	M
Np	H	M	B	M
Nr	M	M	M	M
Psc	H	M	M	B
Pscmax	M	M	M	M
Rac	M	M	B	H
Rls	M	M	M	M
Rlw	M	M	M	M
Rnc	M	H	B	H
Rvc	M	M	B	M
v	M	M	M	M
Vcd	M	H	B	M
Vcs	B	H	B	H
Vg	M	M	M	M
Vp	H	M	M	B
Vs	M	M	M	M
Vw	M	H	M	M

Tableau 7.7: Analyse de variance pour les projets réalisés en FORTRAN

Métrique	DATR89_e	DATR89_f	DATR89_g
Bs	M	M	M
Bsp	M	M	M
e	M	M	B
K	M	H	M
Ls	M	H	B
Nbe	M	H	B
Nbt	M	M	B
Nc	M	M	M
Ncmax	M	M	M
Ncn	M	M	B
Ne	M	M	H
Nel	M	M	B
Nelmax	M	M	B
Nelw	M	H	M
Ni	M	M	M
Nl	M	H	B
Np	M	M	B
Nr	M	M	H
Psc	M	H	B
Pscmax	M	H	B
Rac	M	H	M
Rls	B	H	M
Rlw	M	M	H
Rnc	M	M	B
Rvc	M	M	M
v	M	M	B
Vcd	H	M	B
Vcs	M	H	M
Vg	M	M	B
Vp	B	H	M
Vs	M	M	B
Vw	M	H	B

Tableau 7.8: Analyse de variance pour les projets réalisés en C

Les tableaux nous permettent de voir que certaines métriques présentent des moyennes qui se ressemblent pour tous les projets: Ni, Nr, Vs, Rvc, Bsp, K, Vg, Rls et Ncn. Alors que pour le FORTRAN et le C, les métriques suivantes pourraient être tirées d'une même population.

FORTRAN: Ni, Nl, Nr, Vs, Bsp, Vg, Rls, Rlw, V, E, Ncn.

C: Ni, Rvc, Bs, Bsp, Nc, Ncmax.

Il est dangereux de construire des coefficients de confiance pour les valeurs. Dans un premier temps à cause de la non-normalité et aussi parce que le nombre de projets est faible et que l'on ne peut assumer que les projets obtenus représentent la qualité exemplaire.

L'analyse de ces trois tableaux nous porte quand même à conclure que les projets ne peuvent pas être considérés comme des échantillons. Les résultats trouvés sur ces projets ne peuvent être inférés à l'ensemble des routines réalisées dans le monde de l'informatique.

E. Analyses factorielles classiques

Les analyses factorielles classiques sont effectuées dans le but de mesurer le nombre de dimensions mesurées par des ensembles de métriques. Chaque analyse est effecutée sur un projet entier. Un projet représente donc une population. Aucune inférence ne sera tirée des analyses, l'analyse factorielle classique sera utilisée à titre descriptif seulement.

1. Paramètres du modèle

L'analyse factorielle est effectuée avec la procédure FACTOR du module SAS/STAT de SAS version 6.03. La procédure utilise le modèle des composantes principales. Pour les projets dont on n'a pas les valeurs de chaque observation, on utilise un modèle basé sur la matrice des corrélations.

La somme des valeurs propres des facteurs extraits d'un ensemble de données est égale à P, où P est le nombre de variables. On retient les valeurs des facteurs ayant des valeurs propres supérieures à 1. Ces facteurs expliquent au moins autant de variabilité qu'expliquerait une variable indépendante.

Aucune rotation n'est effectuée, car on ne cherche pas à identifier les facteurs. Des recherches précédentes dans le cadre de ce travail de maîtrise n'ont pas obtenu des résultats homogènes ou même satisfaisant pour l'interprètation des axes. On cherche plutôt à démontrer l'importance des facteurs dans le modèle.

2. Résultats des analyses factorielles classiques

En partant de l'hypothèse de dépendance entre ces métriques émise par bon nombre de chercheurs [LIND89] en génie logiciel, l'application de l'analyse factorielle à un ensemble de métriques devrait résulter en un espace de facteurs considérablement plus petit.

Une analyse factorielle classique a été effectuée sur chacun des projets énumérés dans le tableau 6.1. Les résultats sont présentés par le tableau 7.9. A coté du nom du projet, on retrouve le nombre de facteurs ayant des valeurs propres supérieures à 1, suivi de la proportion de la variance expliquée pour ce nombre de facteurs. Cette proportion est exprimée en terme de pourcentage. Les colonnes qui suivent expriment les proportions de variance exprimée par chacun des facteurs pris individuellement. On note que l'on donne les valeurs que pour les quatre premiers facteurs, les autres valeurs étant peu significatives.

PROJET	FACT. AVEC VP>1	SOM VAR. EXP.	% var. exp. par facteurs			
			1	2	3	4
DATR89_a	6	84.1	57.3	8.3	6.5	4.4
DATR89_b	8	81.9	37.2	12.9	8.8	5.8
DATR89_c	6	88.3	54.7	10.0	7.0	6.7
DATR89_d	5	82.2	55.4	11.3	6.0	5.3
DATR89_e	7	83.8	49.6	9.7	6.3	5.5
DATR89_f	7	83.8	38.0	15.7	11.1	6.0
DATR89_g	7	81.7	39.4	15.2	8.9	5.5
ELSH84	4	81.6	55.3	13.2	7.0	6.0
HENR84	1	79.1	79.1			
LI87	1	90.6	90.6			
LIND89_a	1	76.2	76.2			
LIND89_b	1	79.6	79.6			
SCHR84	2	88.3	71.7	16.5		
SUNO81	1	87.2	87.2			
VERI88_a	4	86.6	60.3	12.1	8.9	5.3
VERI88_b	3	81.2	62.5	10.7	8.0	
VERI88_c	4	84.6	58.4	10.6	8.1	7.5
VERI88_d	4	84.0	58.0	13.4	7.5	5.0
VERI88_e	4	85.4	62.5	10.0	7.3	5.6
Moyennes	4.0	83.7	61.7			

Tableau 7.9: % de variabilité expliquée pour chaque projet

On remarque que le modèle donne de moins bon résultats lorsque l'on tend à augmenter le nombre d'observations ou le nombre de variables. Mis à part les projets DATR89, les autres ensembles de données ont nécessité 4 facteurs ou moins. Le tableau 7.10 donne les projections de chaque métrique sur le facteur principal lorsque cette projection est supérieure à 0.8 pour les projets DATR89. Les métriques non répertoriées dans le tableau n'ont aucune forte corrélation avec le facteur principal. Il est intéressant de remarquer que l'ensemble des métriques du tableau 7.10 sont reliées à la grosseur d'une routine.

Métrique	Données des ensembles de DATR89						
	a	b	c	d	e	f	g
Vs	.97	.96	.96	.97	.95	.94	.95
Vw	.82		.88	.92	.85		
Vcs				.91	.87	.86	
Bs	.83						
K	.84			.89	.85		
Vg	.98	.94	.96	.90	.95	.94	.93
Np	.84		.87			.82	
Nel	.93	.82	.91		.88	.86	.80
Nelmax	.94	.82	.91		.87	.86	.80
Nelw	.92		.91		.87	.84	.81
v	.98	.95	.95	.97	.95	.94	
e	.98	.96	.96	.96	.95	.94	.94
Ncn		.96	.95	.90	.95	.93	.95
Psc	.95	.83	.85				.94
Pscmax	.96	.92	.92		.95	.90	.91
Nbt	.89	.88	.92	.94			
Nbe	.90	.80	.95	.95			

Tableau 7.10: Métriques avec de fortes projections sur l'axe principal pour les projets DATR89

Il est trop fastidieux de décrire d'une façon détaillée les résultats pour chacun des 19 projets. L'essentiel de ces résultats est déjà rapporté dans le tableau 7.9.

Les pages suivantes décrivent les résultats pour deux de ces projets. Ensuite on rapporte les résultats d'une autre expérience où l'on a conservé que des "métriques de volume" du sous-ensemble de métriques DATR89-1.

Les tableaux 7.11 et 7.12 qui suivent présentent les projections des métriques sur les principaux axes pour les ensembles de données HENR84 et LI87.

Le projet HENR84 est un exemple typique de données où l'ajout d'une nouvelle métrique amène de l'information au projet. L'auteur rapporte une matrice de corrélation où l'on peut voir que la dernière métrique INFLOW n'est pas corrélée avec les autres métriques. L'utilisation d'un modèle d'analyse en composantes principales présente les résultats d'une façon beaucoup plus évidente. Une seule colonne démontre que toutes les métriques se projètent sur le même facteur, identifié comme le volume, tandis que la dernière métrique est peu corrélée avec cet axe.

On retient de ce tableau 7.11 que l'analyse factorielle avec le modèle des composantes principales peut être utilisé pour démontrer l'utilité d'une métrique pour certains projets.

PROJECTIONS		
Métriques	Facteur1	Var expliquée
N	0.99	0.98
NHAT	0.93	0.87
V	0.99	0.98
E	0.94	0.89
MCCABE	0.94	0.89
INFLOW	0.40	0.16
VARIANCE EXPLIQUÉE/VARIANCE TOTALE		
	Facteur1	Total
Ratio	0.791	0.791

Tableau 7.11: Résultats avec les données de HENR84

Li [LI87] est souvent référencé dans la littérature à cause du nombre de métriques considérées. L'analyse de ses données comporte un fait remarquable (tableau 7.12). Un seul facteur, identifié ici aussi comme le volume, réussit à expliquer plus de 90% de la variance totale. Ces résultats démontrent que même les métriques textuelles telles que celles d'Halstead peuvent être considérées comme des métriques de volume dans certains cas. Ces métriques n'apportent donc pas plus d'information dans les données pour ce projet.

PROJECTIONS		
Métriques	Facteur1	Var expliquée
STMTS	0.97	0.93
LN-CM	0.95	0.90
NODES	0.97	0.94
EDGES	0.96	0.93
McCBE	0.96	0.92
SCOPE	0.90	0.81
n2	0.96	0.92
N1	0.98	0.95
N2	0.98	0.96
n	0.97	0.93
N	0.98	0.97
N^	0.96	0.93
V	0.99	0.98
IC	0.89	0.80
E^	0.94	0.88
E^^	0.94	0.88
CL	0.93	0.87
KNOT2	0.92	0.85

VARIANCE EXPLIQUÉE/VARIANCE TOTALE		
	Facteur1	Total
Ratio	0.906	0.906

Tableau 7.12: Résultats avec les données de LI87

Un sous-ensemble de métriques associées au volume a été retenu à partir de celles calculées par DATRIX. Ce sous-ensemble est référencé par DATR89-2. Ces métriques sont pour la plupart reliées au graphe de contrôle et leur sélection avait pour but de démontrer la dépendance entre celles-ci. Seuls les quatre projets en FORTRAN ont été considérés pour cette analyse. Le sous-ensemble des métriques est composé de:

{ e, K, Nbe, Nbt, Ncn, Nel, Nelmax, Nelw, Nl, Psc, Pscmax, V, Vg ,Vs }

Les résultats du projet DATR89_a (tableau 7.13) démontrent que l'ensemble des métriques représentées dans un espace orthonormal de 14 dimensions se projètent presque toutes avec des valeurs supérieures à 0.9 sur un axe imaginaire au centre de notre ensemble de métriques. Les métriques sont donc toutes linéairement dépendantes les unes aux autres. On remarque aussi que 86% de la variance de nos observations serait considérée si l'on remplaçait les 14 métriques originelles par la projection de l'observation sur l'axe de ce facteur.

Pour ce projet, une seule métrique suffirait à classer les observations. Cette métrique serait tout simplement définie comme VOLUME puisque toutes les valeurs des métriques augmentent dans le même sens que le nombre de lignes. Les résulats pour ce projet ne font que confirmer les études de corrélations faites par beaucoup de chercheurs [LI87] [LIND89].

PROJECTIONS		
Métriques	Facteur1	Var expliquée
E	0.978	0.956
K	0.855	0.730
NBE	0.897	0.805
NBT	0.890	0.792
NCN	0.983	0.967
NEL	0.940	0.883
NELMAX	0.952	0.906
NELW	0.937	0.877
NL	0.641	0.411
PSC	0.956	0.914
PSCMAX	0.966	0.933
V	0.966	0.933
VG	0.984	0.969
VS	0.974	0.948
VARIANCE EXPLIQUÉE/VARIANCE TOTALE		
	Facteur1	Total
Ratio	0.859	0.859

Tableau 7.13: Résultats pour le projet DATR89_a

Le deuxième projet DATR89_b est celui qui possède le plus de routines parmi les projets FORTRAN. Comme on le remarque souvent avec les analyses de régression, les procédures statistiques tendent à donner de moins bons résultats lorsque la taille de l'échantillon augmente (tableau 7.14). Le facteur 1 explique 72.2% de la variabilité totale, soit le plus faible résultat parmi les 4 projets. Le résultat n'en demeure pas moins étonnant. Un seul facteur est nécessaire ici aussi et il est identifié comme le VOLUME.

En plus, on peut interpréter le deuxième facteur comme un indice de TOPOLOGIE. A son extrémité positive se projètent les métriques du nombre de lignes et du nombre de lignes exécutables, tandis qu'à l'autre extrémité, on retrouve des métriques comme le nombre de croisement et les métriques du niveau d'imbrication.

Le troisième facteur est difficilement interprétable et ne sert qu'à expliquer la variabilité des métriques moins bien représentées par les deux premiers facteurs.

PROJECTIONS				
Métriques	Facteur1	Facteur2	Facteur3	Var expliquée
E	0.967	0.047	-0.218	0.985
K	0.677	-0.332	0.083	0.576
NBE	0.733	0.636	0.196	0.980
NBT	0.454	0.741	0.471	0.978
NCN	0.980	0.018	-0.132	0.978
NEL	0.850	-0.314	0.380	0.965
NELMAX	0.838	-0.289	0.393	0.940
NELW	0.849	-0.221	-0.057	0.773
NL	0.770	0.241	-0.242	0.709
PSC	0.778	-0.335	0.375	0.859
PSCMAX	0.951	0.007	-0.171	0.933
V	0.965	0.093	-0.220	0.988
VG	0.947	-0.030	-0.208	0.940
VS	0.968	0.065	-0.219	0.989
VARIANCE EXPLIQUÉE/VARIANCE TOTALE				
	Facteur1	Facteur2	Facteur3	Total
Ratio	0.722	0.106	0.072	0.900

Tableau 7.14: Résultats pour le projet DATR89_b

Les deux projets DATR89_c (tableau 7.15) et DATR89_d (tableau 7.16) ont le même comportement ne présentant que deux facteurs aux valeurs propres supérieures à 1. Le facteur principal explique respectivement 86.0% et 89.2% pour ces 2 projets.

PROJECTIONS			
Métriques	Facteur1	Facteur2	Var expliquée
E	0.962	-0.261	0.994
K	0.792	0.519	0.897
NBE	0.953	-0.243	0.967
NBT	0.941	-0.286	0.967
NCN	0.950	-0.296	0.991
NEL	0.893	0.427	0.979
NELMAX	0.918	0.370	0.979
NELW	0.905	0.413	0.990
NL	0.941	-0.117	0.899
PSC	0.861	0.460	0.953
PSCMAX	0.963	-0.021	0.928
V	0.952	-0.291	0.991
VG	0.976	-0.203	0.993
VS	0.959	-0.273	0.993
VARIANCE EXPLIQUÉE/VARIANCE TOTALE			
	Facteur1	Facteur2	Total
Ratio	0.860	0.106	0.966

Tableau 7.15: Résultats pour le projet DATR89_c

PROJECTIONS			
Métriques	Facteur1	Facteur2	Var expliquée
E	0.984	-0.043	0.969
K	0.905	0.343	0.937
NBE	0.969	-0.072	0.943
NBT	0.944	-0.024	0.891
NCN	0.947	-0.234	0.951
NEL	0.917	-0.220	0.889
NELMAX	0.889	-0.244	0.850
NELW	0.926	-0.290	0.941
NL	0.617	0.731	0.914
PSC	0.785	0.217	0.663
PSCMAX	0.882	0.296	0.865
V	0.982	0.070	0.969
VG	0.947	-0.234	0.951
VS	0.985	0.001	0.970
VARIANCE EXPLIQUÉE/VARIANCE TOTALE			
	Facteur1	Facteur2	Total
Ratio	0.829	0.079	0.908

Tableau 7.16: Résultats pour le projet DATR89_d

Les analyses factorielles classiques ont démontré pour ces projets que très peu d'information est véhiculée par les ensembles de métriques traditionnelles. De plus, seules les relations linéaires ont pu être déduites avec le modèle, ce qui peut sous entendre qu'encore moins d'information est véhiculé.

L'utilisation de l'analyse factorielle classique doit être considérée lors de l'établissement de nouvelles métriques. Si les résultats déduits ne peuvent être inférés, on pourra quand même vérifier l'apport d'information de la nouvelle métrique pour certains projets.

3. Modèle des résidus

Prenant un projet de 1000 routines pour lequel 20 métriques ont été calculées, l'approche traditionnelle nous conduit à observer 20 000 données. L'étude d'un tel ensemble de données devient une tâche considérable. Utilisant un modèle basé sur l'analyse factorielle classique, nous allons réduire le volume de cet ensemble de données.

Les expériences des analyses factorielles classiques effectuées à la section précédente nous ont donné un ratio de réduction de 5 pour le nombre de variables. Ce qui veut dire qu'en moyenne, le nombre de facteurs obtenus était 5 fois plus petit que le nombre de variables originelles. L'information que l'on retrouve dans l'espace des facteurs possède le désavantage d'être incomplète. Une partie de la variabilité s'est perdue en transférant l'information dans le nouvel espace. Il se peut que certaines observations soient mals représentées dans ce nouvel espace.

La transformation des observations vers l'espace factoriel s'effectue selon l'équation suivante.

TRANSFORMATION 1 PROJECTIONS = DATA * SCORE

Sachant que la matrice inverse de la transformation (PATTERN) nous permet de revenir dans l'espace initial.

TRANSFORMATION 2 DATA = PROJECTIONS * PATTERN' + ERROR

Nous allons observer la matrice ERROR pour chacune des observations. Cette matrice représente l'erreur présentée par le modèle des facteurs. Utilisant la mesure de distance euclidienne (somme des carrés), nous pouvons classer les observations en fonction de leurs modélisations par le modèle des facteurs. On détermine une distance limite acceptable et les observations ayant une distance plus grande sont considérées comme mal représentées par le modèle. Cette limite acceptable peut être une valeur fixe ou pondérée de distance ou tout simplement un pourcentage des observations.

Grâce à ce modèle, l'analyse d'un grand tableau de données devient plus facile. Dans un premier temps, une analyse factorielle classique est effectuée sur un ensemble de N observations et P variables. Cette analyse conduit à un modèle de facteurs où M observations sont mals représentées dans l'espace résultants de Q facteurs. Le practicien examine les projections pour les observations bien modèlisées et les variables originelles pour les autres observations. Le nombre d'éléments de chacune de ces 2 matrices est respectivement:

N * Q - M * Q éléments et

M * P éléments.

Le facteur de réduction R en terme d'éléments est donné par la formule suivante:

$$R = \frac{N * P - (N * Q - M * Q + M * P)}{N * P}$$

Les résultats obtenus avec cette méthode sont intéressants. La qualité des résultats dépend beaucoup du nombre de facteurs. Il est souvent essentiel d'augmenter le nombre de facteurs. On retiendra les facteurs ayant des valeurs propres supérieures à 0.5. Suivant cette méthode, on peut utiliser des rotations pour expliquer le comportement des facteurs en fonction des variables initiales.

L'analyse de gros projets nous a démontré que les routines mal représentées sont souvent celles qui ont un gros volume et une véritable complexité faible. Dans ces cas, environ 10% des observations sont considérées comme mal représentées. Par exemple les routines contenant la liste des messages à être imprimés, série d'initialisations...

La figure 7.2 nous démontre un exemple de la méthodologie à employer. Les deux matrices encadrées représentent les données à être examinées au lieu de la matrice originelle.

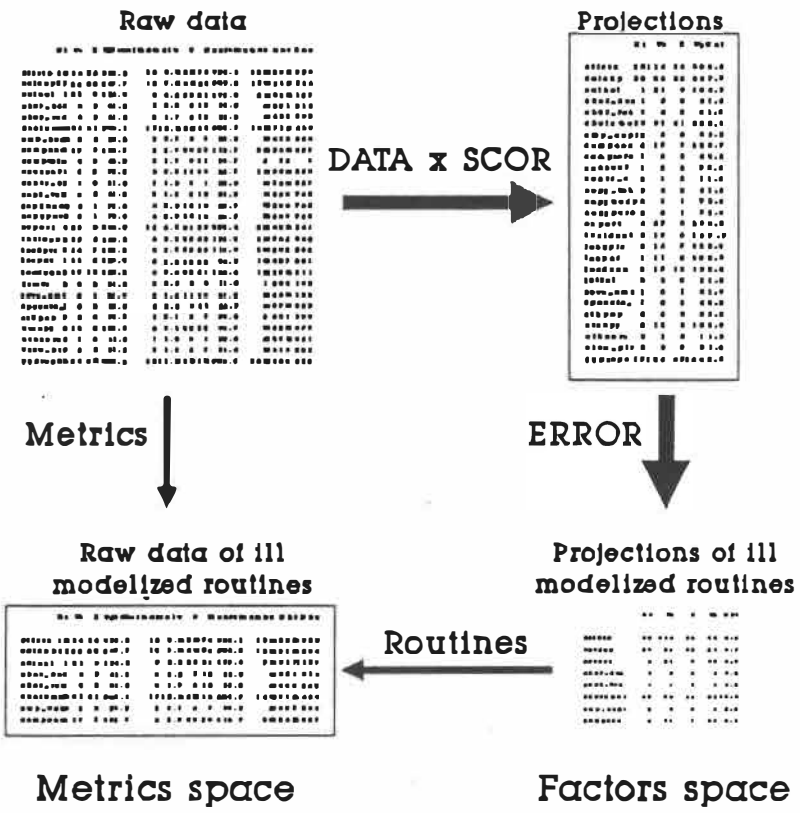

Figure 7.2: Méthode d'étude de grands tableaux à l'aide de l'analyse factorielle

L'exemple qui suit a été effectué à l'aide du projet DATR89_g (503 routines). L'ensemble des métriques retenues est { Vg, Vs, Nl, Nel, Nbe }. Ces métriques sont toutes tirées de l'ensemble DATR89-1.

Dans un premier temps, les observations sont normalisées à N(0,1). Ensuite une analyse factorielle classique est effectuée. Deux facteurs sont conservés. Ils représentent respectivement 70% et 17% de la variance. Le modèle obtenu par la TRANSFORMATION 1 explique donc 87% du modèle des données.

En appliquant la TRANSFORMATION 2, on obtient de nouveau les observations dans l'espace des données. La figure 7.3 montre la distribution des écarts (erreur) pour chaque observation. L'axe des Y représente l'erreur en valeur relative, tandis que l'axe des X projette chacune des 503 observations ordonnées en fonction de leur erreur.

On voit sur cette figure que les 10 premières observations ont une erreur très forte. On décidera ensuite d'éliminer les 20 ou 40 routines suivantes en fonction du degré de confiance que l'on veut avoir sur le modèle. L'évaluation de ces degré de confiance n'est pas explorée dans mon modèle et représente en elle-même un sujet complet.

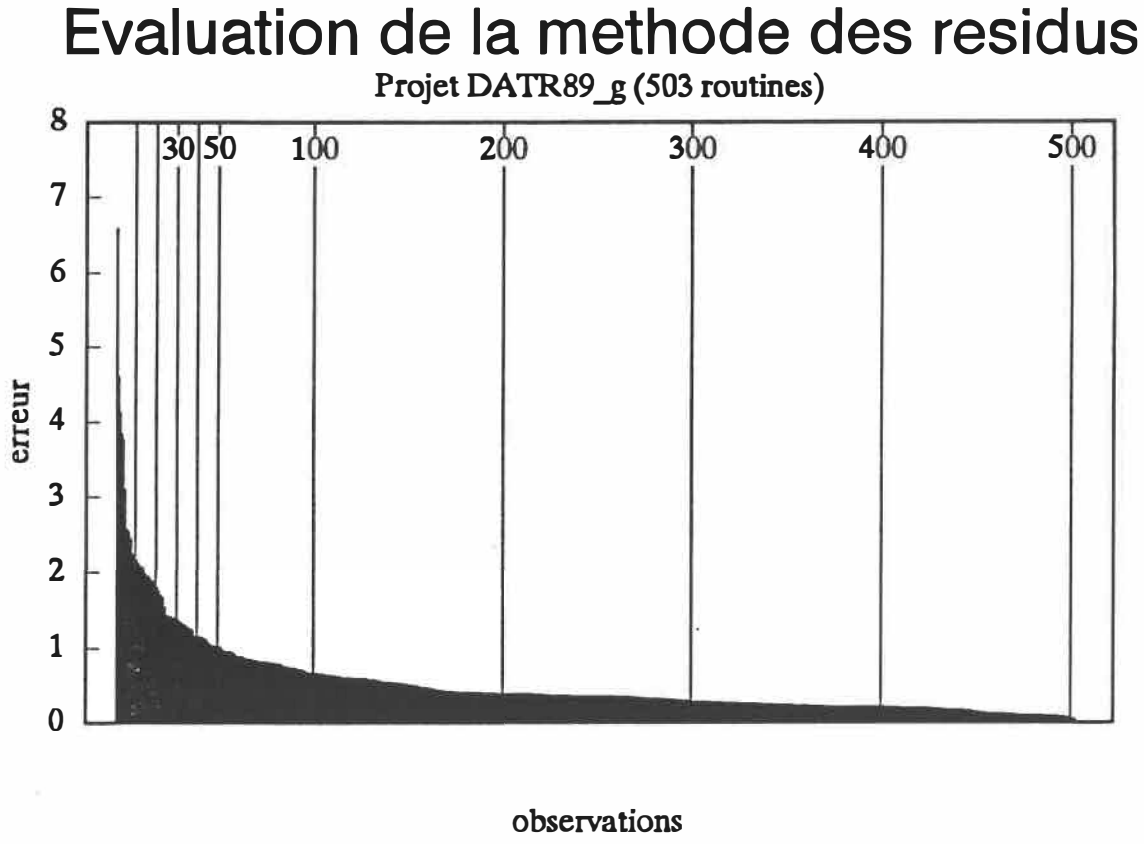

Figure 7.3: Évaluation de la méthode des résidus

L'analyse des 10 routines les plus mals modélisées nous permet de constater que 4 d'entre elles gèrent pratiquement l'ensemble des messages et des menus du logiciel, 3 autres représentent les algorithmes les plus différents du logiciel tandis que 2 routines sont du code produit automatiquement par des outils.

Il faut comprendre que l'identification de ces routines ne correspont en aucun temps aux routines jugées les plus complexes. Il est évident que la tâche de gestion des messages et des écrans est excessivement simple. La caractéristique de ces routines est qu'elles diffèrent de la façon dont les autres routines sont construites.

Obtenant la liste des routines mal modèlisées par l'analyse factorielle, nous évaluons la qualité de ces routines à l'aide des valeurs des métriques. Pour les autres routines, on pourra utiliser les projections dans l'espace factoriel.

4. Conclusions sur les analyses factorielles classiques

On retient de cette section que très peu de dimensions sont mesurées par l'ensemble des métriques actuelles. On obtient une moyenne de 4 dimensions mesurées par environ 20 métriques. Il y a encore beaucoup de place pour de nouvelles métriques. La description d'une nouvelle métrique doit être justifiée par l'apport d'information qu'elle amène. L'analyse factorielle classique est un excellent moyen de vérifier l'information amenée par une nouvelle variable.

L'interprétation des facteurs démontre que le volume est constamment identifié comme le principal axe. L'interprétation des autres facteurs est plus facile si l'on utilise des rotations. Le principal désavantage de ces rotations est de distribuer la variance des principaux axes sur les autres axes, c'est pourquoi elles ont été rejetées lors de la présentation des résultats.

Le modèle des résidus n'est pas approfondi complètement étant donné sa complexité. Il propose néanmoins une façon intéressante de corriger le modèle des analyses factorielles classiques.

F. Analyses discriminantes

Cette section vise à démontrer que l'hypothèse d'indépendance des données ne peut être utilisée si l'on considère un projet comme un échantillon de la population consistant en toutes les routines.

1. Paramètres du modèle

Tel qu'énoncé dans la section décrivant les analyses discriminantes, le modèle utilisé est non-paramétrique, utilisant la méthode des plus proches voisins. Les cinq plus proches voisins sont considérés pour le classement d'une observation. Pour la discrimination selon les auteurs, un modèle avec noyaux uniformes (kernel) est aussi utilisé.

Des sous-ensembles de DATR89-1 sont utilisés comme variables discriminatoires tandis que la variable à discriminer est soit le PROJET, le LANGAGE ou l'AUTEUR d'une routine. On note que la dernière variable n'est disponible que pour les échantillons DATR89_b et DATR89_e. Les chiffres dans les tableaux 7.18, 7.20, 7.21, 7.23 et 7.24 correspondent au pourcentage des observations bien classées par les modèles des analyses.

2. Discrimination sur les projets

Les analyses discriminantes sont utilisées pour reclasser les routines à l'intérieur d'un projet. Seuls les projets réalisés en FORTRAN sont considérés pour éliminer l'apport que peut jouer la variable langage dans l'analyse. Les routines sont distribuées dans les projets DATR89_a à DATR89_d (tableau 7.17).

Projet	Observations
DATR89_a	187
DATR89_b	718
DATR89_c	216
DATR89_d	161

Tableau 7.17: Nombre d'observations pour chaque classe de projets écrit en FORTRAN

Ensemble de variables	Projet				
	DATR89_a	DATR89_b	DATR89_c	DATR89_d	TOTAL
Nbt Vg	84%	67%	90%	72%	79%
Nbt Vg Nc V	87%	71%	86%	81%	81%
Nbt Vg Nc V Nl Nel Rlw	88%	71%	85%	86%	82%
Ensemble DATR89-1	96%	69%	96%	98%	90%

Tableau 7.18: Résultats de l'analyse discriminante pour la classification du langage considérant les différents dialectes

On peut remarquer (tableau 7.18) que seulement 2 variables réussissent à reclasser correctement 78% des observations. En utilisant toutes les variables, on augmente ce pourcentage à 90%. Une approche non-paramétrique utilisant des densités de points avec un noyau (r=0.5) a aussi permi de reclasser globalement 98% des observations sous le bon projet en utilisant toutes les variables.

On peut donc conclure que les métriques permettent d'identifier clairement à quel projet appartient une routine pour ces projets.

3. Discrimination sur les auteurs

Pour les projets DATR89_b et DATR89_e, le code source est disponible. Dans les sources, le nom du programmeur ayant réalisé la routine est inscrit. Cette information permet d'ajouter la variable AUTEUR à ces deux ensembles de données. Le tableau 7.19 montre le nombre d'observations considérées pour chacun des ensembles de données. Les observations dont on ne possède pas l'identification du programmeur sont rejetées.

Projet	Obs. total	Obs. auteurs	Programmeurs
DATR89_b	718	629	9
DATR89_e	255	251	2

Tableau 7.19: Description des ensembles de données pour la discrimination sur les auteurs

Une première analyse discriminante est effectuée sur un projet avec pour but de reclasser les routines selon le programmeur qui a écrit le code. On utilise un modèle linéaire où l'on considère les 5 voisins les plus proches. L'ensemble des variables DATR89-1 est utilisé. Les résultats présentés par le tableau 7.20 démontrent un facteur de discrimination considérable, mais faible si l'on considère que 32 variables sont utilisées pour départager seulement 2 classes.

Projet	Programmeurs		
	1	2	Total
DATR89_e	85%	75%	81%

Tableau 7.20: Résultats de l'analyse discriminante pour la classification des programmeurs avec un modèle non-paramétrique basé sur le nombre de voisins

Un autre modèle a été utilisé. Il s'agit d'une approche non-paramétrique avec noyau (r=0.5) . Le tableau 7.21 démontre les résultats globaux obtenus.

Projet	Classement
DATR89_b	94%
DATR89_e	98%

Tableau 7.21: Résultats de l'analyse discriminante pour la classification des programmeurs avec un modèle non-paramétrique basé sur les noyaux

Il faut conclure ici aussi que le programmeur est une variable sur laquelle on peut discriminer les routines. Les programmeurs laissent une signature dans les routines. Et cette signature peut être mesurée à l'aide des métriques.

4. Discrimination sur les langages

Les 7 ensembles de données DATR89_a à DATR89_g sont réalisés en FORTRAN ou en C. Le FORTRAN utilisé admet 2 dialectes. Le FORTRAN VAX est utilisé pour les projets a et c tandis que le FORTRAN ANSI est utilisé pour les projets b et d. Les projets e, f et g sont réalisés en ANSI_C. Le nombre d'observations pour chaque langage est rapporté par le tableau 7.22

Etant donné que le temps nécessaire à la réalisation de la procédure est fonction du carré du nombre de variables, il n'a pas été possible de toutes les utiliser pour la discrimination. Par exemple, la discrimination utilisant 9 variables a demandé plus de 20 heures d'analyse sur un micro-processeur 80386/20 MHz.

Language	Observations
ANSI_C	2870
VAX_FORTRAN	879
ANSI_FORTRAN	412

Tableau 7.22: Nombre d'observations pour chaque classe de language

Les tableaux des moyennes (tableau 7.1) et des écart-types (tableau 7.2) ont aidé à choisir les variables. Celles-ci ont été choisies de façon à maximiser le pouvoir discriminatoire. Le tableau 7.23 rapporte les résultats de la classification globale pour différents ensembles de métriques. Une méthode non-paramétrique avec 5 voisins est utilisée. Les chiffres dans les tableaux correspondent au pourcentage des observations bien classées par le modèle.

Ensemble de variables	Langage			
	ANSI_C	ANSI_FOR	VAX_FOR	TOTAL
Nbt Vg	83%	92%	75%	83%
Nbt Vg Nc V	89%	94%	74%	86%
Nbt Vg Nc V Nl Nel Rlw	96%	92%	77%	88%
Nbt Vg Nc V Nl Nel Rlw Psc Nl	95%	92%	80%	89%

Tableau 7.23: Résultats de l'analyse discriminante pour la classification du langage considérant les différents dialectes

Les mêmes classifications ont été reprises où l'on a fusionné les classes ANSI_FORTRAN et VAX_FORTRAN. Un modèle non-paramétrique basé sur les 5 voisins les plus proches est encore utilisé. Le tableau 7.24 rapporte les résultats.

Ensemble de variables	Langage		
	ANSI_C	FORTRAN	TOTAL
Nbt Vg	89%	89%	89%
Nbt Vg Nc V	96%	95%	95%
Nbt Vg Nc V Nl Nel Rlw	97%	98%	97%

Tableau 7.24: Résultats de l'analyse discriminante pour la classification du langage

Les tableaux 7.23 et 7.24 démontrent d'une façon claire qu'il existe une signature pour les langages. Cette différence est encore plus marquée lorsque l'on fusionne les différents dialectes d'un langage ensemble. Ce qui porte à croire que la discrimination entre les deux types de language est plus difficile.

Un point intéressant à soulever est que la discrimination s'effectue avec très peu de variables. On peut supposer que la discrimination serait presque parfaite si l'on utilisait l'ensemble des variables. Cette observation est essentielle, puisqu'il existe aussi un facteur discriminatif pour les différents projets. Comme une classe de langage est composée de plusieurs classes de projets, un fort facteur discriminatoire sur les projets ne nous permettrait de conclure quoi que ce soit pour les langages, à moins d'avoir plusieurs langages par projets.

5. Conclusions sur les analyses discriminantes

On conclut cette section sur les analyses discriminantes en rappelant les principales observations faites lors des différentes analyses. Il existe des facteurs discriminatoires distincts pour les classes de langages, de projets et de programmeurs. Ces facteurs discriminatoires peuvent avoir besoin de beaucoup de variables comme c'est le cas pour les programmeurs. Il advient donc que contrairement aux modèles proposés par la section sur les analyses factorielles classiques, un ensemble de métrique complet trouve son utilité pour la discrimination de routines selon certains groupes.

VIII. Expérience sur la perception de la complexité

Au chapitre VI, deux alternatives ont été présentées pour obtenir les données. La première consistait à cueillir des données lors d'une expérience controlée. Cette alternative a été utilisée dans le cadre du cours "génie du logiciel". Une expérience a été tentée à deux reprises; aux sessions d'automne 1988 et d'automne 1989. Cette expérience visait à mesurer la perception qu'on les individus face à la complexité du code source. Les buts sont de:

- trouver les points qui caractérisent la complexité de différentes routines dans les cas de consensus général ou de discordance dans les opinions;
- vérifier l'impact de la réalisation d'une phase particulière sur la perception de la complextité.

A. La démarche

Les étudiants réalisent une phase différente du cycle de vie sur trois projets distincts. Ensuite ils évaluent les codes sources des routines composant ces trois projets. Les trois projets sont réalisés en trois phases par des groupes différents. Si cette approche est innovatrice en elle-même, plusieurs autres ont été décrites par des professeurs dans la littérature. Ces approches se classent principalement dans deux catégories.

La première catégorie comprend les approches où l'on fournit un seul projet identique à tous les groupes. Ces approches traditionnelles sont rapportées par Carver [CARV84], Gilmore [GILM82] et Hayes [HAYE82].

La deuxième catégorie comprend les approches où plusieurs projets sont distribués aux différents groupes de la classe. Henry [HENR83] discute de l'approche où chaque équipe a son propre projet différent des autres, tandis que Beccue [BECC84] présente une description de cours dans laquelle chaque équipe développe une partie différente d'un système global.

Dans le cadre du cours de génie logiciel, chaque équipe complète une partie du travail à partir d'un résultat partiel d'une autre équipe. Le projet à réaliser est décomposé en quatre phases distinctes. Les étudiants reçoivent un énoncé du projet. Dans un premier temps, une analyse doit être réalisée. Suivent ensuite les phases de conception du logiciel, de tests et finalement d'analyse de complexité pendant laquelle la plus grande partie de la collecte des données s'effectue.

Trois projets différents sont constitués, ils sont référencés respectivement par les lettres A, B et C. Les trois projets sont distribués aléatoirement à une dizaine d'équipes. Chaque équipe composée de deux personnes reçoit un seul projet. Les équipes recevant initialement les projets A, B et C sont référencées respectivement par TYPE I, TYPE II et TYPE III.

Chaque équipe, qu'importe son type, effectue la phase de l'analyse sur le projet reçu, la phase de réalisation sur un autre projet et les tests sur un autre projet. Finalement l'équipe effectue une évaluation de la complexité sur les trois projets. Le tableau 8.1 démontre la procédure.

EQUIPE	PHASES			
	ANALYSE	CONCEP.	TESTS	EVALUATION
Type I	A	B	C	A, B et C
Type II	B	C	A	A, B et C
Type III	C	A	B	A, B et C

Tableau 8.1: Exemple de rotation pour les parties à réaliser

L'énoncé initial remis contient une brève description de la fonctionalité désirée, l'ensemble des concepts théoriques reliés au projet, quelques exemples ainsi qu'une bibliographie des articles qui peuvent être consultés. Les étudiants font l'analyse et remettent les spécifications complètes (prototypes d'écrans, algorithmes, structures de données, ...) du projet à réaliser.

La figure 8.1 montre l'échange d'information entre les différentes phases. Le document remis aux étudiants pour la phase de conception est la meilleure analyse effectuée lors de la phase précédente. On trouve la meilleure analyse pour chacun des trois projets. Les étudiants réalisent une version exécutable de l'un des logiciels spécifiés qu'ils remettent avec les documentations interne et externe du produit.

Le meilleur programme de chaque projet est fourni à la phase suivante pour que les étudiants puissent réaliser la phase des tests. La sélection s'effectue en fonction de la concordance entre ce qui a été décrit par les spécifications et ce qui a été réalisé.

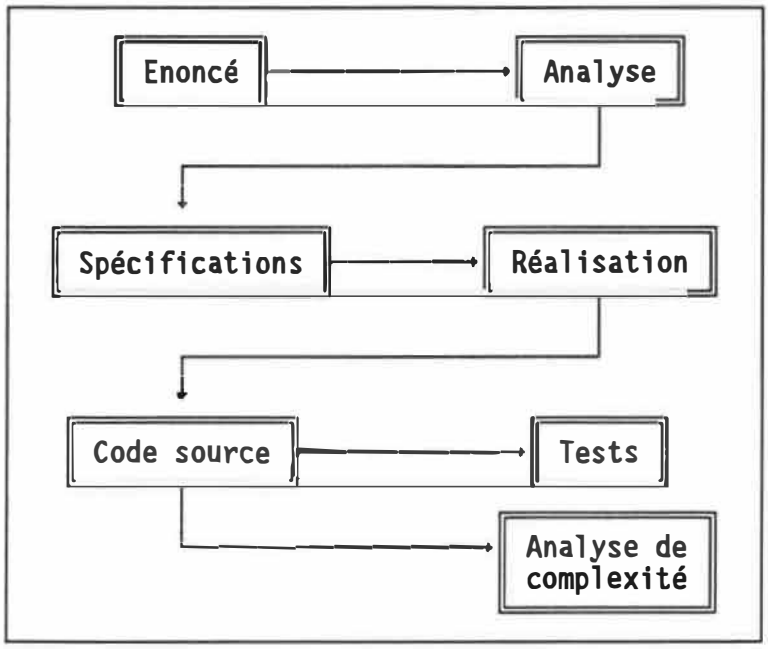

Figure 8.1: Échange d'information entre les différentes phases

La dernière phase du travail s'effectue sur l'ensemble des trois projets par chacune des équipes. Les trois projets sont les mêmes que ceux de l'étape précédente, c'est-à-dire les trois projets retenus lors de la phase de conception.

Pour une équipe qui aurait analysé le projet A, codé le projet B et testé le projet C, le code à observer pour cette dernière partie est le projet C car c'est celui que l'on a testé. Il diffère un peu en B si l'on n'est pas l'équipe qui a fourni le meilleur travail. Ces différences ne sont pas majeures car la fonctionalité et l'algorithme de chaque routine sont distribués dans les spécifications. Le seul code avec lequel les étudiants sont moins familiers est celui du projet analysé car les spécifications déduites peuvent être différentes en plusieurs points des leurs. La décomposition du projet ainsi que les structures de données qu'ils avaient choisies auraient pu conduire à un code source différent.

Dans cette dernière phase du projet, les étudiants ont l'occasion de mesurer la complexité des projets qu'ils ont eux-mêmes réalisés à l'aide d'une gamme de métriques. Le calcul des métriques est automatisé par le logiciel DATRIX. Les données de l'analyse par DATRIX de même que le code source des trois projets sont distribués aux étudiants.

L'étudiant

- classifie les projets en ordre de complexité selon son jugement intuitif;
- classifie les routines en ordre de complexité selon son jugement intuitif pour chaque projet;
- analyse les résultats des métriques;
- propose des critères d'évaluation de complexité;
- classifie les projets en ordre de complexité selon son système de mesure;
- classifie les routines en ordre de complexité selon son système de mesure pour chaque projet;

- discute sur l'évaluation de la qualité et la complexité du code source à l'aide des métrique.

La figure 8.2 montre un exemple de rotation pour les différentes équipes.

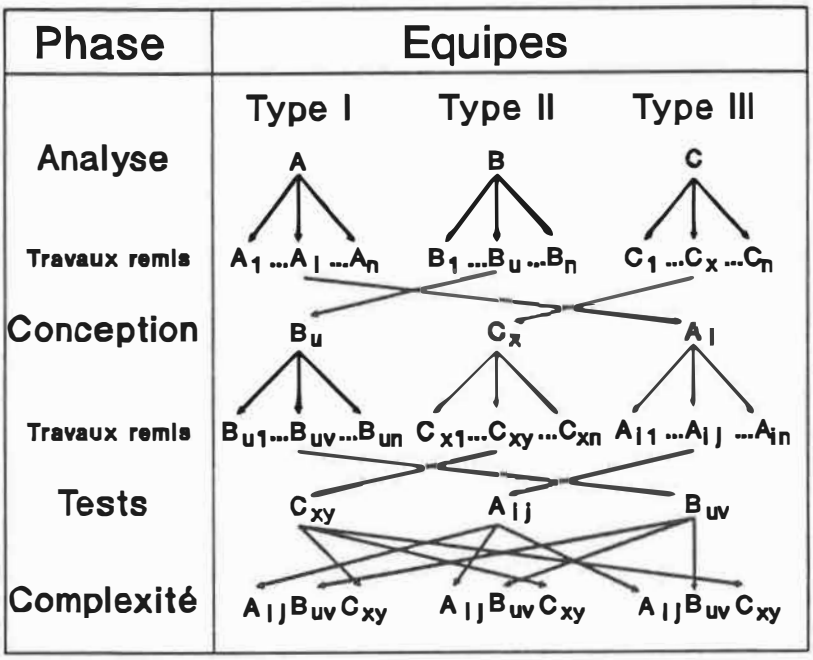

Figure 8.2: Exemple de sélection de travaux pour les différentes phases

B. Formation des individus

Dans le but de déterminer l'expérience des individus de la classe, un petit questionnaire leur est distribué. Ce sondage recueille le niveau de connaissance de chacun des langages qu'ils connaissent. On énumère une liste de langages pour lesquels on demande de compléter les informations suivantes:

- la maîtrise de ce langage de programmation sur une échelle graduée de 0 à 3 correspondant à une connaissance nulle jusqu'à une connaissance excellente;
- le nombre de lignes du plus volumineux programme écrit à l'aide de ce langage;
- les cours ou projets dans lesquels ils ont utilisé ce langage.

Au moment du sondage, 48 étudiants étaient présents pour la session d'automne 1988 tandis que 38 y étaient pour la session d'automne 1989. Une compilation des données de ce sondage nous a donné les résultats reportés dans le tableau 8.2. Un indice de maîtrise global du langage est fourni à la fin du tableau. Cet indice varie de 0 pour une connaissance nulle à la valeur 3 pour une maîtrise excellente du langage.

Année	Langage de prog.	maîtrise				
		excellente	passable	faible	nulle	indice
1988	BASIC	18	19	8	3	2.1
	C	41	6	0	1	2.8
	DBase	10	4	9	25	1.0
	FORTRAN	26	19	3	0	2.5
	Pascal	34	9	3	2	2.6
1989	BASIC	11	19	8	0	2.1
	C	34	3	1	0	2.9
	DBase	8	4	12	14	1.2
	FORTRAN	5	13	18	2	1.6
	Pascal	23	11	4	0	2.5

Tableau 8.2: maîtrise des langages de programmation par les étudiants

Le tableau 8.2 nous permet de conclure que les étudiants des 2 dernières années possèdent un niveau d'expérience équivalent pour l'utilisation des langages évolués de programmation, à l'exception du FORTRAN.

Le cours de génie du logiciel est obligatoire aux étudiants de quatrième année en génie informatique. Plus de 90% de l'ensemble des étudiants inscrits au cours proviennent donc de cette spécialité. Ceux-ci possèdent déjà une base acquise dans les cours précédents:

- informatique de base avec Pascal;
- structures de données avec le langage C;
- systèmes d'exploitation (2 cours);
- architecture d'ordinateurs;
- conception assistée par ordinateur;
- assembleur;
- bases de données;
- etc.

C. Études expérimentales

1. Description des résultats

Les sections 2 et 3 qui suivent présentent respectivement les expériences réalisées au cours des sessions d'automne 1988 et 1989. Pour chaque expérience, nous décrivons les projets sur lesquels les étudiants ont travaillé en plus de rapporter les résultats obtenus par l'évaluation de la complexité (quatrième travail pratique). Une section suit pour discuter de ces résultats.

Dans la section où l'on rapporte les résultats pour une expérience, 4 tableaux sont présentés. Le premier tableau rapporte les résultats des classements des projets par ordre de complexité par les équipes. Les valeurs présentées sont les moyennes des rangs de complexité associées aux projets ou aux routines. Le rang 1 correspond au projet jugé le plus complexe. Plus le nombre est grand, moins le projet est jugé complexe vis-à-vis les autres projets.

Le deuxième tableau contient les résultats d'une analyse factorielle effectuée sur chacun des trois projets. La méthode des composantes principales est utilisée et de plus aucune rotation n'est effectuée.

Le troisième tableau liste les classements des différentes routines par chacune des équipes. Le tableau, désigné par le terme "classement...", rapporte des moyennes des rangs attribués par les étudiants. Ce tableau est ordonné par ordre alphabétique des routines. Les noms des routines sont substitués par la lettre du projet suivit d'un nombre. Le nom de la routine n'est pas pertinent aux résultats.

Le dernier tableau, désigné par le terme "rang...", n'apporte pas d'informations supplémentaires au tableau précédent. Son but est de faciliter la tâche au lecteur pour les recherches d'informations. Il est classé par ordre décroissant de la projection de la routine sur l'axe principal de l'analyse factorielle. Les valeurs rapportées sont les rangs des valeurs du tableau précédent.

2. Expérience réalisée à la session d'automne 1988

Les trois projets sont choisis dans le but de donner une charge de travail équivalente aux différentes équipes. Deux des trois projets ont déjà été réalisés par la personne qui supervisait les laboratoires tandis que l'autre projet a été réalisé individuellement par 2 autres personnes du même niveau. Les trois projets sont jugés équivalents en charge de travail.

a. Projet A88 - Détermination du nombre de chemins dans un graphe

Le projet A88 a pour but de calculer le nombre de chemins indépendants dans un graphe de contrôle de son point d'entrée à son point de sortie. Cette métrique a été décrite par Schneidewind [SCHN79]. On la référence habituellement par l'abréviation de NP.

Pour calculer cette métrique, il s'agit de calculer tous les chemins avec la restriction que les retours de boucles sont empruntés aucune fois ou une seule fois.

Les étudiants sont informés que le graphe peut comporter de la non-structure et par contre une grammaire telle que définie par Nejmeh [NEJM88] devient insuffisante pour calculer la métrique.

Il existe plusieurs façon de résoudre le problème. La façon la plus évidente consiste à parcourir chacun des chemins à l'aide d'une fonction récursive. Cette méthode se révèle beaucoup trop lente dans le cas de graphes volumineux car la distribution de la métrique est exponentielle. Une façon plus performante considère l'ensemble des permutations possibles et calcule l'accessibilité d'un sommet comme la somme des accessibilités des sommets qui pointent sur le sommet en question. Finalement, il existe de nombreuses variantes où l'on se défini des règles tenant compte des patrons que l'on peut rencontrer.

b. Projet B88 - Dérivée symbolique d'une expression

Le projet B88 consiste à calculer la dérivée algébrique d'une expression mathématique. L'ordre des opérations à effectuer lors de la dérivée d'une expression étant fixe, il n'est pas nécessaire d'utiliser une base de connaissances pour réaliser ce problème. Dériver une expression consiste tout simplement à effectuer les opérations les plus prioritaires d'abord.

Les étudiants reçoivent quelques indices sur la façon de stocker l'expression et de réaliser des simplifications. La portée des opérateurs supportés et des simplifications à être effectuées sont laissés à la discrétion des équipes réalisant l'analyse.

c. Projet C88 - Calcul de l'interconnectivité

Le projet C88 est lui aussi relié aux métriques de complexité du logiciel. Il s'agit de calculer l'interconnectivité d'une matrice selon la méthode de Boloix et Robillard [BOLO88]. Cette métrique est basée sur le calcul de l'excès d'entropie. La procédure étant jugée trop complexe versus les autres projets, certaines simplifications sont faites et les étudiants doivent effectuer une partie bien définie de la procédure complète du calcul.

EQUIPE	PHASES			
	ANALYSE	CONCEP.	TESTS	EVALUATION
Type I	A	B	C	A, B et C
Type II	B	C	A	A, B et C
Type III	C	A	B	A, B et C

Tableau 8.3: Rotation des groupes pour la session d'automne 1988

Les métriques présentées aux étudiants sont les suivantes: Bs, Bsp, e, Nbe, Nbt, Ncn, Ne, Nel, Nelmax, Nelw, Ni, Nl, Np, Nr, v, Vg, Vp, Vs.

d. Résultats observés

Le tableau 8.4 démontre que le projet B est jugé le plus complexe. Pour les deux autres projets, seule l'équipe de type I qui a réalisé les tests sur C juge qu'ils sont de complexité égale. Les équipes de type II et III jugent C plus complexe que A.

Projet	Global	Equipe		
		Type I	Type II	Type III
A88	2.7	2.5 (analyse)	2.8 (tests)	3.0 (concep.)
B88	1.0	1.0 (concep.)	1.0 (analyse)	1.0 (tests)
C88	2.3	2.5 (tests)	2.3 (concep.)	2.0 (analyse)

Tableau 8.4: Classement des projets de 1988 de façon intuitive par les étudiants

Le tableau 8.5, qui rapporte les résultats d'une analyse factorielle classique, concorde avec les résultats trouvés précédemment. Les trois premiers facteurs expliquent de 90% à 97% la totalité de la variation des observations. Ici le facteur 1 peut être identifié comme le volume du graphe de contrôle. Pour les projets A88 et B88, on identifie le deuxième facteur au volume lexical, puisqu'il représente le nombre de lignes et d'énoncés.

Projet	Facteur	% var. exp.	Métriques dominantes
A89	1	67	Nl Vs Vg Np Nel Nelmax Nelw v e Ncn
	2	15	Nbe Nbt
	3	9	Nr
B89	1	70	Vs Vg Np Nel Nelmax Nelw v e Ncn
	2	13	Nbt Nbe
	3	7	Ne
C89	1	80	Vs Vg Np Nel v e Ncn Nbt Nbe
	2	17	Nelmax Nelw

Tableau 8.5: Extraction des facteurs principaux pour les métriques des projets de la session d'automne 1988

Les deux tableaux 8.6 et 8.7 démontrent le lien évident entre la complexité et le volume d'une routine. La correspondance pour les projets A et C est étonnante, seul le projet B diffère un peu. La correspondance est bonne sauf que le pouvoir discriminatoire entre 30 routines semble plus difficile. Miller [MILL56] a d'ailleurs énoncé que le pouvoir discriminatoire de l'être humain est limité à 7±2 éléments.

Routine	Proj. sur facteur principal	Global	Phase de l'équipe		
			Analyse	Concep.	Tests
A1	-0.49	6.1	6.3	6.6	5.3
A2	0.63	2.0	2.3	2.1	1.9
A3	0.46	2.8	2.3	2.6	3.4
A4	-1.12	8.2	8.3	8.6	7.6
A5	-0.57	7.2	7.3	7.1	7.3
A6	-0.16	6.8	6.7	6.3	7.7
A7	0.19	4.8	4.7	5.1	4.7
A8	-0.12	5.3	5.7	5.1	5.3
A9	2.29	1.7	1.7	1.6	1.9
B1	-0.27	15.9	14.3	15.8	17.2
B2	1.37	5.5	4.3	6.3	5.4
B3	3.44	1.7	1.0	2.3	1.4
B4	-0.86	25.5	24.8	25.1	26.6
B5	2.11	4.7	4.3	6.7	2.6
B6	1.32	5.5	5.0	5.3	6.0
B7	-0.63	16.9	18.5	14.7	18.2
B8	-0.60	22.8	23.5	20.9	24.6
B9	-0.70	28.0	29.8	26.1	29.0
B10	-0.16	18.7	17.8	18.3	20.0
B11	-0.42	21.9	23.0	22.1	20.8
B12	0.14	7.3	10.0	13.0	17.2
B13	-0.85	18.8	24.8	19.2	22.0
B14	0.75	22.8	6.3	7.5	7.8
B15	-0.48	3.7	19.0	19.8	17.4
B16	-0.69	18.7	21.0	24.1	22.7
B17	1.02	15.8	3.8	4.0	3.2
B18	-0.61	21.5	18.5	18.2	19.6
B19	-0.29	15.3	16.3	16.0	15.2
B20	-0.40	21.5	21.3	21.8	21.3
B21	-0.53	15.3	15.8	17.2	12.8
B22	-0.66	17.9	17.3	19.4	16.6
B23	-0.64	25.8	26.3	27.6	23.2
B24	0.64	7.7	8.6	6.5	8.4
B25	0.89	5.7	5.8	5.0	6.5
B26	0.77	6.5	6.8	6.0	6.9
B27	-0.14	10.9	13.3	10.5	9.4
B28	-0.63	17.1	15.3	18.0	17.6
B29	-0.86	26.4	27.8	25.6	26.2
B30	-0.64	21.0	27.8	22.1	19.2
C1	-0.31	4.9	4.9	5.1	4.5
C2	-0.88	5.1	5.4	4.8	5.2
C3	-0.26	3.3	2.6	3.5	3.8
C4	-0.24	2.3	2.9	2.1	1.8
C5	-0.29	3.4	3.0	3.5	3.8
C6	1.98	2.1	2.3	2.0	1.8

Tableau 8.6: Classement des routines des projets de la session d'automne 1988 de façon intuitive

Routine	Proj. sur facteur principal	Global	Phase de l'équipe		
			Analyse	Concep.	Tests
A9	1	1	1	1	1.5
A2	2	2	2.5	2	1.5
A3	3	3	2.5	3	3
A7	4	4	4	4.5	4
A8	5	5	5	4.5	5.5
A6	6	7	6.5	6	9
A1	7	6	6.5	7	5.5
A5	8	8	8	8	7
A4	9	9	9	9	8
B3	1	1	1	1	1
B5	2	3	3.5	8	2
B2	3	4.5	3.5	6	4
B6	4	4.5	5	4	5
B17	5	2	2	2	3
B25	6	6	6	3	6
B26	7	7	8	5	7
B14	8	8	7	9	8
B24	9	9	9	7	9
B12	10	11	10	11	14.5
B4	11	27	26.5	28	29
B29	11	29	29	29	28
B13	13	23	26.5	19	24
B9	14	30	30	25	30
B16	15	25	21	27	25
B22	16	17	16	20	13
B23	17	28	28	30	26
B30	18	21	23	25	19
B7	19	15	18.5	12	18
B28	19	16	13	16	17
B18	21	18.5	18.5	17	20
B8	22	26	25	22	27
B21	23	12	14	15	11
B15	24	20	20	21	16
B11	25	24	24	25	22
B20	26	22	22	23	23
B19	27	13	15	14	12
B1	28	14	12	13	14.5
B10	29	18.5	17	18	21
B27	30	10	11	10	10
C6	1	1	1	1	1.5
C4	2	2	3	2	1.5
C3	3	3	2	3.5	3.5
C5	4	4	4	3.5	3.5
C1	5	5	5	6	5
C2	6	6	6	5	6

Tableau 8.7: Rangs des routines des projets de la session d'automne 1988 de façon intuitive

3. Expérience réalisée à la session d'automne 1989

Lors de l'année 1988, le projet B s'est avéré moins complexe. Dans le but d'avoir une difficulté équivalente, les trois projets soumis à la session d'automne 1989 avaient déjà été réalisés par les personnes en charge des laboratoires avant la session. Pour être plus précis, ces trois projets étaient des simplifications de tâches réalisées dans un projet de plus grande envergure.

a. Projet A89 - Détermination du nombre de chemins dans un graphe

Le projet soumis à la dernière session ayant conduit à une solution triviale et non performante, nous avons décidé de le reprendre à cette session. L'énoncé demeure globalement le même. Des exemples supplémentaires sont ajoutés ainsi que des indices dans le but d'orienter les étudiants vers une solution plus performante et assurément différente de celle de l'année précédente.

b. Projet B89 - Réduction et concaténation d'un graphe de contrôle

Le graphe de contrôle tel qu'il est utilisé peut être obtenu par simplification et réduction formelles d'un graphe beaucoup plus volumineux. Ce dernier s'obtient par une correspondance biunivoque entre un code source dans un langage donné et des arcs auquels sont attribués des caractéristiques. Le projet à réaliser est de réduire et de concaténer ces graphes initiaux à l'aide de règles très précises qui sont fournies dans l'énoncé du projet.

c. Projet C89 - Graphe d'appels des routines d'un projet

On cherche à représenter les appels entre les différentes routines d'un projet. Le format d'un fichier contenant les relations appelantes-appelées est donné ainsi que celui d'un fichier qui contient la liste des routines qui appartiennent à la librairie du compilateur. Le but est de trouver une représentation permettant de visualiser toutes sortes de caractéristiques intéressantes dans ce genre de graphe.

EQUIPE	PHASES			
	ANALYSE	CONCEP.	TESTS	EVALUATION
Type I	A	C	B	A, B et C
Type II	B	A	C	A, B et C
Type III	C	B	A	A, B et C

Tableau 8.8: Rotation des groupes pour la session d'automne 1989

Les métriques présentées aux étudiants sont les suivantes: Bs, e, Ls, Nbe, Nbt, Ncn, Ne, Nel, Nelmax, Nelw, Ni, Nl, Np, Nr, Rac, Rcc, v, Vcd, Vcs, Vg, Vs

d. Résultats observés

On remarque par le tableau 8.9 que les types d'équipes ont tous choisi le projet qu'ils ont conçu comme le plus difficile. Le deuxième plus difficile est encore ici aussi unanimement le projet qu'ils ont analysé. Finalement le projet testé est jugé comme le plus facile.

Projet	Global	Equipe		
		Type I	Type II	Type III
A89	2.2	2.0 (analyse)	2.0 (concep.)	2.6 (tests)
B89	2.1	2.1 (tests)	2.5 (analyse)	1.7 (concep.)
C89	1.7	1.9 (concep.)	2.5 (tests)	1.7 (analyse)

Tableau 8.9: Classement des projets de 1989 de façon intuitive par les étudiants

Le tableau 8.10 de l'analyse factorielle classique donne les mêmes résultats sur les projets des sessions d'automne 1988 et d'automne 1989. Le premier facteur est identifié comme le volume du graphe de contrôle, tandis que le deuxième facteur est lié au volume lexical. On remarque l'apparition des métriques du volume de commentaires (Vcd et Vcs) et la proportion des noeuds commentés (Rnc) sur le deuxième facteur. Ces métriques ne faisait pas partie de celles distribuées aux étudiants de la session d'automne 1988.

Projet	Facteur	% var. exp.	Métriques dominantes
A88	1	58	Vs Vg Np Nel Nelmax Nlew v e Ncn
	2	20	Vcs Rnc Nbt Nbe
	3	13	Nr Ne
B88	1	50	Nl Vg Nel Nelmax Nelw v e Ncn
	2	28	Ne Nr Bs Nbt Nbe
C88	1	43	Nl Vs Vcs Vg Np Nel Nelmax Nelw v e Ncn
	2	20	Vcd Nbt Nbe
	3	14	Ne Bs

Tableau 8.10: Extraction des facteurs principaux pour les métriques des projets d'automne 1989

Les tableaux de classements 8.11 et de rangs 8.12 démontrent ici aussi la forte correspondance entre l'aspect volume et la complexité de la routine. Les rangs pour chacun des trois projets sont excellents. Lorsque quelques valeurs sont interchangées, on se réfère au tableau des classements 8.11 pour apercevoir les faibles différences entre les projections des routines sur l'axe principal.

Routine	Proj. sur facteur principal	Global	Phase de l'équipe		
			Analyse	Concep.	Tests
A1	-1.42	7.3	7.1	7.3	7.4
A2	-0.01	3.4	3.9	3.5	3.0
A3	0.56	2.7	3.9	1.5	2.3
A4	1.38	2.4	1.6	3.0	2.9
A5	0.18	5.6	4.6	6.0	6.3
A6	-0.63	4.77	5.0	5.3	4.0
A7	-1.11	6.7	7.2	7.0	6.0
A8	1.05	3.3	2.9	2.5	4.1
B1	1.30	2.4	1.5	2.9	2.4
B2	-0.90	8.3	9.0	7.7	8.4
B3	0.06	8.5	10.3	8.3	7.7
B4	-1.56	10.7	10.0	11.0	10.8
B5	0.28	5.1	6.1	4.3	5.4
B6	0.48	4.3	5.3	4.1	4.0
B7	-0.97	9.0	8.6	9.0	9.2
B8	1.68	3.1	3.6	3.4	2.4
B9	0.00	7.9	10.0	6.9	7.8
B10	0.85	4.1	3.0	5.3	3.5
B11	-1.13	9.3	8.1	9.7	9.6
B12	-0.09	5.4	2.5	5.4	6.9
C1	0.56	5.4	5.1	4.1	8.0
C2	-0.93	10.0	10.4	10.0	9.3
C3	1.54	3.8	2.9	2.5	7.8
C4	0.18	4.4	4.3	5.1	3.5
C5	-1.30	11.7	11.7	12.0	11.0
C6	0.20	4.8	5.3	5.6	2.3
C7	-0.18	6.5	6.6	7.9	3.9
C8	-0.39	5.4	6.3	5.3	4.3
C9	-0.29	7.1	8.4	7.4	4.3
C10	-1.54	9.6	8.6	9.7	11.0
C11	1.69	4.6	3.7	3.4	8.4
C12	0.48	4.8	4.7	4.9	4.5

Tableau 8.11: Classement des routines des projets d'automne 1989 de façon intuitive

Routine	Proj. sur facteur principal	Global	Phase de l'équipe		
			Analyse	Concep.	Tests
A4	1	1	1	3	2
A8	2	3	2	2	5
A3	3	2	3.5	1	1
A5	4	6	5	6	7
A2	5	4	3.5	4	3
A6	6	5	6	5	4
A7	7	7	8	7	6
A1	8	8	7	8	8
B8	1	2	4	2	1.5
B1	2	1	1	1	1.5
B10	3	3	3	5	3
B6	4	4	5	3	4
B5	5	5	6	4	5
B3	6	9	12	9	7
B9	7	7	10.5	7	8
B12	8	6	2	6	6
B2	9	8	9	8	9
B7	10	10	8	10	10
B11	11	11	7	11	11
B4	12	12	10.5	12	12
C11	1	3	2	2	9
C2	2	1	1	1	7
C1	3	6	5	3	8
C12	4	4.5	4	4	6
C6	5	4.5	6	7	1
C4	6	2	3	5	2
C7	7	8	8	9	3
C9	8	9	9	8	4.5
C8	9	7	7	6	4.5
C2	10	11	11	11	10
C5	11	12	12	12	11.5
C10	12	10	10	10	11.5

Tableau 8.12: Rangs des routines des projets d'automne 1989 de façon intuitive

D. Analyse des données des deux expériences

L'approche des projets multiples offre beaucoup d'avantages, par contre les avantages doivent être considérés. En particulier la difficulté d'identifier plusieurs projets réalisables dont la complexité est équivalente [CARV85]. L'expérience de 1988 en est la preuve. L'un des trois énoncés a conduit à une solution qui était beaucoup plus volumineuse que les solutions retenues pour les deux autres projets. Par chance, l'expérience de 1989 allait produire trois projets beaucoup plus égaux.

Lorsque l'on a demandé aux étudiants de classer les projets par ordre de complexité, on a remarqué la tendance à percevoir la complexité plus grande pour un projet que l'on a codé et moins grande pour un projet que l'on a testé. La complexité face à un projet semble se refléter selon l'ordre du tableau 8.13.

1 - Phase de conception
2 - Phase de l'analyse
3 - Phase des tests

Tableau 8.13: Ordre de la complexité selon la tâche réalisée

Evidemment pour l'expérience de 1988 (tableau 8.4), le projet B est beaucoup plus volumineux et unanimement les étudiants sont d'accord pour le classer comme le plus complexe. Le projet C est le suivant en ordre de complexité sauf pour les groupes qui ont fait les tests sur C. Ceux-ci ont conclus que les 2 projets étaient de complexité équivalente.

Pour l'année 1989 (tableau 8.9), le projet C est jugé le plus complexe sauf pour les gens ayant effectué les tests sur ce projet. Comme cette phase semble rendre un projet moins complexe aux yeux des programmeurs, ceux-ci jugent ce projet comme le plus simple. Pour les deux autres projets qui ont une côte globalement équivalente, chaque type d'équipe les classent selon l'ordre présenté par le tableau 8.13.

Nous concluons donc que la phase sur laquelle une équipe a travaillé a influencé l'évaluation de la complexité du code source. Cette observation provient probablement du fait que beaucoup ont tendance à négliger les phases d'analyse et de tests au détriment de la phase de conception.

Pour les routines, les tableaux des classements (8.6 et 8.11) démontrent la forte ressemblance des résultats obtenus pour chaque type d'équipe. Il ne semble donc pas y avoir d'influence marquée quant à la partie que les individus ont réalisé.

Les tableaux de rangs (8.7 et 8.12) démontrent clairement que la complexité intuitive des étudiants pour les routines est directement proportionnelle à la projection de la routine sur l'axe principal. Examinant les métriques qui se projètent sur l'axe principal (tableaux 8.5 et 8.10), on se rend compte que ce sont toutes des métriques de volume du graphe de contrôle. Les classements envers le premier facteur sont plus cohérents que pour n'importe quelle métrique prise individuellement. On en conclut donc que la perception première de la complexité est directement influencé par l'aspect volume du graphe de contrôle de la routine. De plus les étudiants ont su quantifier involontairement ce facteur en observant le code des routines.

IX. Conclusion

La recherche rapportée dans ce mémoire nous permet de rapporter des faits sur l'état actuel des métriques du logiciel. La littérature propose beaucoup de métriques de complexité. Les définitions des calculs sont souvent non-formelles, ce qui se traduit par la prolifération de métriques qui mesurent la même chose à quelque delta près. Par exemple, on peut souligner le nombre de lignes (LC, LE, LNE, LOC), le niveau d'imbrication (Band,Nelw) et le nombre de sommets conditionnels (Ncn, DE, CL).

La collecte des données aura démontré la difficulté d'obtenir des observations. De plus pour les observations, on remarque la quasi inexistance des données à modéliser que sont les coûts, le temps et les ressources nécessaires au développement des logiciels.

L'analyse statistique était le coeur de l'étude. On aura compris que les distributions des métriques ne s'appliquent pas aux modèles théoriques et que l'inférence statistique paramétrique ne s'appliquent pas. Encore plus difficile est le fait de reconnaître la dépendance des observations envers les langages de programmation, les projets eux-mêmes et aussi envers les programmeurs. Ce fait nous laisse démuni face à l'inférence statistique et nous réduit à la description des données.

Du coté descriptif, les projets recueillis nous permettent de construire une expertise sur les profils des données. L'analyse factorielle nous aura permis de démontrer que peu de dimensions sont mesurées par les ensembles de métriques traditionnels. Un modèle d'analyse de larges projets est tiré de la procédure statistique. L'analyse discriminante aura pour sa part contribué à démontrer la signature possible des individus et des environnements dans le code source des routines.

La définition des relations entre les données aura permis le développement d'un module d'exportation pour le logiciel DATRIX.

Un expérience sur la complexité perçue par les programmeurs réalisé à deux reprises aura démontré la forte relation avec l'aspect volume des routines. De plus on aura remarqué l'aspect psychologique que la phase du cycle de vie apporte lors de l'évaluation de la complexité.

Bibliographie

[AHO88] Alfred V. Aho, Brian W. Kernighan and Peter J. Weinberger, *The AWK Programming Language*, Addison-Wesley, 1988, 210 p.

[ANDE60] E. Anderson, "A semigraphical method for the analysis of complex problems," *Technometrics*, Vol. 2, pp. 387-391.

[ANDE78] T. W. Anderson and S. L. Sclove, *An Introduction to the Statistical Analysis of Data*, Houghton Mifflin, Boston, 1978.

[ANDE84] T. W. Anderson, *An Introduction to Multivariate Statistical Analysis of Data*, John Wiley & sons, New-York, 1984.

[ANDR72] D. F. Andrews, "Plots of high-dimensional data," *Biometrics*, Vol. 28, pp. 125-136.

[BAKE80] A. L. Baker and S. H. Zweben, "A Comparaison of Measures of Control Flow Complexity," *IEEE Transactions on Software Engineering*, November 1980, pp. 506-512.

[BECC84] Barbara Beccue, "Integration of Methodology and Tools: An Approach to Teaching Systems Development," *SIGCSE Bulletin*, Vol. 16, No. 1, Febrary, 1984, pp. 10-14.

[BEIZ84] Boris Beizer, *Software System Testing and Quality Assurance*, Van Nostrand Reinhold, New Jersey, 1984, 358 p.

[BELA80] B. L. A. Belady, "Software geometry," *Proceedings of Int. Computer Symp.*, Taipei, Republic of China, Dec. 1980.

[BERG70] Claude Berge, *Graphes et hypergraphes*, Dunod, Paris, 1970.

[BERG73] Claude Berge, *Graphs*, North-Holland, Amsterdam, 1973.

[BOEH78] Barry W. Boehm, John R. Brown, Hans Kaspar, Myron Lipow, Gordon J. MacLeod and Michael J. Merritt, *Characteristics of Software Quality*, North-Holland, Amsterdam, Netherlands, 1978.

[BOLO88] Germinal Boloix and P. N. Robillard, "Interconnectivity Metric for Software Complexity," *INFOR*, Vol. 26, No. 1, February 1988, pp. 17-39.

[BOWE78] John B. Bowen, "Are Current Approach Sufficient for Measuring Software Quality," *Software Engineering Notes*, November 1978, pp. 148-155.

[BOXC64] G. E. P. Box and D. R. Cox, "An analysis of transformations," *J. R. Stat. Soc.*, Vol 26, pp. 211-252.

[BRUC78] Lawrence A. Bruckner, "On Chernoff Faces," *Graphical Representation of Multivariate Data*, Academic Press, New-York, 1978, pp. 1-11.

[CARD87] D. N. Card and W. W. Agresti, "Resolving the Software Science Anomaly," *The Journal of Systems and Software*, Vol 7 No 1, March 1987, pp. 29-35.

[CARV84] Doris L. Carver, "Software Engineering for Undergraduates," *SIGCSE Bulletin*, Vol. 16, No. 3, September, 1984, pp. 23-25.

[CARV85] Doris L. Carver, "Comparaison of Techniques in Project-Based Courses," *SIGCSE Bulletin*, Vol. 17, No. 1, March, 1985, pp. 9-12.

[CHAM83] John M. Chambers, William S. Cleveland, Beat Kleiner and Paul A. Tukey, *Graphical Methods for Data Analysis*, Wadsworth International Group, Belmont, California, 1983, 395 p.

[CHER73] Herman Chernoff, "Using faces to represents points in K-dimensional space graphically," *J. Amer. Statist. Assoc.*, Vol. 68, pp. 361-368.

[CHER78] Herman Chernoff, "Graphical Representation as a Discipline.", *Graphical Representation of Multivariate Data*, Academic Press, New-York, 1978, pp 1-11.

[CLEM86] Bernard Clément, *Analyse statistique*, Ecole Polytechnique de Montréal, Montréal, 1986.

[CLEM88] Bernard Clément, *Analyse statistique multidimensionnelle*, Ecole Polytechnique de Montréal, Montréal, 1988.

[COAL87] François Coallier, *La caractérisation de la structure de programmes sources*, M. A. Sc. thesis, Université de Montréal, Ecole Polytechnique, 1987.

[COAL89] F. Coallier, P. N. Robillard, A. Beaucage, M. Simoneau and D. Coupal, "DATRIX: A Software Workmanship Evaluation Tool," *Proceedings of the 1989 IEEE Canadian Conference*, Montréal, 1989.

[CONT86] S. D. Conte, H. E. Dunsmore and V. Y. Shen, *Software Engineering Metrics and Models*, Benjamin/Cummings, California, 1986, 396 p.

[COOK82] Michael L. Cook, "Software Metrics: An Introduction and Annotated Bibliography," *ACM Software Engineering Notes*, April 1982, pp. 41-60.

[COTE88] V. Coté, P. Bourque, S. Oligny and N. Rivard, "Software Metrics: An Overview of Recent Results," *The Journal of Systems and Software*, March 1988, pp. 121-131.

[COUP87] Daniel Coupal, *POLEMICS: Projet d'un Outil Logiciel pour l'Évaluation de la Métrique d'Interconnectivité du Code Source*, Rapport de PFE, Université de Montréal, École Polytechnique, 1987.

[COUP90a] Daniel Coupal et Pierre N. Robillard, "Premilinary Results of Factor Analysis of Source Code Metrics," *Proceeding of the 2th Annual Oregon Workshop on Software Metrics*, Portland, Oregon, March 19-20, 1990.

[COUP90b] Daniel Coupal et Pierre N. Robillard, "Factor Analysis of Source Code Metrics," A être publié dans le Journal of Systems and Software, juillet 1990.

[CURT79] B. Curtis, S. B. Sheppard, P. M. Milliman, M. A. Borst and T. Love, "Measuring the Psychological Complexity of Software Maintenance Tasks with the Halstead and McCabe Metrics," *IEEE Transactions on Software Engineering*, March 1979, pp. 96-104.

[DUNS84] H. E. Dunsmore, "Software Metrics: An Overview of an Evolving Methodology," *Information Processing and Management*, Vol 20 no 1 & 2, 1984, pp. 183-192.

[DUTO86] S. H. C. du Toit, A. G. W. Steyn and R. H. Stumpf, *Graphical Exploratory Data Analysis*, Springer-Verlag, New-York, 1986, 314 pages.

[ELSH84] J. L. Elshoff, "Characteristic Program Complexity Measures," *Proceedings of the International Conference on Software Engineering*, 1984, pp. 288-293.

[EVAN83] W. M. Evangelist, "Software Complexity Metric Sensitivity to Program Structuring Rules," *The Journal of Systems and Software*, 1983 No 3, pp. 231-243.

[EVER78] B. S. Everitt, *Graphical Techniques for Multivarialte Data*, Heinemann Educational Books, London, 1978, 117 pages.

[FLUR81] B. Flury and H. Riedwyl, "Graphical representation of multivariate data by means of assymetrical faces," *J. Amer. Statist. Assoc.*, Vol. 76, pp. 757-765.

[GILB77] T. Gilb, *Software Metrics*, Winthrop, Cambridge, Mass., 1977.

[GILM82] Pat Gilmore, Kendall Griggs and Arless Eleirts, "DP Projects Provide Bridge Between Classroom and Job," *Interface*, Fall, 1982, pp. 68-70.

[HALS77] M. H. Halstead, *Elements of Software Science*, North-Holland, New-York, 1977.

[HANS78] Wilfred J. Hansen, "Measurement of Program Complexity by the Pair," *SIGPLAN Notices*, Vol. 13, No. 3, March 1978, pp. 29-33.

[HARR81] Warren A. Harrison and Kenneth I. Magel, "A Complexity Measure Based on Testing Level," *SIGPLAN Notices*, March 1981, pp. 63-74.

[HARR87] Warren A. Harrison and Curtis Cook, "A Micro/Macro Measure of Software Complexity," *The Journal of Systems and Software*, 1987, pp. 213-219.

[HART75] J. A. Hartigan, "Printer graphics for clustering," *J. of Statist. Computation and Simulation,* Vol. 4, pp. 187-213.

[HAYE82] Helen Hayes, "Using Team Projects to Teach Modular Design," *Interface*, Winter, 1982-83, pp. 18-22.

[HENR81] Sallie Henry and Dennis Kafura, "Software Structure Metrics Based on Information Flow," *IEEE Transactions on Software Engineering*, September 1981, pp. 510-518.

[HENR83] Sallie Henry, "A Project Oriented Course on Sotfware Engineering", *SIGCSE Bulletin*, Vol. 15, No. 1, February, 1983, pp. 57-61.

[HENR84] Sallie Henry and Dennis Kafura, "The Evaluating of Software Systems' Structure Using Quantitative Software Metrics," *Software-Practice and Experience*, Vol. 14, No. 6, June 1984, pp. 561-573.

[HUFF78] David L. Huff and William Black, "A Multivariate Graphic Display for Regional Analysis", *Graphical Representation of Multivariate Data*, Academic Press, New-York, 1978, pp. 1-11.

[JACO78] Robert J. K. Jacob, "Facial Representation of Multivariate Data", *Graphical Representation of Multivariate Data*, Academic Press, New-York, 1978.

[JENS82] H. Jensen, *An investigation of software metrics for real time software*, M. A. Thesis, Dep. Elec. Eng. and Computer Sci., Univ. of Wisconsin, Milwaukee, Aug. 1982.

[JENS85] H. A. Jensen and K. Vairavan, "An Experimental Study of Software Metrics for Real-Time Software," *IEEE Transactions on Sortware Engineering*, Vol SE-11 No. 2, February 1985, pp. 231-234.

[KOKO88] P. Kokol, B. Ivanek, and V. Zumer, "Software Effort Metrics: How to Join Them," *ACM Software Engineering Notes*, April 1988, pp. 55-57.

[LAUR82] Timo Laurmaa and Markuu Syrjanen, "APL and Halstead's Theory: A Measuring Tool and some Experiments," *Performance Evaluation Review (ACM Sigmetrics)*, Vol. 11, No. 3, Fall 1982, pp. 32-47.

[LI87] H. F. Li and W. K. Cheung, "An Empirical Study of Software Metrics," *IEEE Transactions on Software Engineering*, June 1987, pp. 697-708.

[LIND89] Randy K. Lind and K. Vairavan, "An Experimental Investigation of Software Metrics and Their Relationship to Software Development Effort," *IEEE Transactions on Software Engineering*, Vol SE-15 No 5, May 1989, pp. 649-653.

[MAGE81] Kenneth Magel, "Regular Expressions in a Program Complexity Metric," *ACM Sigplan Notices*, July 1981, pp. 61-65.

[MART83] James Martin and Carma McClure, *Software Maintenance: The problem and its solutions*, Prentice-Hall, New-Jersey, 1983, 465 p.

[MCCA76] Thomas J. McCabe, "A Complexity Measure," *IEEE Transactions on Software Engineering*, December 1976, pp. 308-320.

[MCCL76] Carma McClure, *Reducing COBOL Complexity through Structured Programming*, Van Nostrand Reinhold, 1976.

[MERT84] Robert K. Merton, David L. Sills and Stephen M. Stigler, "The Kelvin dictum and social science: an excursion into the history of an idea," *Journal of the history of the Behavioral sciences*, Vol 20, pp. 319-331.

[MEZZ78] Juan E. Mezzich and David R. L. Worthington, "A Comparaison of Graphical Representations of Multidimensional Psychiatric Diagnostic Data", *Graphical Representation of Multivariate Data*, Academic Press, New-York, 1978.

[MILL56] Georges A. Miller, "The magical number seven, plus or minus two: some limits on our capacity for processing information", *The Psychological Review*, Vol. 63, No. 2, March 1956, pp. 81-98.

[MUNS89] John C. Munson and Taghi M. Khoshgoftaar, "The Dimensionality of Program Complexity," *Proceedings of the 11th International Conference on Software Engineering*, Pittsburgh, May 1989, pp. 245-253.

[MYER77] Glenford J. Myers, "An Extension to the Cyclomatic Measure of Program Complexity", *ACM SIGPLAN Notices*, Octber 1977, pp. 61-64.

[NEJM88] Brian A. Nejmeh, "NPATH: A Measure of Execution Path Complexity and its Applications," *Communications of the ACM*, February 1988, pp. 188-200.

[PCMA89] "Statistical Analysis: Advanced," *PC Magazine*, March 14, 1989, pp. 121-200.

[PELL66] Roger Pellet, *Initiation à la théorie des graphes: vocabulaire descriptif*, Entreprise moderne d'édition, Paris, France, 1966.

[PERL81] Alan Perlis, Frederick Sayward and Mary Shaw, *Software Metrics: An Analysis and Evaluation*, MIT Press, Cambridge, Massachussetts, 1981.

[RAMA84] K. V. S. Rama Rao and S. Iyengar, "A General Measure for Program Complexity," *Software Engineering Workshop*, Nice, France, June 1984.

[ROBI85] P. N. Robillard, *LE LOGICIEL: de sa conception à sa maintenance*, Gaëtan Morin éditeur, Chicoutimi, Québec, 1985.

[ROBI85b] Pierre-N. Robillard, Software Engineering Education in Academia and Industry: Synthesis of Some Practical Experience, *4th World Conference on Computers in Education*, July 1985, North Holland, pp. 753-756.

[ROBI88a] P. N. Robillard and Daniel Coupal, "Data on the Automation of Program's Comments," *6th Annual Pacific Northwest Software Quality Conference*, Portland, Oregon, September 19-20, 1988, pp. 405-417.

[ROBI88b] Pierre-N. Robillard and Daniel Leblanc, "The Simulated Working Environment in a Project-Based Software Engineering Course," *Computers and Education*, Vol. 12, No. 4, 1988, pp. 471-477.

[ROBI89] P. N. Robillard and G. Boloix, "The Interconnectivity Metrics: A New Metric Showing How a Program is Organized," *The Journal of Systems and Software*, 1989.

[ROUV88] Christian Rouve, Michel Viala and Jacques Meekel, "Logiscope: A Maintenance Tool," *Proceedings of Software Engineering & its Applications*, Toulouse, France, December 1988, pp. 1013-1020.

[SASP85] *SAS Procedures Guide*, SAS Institute Inc., Cary, NC, 1985, pp. 343-358.

[SASS87] *SAS/STAT Guide for Personal Computers*, Sas Institute Inc., Cary, NC, 1987, pp. 449-492.

[SCHN79] N. F. Schneidewind, "Software Metrics for Aiding Program Development and Debugging," *AFIPS Conference Proceeding*, Vol 48, 1979, pp. 989-994.

[SCHN82] Karl Ernst Schnurer, "Product Assurance Program Analyzer (P.A.P.A.). A Tool for Program Complexity Evaluation.", *Performance Evaluation Review*, Vol. 11, No. 3, 1982, pp. 73-74.

[SCHR84] A. Schroeder, "Integrated Program Measurement and Documentation," *Proceedings of the International Conference on Software Engineering*, 1984, pp. 304-313.

[SEBE84] G. A. F. Seber, *Multivariate Observations*, John Wiley & Sons, New-York, 1984, 687 p.

[SHAP65] S. S. Shapiro and M. B. Wilk, "An analysis-of-variance test for normality," *Biometrika*, Vol. 52, pp. 591-611.

[SHAW81] M. Shaw, "Annotated Bibliography on Software Metrics," *Software Metrics: An Analysis and Evaluation*, MIT Press, Cambridge, Massachussets, 1981.

[SHEP88] Martin Sheppard, "A Critique of Cyclomatic Complexity," *Software Engineering Journal*, March 1988, pp. 30-36.

[SIMO89] Mario Simoneau, André Beaucage et Pierre N. Robillard, *Définition d'un modèle pour conserver l'information des programmes sources (modèle DATGRAPH) pour le logiciel DATRIX*, Ecole Polytechnique, Montréal, 1989, 61 p.

[STIG86] Stephen M. Stigler, *The History of Statistics*, Belknap Press, Cambridge, Massachusetts, 1986.

[SULL75] J. E. Sullivan, *Measuring the complexity of Computer Software*, NTIS AD-A007 770, 1975.

[SUNO81] T. Sunohara, A. Takano, K. Uehara and T. Ohkawa, "Program Complexity Measure For Software Development Management," *Proceedings of the 5th International Conference on Software Engineering*, San Diego, California, March 9-12, 1981, pp. 100-106.

[TIDM77] F. E. Tidmore and D. W. Turner, "Clustering with Chernoff-types faces," *Comm. Statist. - Theor. Meth.*, Vol. 12, pp. 397-408.

[TUKE57] J. W. Tukey, "On the comparative anatomy of tranformations," *Ann. Math. Stat.*, Vol. 28, pp. 602-632.

[VERI88] *Logiscope - Measurement Study*, Verilog, October 1988, pp. 1.4-1.13.

[VERI89] *Logiscope - Technical Presentation*, Verilog, January 1989.

[WAGU87] Leslie J. Waguespack Jr. and Sunil Badlani, "Software Complexity Assesment: An Introduction and Annotated Bibliography," *Software Engineering Notes (ACM Sigsoft)*, Vol 12 No 4, October 1987, pp. 52-71.

[WANG78] Peter C. C. Wang, *Graphical Representation of Multivariate Data*, Academic Press, New-York, 1978, 278 p.

[WOOD79] M. R. Woodward, M. A. Hennel and D. Hedley, "A measure of control flow complexity in program text," *IEEE Trans. Software Eng.*, Vol. SE-5, Jan. 1979, pp. 45-50.

[WOOD83] Scott Woodfield and James Collofello, "Some Insights and Experience in Teaching Team Project Courses," *SIGCSE Bulletin*, Vol. 15, No. 1, February 1983, pp. 62-65.

[ZUSE90] Horst Zuse, "Methods to Investigate and Evaluate Software Complexity Measures," *Proceeding of the 2th Annual Oregon Workshop on Software Metrics*, Portland, Oregon, March 19-20, 1990.