PolyPublie
Polytechnique Montréal

POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

Titre: Analyse du processus de mesure de la métrique des points de fonction
Title:

Auteurs: Alain Abran, & Pierre N. Robillard
Authors:

Date: 1992

Type: Rapport / Report

Référence: Abran, A., & Robillard, P. N. (1992). Analyse du processus de mesure de la métrique des points de fonction. (Rapport technique n° EPM-RT-92-01). https://publications.polymtl.ca/9540/
Citation:

Document en libre accès dans PolyPublie
Open Access document in PolyPublie

URL de PolyPublie: https://publications.polymtl.ca/9540/
PolyPublie URL:

Version: Version officielle de l'éditeur / Published version

Conditions d'utilisation: Tous droits réservés / All rights reserved
Terms of Use:

Document publié chez l'éditeur officiel
Document issued by the official publisher

Institution: École Polytechnique de Montréal

Numéro de rapport: EPM-RT-92-01
Report number:

URL officiel:
Official URL:

Mention légale:
Legal notice:

Ce fichier a été téléchargé à partir de PolyPublie, le dépôt institutionnel de Polytechnique Montréal
This file has been downloaded from PolyPublie, the institutional repository of Polytechnique Montréal

https://publications.polymtl.ca

07 FEV. 1992

EPM/RT - 92/01

ANALYSE DU PROCESSUS DE MESURE DE LA MÉTRIQUE DES POINTS DE FONCTION

gratuit

par

Alain ABRAN, M.Sc.A., Étudiant Ph.D.
Pierre N. ROBILLARD, Ph.D., Ing., Directeur de thèse

École Polytechnique de Montréal
Janvier 1992

Tous droits réservés. On ne peut reproduire ni diffuser aucune partie du présent ouvrage, sous quelque forme que ce soit, sans avoir obtenu au préalable l'autorisation écrite de l'auteur.

Dépôt légal, 1er trimestre 1992
Bibliothèque nationale du Québec
Bibliothèque nationale du Canada

Pour se procurer une copie de ce document, s'adresser:

Les Éditions de l'École Polytechnique
École Polytechnique de Montréal
C.P. 6079, Succursale "A"
Montréal, (Québec) H3C 3A7
Tél: (514)340-4473

Compter 0,10$ par page et ajouter 3,00$ pour la couverture, les frais de poste et la manutention. Régler en dollars canadiens par chèque ou mandat-poste au nom de l'École Polytechnique de Montréal.

Nous n'honorerons que les commandes accompagnées d'un paiement, sauf s'il y a eu entente préalable dans le cas d'établissements d'enseignement, de sociétés ou d'organismes canadiens.

SOMMAIRE

Depuis le début de l'informatique beaucoup de métriques ont été proposées pour mesurer la productivité du développement informatique. Toutefois aucune n'a obtenue une acceptation généralisée dans l'industrie à cause de la non transmutabilité des résultats d'une équipe de développement à l'autre, d'une technologie à l'autre et d'une industrie à l'autre. La métrique des Points de Fonction, introduite en 1979 par Al. Albrecht, a corrigé certaines de ces faiblesses, mais est encore loin d'être reconnue comme un standard à cause principalement de son manque actuel de fondement théorique validé démontrant clairement les relations entre les diverses composantes du domaine d'application de la métrique. En fait, la métrique des points de fonction n'a pas été encore modélisée de façon explicite et contient même un certain nombre de modèles implicites en lieu de relations scientifiquement arrimées à un systèmes de relations numériques.

La première partie de la revue de littérature aborde le thème des métriques en génie logiciel dans le contexte des modèles classiques de productivité et d'estimation. Cette étape permet de préciser les définitions qui sont utilisées tout au long des travaux de recherche, les contraintes et restrictions spécifiques aux modèles classiques de productivité et d'estimation de la taille des logiciels de même que des conditions d'homogénéité des expérimentations.

La structure de la métrique des points de fonction est ensuite décrite dans une perspective d'évolution historique, puis un modèle de la métrique est élaboré à partir de la perspective d'un processus de mesure. C'est d'ailleurs cette perspective qui permet de poser le diagnostic suivant: les points de fonction sont une métrique synthétique qui souffre des désavantages de toutes les métriques de ce type:

- Les métriques synthétiques sont difficiles à interpréter parce que les raisons pour lesquelles sont assignées des valeurs particulières (poids) ne sont pas précisées.

- La valeur en sortie de la formule est utile seulement si la formule est basée sur une théorie sous-jacente solide, telle l'économétrie ou la physique. Or ce n'est pas le cas des métriques de logiciel actuels.

- Il est nécessaire d'emmagasiner les valeurs de base des mesures qui constituent la métrique synthétique, pour garder le contrôle du processus de mesure.

La revue de littérature spécifique aux points de fonction est ensuite présentée à partir de cette perspective et identifie les quelques volets actuellement couverts par des recherches ainsi que certaines des forces et faiblesses de cette métrique: répétivité et fiabilité, modèles de

productivité et d'estimation. Cette revue de littérature illustre le peu de recherches fondamentales sur les points de fonction en tant que métrique, et également les cadres très restreints qui ont été analysés tel, par exemple celui de la validité du pointage final qui ne s'est pas penché sur la validité des étapes intermédiaires. Ceci laisse donc planer un doute sur les résultats finaux qui pourraient n'être qu'une somme d'approximations en non pas un véritable processus de mesure. Quelques métriques dérivées sont aussi examinées, telles MARK II, Feature Points et ASSERT-R. Il est souligné que quoique ces métriques dérivées soient intéressantes parce qu'elles s'adressent à des domaines d'application qui ne sont pas bien couverts actuellement par les points de fonction, ces nouvelles métriques dérivées ont tout autant de faiblesses au niveau méthodologique et font même appel à un plus grand nombre de modèles implicites non validés.

Enfin un sommaire des problématiques est établi afin de bien positionner le problème par rapport aux biais structurels des descriptions de la métrique, aux problèmes organisationnels d'implantation, aux structures des algorithmes de mesure et à la pertinence et aux domaines des modèles implicites. Ceci permet alors d'analyser correctement la structure elle-même des points de fonction en tant que processus de mesure. Il y est démontré que, de la seule perspective de l'utilisation des échelles de mesure à l'intérieur des étapes de la métrique, les résultats finaux (points de fonction non ajustée ou ajustés) sont très difficiles d'interprétation. Il y a en effet tellement de dimensions impliquées et tellement d'utilisations des différents types d'échelle de mesure, que ce qui semble simple et intuitivement raisonnable est, en fait, un pot-pourri qui pourrait ne pas avoir une signification mathématique valide, à moins évidemment d'avoir démontré, théoriquement et expérimentalement, la validité de chacune des transformations.

Toutefois la revue de littérature a illustré que des relations empiriques avaient été établies entre la taille des applications, mesurée en points de fonction, et la variable des efforts de développement, et ce à l'intérieur de contextes de référence. De là, les hypothèses de recherche suivantes sont formulées:

Hypothèse 1:

Si la métrique est un système de mesure, et non pas seulement une recette pour construire un nombre sans dimension, alors chacune des étapes de la mesure, du point de départ au résultat final, a une signification et une contribution. Dans un tel cas, chacune des étapes ajoute de l'information: la transposition sémantique de l'étape transcende alors l'opération mathématique et rend l'étape valide par l'entremise de cette transformation.

Hypothèse 2:

Si chacune des étapes est valide, il n'est pas nécessaire de se fier uniquement aux résultats de la dernière étape et il est alors plausible que des résultats intermédiaires aient des relations équivalentes. Il se pourrait même que certaines soient plus significatives si, dans la métrique, les étapes subséquentes font perdre de l'information au lieu d'en ajouter.

Hypothèse 3:

Si les deux hypothèses ci-dessus sont vérifiées, il est alors possible d'utiliser les connaissances acquises de la structure de la métrique pour approfondir la notion de contexte de référence et en étudier des relations supplémentaires dans ce contexte. Dans ce cas, la métrique des points de fonction pourrait être utilisée non plus uniquement comme une métrique de taille, mais également comme une métrique de la structure fonctionnelle du logiciel.

Hypothèse 4:

Si la métrique des points de fonction est également une mesure de la structure fonctionnelle de l'application, cette information peut alors être utilisée pour améliorer le processus de modélisation pour les analyses de productivité et d'estimation.

Le document se termine par une proposition de designs exprimentaux en vue de valider ou d'infirmer ces hypothèses de recherche. Le premier design expérimental comprendra une analyse structurelle à cinq fonctions de la métrique et les premiers résultats expérimentaux seront présentés à partir d'une analyse d'une base de données expérimentales couvrant la période 1986 à 1988. Les résultats de modèles de régressions (linéaires, non linéaires et multiples) seront analysés. A partir de là, un nouveau de modèle de référence sera précisé afin de permettre les prochaines analyses selon chacun des niveaux structurels de la métrique.

Le deuxième design expérimental utilisera un nouvel ensemble de données de la période 1986 à 1990, et comprendra une étape d'homogénéisation des sous-ensembles de données utilisées pour les analyses structurelles à plusieurs niveaux. Ce design expérimental comprendra trois étapes: une analyse avec les données primaires, une analyse selon la moyenne de distribution avec pointage, puis une analyse selon la moyenne de distribution avant l'intervention du processus algorithmique de la métrique.

REMERCIEMENTS

L'auteur tient à remercier toutes les personnes qui de près et de loin ont su le motiver à entreprendre cette recherche et l'ont aidé de leurs conseils.

Plus particulièrement ces remerciements s'adressent à Monsieur Jean-Marie Saint-Pierre, pour ses critiques positives, à Monsieur Hong Nguyenkim pour la gestion quotidienne du programme de métriques, au Dr. Pierre N. Robillard pour les conseils de direction de la recherche.

Cette recherche a été réalisée dans le cadre d'une collaboration entre le Montréal Trust et la Direction de la recherche organisationnelle du Centre canadien sur l'informatisation du travail (CCRIT), du ministère des Communications du Canada. Cette collaboration a été rendue possible grâce au Programme national d'échange de personnel spécialisé. Les opinions émises dans ce rapport n'engagent que son auteur.

TABLE DES MATIERES

SOMMAIRE i

REMERCIEMENTS iv

TABLE DES MATIERES v

LISTE DES TABLEAUX ix

LISTE DES FIGURES xi

AVANT-PROPOS xii

Chapitre 1 : INTRODUCTION 1
1.1 Les définitions des points de fonction 1
1.2 Le processus de mesure et les modèles 2
1.3 Les métriques en génie logiciel 4
1.4 Organisation du document 5

Chapitre 2: MÉTRIQUES ET MODELES DE PRODUCTIVITÉ 7
2.1 Les définitions 7
2.1.1 Mesures et métriques. 7
2.1.2 Logiciel 8
2.1.3 Génie logiciel. 8
2.2 Caractéristiques 9
2.2.1 Métriques synthétiques 9
2.2.2 Propriétés des données de logiciel 10
2.3 Les modèles de productivité et d'estimation 10
2.3.1 Définition générale de la productivité 11
2.4.2 Classification des modèles 12
2.4.3 Les limites des modèles 14
2.4 Pertinence des comparaisons: environnement non-homogènes 15
2.4.1 Points de rupture et pathologies 15
2.4.2 Impact des contextes multi-environnements 17
2.5 Sommaire 18

Chapitre 3: DESCRIPTION ET ÉVOLUTION DE LA MÉTRIQUE 19
3.1 Objectifs initiaux 19
3.2 Évolution historique 20
3.2.1 Albrecht 1979 20
3.2.2 Albrecht 1983 21

3.2.3 GUIDE 1985 23
3.2.4 IFPUG 25
3.2.4.1 IFPUG 1986 25
3.2.4.2 IFPUG 1988 25
3.2.4.3 IFPUG 1990 26
3.3 Le concept de contexte de référence 28
3.4 Sommaire 30

Chapitre 4: LE MODELE DU PROCESSUS DE MESURE 32
4.1 Le modèle 32
4.2 Le processus de mesure des données 34
4.3 Le processus de mesure des transactions 36
4.4 Le processus de mesure de l'ajustement 38
4.5 Sommaire 40

Chapitre 5: RECHERCHES EXPÉRIMENTALES: FIABILITÉ DE LA MÉTRIQUE 41
5.1 Classification des recherches expérimentales 41
5.2 Analyse de la fiabilité de la mesure 42
5.2.1 Design expérimental 42
5.2.2 La répétitivité 44
5.2.3 L'exactitude 46
5.2.4 Niveaux intermédiaires de pointage 48
5.3 Sommaire sur les analyses de fiabilité 50

Chapitre 6: RECHERCHES EXPÉRIMENTALES: Productivité et estimation 51
6.1 Caractéristiques des recherches expérimentales 51
6.2 Problématique de l'homogénéité des données 53
6.3 Analyse des relations 55
6.3.1 Relations à deux facteurs 55
6.3.2 Économies d'échelle. 55
6.4 Analyse d'impact des environnements et facteurs de productivité 57
6.4.1 Applications interactives versus différées. 57
6.4.2 Développement versus Additions-Modifications 58
6.4.3 Utilisation d'une méthodologie de développement 58
6.4.4 Facteurs de niveaux de langages de programmation. 59
6.4.5 Impact des outils d'aide au développement 60
6.5 Analyse par regroupement de facteurs 60
6.5.1 Concept des caractéristiques dominantes 60
6.5.2 Analyse par composantes principales 61
6.6 Analyses d'estimation 62
6.6.1 Modèles publics 62
6.6.2 Analyses de sensibilité statistique 64
6.6.3 Modèles privés 64

6.7 Sommaire 66
6.7.1 Analyses de productivité 66
6.7.2 Analyses d'estimation 66

Chapitre 7: RECHERCHES EXPÉRIMENTALES - MÉTRIQUES DÉRIVÉES . . . 67
7.1 Positionnement 67
7.2 Le modèle de Symons 68
7.2.1 La structure 68
7.2.2 Complexité 69
7.2.3 Facteurs de productivité 70
7.3 Les domaines d'applications autres que M.I.S. 70
7.3.1 Jones 71
7.3.2 Reifer 72
7.4 Sommaire 74

Chapitre 8: RECHERCHES EXPÉRIMENTALES - PROCESSUS DE MESURE . 75
8.1 Design expérimental 75
8.2 Les facteurs de productivité. 76
8.3 Le processus de mesure de l'ajustement. 76
8.4 La structure du modèle 77
8.4.1 Une structure inspirée d'Halstead 77
8.4.2 L'algorithme d'Albrecht 80
8.4.3 L'identification des composantes primaires 83
8.5 Sommaire 85

Chapitre 9: IDENTIFICATION DES PROBLÉMATIQUES 86
9.1 Les biais structurels historiques 86
9.1.1 Biais environnementaux du contexte initial 86
9.1.2 Biais M.I.S. 87
9.1.3 Séparation des concepts: taille, complexité et productivité 87
9.2 Les problèmes organisationnels d'implantation 87
9.2.1 La maturité organisationnelle 88
9.2.2 Simplicité apparente et courbe d'apprentissage 88
9.2.3 Exigences de pré-requis: la documentation 89
9.3 Les structures des algorithmes de mesure 89
9.3.1 Les valeurs discontinues des intervalles 89
9.3.2 Les poids 90
9.3.3 L'additivité des fonctions 90
9.3.4 Pot-pourri d'échelles de mesure 90
9.4 La pertinence et les domaines des modèles implicites 91
9.4.1 L'identification des modèles implicites 91
9.4.2 Couverture incomplète des fonctions dans le domaine M.I.S. 92
9.4.3 L'identification des composantes primaires des fonctions 92
9.4.4 Les limites du domaine 93

9.5 Sommaire 94

Chapitre 10: LA PROBLÉMATIQUE DE LA RECHERCHE 95
10.1 Analyse des échelles de mesure 95
10.1.1 Hypothèse de base 95
10.1.2 Propriétés des échelles de mesure 95
10.2 Le processus de mesure des données 96
10.3 Le processus de mesure des transactions 100
10.4 Le processus des ajustements 101
10.5 Définition de la problématique de recherche 102
10.6 Proposition de la méthodologie de recherche 103
10.6.1 Site expérimental 103
10.6.2 Approche 103
10.6.3 Design expérimental 104

BIBLIOGRAPHIE 106

LISTE DES TABLEAUX

TABLEAU 1 Définitions des points de fonction dans la litérature 2

TABLEAU 2.1 Définitions - Mesure/Métrique 7
TABLEAU 2.2 Propriétés des données Gausssiennes et non-Gaussiennes. 10
TABLEAU 2.3 Critères pour les métriques de productivité - Blake 87 11
TABLEAU 2.4 Classification MERMAID des modèles algorithmiques 12
TABLEAU 2.5 Critères de performance statistique 13

TABLEAU 3.1 Définitions de base d'Albrecht 19
TABLEAU 3.2 Albrecht 79 - Poids 20
TABLEAU 3.3 Albrecht 79 - Facteurs 20
TABLEAU 3.4 Albrecht 79 - Exemple de calcul. 21
TABLEAU 3.5 Albrecht 83 - Poids 22
TABLEAU 3.6 Albrecht 83 - Facteurs 22
TABLEAU 3.7 Albrecht 83 - Niveaux de complexité - Fonction de type ENTRÉE. . . . 23
TABLEAU 3.8 GUIDE 85 - Types de composantes primaires. 23
TABLEAU 3.9 GUIDE 85 - Dimensions des tables de décision, par type de fonction . . 24
TABLEAU 3.10 GUIDE 85- Structure des tables de décision 24
TABLEAU 3.11 IFPUG 90 - OBJECTIFS 26
TABLEAU 3.12 Innovations du manuel IFPUG 1990 26
TABLEAU 3.13 Online Data Entry Characteristics: Degree of Influence 27
TABLEAU 3.14 Albrecht 79 - Description du contexte global 28
TABLEAU 3.15 Albrecht 79 - Critères de sélection de projets admissibles 28
TABLEAU 3.16 Définition d'un point de fonction. 29
TABLEAU 3.17 Signification des termes de la définition d'un point de fontion. 29
TABLEAU 3.18 Liste des versions officielles. 31

TABLEAU 5.1 Classification des recherches sur les points de fonction. 41
TABLEAU 5.2 Analyses de fiabilité - Design expérimental 42
TABLEAU 5.3 Caractéristiques des recherches expérimentales de fiabilité 43
TABLEAU 5.4 Sources de variations inter-pointeurs 44
TABLEAU 5.5 Variations inter-pointeurs - Kemerer 90 45
TABLEAU 5.6 Variations inter-pointeurs - Sommaire des recherches. 45
TABLEAU 5.7 Sources d'inexactitudes dans les pointages. 46
TABLEAU 5.8 Analyse critique des facteurs d'inexactitude de Low et al. 47

TABLEAU 6.1 Sommaire des variations de taille et de coûts. 51
TABLEAU 6.2 Analyses de productivité sur les points de fonction - Caractéristiques des ensembles de données expérimentales 52
TABLEAU 6.3 Les meilleurs corrélations de productivité - Emrick 88 53
TABLEAU 6.4 Analyses de productivité sur les points de fonction - Identification des relations (Coeff. de régression & corrélation) 56
TABLEAU 6.5 Caractéristiques des échantillons de projets Différés/Interactifs - Emrick 88 57
TABLEAU 6.6 Comparaison des échantillons Nouvelles Applications / Add.-Modif. - Emrick 88 58
TABLEAU 6.7 Caractéristiques dominantes - Johson 91. 60
TABLEAU 6.8 Comparaison des variables d'estimation 62
TABLEAU 6.9 Comparaison des modèles d'estimation - Kemerer 87. 63
TABLEAU 6.10 Comparaison des modèles d'estimation - Informétric 88. 65
TABLEAU 6.11 Comparaison des divers ensembles de données expérimentales 65

TABLEAU 7.1 Positionnement des métriques dérivées (Selon le modèle de Porter) 67
TABLEAU 7.2 Critères et poids de complexité - Jones 71
TABLEAU 7.3 Caractéristiques des domaines d'applications logicielles - Reifer 72
TABLEAU 7.4 Description des variables de l'équation - Reifer 72
TABLEAU 7.5 Liste des estimateurs - Reifer. 73

TABLEAU 8.1 Caractéristiques des recherches expérimentales - Processus de mesure 75
TABLEAU 8.2 Variables d'estimation explorées - Albrecht 83. 78
TABLEAU 8.3 Résultats des régressions - Albrecht 83 79
TABLEAU 8.4 Facteurs du modèle dérivé - Bock 81
TABLEAU 8.5 Tests statistiques - Bock 81
TABLEAU 8.6 Comparaison des poids (Bock versus Albrecht). 82

TABLEAU 10.1 Échelles de mesure - Transformation admissibles 96
TABLEAU 10.2 Rangs des fichiers logiques. 97
TABLEAU 10.3 Identificateurs des rangs. 98
TABLEAU 10.4 Identificateurs de rangs sur 2 axes. 98
TABLEAU 10.5 Matrice de complexité fonctionnelle. 99
TABLEAU 10.6 Poids des fichiers logiques. 100

LISTE DES FIGURES

FIGURE 4.1 Modèle de mesure des *points de fonction*. 33
FIGURE 4.2 Modèle de mesure des *données* 35
FIGURE 4.3 Modèle de mesure des *transactions*. 37
FIGURE 4.4 Modèle de mesure de l'*ajustement* 39

AVANT-PROPOS

"I often say that 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: it may be the beginning of knowledge, but you have scarcely, in your thoughts, advanced to the stage of ***science****, whatever the matter may be."*

Lord Kelvin, inventeur de l'échelle de valeur absolue pour la mesure des tempétatures.[1]

---

[1]Lord Kelvin, "Popular Lectures and Addresses", Volume 1, 1889.

Chapitre 1 : INTRODUCTION

Ce premier chapitre présente un survol de la problématique analysée au cours de cette recherche, à savoir la pertinence des points de fonction comme instruments de mesure dans le domaine des métriques en génie logiciel. Les trois thèmes suivants sont abordés:

1- Les interprétations des "points de fonction"
2- Le processus de mesure et les modèles
3- Les métriques en génie logiciel

Il y sera vu qu'il existe une confusion considérable sur la nature des points de fonction, et par conséquent sur ses propriétés. Toutefois, le problème n'est pas unique aux points de fonction: il est générique à tout le domaine des métriques en génie logiciel qui ne possède pas encore de bons modèles de ses processus et encore moins de systèmes de mesure approprié. En fait, le processus du développement de logiciel est encore loin d'être une science et en est encore, comparé aux autres domaines des sciences et d'ingénierie, à ses débuts. Une fois ce positionnement effectué, la démarche de l'analyse de la métrique des points de fonction est présentée en fin de chapitre.

1.1 Les définitions des points de fonction

La métrique des points de fonction a été introduite en 1979 et, bien qu'utilisée dans plus de 500 organisations, la signification de cette métrique est encore ambiguë. Les points de fonction sont en effet représentés tour à tour comme une métrique de la taille d'une application, d'unité de productivité, de complexité fonctionnelle, de fonctionnalité, de la valeur de l'application pour les usagers, comme nombre sans dimension, comme un indice et enfin comme une mesure multidimensionnelle. Le tableau 1 présente une douzaine de définitions de divers auteurs, regroupées selon les mêmes thèmes.

Tableau 1 : Définitions des points de fonction dans la littérature

Thème	Description/Définition & Auteurs
Taille	1- Une métrique de taille (Wrigley, 1988) 2- Une mesure "surrogate" de la taille (ESPRIT, MERMAID 1, 1989)
Productivité	3- Une mesure de la taille du produit en vue de définir une mesure de productivité (ESPRIT, MERMAID 1, 1989) 4- Une mesure de productivité définie comme la somme pondérée des unités fonctionnelles produites (Shen et al., 1986) 5- Une unité économique de production (Jones, Met. Life 1988)
Complexité	6- Une mesure de complexité des projets (Martin, 1991)
Fonctionnalité	7- La fonctionnalité ou l'"*utility*" d'un programme (Pressman, 1987) 8- Une mesure indirecte de la fonctionnalité créée par une application logicielle (Pressman, 1988)
Livrables - Usager	9- Une assignation de valeur numérique à des livrables usagers (Johnson, 1991)
Nombre sans dimension	10- Les résultats des calculs n'ont ni une signification directe ni interprétation ou analogie physique (Pfleeger, 1989) 11- Un nombre sans dimension. Un nombre abstrait et synthétique mais un indicateur de performance pratique et utile. Une métrique artificielle similaire au concept de la moyenne du Dow Jones industriel (Jones, 1986, Computerworld 1988)
Multi dimensions	12- Un paramètre pour quantifier le comportement global ("*overall behavior*") (Dunn, 1990)

En fait, la confusion vient de la définition pot-pourri initiale d'Albrecht lui-même qui englobe toutes les dimensions mentionnées au tableau 1:

Un nombre sans dimension qui est une mesure efficace relative de la valeur de la fonction livrée à l'usager ("*This gives a dimensionless number defined in function points which we have found to be an effective relative measure of function value delivered to our customer*") (Albrecht, 1979).

Qu'en est-il vraiment? Pour apporter des éléments de réponse à ces questions, ce chapitre commence par relever quelques éléments fondamentaux spécifiques aux systèmes de mesure afin d'identifier les faiblesses à cet égard des métriques en génie logiciel.

1.2 Le processus de mesure et les modèles

Les processus des mesures font partie intégrante de la démarche scientifique. Pour

comprendre les phénomènes il faut construire des modèles descriptifs des composantes de la discipline pour améliorer la compréhension (Basili et al., 1991):

- de la nature et des caractéristique des processus et des produits
- les variations entre celles-ci
- les forces et faiblesses de chaque caractéristique
- les mécanismes pour les prédire et les contrôler

Qu'est-ce qu'une métrique si c'est seulement un nombre (Dijkstra, 1990)? Avant de pouvoir mesurer, il faut avoir un concept clair d'un attribut et de l'ensemble des entités à mesurer qui possèdent cet attribut. En fait le concept dans chaque cas précède la mesure (Fenton, 1991).

Les mesures doivent être associées à un processus de modélisation. Une fois un modèle élaboré, il faut des mesures pour vérifier si le modèle performe tel que prévu. Dans le passé, les mesures en génie logiciel ont été trop orientées sur les métriques, et pas assez sur les modèles (Basili et al., 1991). En fait il s'est agit souvent de collecter beaucoup de données, sans avoir d'objectifs précis, sans modèle et sans contexte: en fait, des éléments de mesure sans suffisamment d'information contextuelle, sont peu utiles pour l'interprétation des données.

Le but devrait être un ensemble de mesures qui puissent être justifiées théoriquement, qui puissent être supportées empiriquement et qui puissent être utilisées avec confiance et par les programmeurs et par les chefs de projets: *The goal should be a set of measures that can be justified theoretically, that can be used with confidence by both programmers and project managers* (Schen et al., 1983).

Dans les autres sciences, des instruments de mesure ont été développés à partir des connaissances approfondies sur les diverses caractéristiques, connaissances acquises à travers les siècles. Une mesure n'est pas en soi un nombre, mais une assignation d'un nombre: il s'agit donc d'un arrimage ("*mapping*") entre des entités et la caractéristique sous observation. Il se pose alors plusieurs questions fondamentales dans la cadre des mesures (Fenton, 1991):

1- Jusqu'à quel niveau faut-il bien connaître une caractéristique avant qu'il soit raisonnable de considérer pouvoir la mesurer?

2- Comment savons-nous si nous avons réellement mesurer la caractéristique que nous voulions mesurer?

3- En utilisant la mesure, que pouvons nous dire à propos de cette caractéristique et de l'entité mesurée

4- Quel type d'opération peuvent être opérées sur ces mesures qui puissent être significatives?

Ceci consiste essentiellement à être d'accord qu'il y a une sorte de modèles des objets à être mesurés. Donc lorsque l'on parle de mesurer une caractéristique, ceci doit toujours être par rapport à un système particulier de relations (Fenton, 1991): mesurer une caractéristique, c'est l'arrimage (*mapping*) d'une relation empirique dans un système de relations numériques. Les mesures doivent également s'appuyer sur la discipline mathématique de la théorie de la mesure: sur quelle structure (espace) la mesure est-elle appliquée? Quels modèles sont employés pour la mesure (Eijogu, 1990).

1.3 Les métriques en génie logiciel

Les domaines du développement et de l'entretien du logiciel sont des domaines très complexes. Toutefois la démarche scientifique dans le domaine du logiciel en est encore à ses débuts. Quoiqu'il existe beaucoup d'outils et de techniques pour supporter les processus de développement et de maintenance, il manque encore de maturité dans deux des domaines qui se sont révélés très importants dans les autres sciences: d'une part, l'élaboration des modèles des processus et, d'autre part, l'élaboration de mesures pour contrôler, analyser et améliorer les processus de génie logiciel.

Il existe bien un nombre considérable de métriques en génie logiciel, mais il n'en demeure pas moins beaucoup de questions quant à l'utilité de ces métriques. En fait, il semble bien que le focus ait été souvent mis sur les métriques pour eux-mêmes, plutôt que sur leur signification et leur utilité. Il y a déjà dix ans, Brown et al. (1981) avaient noté que l'absence de principes de base en génie logiciel non seulement rend difficile la formulation de modèle cohérents ou de relations entre les variables d'un système, mais rend également difficile la spécification de systèmes de mesure dans ce domaine.

Le domaine du logiciel est très complexe, et représente une multiplicité de dimensions, chacune étant inter-reliée (Balzer, 1990; Nejmeh, 1990). Une étude de n'importe quel phénomène en génie logiciel, que ce soit la fiabilité, la productivité, l'estimation, la maintenabilité, doit tenir compte simultanément d'un nombre très grands de dimensions et de niveaux de complexité et de conceptualisation. De plus, aucun attribut n'est indépendant, et chacun influence les autres : ils sont inter-reliés et donc difficile d'observation parce que dans la réalité trop de facteurs varient en même temps, ce qui rend difficile l'étude de phénomène en isolation (MacDonell, 1991).

Or même face à la complexité de la tâche, il faut modéliser, mesurer et gérer le processus de développement du logiciel si on veut pouvoir l'améliorer progressivement pour en venir à l'optimiser. Pour cela il faut isoler, identifier, et catégoriser les différentes caractéristiques de la discipline, définir des notations pour les représenter et spécifier les inter-relations entre elles.

Le domaine des métriques en génie logiciel a fait l'objet de plusieurs publications récentes. Zuse (1990) a présenté les six principaux domaines des mesures actuelles en génie

logiciel, ces six ensembles n'étant pas disjoints:

- Les modèles d'estimation des coûts
- les mesure de productivité
- les modèles de qualité
- les modèles de fiabilité
- les mesures de complexité
- la complexité algorithmique

Zuse s'est principalement concentré sur les métriques de la complexité au niveau de la programmation et il a présenté un inventaire commenté du domaine de l'analyse de la complexité des lignes de code, incluant une analyse détaillée de près d'une centaine de métrique présentée dans la littérature spécialisée.

Card et al. (1990) examinent le domaine de la mesure de la qualité du design tandis que Musa et al. (1987) concentrent leurs recherches sur la fiabilité du logiciel. Fenton (1991) et Eijogu (1991) présentent tous les deux des analyses sur divers aspects des utilisations des concepts des systèmes de mesure dans le domaine des métriques de logiciel. Il s'est développé également un domaine d'analyses de productivité afin de s'adresser à un autres des besoins pressants en génie logiciel, à savoir l'urgence de développer des modèles d'estimation rapide et fiable (ESPRIT - MERMAID 1, 1989; Pfleeger, 1989).

Les processus de mesures représentent un niveau de maturité d'une discipline. Mais, selon Ejiogu (1991), il n'existe pas encore de métriques efficaces en génie logiciel. Ses principales critiques vis-à-vis les métriques actuelles en génie logiciel sont les suivantes:

1- Un manque d'appel à l'intuition (lack of intuitive persuasion)
2- Des règles incongrues de comptage
3- Des définitions imprécises des paramètres de mesure
4- Absence d'espace mathématique mesurable qui puisse être flexible suffisamment pour être extensionnés à d'autres métriques
5- Dans les modèles d'estimation, leur modèle mathématique viole potentiellement les règles d'analyse de dimensionnalités

1.4 Organisation du document

La démarche d'analyse de la métrique des points de fonction est organisée de la façon qui suit. Après quelques définitions de base sur les métriques en génie logiciel, le chapitre deux (2) passe en revue les caractéristiques générales des métriques de logiciel et des modèles de productivité et d'estimation de la taille des logiciels, en insistant en particulier sur la pertinence (limites et contraintes) des comparaisons pour des environnements non-homogènes. Une fois ce contexte établi pour les métriques de génie logiciel en général, le chapitre trois (3) présente une description de la métrique des points de fonction à partir de la perspective de son évolution

historique de façon à illustrer les améliorations qui y ont été apportées afin de rendre le processus moins sujet à des interprétations et à lui établir une base objective de critères. Le contexte de référence dans le quel cette métrique a été développée est également présenté puis, au chapitre quatre (4) des modèles de la métrique sont développés et proposés à partir strictement d'une perspective de l'analyse des processus de mesure incorporés à cette métrique. Ceci comprend non seulement le modèle général mais une décomposition en trois processus distincts de mesures: mesure de données, mesure des transactions et mesure de la valeur du facteur d'ajustement, le tout ensuite recombiné pour donner un total de points de fonction.

L'ensemble des recherches sur la métrique des points de fonction sont passées en revue dans les trois chapitres qui suivent. Le chapitre 5 propose une classification de ces recherches: la fiabilité (chapitre 5), les modèles de productivité et les modèles d'estimation (chapitre 6), les métriques dérivées (chapitre 7) puis les analyses des structures du processus de mesure de la métrique (chapitre 8). En fait, cette revue permet de réaliser que bien peu de recherches fondamentales ont été effectuées par rapport, et avec cette métrique.

Après cette analyse de la littérature, un ensemble de problématiques est inventorié au chapitre huit (8), incluant: les biais structurels historique, les problèmes organisationnels d'implantation, les structures des algorithme de mesure et la pertinence et les domaines des modèles implicites dans la métrique.

Parmi toutes ces problématiques, une seule est choisie (chapitre 10) pour une analyse plus approfondie, soit l'utilisation des échelles de mesure à travers les processus de mesure qui avaient été modélisés au chapitre quatre (4). De là, des hypothèses de recherche sont formulées et une méthodologie de recherche est proposée.

Chapitre 2: MÉTRIQUES ET MODELES DE PRODUCTIVITÉ

2.1 Les définitions

Avant d'aborder spécifiquement la description de la métrique des points de fonction, ce chapitre présente un tour d'horizon de la littérature sur diverses problématiques génériques aux métriques en génie logiciel, lesquelles sont susceptibles de se retrouver également chez les points de fonction. Cette première section établit les définitions de base qui seront utilisées tout au long de cette analyse car il n'y a pas de définitions normalisées du vocabulaire de base pour les métriques en génie logiciel:

- les mesures & métriques
- le logiciel
- le génie logiciel

2.1.1 Mesures et métriques.

De Marco (1982) définit une métrique de la façon suivante: "*A metric is the number you attach to an idea. More precisely, a metric is a measurable indication of some quantitative aspect of a system*". Voici quelques autres exemples de définitions au Tableau 2.1, et il est évident, de par ce tableau, que les nuances entre mesures et métriques ne sont pas claires dans la littérature courante.

Tableau 2.1 : Définitions - Mesure/Métrique

AUTEURS	MESURES	MÉTRIQUES
Applied Info. Dev.[2]	Unité fondamentale de mesure	Un indice calculé, ou combiné, basé sur deux ou plusieurs mesures
Meredith (1991)	*Measure: A basic quantitative value that describes the magnitude of a given element of the software engineering process*	*Metric: A composite of two or more measures resulting in a meaningful and predictive value that is useful in defining a characteristic or standard*
ICE[3] (Proposition de normalisation)	Évaluation des caractéristiques quantitatives	Évaluation qualitative et/ou prévision quantitative
IFPUG[4]	*Measure (Synomym: metric): A quantified rule for evaluating an object*	*Measurement: An occurence or instance of a metric*

[2]Applied Information Development Inc., The Hierarchy of Metrics

[3]ICE - 56(bureau Central)158, Tableau 1, page 7

[4]IFPUG, "Function Points as Assets", 1990

La définition de Meredith sera donc utilisée afin d'exprimer cette relation, à savoir que l'évaluation qualitative et/ou prévision quantitative est un assemblage structuré d'une ou plusieurs mesures en vue d'obtenir une valeur utile pour définir une caractéristique de l'objet mesuré.

2.1.2 Logiciel

Une définition intéressante du terme logiciel est présentée par Mills (1987) dans sa proposition de cours sur les métriques pour le Software Engineering Institute:
"*Le produit logiciel peut être considéré comme un objet abstrait qui a évolué à partir d'un énoncé d'un besoin pour se finaliser par un logiciel, lequel comprend tout aussi bien le code, objet ou source, que les diverses formes de documentation produites tout au long du processus de développement*".

Cette définition du "logiciel" de Mills est celle qui est utilisée tout au long de ce texte. Dans ce texte les expressions "applications logicielles", "systèmes logiciels" et "projets logiciels" seront utilisées indifféremment pour exprimer le même concept.

2.1.3 Génie logiciel.

Dans son livre Software Engineering, Pressman (1987) rappelle une des premières définitions du génie logiciel:

"*C'est l'établissement et l'utilisation de principes éprouvés d'ingénierie pour obtenir de façon économique un logiciel qui est fiable et qui fonctionne avec efficience sur des machines réelles*".

L'on retrouve dans cette définition les principaux éléments à la base de toute étude de productivité, à savoir les intrants, les extrants et les relations entre les deux.

Quant à la définition de génie logiciel la plus connue, celle de Boehms (1981), elle a été dérivée des définitions de base du dictionnaire Webster:

"*Logiciel: l'ensemble complet des programmes, des procédures et de la documentation connexe associée à un système, et particulièrement à un système informatique*".

"*Génie : l'application de la science et des mathématiques grâce auxquelles les propriétés de la matière et des sources d'énergie de la nature sont rendues utiles à l'homme par des structures, des machines, des produits, des systèmes et des processus*".

Boehms combine alors les deux définitions ensemble pour obtenir la définition suivante du génie logiciel:

"*C'est l'application de la science et des mathématiques grâce auxquelles les capacités de l'équipement informatique sont rendues utiles à l'homme via des programmes informatiques, des procédures et de la documentation associée*".

2.2 Caractéristiques

Il y a des métriques directes et des métriques indirectes du logiciel. Les métriques directes sont effectuées à partir du comptage des unités de base, telles les lignes de code. La plupart du temps ces métriques sont utilisées pour tenter de mesurer un autre phénomène, dont deux des plus souvent cités sont la complexité et la qualité du code. Par exemple, les métriques indirectes du code ont comme objectif d'établir, en partant des lignes de codes et de l'examen de leur contenu et de leur structure, des liens et des relations avec diverses caractéristiques des programmes telles:

- la taille
- le type
- la complexité
- l'exactitude
- la fiabilité

Ces métriques indirectes sont souvent calculées à partir de formules: des unités de base sont rassemblées, puis utilisées comme input à une formule. On peut prendre comme exemple les métriques de Halstead dont les formules sont basées sur le comptage d'opérandes et d'opérateurs, et les métriques des Points de fonction qui utilisent le comptage de cinq types de fonctionnalités.

Ces métriques indirectes sont également qualifiées de métriques dérivées ou de métriques synthétiques. Dans le document de synthèse du programme de recherche européen ESPRIT (projet MERMAID 2, 1989), ces métriques algorithmiques, ou indirectes, sont plutôt nommées métriques synthétiques.

2.2.1 Métriques synthétiques

Toutefois les chercheurs du projet ESPRIT - MERMAID note qu'il y a peu d'évidence que les métriques de logiciels soient dérivées de théories prouvées et ils portent les jugements suivants sur les métriques synthétiques:

- Les métriques synthétiques sont difficiles à interpréter parce que les raisons pour lesquelles sont assignées des valeurs particulières (poids) ne sont pas précisées.

- la valeur en sortie de la formule est utile seulement si la formule est basée sur une théorie sous-jacente solide, telle l'économétrie ou la physique. Or ce n'est pas le cas des métriques de logiciels actuels.

- il est nécessaire d'emmagasiner les valeurs de base des mesures qui constituent la métrique synthétique, pour garder le contrôle du processus de mesure.

Un autre des principaux problèmes des métriques de logiciel c'est qu'elles n'ont pas d'échelle d'interprétation, comme le zéro degré de congélation de l'eau et le 100 degré du point d'ébullition. Les métriques de logiciel ne peuvent donc être utilisées que de façon relative:

- par rapport à des plans et des cibles
- par comparaison avec des projets similaires
- par comparaison avec des composantes semblables
- par rapport à des déviations de normes, de moyennes.

2.2.2 Propriétés des données de logiciel

Les membres du projet MERMAID note qu'en général les ensembles de données de métriques de logiciels ont des propriétés indésirables (Voir le tableau qui suit). Ceci implique qu'il est souvent irréaliste d'utiliser les techniques conventionnelles de statistiques basées sur l'hypothèse que les données ont une distribution de fréquence normale Gaussienne (en forme de cloche). Avec des données non-Gaussiennes il faut alors utiliser la vraie distribution, une bonne technique statistique robuste, des techniques non-paramétriques, indépendantes de la distribution et appliquer des transformations aux données pour les forcer près d'une distribution gaussienne.

Tableau 2.2 : Propriétés des données Gaussiennes et non-Gaussiennes

Données	Gaussiennes	Données	Non-Gausssiennes
Caractéristiques	Techniques statistiques	Caractéristiques	Techniques statistiques
Symétriques	Régressions	Heavily skewed	Utiliser la vraie distribution
Continues	Corrélations	Valeurs discrètes	Techniques non-paramétriques
Distribution normale		Non négatives	Appliquer des transformations aux données
		Un nombre substantiel de "outliers"	Analyses de variance

Les techniques d'analyse de variance sont donc plus appropriées que les régressions et les techniques non-paramétriques ont l'avantage de réduire l'effet des *outliers*. Pour les analyses *multi-variate* la technique de la régression multi-variée semble acceptable. Pour un traitement en profondeur des méthodes statistiques des métriques de logiciel il faut se référer à l'ouvrage de Conte (1986) qui constitue un classique dans ce domaine.

2.3 Les modèles de productivité et d'estimation

2.3.1 Définition générale de la productivité

Le processus, tant du développement du logiciel, que de son entretien, a besoin d'être analysé, compris, géré et contrôlé. Or tant dans le domaine des sciences que dans le domaine de la gestion, la compréhension et le contrôle d'un phénomène, passe par des métriques. Mais ce n'est pas tout d'avoir des mesures, il faut également que ces mesures soient intégrées à l'intérieur d'un système de métriques cohérentes, avec une base de référence solide et normalisée. Le domaine du génie logiciel a produit un nombre considérable de modèles de productivité et d'estimation, mais par contre aucun de ces modèles n'est basé sur un système de référence rigoureux.

Le concept qui permet de relier ensemble les diverses facettes multidimensionnelles d'un même phénomène de production, c'est le concept de la productivité. De façon générale, la productivité est définie par la relation entre les résultats produits (ou les services rendus) et l'utilisation des ressources mises en oeuvre lors de la production (Filion, 1988). De façon arithmétique, la productivité s'exprime comme suit:

$$\frac{\text{EXTRANTS}}{\text{INTRANTS}} = \frac{\text{RESULTATS}}{\text{RESSOURCES}}$$

Il s'agit donc d'une métrique selon une échelle de **ratio**, l'échelle qui permet la meilleure analyse, mais également l'échelle la plus difficile à établir. Blake (1987) a identifié cinq critères importants pour les métriques de productivité:

Tableau 2.3 : Critères pour les métriques de productivité - Blake 87

Critères
1- Refléter avec justesse les changements dans la productivité; 2- Permettre de tenir compte de toutes les composantes tant de l'intrant que de l'extrant de la productivité totale; 3- Justifier les coûts qu'il engendre et donc ne doit pas, dans la mesure du possible, déranger le cours normal des activités de l'unité étudiée; 4- Favoriser l'objectivité plutôt que la subjectivité; 5- Permettre les comparaisons entre les unités de même type ainsi qu'à travers le temps.

Dans la définition ci-dessus de la productivité il existe plusieurs notions qu'il a été assez difficile de cerner jusqu'ici dans le domaine du génie logiciel. En effet, il faut faire une distinction très nette entre le produit et les processus de production, c'est-à-dire entre les résultats et les activités. Pour vraiment analyser et comprendre la productivité du développement d'une application informatique il faut approfondir simultanément les deux concepts: le produit logiciel et le processus de production du logiciel.

En partie à cause de l'absence de bonnes métriques pour mesurer la taille d'un logiciel, il y a peu de recherches sur la productivité du génie logiciel qui ont isolé séparément ces deux aspects de la productivité et qui ont analysé l'ensemble des facteurs et de leurs interactions en

tant que groupes à l'intérieur de chacun de ces deux aspects.

Les processus de production du logiciel ont fait l'objet de multiples études depuis les débuts de l'informatique. Les premières études dans les années 1960 et 1970 ont tout d'abord mis le focus sur le principal facteur de production du logiciel, à savoir le programmeur, et comme corollaire les activités de programmation[5].

Puis à mesure que les logiciels se faisaient plus volumineux et plus complexes le focus s'est déplacé sur l'ensemble des facteurs qui étaient susceptibles d'influencer la productivité des équipes de développement. Cela a mené à toute une série de propositions de modèles de productivité, selon les points de vue choisis par les promoteurs de ces modèles.

2.4.2 Classification des modèles

Les modèles algorithmiques, ou paramétriques, sont définis comme les modèles qui utilisent des données historiques pour un ou plusieurs algorithmes pour produire des estimés des efforts de développement. Cette approche utilise un "facteur de productivité" moyen. Il existe un vaste champs de publications pour les modèles de ce type. Voici une classification, proposée par MERMAID, de ces modèles algorithmiques: basés sur des statistiques, sur des données historiques, sur des théories ou encore des modèles composés.

Tableau 2.4 : Classification MERMAID des modèles algorithmiques

Statistiques	Données historiques	Théories	Modèles composés
- SDC (US Air Force) 1964 - ARON (IBM) 1969	- Wolverton (TRW) 1974 - Doty (US Air Force) 1977 - Warson & Felix (IBM) 1977 - Bailey & Basili 1981	- Norten (IBM) 1963 - SLIM - Putnam 1978	- COCOMO Boehm 1981 - Jensen 1983 - Kulkarni 1988

Ces modèles assument en général soit une forme linéaire, soit une forme non-linéaire:

- modèles linéaires: E = Co + (SUM CiXi)
- modèles non-linéaires: E = A + B*S**c

Il est à remarquer que la qualité de ces modèles algorithmiques est très dépendante des facteurs suivants:

- un jugement d'expert pour l'estimation de la taille et de la complexité de chacune des composantes individuelles.

---

[5]Voir les études des années 1960 qui étaient centrées sur le programmeur.

- des données historiques sur lesquelles les modèles paramétriques sont basés. Ces données sont intimement liées aux conditions des environnements de développement. C'est bien souvent l'environnement qui est modélisé, et pas simplement l'effort requis pour développer un produit particulier (ESPRIT, Mermaid 1, 1989).

- de la qualité des données reliées au niveau de maturité et la capacité de mesurer, et de bien mesurer.

Donc les modèles paramétriques peuvent être appliqués avec succès seulement par des analystes très familiers avec les "requirements" du produit à développer et avec les caractéristiques de l'environnement de développement, et dans un environnement avec un bon niveau de maturité. Pour des analyses plus détaillées de ces modèles, il est possible de se référer aux sources suivantes: Conte et al. (1986), Laranjeira (1990), ESPRIT (1989), Wrigley (1986), Oligny et al. (1988) et Kemerer (1987).

2.4.2 Validité statistique

Il a été vu précédemment que pour la plupart des modèles relativement peu d'informations scientifiques étaient disponibles, dû à leur origine dans le domaine commercial où ces modèles étaient traités comme des avantages concurrentiels. De la même manière il y a peu d'information sur la validité des méthodes de collecte des données et de leur degré de précision. Il y a néanmoins quelques études qui ont tenté d'évaluer la qualité de ces modèles, de même que leur performance relative dans les estimés.

L'ouvrage de référence le plus intéressant et le plus détaillé sur la discussion des modèles et de leur approche statistique est sans contredit l'ouvrage de Conte et al. (1986). Dans ce volume ces auteurs révisent en détail l'approche statistique de chacun des modèles et en examine les hypothèses et les résultats statistiques en fonction des données de recherche de chaque modèle. Les critères pour déterminer une performance acceptable de ces modèles utilisent les concepts statistiques suivants:

Tableau 2.5 : Critères de performance statistique

	Coeff.	Description
1	R**2	Coefficient de détermination multiple
2	RE	Erreur relative moyenne
3	MRE	Ordre de grandeur de l'erreur relative moyenne
4	PRED	Facteur de prédiction
5	EQR	Racine carré de l'erreur relative

Chacun des modèles est examiné en tenant compte des critères de validité statistique, tant à partir des données initiales des publications, qu'à partir d'autres ensembles de données qui ont été utilisées pour d'autres modèles.

Il faut toutefois se rappeler que tous les tests statistiques de même que les ensembles de données disponibles ne tiennent compte que de facteurs *a posteriori*: pour chacun des modèles et chacune des comparaisons, il n'y a pas d'information de disponible sur le comportement de ces modèles par rapport aux estimations initiales *ex ante*. Toutes les comparaisons sont faites uniquement entre la performance de prédiction du modèle et les données post-projet, lorsque ceux-ci sont terminés et que 100% de l'information est connue, tant sur le contenu du produit que sur le processus de développement qui a mené au produit complété.

Le problème de la justesse des prévisions faites avant le début du projet par rapport aux actuels post-projets n'a pas été examiné. Aucune référence n'a été trouvée dans ce domaine. De la même façon, il n'y a pas de référence, ni de données de recherche, sur les hypothèses qui auraient pu être faites *ex ante*, et de la réalité *a posteriori* sur la justesse de ces hypothèses, et encore moins de leur impact. Par exemple, quelle est la performance de ces modèles dans une situation *ex ante* où seulement 10% de l'information est connue, ou bien 20% ou encore 30%, comme c'est la majorité des cas lorsque des estimés de projets doivent être faits?

Le seul secteur où cet exercice a été fait ne concerne qu'une seule dimension, à savoir les estimés de la taille des systèmes exprimée en lignes de code, et la fiabilité de ces estimés. Et encore, ces estimés sont faits en général relativement tard dans le cycle de projet, lorsque pratiquement 100% du produit est maintenant connu, que l'architecture et le design sont complétés et que toute l'analyse, et les coûts qui y sont reliés ont déjà été comptabilisés.

Ceci ne veut pas dire qu'il n'y a aucune information sur ces dimensions de l'estimation, mais plutôt que cette information est strictement empirique au niveau de la pratique et qu'elle n'est pas disponible, ni publiée au niveau de la recherche. Le problème de la validité et des niveaux de confiance de ces données se posent également.

2.4.3 Les limites des modèles

Parmi les faiblesse des modèles il faut en mentionner les principales faiblesses, à savoir:

1- Les équations des modèles s'appuient sur une donnée qui, elle-même n'est qu'un estimé, à savoir l'estimé de la taille du logiciel, exprimé par un estimé en lignes de code;
2- Ces modèles ne tiennent compte que des temps de programmation et test;
3- Ces modèles ne tiennent pas compte de l'aspect dynamique des systèmes, à savoir la gestion des changements.
4- Les LOC ne sont pas une mesure naturelle pour prédire la taille des phases de développement qui ne sont pas orientées vers la programmation.

La majorité des modèles s'appuie sur une donnée fondamentale, à savoir la taille du logiciel, exprimée habituellement en lignes de code. Or si cette donnée est connue de façon précise à la fin des projets (c'est-à-dire *a posteriori*), ce qui permet d'avoir des données historiques pour trouver les coefficients des équations, cette donnée sur la taille n'est généralement pas connue au début du projet, c'est-à-dire *ex ante*, et elle doit être estimée.

Or aucun des modèles proposés ne propose de méthodologie pour déterminer la taille *ex ante*, du degré de précision et de l'impact de cette estimation. Deux études ont mentionné ce problème fondamental, soient Laranjeira (1990) et le projet MERMAID qui a consacré tout un projet à une recherche de littérature sur les façons d'estimer la taille des logiciels.

Ces modèles seront donc très sensibles aux erreurs d'estimation des lignes de code. Par exemple avec l'équation

$$E = A + B*(KLOC)**C$$

une erreur d'estimation de 50% pour les lignes de code aura un impact de 63% sur les estimés des efforts requis pour le projet.

Les commentaire du projet ESPRIT sur l'estimation sont les suivants:

- L'estimation de l'effort et du coût n'est pas un processus déterministique. Les interactions humaines et le jugement sont essentiels tout au long du processus.

- L'estimation de la taille exige des compromis entre certains aspects des spécifications du système et de la qualité du logiciel avec les contraintes de ressources et de durée.

- Transmutabilité des modèles

- Les différents modèles produisent des estimés d'efforts qui varient substantiellement, un phénomène qui ne peut être expliqué facilement.

- impact de l'environnement sur les facteurs: il est difficile d'isoler l'impact d'un facteur en particulier, parce que les facteurs sont inter-dépendants entre eux.

Les modèles développés dans un environnement particulier, ne peuvent capturer l'impact que des facteurs qui sont variables dans cet environnement, alors que les autres qui y sont constants ne provoquent pas de variation dans la productivité; mais ils peuvent avoir un tout autre effet dans un autre environnement.

2.4 Pertinence des comparaisons: environnement non-homogènes

2.4.1 Points de rupture et pathologies

Les concepteurs des modèles, et ceux qui les évaluent, n'ont pas délimité de façon explicite les contraintes de leurs modèles: Ils n'ont pas défini les contraintes d'utilisation de leur modèle en définissant par exemple les conditions de normalités et d'anormalités.

De la même façon ils n'ont pas défini non plus les conditions sous lesquelles le processus du développement est considéré sous contrôle ou hors contrôle. Par exemple dans la construction d'un modèle, toutes les données sont utilisées telles quelles, sans considérer s'il y a des aberrations ou non, ni sans normalisation des données si cela devrait être fait selon certaines conditions. Ainsi un projet qui aurait été sujet à des problèmes très importants de dépassements de coûts dus à des erreurs d'analyse des besoins ou de conception architecturale ne fait pas l'objet d'une normalisation lorsque ses données de temps sont utilisées dans les modèles actuels.

De même l'utilisation des données de productivité d'un projet soumis à des contraintes excessives et déraisonnables de limites de temps, telle qu'utilisé dans les articles d'Abdel-Hamid (1990), ne peuvent invalider un modèle. Ceci peut être illustré par deux exemples différents:

> 1- Le comportement élastique des matériaux est prévisible, et en général exprimé sous forme d'équation, à l'intérieur de certaines contraintes et de certains paramètres. Toutefois au delà de certaines limites bien précises, il y a un point de rupture qui en change radicalement le comportement et au-delà duquel l'équation initiale n'est plus valide et doit être remplacée par une équation d'une tout autre forme.
>
> 2- En médecine, une personne saine réagit de façon prévisible à une foule de stimulus, mais dans les cas pathologiques, les règles du jeux ne sont plus du tout les mêmes et pensons uniquement au cancer (et aux différentes formes de cancer) et au S.I.D.A.

Or il en est de même pour les projets de génie logiciel où il existe des points de rupture et des contraintes, mais lesquels n'ont pas été identifiés de façon précise. Le modèle de Putnam avec la courbe de compression de temps est le seul type de modèle qui s'approche de ce concept avec l'exploration de la compression de certaines contraintes et de l'analyse du comportement sous certaines contraintes. Toutefois les points de rupture ne sont pas vraiment identifiés, ni l'impact de ces ruptures et des changements radicaux de comportement.

Basili & Ramsey (1989) soulignent l'importance des conditions d'homogénéité de l'environnement et mentionnent des conditions telles que:

- types de projets similaires pour des mêmes secteurs d'application
- modèle standard de processus développé au cours des années
- la méthodologie de développement est similaire pour les projets
- il y a beaucoup de ré-utilisation à partir des projets précédents.

Dans un environnement homogène, et avec des métriques homogènes, les déviations par rapport à une métrique indiquent que quelque chose de différent par rapport à la norme s'est passée. Les experts, qui en général travaillent dans cet environnement homogène depuis plusieurs années, déterminent certaines explications: de bons essais, ou des spécification instables. C'est une approche bottom-up: manifestation à interprétation, à cause. Basili mentionne cependant les limites des expérimentations actuelles en génie logiciel:

- les connaissances et les relations ne sont pas claires
- les experts eux-mêmes ne sont pas en accord
- trop de variables dans les modèles, et pas assez d'observations dans les échantillons des expérimentations
- les métriques utilisés ne sont pas idéales
- plusieurs des interprétations sont subjectives
- il peut y avoir des différences d'interprétation selon les phases de développement.

Au niveau macro-économique Cusamano et Kemerer (1989) se sont penchés sur une comparaison de la productivité du développement informatique des USA versus celle du Japon et ils mentionnent que le problème fondamental des analyses de productivité, et d'estimation, en génie logiciel est le suivant:

> Ni les clients ni les producteurs n'ont défini précisément les normes du produit, ni les normes du processus du logiciel

L'approche des chercheurs du M.I.T. est particulièrement intéressante parce qu'elle s'intéresse spécifiquement à la productivité et que dans ce cadre ils ont regardé au niveau macro-économique plusieurs aspects comme les données sur le temps d'analyse, la répartition du temps selon les étapes du cycle de développement et les impacts de cette répartition sur la productivité et la qualité de même que de divers autres aspects de la productivité. Leurs méthodes d'analyses statistiques sont également très intéressantes mais, comme mentionné, la faiblesse réside principalement dans la qualité de leurs données de recherche, lesquelles données ne semblent pas avoir été contrôlées.

2.4.2 Impact des contextes multi-environnements

Plusieurs des problèmes méthodologiques ci-dessus ont été relevés par Moreau et al. (1990) qui s'est attaché plus précisément à l'analyse des environnements de programmation comme facteur d'influence de la productivité, tout aussi bien qu'à une identification du problème de l'identification des caractéristiques des applications logicielles. Il critique de la façon suivante les méthodologies actuelles:

1- L'information recueillie est de peu d'utilité pour évaluer les environnements de programmation très complexes où les interactions entre les facilités de support

est un facteur majeur de la productivité du développement.

2- Certaines des approches actuelles font des hypothèses de base qui rendent difficiles les évaluations d'environnements de développement très hétérogènes.

3- Certaines de ces approches sont basées sur des évaluations d'experts des critères d'évaluation.

4- Toutes ces approches actuelles n'identifient pas les métriques primaires qui caractérisent la structure des applications logicielles.

5- Les modèles qui s'inspirent de COCOMO proposent des listes de critères subjectifs sur les caractéristiques et des fonctions des environnements de développement. Avec ces approches non seulement les définitions opérationnelles des critères sont subjectives, mais également les mesures de ces critères (Weiderman 87). Il note qu'il est très difficile d'appliquer cette méthode de façon consistante.

2.5 Sommaire

Les modèles actuels d'estimation en génie logiciel souffrent de faiblesses importantes: en fait ce sont en général des modèles de productivité qui n'ont été validés que dans des conditions "a posteriori". Leurs limites et contraintes d'utilisation d'ont pas été de plus clairement définies ce qui rend difficile de déterminer le type de situations pour lesquelles ils sont valides, ou invalides. Enfin ces modèles sont à la fois des modèles de type algorithmique, et sont simultanément basés sur des métriques algorithmiques. Ils ont donc les faiblesses suivantes:

- Ils sont difficiles à interpréter parce que les raisons pour lesquelles sont assignées des valeurs particulières (poids) ne sont pas précisées.

- La valeur en sortie de la formule est utile seulement si la formule est basée sur une théorie sous-jacente solide, telle l'économétrie ou la physique. Or ce n'est pas le cas en général en génie logiciel actuellement.

Chapitre 3: DESCRIPTION ET ÉVOLUTION DE LA MÉTRIQUE

3.1 Objectifs initiaux

En octobre 1979, Allan J. Albrecht présente pour la première fois les résultats de sa nouvelle technique de mesure des logiciels, appelée *Function Points Analysis*, à l'occasion d'une rencontre des groupes SHARE et GUIDE D'IBM à Monterey, Californie. Cette métrique a été élaborée initialement pour répondre à un besoin spécifique d'une mesure de productivité applicable à un contexte précis.

L'objectif était d'établir un système de mesure pour mesurer l'augmentation de productivité dans un ensemble de 24 projets réalisés sur une période de 5 ans (1974 à 1978), lesquels projets avaient été développés avec des langages de programmation et des outils différents. La métrique associée à la mesure d'un logiciel devait être valide quel que soit le langage de programmation employé et devait représenter les services fournis à l'utilisateur. Comme les métriques basées sur les lignes de code n'étaient pas appropriées à ce travail, Albrecht a donc cherché à développer une nouvelle approche plus conforme aux besoins exprimés pour la mesure de l'augmentation de productivité sur cet ensemble de projets.

Pour mesurer la productivité, un produit et un coût doivent être définis et mesurés. Connaissant d'une part les coûts des projets (exprimés en jours de travail), le problème résidait dans la mesure du produit, dans ce cas ci les projets à mesurer. Dans cette optique, le produit d'un projet devait pouvoir être quantifié en termes des valeurs fonctionnelles livrées à l'usager, lesquelles devaient être mesurées objectivement par ses manifestations externes, telles que vues et perçues par les usagers. Voici les définitions de base d'Albrecht (1983) dont deux des mots-clefs sont certainement les mots "produit" et "usager".

Tableau 3.1: Définitions de base d'Albrecht

1	Les points de fonction sont une mesure du produit du processus du développement et d'entretien basée sur la fonction usager du traitement de l'information.
2	Les points de fonction mesurent une application en quantifiant la fonctionnalité du traitement de l'information associée avec les principaux intrants de données externes ou de contrôle, les sorties et les types de fichiers.
3	Ce traitement spécifique d'information est ensuite ajusté pour la fonction générale du traitement de l'information en appliquant un ajustement basé sur les caractéristiques générales de l'application.

3.2 Évolution historique

3.2.1 Albrecht 1979

Le modèle initial de 1979 est relativement simple: il comprend quatre types de fonctions, un seul poids par fonction (Tableau 3.2), et dix facteurs d'ajustement de complexité maximum de +/- 25% pour l'ensemble du comptage final de l'application (Tableau 3.3). Les entrées, les sorties, les requêtes et les fichiers livrés sont alors comptés, pondérés, additionnés et ajustés par des poids. Selon Albrecht, ces quatre groupes de fonctions devaient pouvoir tenir compte de toutes les fonctionnalités livrées à l'usager.

Tableau 3.2 : Albrecht 79 - Poids

	Types de fonction	Poids	Facteurs d'ajustement de complexité
1	Fichiers	10	+/- 25%
2	Entrées	4	
3	Sorties	5	
4	Requêtes	4	

Tableau 3.3 : Albrecht 79 - Facteurs

Facteurs d'ajustement de complexité	Degré d'influence par facteur
1- Backup 2- Data communications 3- Distributed processing 4- Performance issues 5- Heavily utilized configuration 6- On-line data entry 7- Conversational data entry 8- On-line update of master files 9- Complex functions 10-Internal processing complex	0- Aucun 1- Accessoire 2- Modéré 3- Moyen 4- Significatif 5- Essentiel
Ajustement total de complexité	= (0.75 +/- 25% max)

L'intention d'allouer des poids différents aux différents types de fonctions était de reconnaître que chaque fonction représentait des valeurs fonctionnelles différentes selon le point de vue des usagers. Albrecht mentionne que, bien qu'il ait effectué des analyses statistiques

initialement, les poids ont été basés principalement selon des jugements d'experts ("*by debate and trial*"), et ne découlent pas d'une analyse statistique systématique. Dans ce contexte, l'ensemble du processus donne alors un nombre, sans dimension, défini en "points de fonction" et qui, selon l'expérience d'Albrecht, donne une mesure relative et efficiente des valeurs fonctionnelles livrées à l'usager ("*This gives a dimensionless number defined in function points which we have found to be an effective relative measure of function value delivered to our customer*",Albrecht, 1979).

Voici un exemple du calcul des points de fonction pour une application comprenant 3 fichiers, 2 entrées, 2 sorties, 5 requêtes et dont les sept premiers facteurs d'ajustement de complexité ont une valeur zéro, et les trois derniers la valeur 4 pour leur degré d'influence. Tel qu'illustré au tableau 3.4, cette application obtient une valeur de 59 points de fonction.

Tableau 3.4 : Albrecht 79 - Exemple de calcul

Types de fonction	No. Fonctions * Poids	Facteurs d'ajustement de complexité
Fichiers	3 * 10 = 30	Facteurs 1 à 7 = 0
Entrées	2 * 4 = 8	Facteur 8 = 4
Sorties	2 * 5= 10	Facteur 9 = 4
Requêtes	5 * 4 = 20	Facteur 10 = 4
Total	68	Total = 12
Points de fonction =	68 * (0.75 + 12)	= 59 Points de fonction

3.2.2 Albrecht 1983

En novembre 1983, une version modifiée de la méthode est présentée par Albrecht (Tableau 3.5). La notion de complexité des fonctions est alors introduite avec l'introduction de trois niveaux de complexité, de règles pour évaluer la complexité par type de fonction, et un tableau avec des poids correspondants. La version de 1983 introduit également un cinquième type de fonction, les fichiers logiques externes (interfaces), lesquelles étaient auparavant agrégées à l'intérieur des fichiers. Encore une fois, la détermination des poids à l'intérieur des matrices 3x3 de complexité est basée sur une évaluation des experts, de même que la classification des fonctions en simple, moyenne et complexe en fonction des jugements des experts sur leur niveau de complexité.

Tableau 3.5 : Albrecht 83 - Poids

	Types de fonction	Poids simple	Poids moyen	Poids complexe	Facteurs d'ajustement
1	Fichiers logiques internes	7	10	15	
2	Fichiers logiques externes (interfaces)	5	7	10	
3	Entrées externes	3	4	6	+/- 35%
4	Sorties externes	4	5	7	
5	Requêtes externes	3	4	6	

La définition du premier des dix facteurs d'ajustement de complexité change de *backup* à *reusability* et le nombre de facteurs d'ajustement augmente à quatorze (Tableau 3.6), avec la même structure pour les degrés d'influence, ce qui fait passer le maximum d'ajustement de (+/- 25%) à (+/- 35%).

Tableau 3.6: Albrecht 83 - Facteurs

Facteurs de complexité	Degré d'influence par facteur
1- ~~Backup~~ *Reusability* 2- Data communications 3- Distributed processing 4- Performance issues 5- Heavily utilized configuration 6- On-line data entry 7- Conversational data entry 8- On-line update of master files 9- Complex functions 10-Internal processing complex *11-Installation ease* *12-Operational ease* *13-Multiple sites* *14-Facilitate change*	0- Absent ou aucune influence 1- Non significative 2- Modérée 3- Moyenne 4- Significatif 5- Forte, dans l'ensemble
Ajustement total de complexité	= (0.75 +/- 35% max)

Comme illustration, le tableau 3.7 présente les définitions des niveaux de complexité pour une fonction de type ENTRÉE.

Tableau 3.7 : Albrecht 83 - Niveaux de complexité - Fonction de type ENTRÉE

Complexité	Description
Simple	Il y a peu de types d'éléments de données, et il y a peu de types de fichiers logiques internes référencés par le type d'entrée externe. Les considérations des facteurs humains pour les usagers ne sont pas significatives pour le design du type d'entrée externe.
Moyenne	Le type d'entrée externe n'est pas clairement ni simple, ni complexe.
Complexe	Plusieurs types d'éléments de données sont inclus dans le type d'entrée externe, et plusieurs types de fichiers logiques internes sont référencés par le type d'entrée externe. Les considérations des facteurs humains des usagers affectent de façon significative le design du type d'entrée externe.

3.2.3 GUIDE 1985

Comme ces classifications de complexité pouvaient être interprétées de façons différentes par plusieurs personnes entre diverses organisations (Pressman, 1987), et à l'intérieur d'une même organisation (donc sujet à interprétation et, par conséquent, subjectif), un groupe du GUIDE (GPP-134, 1985) reprit les définitions de base d'Albrecht pour établir, clarifier et préciser les règles en ce qui concerne les définitions des cinq types de fonctions, codifier les règles courantes et présenter des exemples de ce qui devait être compté, ou ne pas être compté.

Ils introduisent également une nouveau concept de tables de décisions pour identifier de façon précise ce que voulaient dire les termes tels que "peu" et "plusieurs" de façon à obtenir des critères précis pour classer chacune des fonctions à l'intérieur des trois classes (simple, moyenne et complexe). Ces tables de décisions sont basées sur l'identification et le comptage de composantes primaires à l'intérieur de chaque fonction. Trois types de composantes primaires sont alors identifiés (Tableau 3.8).

Tableau 3.8 : GUIDE 85 - Types de composantes primaires

Description	Abréviation	Description en anglais
Types d'éléments de données	DET	Data Element Types
Types d'enregistrements	RET	Record Types
Types de fichiers référencés	FTR	File Types Referenced

A partir de ces composantes primaires, deux types de tables de décision à deux dimensions sont construites (Tableau 3.9), le premier type pour les fonctions de type fichiers

logiques (internes et externes), le deuxième pour les fonctions de type transactions (les entrées, les sorties et les requêtes).

Tableau 3.9 : GUIDE 85 - Dimensions des tables de décision, par type de fonction

Fonctions de type *Fichiers*		Fonctions de type *Transactions*	
Dimension 1	Dimension 2	Dimension 1	Dimension 2
Nombre de types d'enregistrements	Nombre de types d'éléments de données	Nombre de fichiers logiques référencés	Nombre de types d'éléments de données

A partir de cette structure de tables de décision, des intervalles de nombre de composantes primaires pour chaque dimension sont alors déterminés par les experts du group GUIDE afin d'établir les classements (simple, moyen et complexe). Trois tables à deux dimensions sont ainsi construites (Tableau 3.10): une pour les fichiers logiques (internes et interfaces), et deux pour les transactions avec des valeurs d'intervalle différentes pour les entrées et les sorties. Il est à noter qu'un autre concept est introduit: pour les transactions de requêtes, il faut calculer à la fois la partie entrée et la partie sortie de la requête, puis choisir comme valeur finale la plus élevée des deux.

Tableau 3.10 : GUIDE 85 - Structure des tables de décision

FICHIERS				TRANSACTIONS							
				Entrée				Sorties			
RET	1 à 4 DET	5 à 15 DET	16+ DET	FTR	1 à 19 DET	20-50 DET	51+ DET	FTR	1 à 5 DET	6-19 DET	20+ DET
0-1	S	M	M	1	S	S	M	0-1	S	S	M
2	S	M	C	2 à 5	S	M	C	2 à 3	S	M	C
3+	M	C	C	6+	M	C	C	4+	M	C	C
S = Simple M = Moyen C = Complexe											

Une fois que les types d'éléments de données et les fichiers et enregistrements logiques utilisés sont comptés, le résultat est ensuite positionné dans le tableau de dimension 3 X 3 correspondant au type de fonction, lequel tableau leur attribue alors un niveau de complexité (Tableau 3.10) et un poids correspondant (Tableau 3.5). Grâce à ces tables de décisions, des personnes différentes peuvent alors utiliser des règles qui leur permettent de mesurer et d'obtenir les mêmes classements (Simple, moyen et complexe) pour l'attribution des points de fonction

bruts.

Voici un exemple de calcul pour un fichier logique externe ayant 2 types d'enregistrements logiques différents et 13 éléments de données. Ce fichier a donc dans la première dimension 2 RET et dans la deuxième 13 DET, ce qui lui assigne une classification M (Moyenne) dans la table de décision (Tableau 3.10). En consultant la matrice des poids (Tableau 3.5) pour un fichier logique externe classé moyen, ce fichier obtient un poids égal à 7.

3.2.4 IFPUG

A l'automne 1984, un groupe d'utilisateurs de la technique des points de fonction se forme à Toronto et s'identifie sous le vocable "INTERNATIONAL FUNCTION POINT USERS GROUP", plus connu sous l'acronyme IFPUG[6]. Le mandat du groupe d'usagers est de diffuser la méthodologie, d'en coordonner l'évolution, de suivre l'implantation dans l'industrie et de promouvoir l'homogénéité de l'application des règles. Le groupe d'usagers, en plus d'organiser des conférences spécialisées semi-annuelles, met sur pied un comité pour examiner l'application des règles de pointage et pour établir les normes (le *Counting Practices Committee*). Ce comité recommande que les définitions d'Albrecht deviennent les normes de l'industrie mais il reconnaît également qu'un guide de pointage doit être préparé afin de préciser les définitions, fournir des exemples pour refléter les changements qui surviennent dans l'industrie et pour ajuster le vocabulaire et les concepts à des environnements de développement autres que ceux d' IBM.

3.2.4.1 IFPUG 1986

La version 1 du manuel de pointage de IFPUG est publiée en févier 1986 et, pour chacun des cinq types de fonction, le manuel comprend des définitions alternatives, des candidats potentiels, des guides d'interprétation et des exemples. A la dernière section du manuel IFPUG version 1, se trouve une série de points en suspens pour lesquels il n'y avait pas encore eu de consensus, comme par exemple les règles pour les menus de navigation, les messages d'erreurs et les générateurs de rapports.

3.2.4.2 IFPUG 1988

La version 2 du manuel IFPUG est publiée en avril 1988. Le manuel est alors considéré comme le focus pour l'accumulation des directives d'interprétations (*guidelines*), des exemples de ces interprétations et des suggestions pour ceux qui doivent compter les points de fonction. Il y est mentionné que ce n'est pas un document figé dans le temps: les définitions y sont relativement stables, mais leur interprétation doit tenir compte des changements dans l'environnement des affaires (*business*) et des environnements techniques. Cette version 2 inclut

---

[6]Le groupe IFPUG s'incorpore en 1988 dans l'état de l'Ohio, USA.

deux types de guides d'interprétation (*guidelines*) pour chacun des cinq types d'éléments:
- des guides d'interprétation au niveau logique
- des guides d'interprétation au niveau physique

3.2.4.3 IFPUG 1990

La version 3 est publiée en mars 1990 et représente un effort majeur pour transformer les diverses interprétations des manuels précédents en un tout cohérent qui représente le consensus des experts sur les règles d'identification et de mesure des points de fonction. Les objectifs du comptage des points de fonction y sont aussi précisés (Tableau 3.11).

Tableau 3.11 : IFPUG 90 - OBJECTIFS

1- Mesurer ce que l'usager a demandé et obtenu; 2- Mesurer de façon indépendante de la technologie utilisée pour l'implantation; 3- Obtenir une métrique de la dimension (taille) pour supporter l'analyse de la qualité et de la productivité; 4- Obtenir un véhicule pour l'estimation du logiciel; 5- Obtenir un facteur de normalisation pour la comparaison du logiciel.

Cette version 1990 du manuel contient plusieurs améliorations qui visent à rendre le processus de mesure plus facile, simple et concis de façon à permettre la consistance dans le temps, les projets et les praticiens. Il est à noter que dans le vocabulaire IFPUG l'expression "caractéristiques générales de l'application" (*General System Characteristics*) correspond à l'expression "facteurs d'ajustement de complexité" utilisée dans la version Albrecht 79. La version 1990 contient plusieurs innovations telles qu'illustrées au tableau 3.12.

Tableau 3.12 : Innovations du manuel IFPUG 1990

1- Le vocabulaire est normalisé et défini de façon précise dans un glossaire; 2- Les interprétations sont reformulées sous forme de règles; 3- Les règles qui sont conformes au document d'Albrecht 1984 sont identifiées comme telles; 4- Les règles qui n'étaient pas dans le document d'Albrecht 1984, ou qui ont fait l'objet de décisions différentes du comité international sont également identifiées afin de permettre de suivre l'évolution de la méthodologie des points de fonction, et d'en assurer une interprétation et normalisation. 5- Les règles pour la détermination des valeurs (de 0 à 5) pour chacune des 14 caractéristiques générales d'une application sont précisées pour permettre d'obtenir une notation plus objective.

Comme exemple de ces améliorations, une comparaison des versions Albrecht 84 et

IFPUG 90 pour la caractéristique numéro six *On-line Data Entry*[7] est présentée au Tableau 3.13 et illustre que les critères descriptifs (et relativement subjectifs) de la version Albrecht 84 on fait place à des critères quantitatifs très précis (et donc plus objectifs).

Tableau 3.13 : Online Data Entry Characteristics: Degree of influence criteria

Degree of influence	Albrecht 84	IFPUG 90
0	Not present, or no influence	All transactions are processed in batch mode
1	Insignificant influence	1% to 7% of transactions are interactive data entry
2	Moderate influence	8% to 15% of transactions are interactive data entry
3	Average influence	16% to 23% of transactions are interactive data entry
4	Significant influence	24% to 30% of transactions are interactive data entry
5	Strong influence throughout	Over 30% of transactions are interactive data entry

Bien que se voulant totalement indépendante des technologies, cette métrique a néanmoins été introduite avant la généralisation des méthodologies structurées et de la modélisation des données. Le vocabulaire est donc encore influencé par la méthode de développement avec le type de bases de données en usage courant au début des années 1980. La version IFPUG 1990 des règles officielles de la métrique fait une place, dans sa structure, à une seconde approche, basée sur le modèle Entité/Relation, pour l'identification des données primaires de la métrique mais il s'agit tout au plus d'une reconnaissance de l'intérêt pour cette approche et aucun détail n'en est encore publié. L'approche basée sur le modèle Entité/Relation (IFPUG E/R*) fait actuellement l'objet de discussions afin d'en évaluer la valeur de même que l'impact du changement que cela provoquera dans la base de pointage déjà établie mais ne fait pas encore partie des documents officiels d'IFPUG. C'est dans ce contexte qu'une étude a été commanditée au M.I.T. afin d'évaluer les impacts de ce changement de méthodologie (Kemerer 1990). Cette étude est basée sur la version Desharnais (1988). Dans cette approche, moins dépendante de l'implantation physique des applications, le concept d'entité logique est utilisé, plutôt que celui des fichiers ou enregistrements logiques pour lesquels il n'existe pas de définitions normalisées dans l'industrie.

[7]Dans la méthodologie IFPUG 90, la caractéristique *On-line Data Entry* est définie de la façon suivante: "On-line data entry and control functions ... provided in the application".

3.3 Le concept de contexte de référence

Cette métrique a l'avantage d'avoir des normes reconnues et des procédures détaillées qui permettent de compter de la même façon et d'établir une base de référence pour des comparaisons, pourvu que les règles de calcul et de comptage soient appliquées de façon correcte, bien évidemment. Il s'agit cependant d'**un modèle empirique**, dont les poids ont été établis et choisis selon les critères de l'expertise.

L'article original d'Albrecht (1979), décrit d'une part le contexte global (Tableau 3.14), puis les critères de sélection des projets admissibles (Tableau 3.15) pour l'application de la métrique, le tout définissant le contexte de référence dans lequel, et pour lequel, la métrique des points de fonction avait été élaborée.

Tableau 3.14: Albrecht 79 - Description du contexte global

- L'organisation comprend 450 ressources en développement d'application - Développement sous contrat pour des clients d'IBM - Les ressources et les clients sont dispersés dans tout les USA - En tout temps de 150 à 170 contrats sont en cours - La *taille* d'un contrat moyen est de 2 à 3 personnes - Chaque contrat est fait à l'intérieur du cadre d'une méthodologie de développement - La majorité des contrats ne comprennent que certaines phases de la méthodologie - Basé sur leur expérience, la phase de design prend 20% des heures de travail (*work-hours*), et la phase d'implantation, 80% - Il faut mesurer tout le processus, incluant le design... des coûts sont encourrus dans le design - Les projets ont été complétés de la mi-1974 au début 1979 - La *taille* des projets varie de 500 à 105,000 heures de travail - Sur environ 1500 contrats pour la période, seulement 22 ont rencontrés les critères de sélection

Tableau 3.15 : Albrecht 79 - Critères de sélection de projets admissibles

1	Seulement les projets qui ont passé par toutes les phases de la méthodologie (de la définition des besoins à l'implantation) et qui ont livré un produit au client
2	Tout le projet a été fait selon la méthode de gestion de projet avec des définitions de tâches et des processus de gestion consistants
3	Toutes les heures de travail, tant d'IBM que du client, devaient être connues et avoir été mesurées avec soin
4	Les facteurs fonctionnels devaient être connus

Il est fait également référence aux points de fonction comme un nombre "sans dimension". Il existe toutefois une certaine contradiction à exprimer la "taille" d'une application à l'aide d'un nombre "sans dimension". Dans ce contexte il y a alors lieu de se poser des questions sur la vraie signification des points de fonction en tant que concept, et d'en examiner la sémantique du point de vue des systèmes de mesure.

Il est important de bien connaître le contexte de référence dans lequel une fonction type est développé. Si ce qui compose le contexte de référence est connu, il est alors possible d'une part de l'analyser et d'autre part, en définir (par rapport à lui) les nouveaux contextes: c'est ce qui fait qu'une fonction type devient transportable d'un contexte à l'autre. Dans un tel cas, si on change de contexte, la transformation devrait être linéaire. Dans un tel contexte, il serait peut-être plus approprié de suggérer la définition suivante (Tableau 3.16) qui sera utilisée tout au long de cet ouvrage:

Tableau 3.16 : Définition d'un point de fonction

Un point de fonction est une mesure de l'effort nécessaire pour développer "la fonction type" dans le contexte de référence.

Il est également nécessaire de préciser et de clarifier dans quel sens certains termes et expressions sont utilisés dans cette définition (Voir Tableau 3.17).

Tableau 3.17 : Signification des termes de la définition d'un point de fonction

Expression	Description
Mesure	Méthodologie qui respecte les concepts de la théorie de la mesure
Développer	Par développer, il faut entendre tout aussi bien créer ou modifier.
Fonction type	Il s'agit d'un étalon pour le système de mesure
Contexte de référence	Caractéristiques reliées à la fois au type d'application développée de même qu'à l'environnement et aux conditions de développement

Il est à noter que cette définition d'un point de fonction diffère sensiblement de la perspective usuelle qu'un point de fonction n'a pas de signification. Il est également à noter qu'il n'y a pas de référence à la vision usager, ni à la valeur d'une fonction ou d'un point pour l'usager. En fait, ni la vision de l'usager, ni la valeur pour l'usager ne sont des notions qui ont été clairement définies, et encore moins quantifiables. Ces visions, quoique intéressantes, ne

sont pas actuellement utilisables dans un contexte de mesure quantitative. Par contre, la notion d'effort nécessaire pour développer la fonction type, est déjà bien comprise et intégrée dans un système de mesure du temps des ressources, et donc mesurable sans problème conceptuel. La majorité des recherches ont d'ailleurs utilisé ces notions de temps comme argument central de recherches, et même de base, des systèmes de mesure de la taille d'un logiciel. Bien qu'elle ne soit pas explicite dans la méthode d'Albrecht, est elle néanmoins présente dans la création des poids. Quant à la notion de valeur, elle n'est pas définie de façon explicite dans la métrique. Toutefois l'approche d'Albrecht 1979 la définit implicitement:

- To measure productivity we had to measure a product and a cost...
- The product that was analyzed was function value delivered...
- The objective was to develop a function value delivered to the user that was independent of the particular technology or approach used.
- We found that the basic value of the application function was consistently proportional to a weighted count of the number of external user inputs, outputs, inquiries and master files.
- These count are weighted by numbers designed to reflect the function value to the customer.
...
- The cost used was the work-hours.

L'effort, exprimé en termes d'heures de travail, constitue donc la base étalon d'une fonction type dans le cadre de référence, ce qui permet alors d'établir des comparaisons de productivité entre différentes technologies (COBOL, PL/1 et ASSEMBLER) et sur la période d'étude (1974 à 1979). Il est à noter que ce cadre est consistant avec les concepts économiques de productivité qui utilisent la productivité par heure de travail comme base de comparaison historique de l'évolution de la productivité.

3.4 Sommaire

Il a été vu dans les sections précédentes que la méthodologie de la métrique des points de fonction a évolué de 1979 à 1990 (Tableau 3.18) et que, progressivement, les règles ont été précisées de façon à rendre plus objectif le processus de mesure. D'une part les versions d'Albrecht 79 à IFPUG 85 ont extensionné la structure elle-même de la métrique pour permettre de tenir compte de la complexité relative entre les diverses fonctions et de distinguer entre des fonctions (*Simple, Moyen et Complexe*) par l'attribution de poids correspondants attribués à partir du comptage des composantes primaires de chaque fonction. Quant aux versions IFPUG, elles n'ont aucunement touché à la structure de la métrique, mais uniquement à la précision et clarification des règles d'identifications des composantes primaires et des critères. Ceci permet alors d'éliminer beaucoup de subjectivité dans l'application des règles.

Tableau 3.18 : Listes des versions officielles

	Version
1	Albrecht 79
2	Albrecht 84
3	GUIDE 1985
4	IFPUG 86
5	IFPUG 88
6	IFPUG 90
7	IFPUG E/R*

* IFPUG E/R*: Version en cours de discussion (Pas encore officielle).

Dans la revue de littérature des résultats des recherches expérimentales, il est donc important d'identifier la version utilisée par les auteurs: ceci permet d'interpréter et d'évaluer la pertinence actuelle des expérimentations et de leurs résultats. Il doit en être de même pour les praticiens qui désirent comparer la productivité interne de leur processus de développement par rapport à l'externe: il faut non seulement comparer les résultats obtenus, mais également la version utilisée.

Chapitre 4: LE MODELE DU PROCESSUS DE MESURE

4.1 Le modèle

Dans ce chapitre la relation entre la métrique des points de fonction et les modèles de mesure est explorée. Le besoin d'analyser la relation entre les modèles de logiciels et les métriques a été soulevé par Tate et al. (1991):

> "*In a sense all metrics relate to some, perhaps implicit model... The relationship between models and metrics is not a one-way street, however. Not only metrics require models. Models, unless they are to be purely conceptual, require metrics to give them quantitative meaning and to make them useful for accounting and management purposes.*"

La perspective de ce travail, à cette étape-ci, ne sera pas de s'attaquer à clarifier ce problème par rapport à son champ d'application, le domaine du produit logiciel, mais plutôt d'examiner la métrique dans son état actuel et de la modéliser afin d'en mieux analyser la pertinence et la validité intrinsèque en tant qu'instrument de mesure.

Le niveau le plus général de cette métrique a déjà été décrit sous une forme de modèle par Symons (1988), mais les autres niveaux de ce processus de mesure n'ont pas été modélisés. Comme vu dans les sections précédentes, la métrique est décrite surtout en termes de définitions, règles, tables de décision, guides d'interprétation, règles et exemples, et il n'y a pas de modèle détaillé de cette métrique. Les modèles qui suivent sont proposés comme outils d'analyse des points de fonction en tant que système de mesure. L'objectif de ces modèles est d'identifier correctement les différents processus inclus dans cette métrique, ce qui permettra dans une première étape d'en identifier les faiblesses, et dans des étapes ultérieures de proposer des avenues de recherche afin de palier à celles-ci ou de les interpréter et valider/infirmer à l'aide de concepts supplémentaires.

Le niveau général (Figure 4.1) comprend deux composantes principales: la taille fonctionnelle (*Functional Size*) et la valeur du facteur d'ajustement (*Value Adjustment Factor*). D'une part, la taille fonctionnelle est calculée à partir du pointage de toutes les fonctions individuelles de l'application mesurée, tandis que la valeur du facteur d'ajustement est calculée à partir de l'addition des valeurs attribuées aux caractéristiques générales de l'application dans son ensemble (ou sa totalité). Le premier résultat représente l'addition des parties du tout et il y est référée dans la littérature comme les points de fonction non-ajustés (*Unadjusted Function Points*, ou parfois encore comme *Function Counts*), tandis que le résultat de la multiplication de la somme des parties par la valeur du facteur d'ajustement du tout donne le pointage final en points de fonction (*Function Points*, *Adjusted Size* ou *Adjusted Function Points*). Ce niveau de la métrique sera intitulé le "Modèle de mesure des points de fonction" (Function Point Measurement Model).

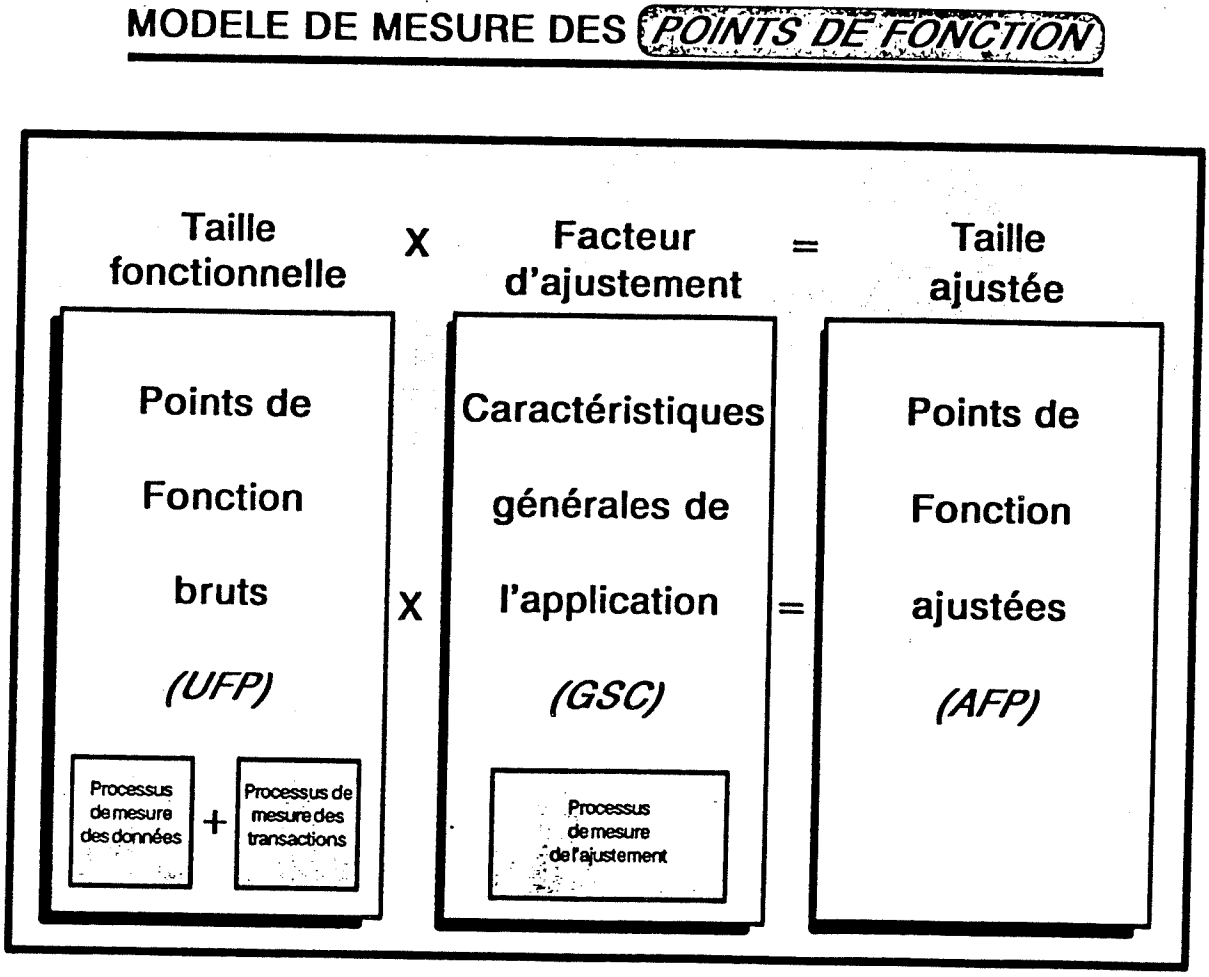

Figure 4.1: Le modèle de mesure des points de fonction

Le deuxième niveau du modèle examine le processus lui-même de la mesure des points des fonctions individuelles, ou la taille fonctionnelle mesurée en points de fonction "bruts", ou "non ajustés". Ce processus se décompose en deux opérations distinctes et successives, le processus de mesure des données et le processus de mesure des transactions de l'application. L'addition des résultats des deux mesures donnent les points de fonction bruts (Figure 4.1).

Le pré-requis à toute utilisation ou analyse de la métrique est la sélection de l'approche de base à utiliser pour l'identification des données primaires, par exemple Albrecht 79, Albrecht 84 et IFPUG 1990, cette dernière étant la version officielle. Il est à noter que cette sélection influence non seulement les règles d'identification des composantes primaires, mais également celles de l'identification des fonctions de même que des tables de poids à utiliser pour le calcul des points de fonction. Pour des fins de consistance, l'approche officielle IFPUG 90 est utilisée tout au long de ce document.

4.2 Le processus de mesure des données

Le processus de mesure des données se décompose lui-même en cinq étapes Fi, tel qu'illustré à la Figure 4.2. Les intrants de chacune des étapes sont indiqués dans la colonne de gauche, tandis que les extrants de chacune sont dans la colonne de droite. Si un extrant d'une étape Fi est ré-utilisée en tant qu'intrant à l'étape F(i+1), cela est indiqué par une flèche en pointillé vers l'étape suivante F(i+1).

F1) La première étape utilise la documentation de l'application ou du projet et, à partir de celle-ci et de la version de la métrique utilisée, identifie les groupes logiques de données.

F2) Cette étape ré-examine la documentation de projet et détermine les frontières entre l'application, ou projet mesuré, et les autres applications externes de façon à classifier les fichiers logiques soit en fichiers internes, soit en fichiers externes (interfaces).

F3) Dans la quatrième étapes chaque type d'éléments de données (DET) et chaque type d'enregistrements logiques (RET) est identifié et compté pour chaque fichier logique.

F4) Dans la cinquième étape l'algorithme des points de fonction est appliqué en utilisant les cinq types d'entrées suivantes: les sous-listes de fichiers, le comptage des données, les comptages des enregistrements, les tables de décision des niveaux de complexité et les tables de poids pour les fichiers logiques.

F5) La dernière étape consiste en l'addition de tous les points calculés pour chaque fichier interne et d'interface. Le résultat donne des points bruts provenant des fichiers internes et des interfaces.

Il est à remarquer que ce processus de mesure des données fait appel à plusieurs modèles implicites (c'est-à-dire à des relations *intuitives* entre des objets différents), tels les modèles de la documentation, les frontières des applications, les structures de données, la complexité et le poids des données. Aucune des relations de ces modèles ne sont toutefois basées sur des théories prouvées expérimentalement dans des cadres précis.

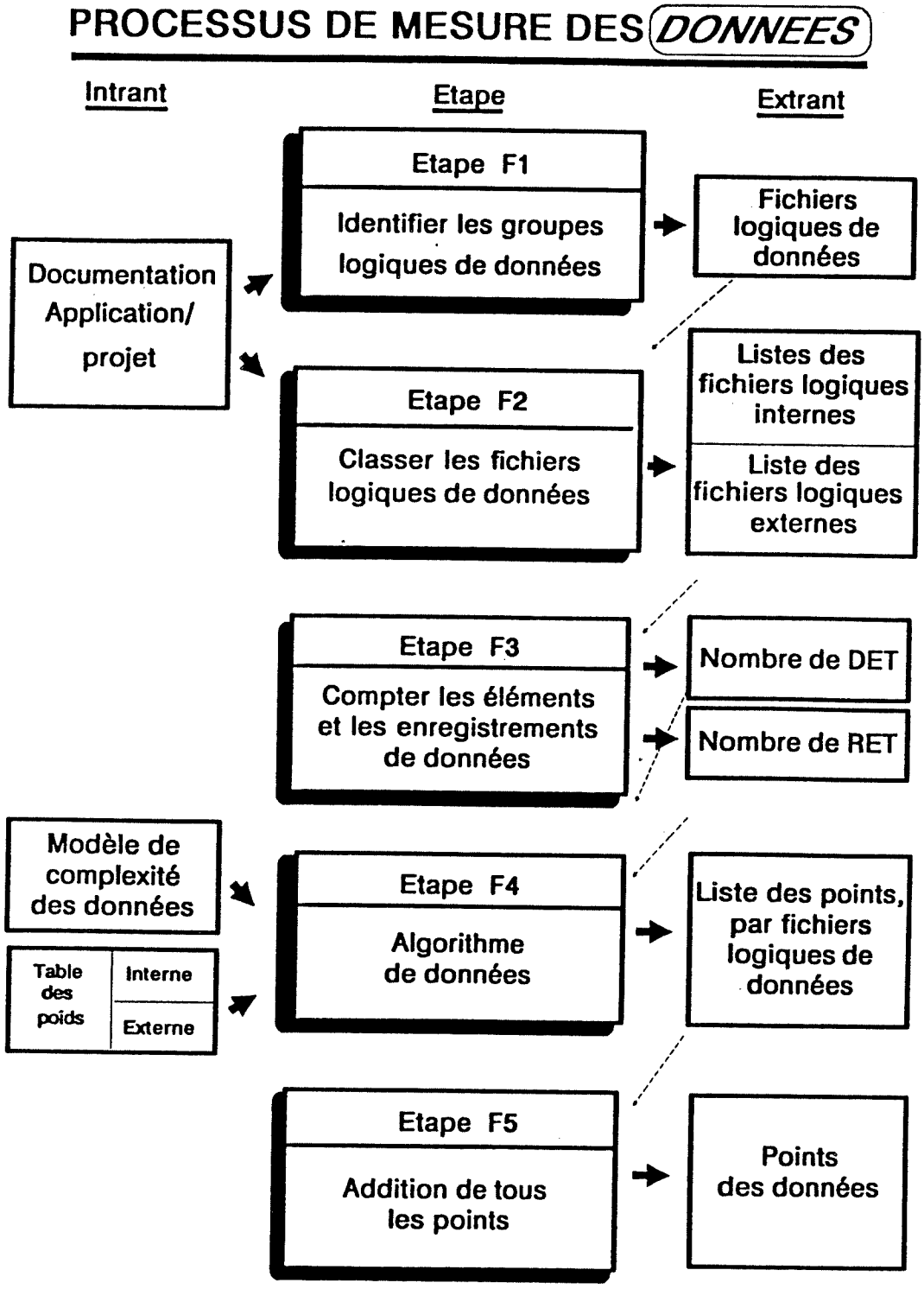

Figure 4.2 : Le processus de mesure des données

4.3 Le processus de mesure des transactions

Le processus de mesure des transactions se décompose lui aussi en plusieurs étapes Ti, tel qu'illustré à la Figure 4.3. Les intrants de chacune des étapes sont indiqués dans la colonne de gauche, tandis que les extrants de chacune sont dans la colonne de droite. Si un extrant d'une étape Ti, est ré-utilisé en tant qu'intrant à l'étape suivante T(i+1), cela est indiqué par une flèche en pointillé vers l'étape suivante T(i+1).

T1) La première étape en intrant la documentation de l'application ou du projet et, à partir de celle-ci et de la version de la métrique, identifie les fonctions de transactions visibles à l'usager. Ceci produit les listes des entrées, des sorties et des requêtes.

T2) La deuxième étape reprend les listes de fonctions précédentes et y applique le modèle des transactions de la méthodologies des points de fonction pour produire les liste de transaction à compter. Il est à noter qu'avec ce modèle une fonction d'entrée peut produire jusqu'à trois transactions chacune, et une fonction de requêtes, nécessairement deux transactions (exemple: la transaction d'entrée et la transaction de sortie d'une requête).

T3) Dans la troisième étape les listes des fichiers logiques et des types d'éléments de données produites aux étapes F2) et F3) sont utilisées pour identifier et compter les fichiers référencés et les éléments contenus dans chaque transaction.

T4-A) La quatrième sous-étape applique l'algorithme des points de fonction avec les intrants suivantes: les listes de transactions, les fichiers comptés, les éléments comptés, la formule de complexité fonctionnelle et les poids des transactions.

T4-B) Dans cette sous-étape, la valeur la plus petite de chaque fonction de requêtes est éliminée (soit la transaction d'entrée de la requête, soit celle de la sortie).

T5) La dernière étape consiste en l'addition de tous les points calculés pour chaque transaction de l'étape T4). Le résultat donne les points bruts provenant des transactions.

L'addition des points de données provenant de F5) et de ceux des transactions provenant de T5) donne alors le total des points de fonction bruts pour produire la taille fonctionnelle (*Functional Size*) de l'application, tel qu'illustré dans la Figure 4.1.

Il est à remarquer qu'ici aussi le processus de mesure des données fait appel à plusieurs modèles implicites (c'est-à-dire à des relations *intuitives* entre des objets différents), tels les modèles de la documentation, la sous-division des fonctions en transactions (add., modif., retrait, sélection, affichage), la complexité et le poids des transactions, la sélections des poids pour les transactions de requêtes, etc. Aucune des relations de ces modèles ne sont toutefois basées sur des théories prouvées expérimentalement dans des cadres précis.

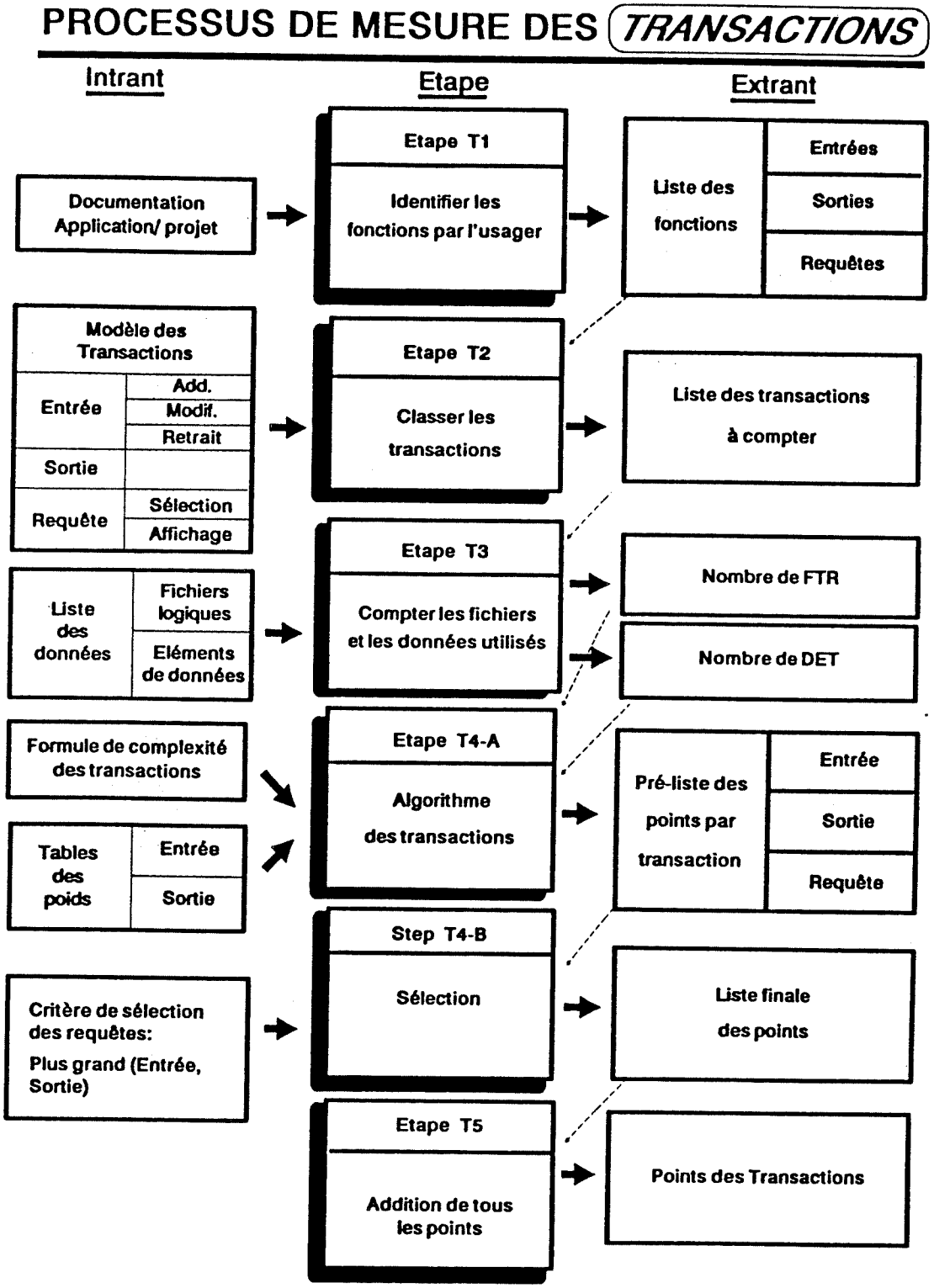

Figure 4.3: Le processus de mesure des transactions

4.4 Le processus de mesure de l'ajustement

Le facteur d'ajustement détermine la complexité du traitement et de l'environnement de l'application. La valeur de ce facteur est déterminée par le processus de mesure modélisé à la Figure 4.4.

V1) La première étape prend en intrant la documentation de projet de même que les définitions de chacune des quatorze caractéristiques générales (*General System Characteristics - GSC*) de l'application. Dépendamment de la spécificité de chaque caractéristique, ce sous-processus produit un ou plusieurs critères.

V2) La deuxième étape assigne ensuite un ordre croissant, c'est-à-dire un classement, de zéro à cinq à chaque caractéristique en se basant sur la table de décision propre à chacune.

V3) Dans cette étape, une valeur (D^i = *Degree of Influence for factor i*) est assignée à chaque classement de l'étape précédente. Il y a actuellement dans la version IFPUG 90 une fonction d'équivalence entre le classement et la valeur attribuée (de zéro à cinq).

V4) La quatrième étape consiste en l'addition de chacune des valeurs pour les quatorze facteurs (*Total Degree of Influence* = *N*).

V5) La dernière étape consiste dans l'application de la formule du degré total d'influence:

$$(0.65 + 0.01 * (N))$$

Cette formule permet alors d'appliquer un facteur qui permet d'ajuster la taille brute d'une application avec un impact maximum de +/- 35%.

Ici encore le processus de mesure de l'ajustement fait appel à plusieurs modèles implicites, tels le modèle et le domaine des ajustements, la structure et les classes de critères de même que les tables d'équivalence des degrés d'influence et les modèles de leur addition et de l'algorithme. Aucune des relations de ces modèles ne sont toutefois basées sur des théories prouvées expérimentalement dans des cadres précis.

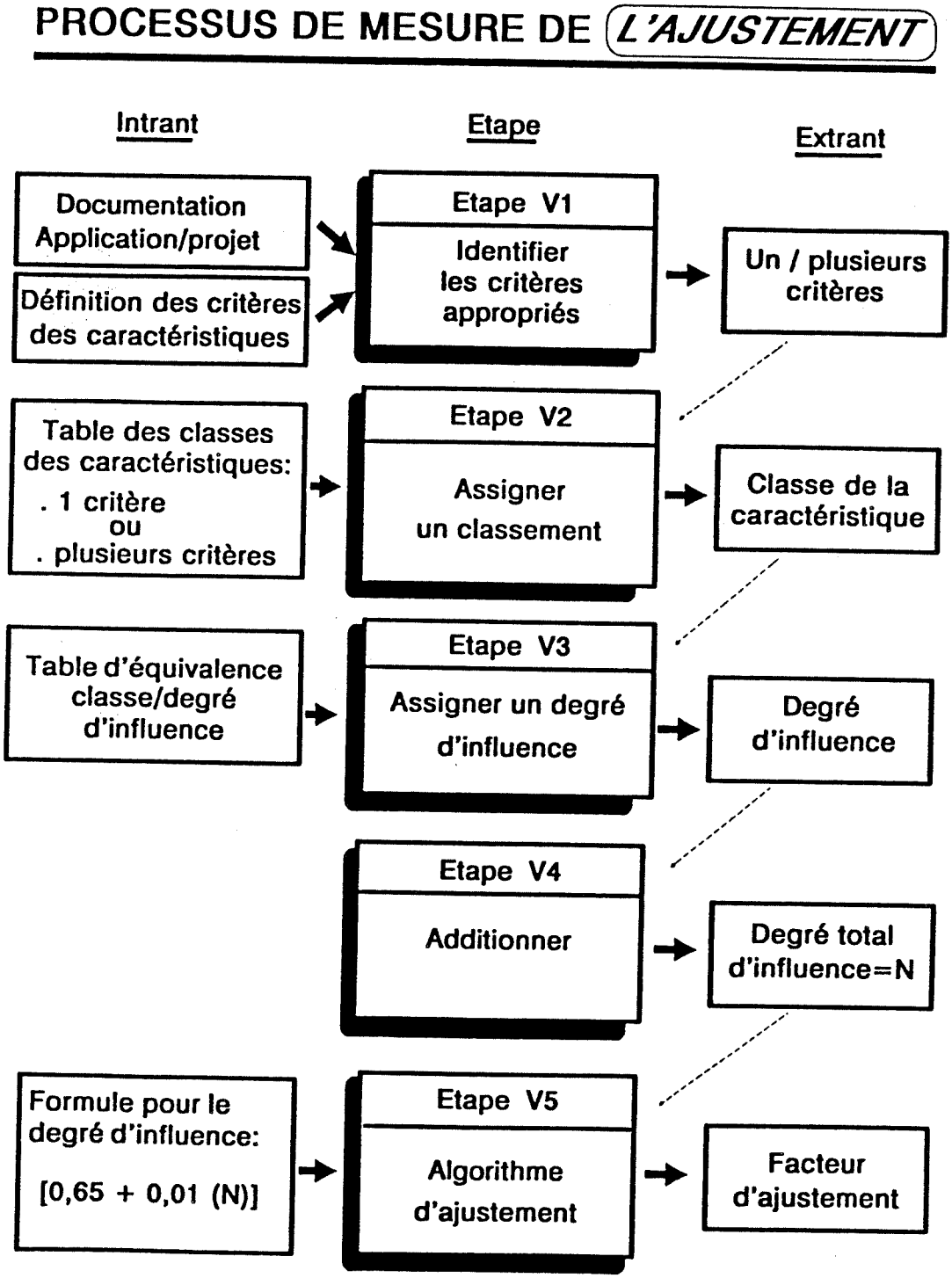

Figure 4.4 : Le processus de mesure de l'ajustement

4.5 Sommaire

Dans ce chapitre, la technique de modélisation à été utilisée afin de rendre plus explicite le processus de mesure intégré à la métrique des points de fonction: trois principaux sous-modèles de mesure ont été dégagés, à savoir les sous-modèles de mesure des données, de mesure des transactions et de mesure de l'ajustement. A l'intérieur de chacun de ces sous-modèles, les diverses étapes ont également été illustrées.

Il a été également noté que cette métrique fait appel à des modèles implicites du produit logiciel à de multiples niveaux (fonctions, données, transactions et facteurs d'ajustement) mais que ces modèles sont implicites. Quoiqu'il soit très important d'examiner ces modèles implicites pour étendre le champ d'application de la métrique des points de fonction en dehors de son domaine initial d'application des applications M.I.S., ce volet de recherche ne sera pas approfondi au cours de cette recherche. Ceci permet alors de se concentrer plutôt sur l'analyse du processus de mesure lui-même.

Chapitre 5: RECHERCHES EXPÉRIMENTALES: FIABILITÉ DE LA MÉTRIQUE

5.1 Classification des recherches expérimentales

Une classification des recherches expérimentales est proposée dans le tableau 5.1. On y retrouve dans la colonne 1 les cinq classes: la fiabilité de la mesure, l'analyse de la productivité, la fiabilité de l'estimation, les métriques dérivées et l'analyse du processus de mesure lui-même. La deuxième colonne du tableau indique des sous-classes tandis que la colonne de droite indique les auteurs qui ont traité de ces sujets. Ce chapitre ne traite que de la première classe de recherches expérimentales, la fiabilité de la métrique. Les recherches sur l'utilisation de la métrique pour les modèles de productivité et d'estimation sont présentées au chapitre 6 et les recherches sur les métriques dérivées et sur le processus de mesure lui-même de la métrique sont analysées au chapitre 7.

Tableau 5.1 : Classification des recherches sur les points de fonction

Classes	Sous-Classes	Auteurs
1- Fiabilité de la mesure	- Répétitivité - Exactitude	- Rudolph 83 & 89 - Low & Jeffery 90 - Kemerer 90 - Whisehunt 90
2- Analyse de productivité	- Gains de productivité - Économie d'échelle - Comparaison inter-industries - Facteurs de productivité	- Albrecht et al. 83 - Banker et al. 89 - Behrens 83 - Emrick 88 - Saindon 89
3- Fiabilité du processus d'estimation	- Comparaison avec autres modèles - Comparaison avec Lignes de Code	- Kemerer 87 - Albrecht et al. 83 - Informetrics 88 - Saindon 89 - Desharnais 88
4- Métriques dérivées	- Modèle simplifié - Feature Points - MARK II - Temps réels	- Bock 90 - Jones 88 - Symons 88 - Reifer 91
5- Analyse du processus de mesure	- Évaluation subjective - Méthodologie E/R - Échelle de Mesure - Modèle global	- Navlaka 86 - Desharnais 88 - Symons 88 - Abran 91

5.2 Analyse de la fiabilité de la mesure

5.2.1 Design expérimental

Une bonne métrique doit posséder parmi ses attributs les caractéristiques suivantes (Navlaka, 1986): exactitude et répétitivité. En effet, à partir des mêmes données et règles, on doit obtenir les mêmes résultats (exactitude), quelles que soient les personnes qui font les mesures, et ces mesures doivent donner les mêmes résultats dans le temps (répétitivité). Ces phénomènes sont examinés dans les deux sous-sections qui suivent, mais avant il est nécessaire de comparer les différents designs expérimentaux utilisés pour ces recherches.

Trois auteurs (Rudolph, 1983 & 1989; Low et al. 1990; Kemerer, 1990) ont effectué des recherches expérimentales pour mesurer l'ampleur de la variation des mesures et pour identifier et analyser l'impact des sources potentielles de différence. Le tableau 5.2 présente un sommaire du design expérimental de ces recherches.

Tableau 5.2 : Analyses de fiabilité - Design expérimental

Auteurs	Design expérimental
Rudolph	- Mêmes applications - Même méthodologie contrôlée - Multiples sujets (individus)
Low	- Mêmes applications - Méthodologie non contrôlée - Multiples sujets
Kemerer	Design expérimental 1: - Différentes applications - Même guide de méthodologie IFPUG 90 - Pour chaque application, mesure par deux sujets
	Design expérimental 2: - Différentes applications - Même guide de méthodologie E/R* Pour chaque application, mesure par deux sujets.
	Design expérimental 3: - Mêmes applications pairées - Mêmes guides de méthodologies - Différentes méthodologies (IFPUG 90 & IFPUG E/R*) - Pour chaque application, mesure par deux sujets.

Quant au tableau suivant (Tableau 5.3) il présente, pour chacun des trois auteurs, une comparaison d'ensemble des principales caractéristiques des conditions d'expérimentation et de la démographie des échantillons de projets étudiés: les sujets qui ont pris les mesures (expérience et formation), les caractéristiques démographiques des applications mesurées (nombre et taille

des applications mesurées) , les mécanismes de contrôle des mesures (méthodologie de pointage, base de référence, de comparaison et de validation), les hypothèses et méthodes statistiques et finalement les résultats des expérimentations en fonction des variations notées en expérimentation.

Tableau 5.3 : Caractéristiques des recherches expérimentales de fiabilité

Caractéristiques	Rudolph 89	Low 90	Kemerer 90
Sujets & niveau d'expérience	Analystes expérimentés (USA,Asie, Europe,...)	- 22 analystes - 7 organisations	De 2 à 4 analystes dans plusieurs organisations
Formation	Contrôlée: 3 jours de formation par le même formateur	20 individus formés par les chercheurs	Manuel d'instruction et guides de pointages fournis par les chercheurs
Nombre d'applications mesurées et de pointages	9 applications et 373 pointages	- 2 programmes - 22 pointages chacun	27 applications 90 pointages
Taille des applications	Min: 90 Points Max: 723 Points Moyenne:	Min: 40 points Max: 58 points Moyenne:	Min: Non disponible Max: Non disponible Moyenne: 450
Méthodologie utilisée	Rudolph 89: IFPUG 88 (Rudolph 83: Albrecht 79)	Non précisée (Non contrôlée)	IFPUG 90 (Contrôlée partiellement)
Base de référence pour l'analyse des comptages	La valeur suggérée pour les points bruts	Non spécifiée	Aucune
Base de comparaison	Par rapport à la référence validée	Par rapport à la moyenne	Comptage pairé indépendant *(Average Relative Error = ARE)*
Validation du pointage	Oui	Non	Non
Hypothèses pour les statistiques	Aucune	Aucune	Indépendance et termes d'erreurs non biaisés
Méthodes statistiques	- *Average judgement* - Moyenne, Écart-type et différences des moyennes	- *Average judgement* - Moyenne, Écart-type et différences des moyennes	- Tests statistiques H_0 et H_1 - *Average Relative Error (ARE)*
Variation de pointage - stade planification - stade post- implantation	Planif.: 20% Post : 10%	Pas précisé	Pas de différence statistique
Variations inter-pointeurs	Albrecht 79: 30% Albrecht 88: 10%	30%	IFPUG 90: ARE = 10.78% IFPUG E\R*: ARE = 17.30% Inter-métho.: ARE = 8.48%

5.2.2 La répétitivité

Les méthodologies de recherche expérimentale utilisées, de même que les résultats, sont commentés en fonction des deux dimensions suivantes de la fiabilité de la mesure: la répétitivité et l'exactitude. Cette sous-section traite uniquement de la répétitivité alors que la section suivante traite de l'exactitude des pointages.

Le tableau suivant (Tableau 5.4) indique les sources de différences inter-pointeurs qui ont été examinées pour trois groupes de recherches (un X dans une colonne indique que le chercheur a noté l'influence de cette caractéristique).

Tableau 5.4 : Sources de variations inter-pointeurs

	Sources de différences	Rudolph 89	Low 90	Kemerer 91
1	Sujets	x	x	
2	Organisations	x	x	
3	Formation/expérience en pointage	x	x	x
4	Expérience professionnelle	x	x	x
5	Connaissance de l'application	x	x	x
6	Niveau de la documentation	x	x	x
7	Régional (pays)	x		

Suite à son expérimentation, Rudolph (1989) indique que dans de bonnes conditions de pointage l'erreur présumée de mesure (CV = coefficient de variation = écart-type * 100 / moyenne) est de 10% (Tableau 5.3, dernière ligne). Selon Rudolph, les variations trouvées sont considérées comme acceptables avec une marge inférieure de plus ou moins de 10% pour les applications complétées et avec une marge de plus ou moins de 20% pour celles qui sont encore au stade de la planification et dont la documentation n'est pas assez précise. Rudolph compare ce résultat de 20% par rapport à un résultat d'expérimentation d'une variation de 30% par rapport à la moyenne de la taille pour des application au stade de la planification. Il explique également l'amélioration de la fiabilité de la méthode par la différence des versions de méthodologie utilisées, son expérimentation de 83 utilisant la version Albrecht 79, laquelle est considérée moins objective que la version officielle la plus courante, telle que documentée de façon beaucoup plus précise par IFPUG 90.

Il est à noter que les résultats de Low et al. (1990), qui semblent utiliser eux aussi une

version pré-1984, rapportent une erreur de +/- 30%, ce qui est du même ordre de grandeur que Rudolph 83. Quant aux résultats de +/- 10.78% mentionnés par Kemerer (1991), ils concordent avec ceux de Rudolph 89, les deux recherches expérimentales utilisant à peu de choses près la même version de méthodologie de pointage.

Il faut toutefois souligner qu'il y a des différences considérables entre les méthodes d'analyse de Kemerer et celles des deux autres recherches. D'une part, Rudolph et Low présentent des observations à partir de jugement d'expert (*expert judgement*) sur l'impact de multiples sources potentielles de variations de pointage, tandis que Kemerer présentent des conclusions d'analyses statistiques sur un seul aspect à la fois: les variations inter-pointeurs avec pointage pairé. Son design expérimental lui permet aussi de vérifier l'ampleur des variations avec l'utilisation indépendante de deux méthodes de pointage, IFPUG 90 et l'approche Entité/Relation (IFPUG E/R*), qu'entre les deux méthodes, avec les résultats suivants pour les critères *Average Relative Error* (ARE):

Tableau 5.5 : Variations inter-pointeurs - Kemerer 90

Méthodologie de pointage	Variations inter-pointeurs
IFPUG 90	ARE = 10.78%
IFPUG E/R*	ARE = 17.30%
Inter-méthodes	ARE = 8.48%

Kemerer rapporte une erreur plus grande de 17.3% avec l'approche Entité/Relation. Il émet l'hypothèse que cette plus grande variation pourrait être due au fait que plusieurs des compteurs ne connaissaient pas l'approche Entités/Relation et que cela a pu leur causer des problèmes de compréhension du guide de pointage qui leur avait été transmis. Kemerer ne commente toutefois pas le fait que les variations inter-méthodes (IFPUG 90 versus IFPUG E/R*) sont plus petites de façon significative: 8.48%.

En résumé, les résultats de ces recherches sur la répétitivité des pointages par des individus différents concordent, tels qu'illustrés au tableau 5.6.

Tableau 5.6 : Variations inter-pointeurs - Sommaire des recherches

	Version de la métrique	Auteurs	écart
1	Albrecht 79 et Albrecht 84	Rudolph 83 et Low & Al. 90	+/- 30%
2	IFPUG 88 et IFPUG 90	Rudolph 90 et Kemerer 90	+/- 10%

5.2.3 L'exactitude

Trois recherches expérimentales ont présenté des observations sur l'exactitude des pointages, à savoir Rudolph (1989), Low et al. (1990) et Whisehunt (1990).

De par leur design expérimental à partir du pointage des mêmes applications avec des pointeurs différents, Rudolph et Low ont pu, d'une part, déterminer l'ampleur des variations au niveau du pointage global, d'autre part, identifier des sources de variations sans nécessairement quantifier les effets individuels de ces variations. Le Tableau 5.7 présente une liste des sources d'inexactitude identifiées par différents auteurs.

Tableau 5.7 : Sources d'inexactitude dans les pointages

Auteurs	Sources d'inexactitude
Rudolph	1- Plus d'erreurs de mesure pour des applications complexes 2- Non familiarité avec l'application 3- Manque d'attention au détail pour le pointage
Low & Jeffery	1- Perception de la complexité des fonctions 2- Des jugements de valeurs 3- Des différences d'interprétations des spécifications 4- La perception des frontières du projet 5- Les objectifs de celui qui compte 6- L'expérience de compter *a priori* 7- Variations organisationnelles 8- Variation immédiatement après la formation
Whisehunt	Les types d'erreurs relevées étaient les suivants: 1- Tous les fichiers avaient été comptés, et non pas seulement les fichiers à l'intérieur du domaine de l'application mesurée; 2- Il y avait des feuilles de travail manquantes ou incomplètes; 3- Des erreurs de calcul mathématiques; 4- Des nombres aux mauvais endroits sur les formulaires; 5- Mauvaises transcriptions de données; 6- Certains formulaires démontraient que la personne qui les avait remplis ne savait pas ce qu'elle faisait.

Toutefois, une analyse des observations de Low pour illustrer les cinq premiers points ci-dessus indiquerait que plusieurs des points soulevés par Low comme problèmes seraient maintenant réglés si la version IFPUG 90 avait été utilisée comme grille d'expérimentation et d'analyse plutôt que celle probablement utilisée (Albrecht 84) dans leur recherche, et que ces différences sont maintenant plutôt considérées comme des erreurs de pointage. Ceci est illustré dans les commentaires 1 à 5 du tableau 5.8. Quant aux sources d'erreurs notées par Whisehunt, elles confirment que le processus lui-même d'enregistrement des données de mesure est également une source d'erreurs (Tableau 5.7): un exercice de validation effectué sur ses données a révélé que pour 98 pointages sur une période de 4 mois plus de 54% des pointages avaient été mal faits et calculés.

Tableau 5.8 : Analyse critique des facteurs d'inexactitude de Low et al.

	Facteurs	Analyse critique des exemples de Low & Jeffery
1	Perception de la complexité des fonctions	Les règles IFPUG 90 donnent l'algorithme précis pour déterminer la complexité des fonctions, basée sur le nombre de types d'éléments et de fichiers accédés; c'est exactement le genre d'information que l'on retrouve dans les spécifications de programme. D'après les commentaires des auteurs, ces règles ne semblaient pas être connues par leurs sujets car ils ont assigné leurs points basés sur leur jugement de la complexité des fonctions, c'est-à-dire la version Albrecht 84 de la métrique.
2	Jugements de valeurs	Le cas suivant est classé comme un jugement de valeur: "Le programme doit générer 3 rapports différant uniquement par un ordre de tri". Low et al. mentionnent alors qu'un analyste a porté un jugement de valeur en considérant que les deux rapports identiques à un tri près ne devaient être comptés qu'une fois. Or il ne s'agit pas d'une erreur de jugement, mais au contraire d'une erreur de comptage par rapport aux règles décrites aux pages 19 et 20 du document officiel[8].
3	Différences d'interprétation des spécifications	Les cas rapportés comme des différences d'interprétations sont nettement des erreurs: par exemple l'analyste qui a compté les "control records et les trailer records", n'a pas fait une erreur d'interprétation: il a directement contrevenu à une règle décrite à la page 21 du document officiel.
4	Perception des frontières du projet	Les auteurs rapportent que 14 des 22 analystes ont établi les types des fichiers en fonction de l'application et non pas en fonction du programme mesuré. Ici encore il ne s'agit pas d'un problème de perception mais d'une erreur par rapport à la règle sur les frontières de l'application qui doivent être établies en fonction du produit du travail (*work product*) qui doit être mesuré (règle à la page 7 du document officiel).
5	Objectifs de celui qui compte	Le cas mentionné est celui de l'analyste qui n'a pas compté les rapports de mise-à-jour parce qu'il estimait qu'ils étaient simples et faisaient partie des entrées. Dans ce cas-ci également il s'agit d'une erreur de la part de l'analyste.
6	Expérience de compter *a priori*	Les auteurs mentionnent que tous les analystes avaient déjà compté des applications "a posteriori", mais que plusieurs ne les avaient jamais comptées "a priori" à partir uniquement des documents de spécifications.
7	Variations organisationnelles	Ils mentionnent que trois pointeurs provenaient de la même organisation (no. 2) qui ne supportait pas activement la métrique, qu'ils étaient responsables pour les valeurs maximum et qu'ils avaient les plus grandes variations dans leurs estimés (Pour le programme 1: min-max de 34-159 par rapport à des min-max de 40 à 73 pour les 4 autres organisations avec plus d'un analyste). En excluant les données de l'organisation 2, le pourcentage de l'écart-type par rapport à la moyenne s'améliore en diminuant de 46% à 22% (programme no. 1).
8	Variations immédiatement après la formation	Le tableau 6 de l'article indique que l'expérience dans le pointage a un effet significatif pour améliorer la justesse du pointage: le ratio des écarts-types par rapport à la médiane est de 22% pour les analystes qui ont de l'expérience en pointage (excluant l'organisation no. 2) par rapport à 42% pour ceux qui venaient de prendre le cours sur les points de fonction.

[8]IFPUG Function Point Counting Practices Manual, April 1990.

Il ne faut pas oublier que l'étude de Kemerer ne visait qu'à évaluer la variation globale du pointage à l'intérieur des mêmes organisations. Dans son design expérimental, il n'y avait pas de mécanisme de contrôle pour vérifier l'exactitude du pointage. En fait, le design expérimental de Rudolph semble le plus pertinent au niveau de l'observation de l'exactitude de la mesure; par contre sa méthodologie d'analyse est faible en termes d'utilisation des méthodes statistiques pour l'interprétation des résultats.

Il est à noter que la conclusion no. 8 de Low (Variations basées sur l'expérience, et sur le niveau de documentation disponible, Tableau 5.8) est contredite par une des conclusions statistiques de Kemerer (1990).

5.2.4 Niveaux intermédiaires de pointage

Il est à noter que toutes les recherches pré-citées se sont concentrées sur des analyses par rapport au résultat du pointage brut final. Il ne semble pas y avoir de publication sur l'analyse au niveau détaillé ou intermédiaire du pointage, ce qui laisse planer des incertitudes majeures sur la fiabilité de tout le processus, sachant qu'il existe une possibilité que le pointage final soit le résultat de pointages intermédiaires avec des erreurs de sur-pointages et de sous-pointages à effets contraires qui s'annulent dans le processus d'agrégation.

Il est d'ailleurs surprenant que cet argument soit utilisé, en particulier pour justifier l'utilisation des points de fonction que pour des applications de plus de 100 points. La position citée par Kemerer (1990, page 16) est que pour des petits pointages, ces erreurs de variation ont des effets relatifs trop importants: *It is suggested that the higher reliability score reflects the use of actual medium-sized systems, where small errors are less important, on a percentage basis, than they would be on counts of small programs.* Rudolph (1989), quand à lui, émet l'opinion contraire que plus la taille de l'application est grande, plus il y a des possibilités de faire des erreurs.

Il est à remarquer qu'en termes d'instrumentation de mesure, et en termes de leur pertinence comme instrument de recherche scientifique, il est très important que tous les niveaux du processus de mesure soient les plus précis afin d'analyser les effets de causalité. On ne peut pas partir scientifiquement d'une somme d'erreurs pour arriver à la vérité, et en espérer des conclusions pertinentes!

Afin d'illustrer l'impact potentiel de telles erreurs, voici un exemple issu d'une expérimentation connexe au projet de Kemerer. Le Montréal Trust a fourni des données de recherches expérimentales, et cela en suivant de façon très exacte les contrôles expérimentaux tels que précisés par ce chercheur en fonction de son design expérimental. Toutefois des contrôles additionnels d'expérimentation ont été ajoutés au Montréal Trust, lesquels contrôles sont nécessaires pour les analyses au niveau de détail. En fait, des contrôles de la qualité de la mesure ont été effectués sur les données de recherche transmises au M.I.T. par le Montréal Trust. Il est à noter que ces contrôles ont été faits de façon indépendante afin de ne pas affecter

la fiabilité expérimentale des données transmises au M.I.T.

Dans les données d'expérimentation fournies par le Montréal Trust, deux applications ont été mesurées, chacune avec les deux méthodologies proposées. Dans le cas de la première application de 109 points, la différence entre les deux méthodes a été de 9% seulement, et aucune contre-validation n'a été effectuée. Dans le cas de la deuxième application, la différence de comptage a été majeure:

- 391 points avec la méthode officielle IFPUG 90
- 273 points avec la méthode E\R.

La contre-validation expérimentale a été effectuée par un expert reconnu de l'industrie afin d'établir une base de référence et d'analyser le pourquoi des différences. Cet expert a établi la base de référence à 311 points et, après analyses, les variations ont été expliquées par les deux causes qui suivent:

1- Le 25% de différence entre la méthode E/R et la version IFPUG 90 dans le cas étudié provient des façons différentes de compter les sorties et les requêtes:

1.1 Sorties: les messages d'erreur sont comptés comme des Sorties dans la version officielle, mais ne sont pas comptés dans la version E/R;

1.2 Requêtes: les message d'aide (Help) sont comptés comme des requêtes dans la méthode officielle, mais ne sont pas comptés dans le modèle E/R.

2- Quant à la différence de 38 points entre le comptage du Montréal Trust avec la méthode E/R, par rapport au comptage de référence effectué directement par l'expert, cela provient d'une erreur de comptage du Montréal Trust, Cette erreur corrigée, la différence n'est plus alors que de 3% par rapport à la référence exacte telle qu'établie par l'expert.

Ces contrôles de qualité avec une différence de seulement 3% est un exemple que l'erreur moyenne relative de 17.3% du M.I.T. pour la méthode E/R peut comprendre non seulement des différences entre les méthodes mais potentiellement, et peut-être même surtout, des erreurs dans le comptage, telles que mentionnées dans les études de Low, Whisehunt et par Kemerer lui-même comme explication potentielle de cet écart. Donc ce ne serait pas nécessairement des imprécisions méthodologiques.

Il s'agit donc principalement d'un problème de disponibilité d'instrumentation de mesure approprié qui permettrait d'augmenter considérablement la qualité du processus de mesure. Whisehunt souligne de plus certains problèmes associés aux erreurs d'enregistrements des données de mesure. Toutefois Whisehunt (1990) indique également qu'avec la mise sur pied de mécanismes de validation des mesure, le taux d'exactitude et de vérifiabilité des pointages s'élève alors à +/- 5%.

5.3 Sommaire sur les analyses de fiabilité

Des recherches citées, on peut retenir les observations suivantes:

1- Rudolph et Kemerer ont démontré qu'en utilisant les versions IFPUG (1988 ou 1990), les marges d'erreurs ou les variations inter-pointeurs n'étaient que de 10%.

2- Les recherches de Low et Whisehunt ont démontré qu'en faisant des erreurs, on arrive à des résultats différents, quoi de plus normal!

3- Le mérite de l'article de Low et al. a été de démontrer qu'en dépit de toutes les erreurs de comptage commises par les analystes qui ont participé à l'expérience, la marge d'erreur (ratio des écarts-types par rapport à la médiane) en utilisant l'ancienne version de la métrique (Albrecht - 79 ou 83) n'était que de 30%, résultat consistant avec les recherches de Rudolph (1983), dans des circonstances et une méthodologie de recherche similaire.

4- Kemerer, Low et Whisehunt ont démontré qu'il y avait un sérieux besoin de formation chez les organisations qui avaient déjà implanté les points de fonction.

5- Low a illustré le fait que ceux qui venaient de prendre le cours de formation avaient encore besoin de pratique avant de pouvoir appliquer la méthode correctement.

L'analyse des résultats de ces recherches expérimentales porte à croire que les différences dans le comptage des points de fonction ne proviennent pas tant de règles ambiguës et subjectives de la métrique version IFPUG 90, mais plutôt des causes suivantes:

- une méconnaissance des règles par les analystes;
- des organisations qui ne fonctionnent pas à partir des même règles officielles telles que celles adoptées par l'IFPUG.
- un manque de rigueur dans l'application du processus de mesure
- un manque de rigueur dans le mécanisme d'enregistrement des données de mesure.

Enfin un problème fondamental a été soulevé dans la dernière sous-section, à savoir l'absence de recherches expérimentales sur la fiabilité des mesures prises à chacune des étapes intermédiaires ce qui laisse planer une incertitude sur la fiabilité de tout le processus, sachant qu'il existe une possibilité que le pointage final soit le résultat de pointages intermédiaires avec des erreurs de sur-pointages et de sous-pointages à effets contraires qui s'annulent dans le processus d'agrégation.

Chapitre 6: RECHERCHES EXPÉRIMENTALES: Productivité et estimation

6.1 Caractéristiques des recherches expérimentales

Plusieurs chercheurs ont utilisé la métrique des points de fonction comme design expérimental pour analyser la productivité du processus de développement informatique: Albrecht (1983), Banker et al. (1989), Behrens (1983), Emrick (1988), Desharnais (1988), Informetric (1988), Kemerer 87, Saindon (1989). Le tableau 6.2 de la page qui suit présente les principales caractéristiques des ensembles de données expérimentales utilisés pour les analyses de productivité basées sur la métrique des points de Fonction.

L'analyse de ce tableau donne un aperçu des difficultés à comparer les résultats de ces diverses études car il y a relativement peu d'homogénéité entre les divers ensembles de données. En fait, il ne semble y avoir d'homogénéité qu'au niveau du domaine des applications mesurées, c'est-à-dire le domaine des application M.I.S. En ce qui concerne toutes les autres caractéristiques, il y a une disparité considérable.

Par exemple au niveau des méthodes de pointages, il y a tous les choix et combinaisons possibles, partant de la méthodologie originale d'Albrecht 79 à la méthode proposée IFPUG E/R*, entre les ensembles des données entre-eux, et ensuite à l'intérieur même des données pour les études inter-industries (Emrick et Desharnais). Il en est de même pour les variations de la taille des applications mesurées, des coûts de projets et des coûts unitaires et le tableau 6.1 présente un aperçu de l'étendue des variations (minimum, maximum et moyenne).

Tableau 6.1 : Sommaire des variations de taille et de coûts

	Caractéristiques	Minimum	Maximum	Moyenne
1	Taille (Pointage)	4 P.F.	10,857 P.F.	174 à 997 P.F.
2	Coûts de projet	100 heures	105,200 heures	non disponible
3	Coûts unitaires	0.2 Hr\P.F.	145.5 Hr\P.F.	non disponible

Il est également à noter que les recherches publiées en 1983 portaient sur des environnements homogènes d'une seule compagnie ou organisation, tandis que celles postérieures à 1983 comprennent des données provenant de plusieurs organisations. En termes de design expérimental, ceci impose alors des contraintes de disponibilité et de pertinence des données expérimentales disponibles: à l'intérieur d'un même organisme, il n'y a pas suffisamment de données pour des analyses statistiques portant sur plusieurs variables de productivité, tandis que pour les recherches inter-compagnies, il n'y pas suffisamment d'homogénéité dans les environnements et dans le processus de collecte des données.

Tableau 6.2 : Analyses de productivité avec les points de fonction - Caractéristiques des ensembles de données expérimentales

Caractéristiques	Albrecht 83	Behrens 83	Kemerer 87	Emrick 88	Desharnais 88	Informetric 88	Saindon 89	Abran 91
Pointeurs	Homogène			Non Homo.	50% Homo.		Non homo.	Homogène
Années des données	74 à 78	80 & 81			83 à 88			86 à 90
Méthodologie utilisée	Albrecht 79	79 & 83: Interf. & 25%	Non précisé	Non Homogène	E/R (88)	E/R	E/R	E/R (84 & 88)
Nombre d'applications	Cobol: 18 Autres: 6 Total : 24	Total: 25	Cobol : 12 Autres: 03 Total : 15	Cobol :238 Autres:125 Total: 363	Cobol : 47 Autres: 34 Total : 81	Total : 35	Total : 76	Total :
Environnement (Homogène-Hétérogène)	1 compagnie, 1 division	1 compagnie	Plusieurs cies Même métho. de dév.	18 compagnies		6 cies gvnt	8 organismes publics	
Domaine des applications	MIS	MIS	MIS	Majorité MIS	MIS	MIS	Majorité MIS	MIS
Taille en Point de Fonction	Min: 199 Max: 1902 Moyen.: 647	Min: 22 Max: 599 Moyen.: 174	Min: 100 Max: 2307 Moyen.: 997	Min: 4 Max: 1,250 Moyen.:316	Min: 73 Max: 793 Moyen.: -		Min: 115 Max: 10,857	Min: 39 Max: 1,542 Moyen.: 362
Coûts (heures)	Min: 0.5K Max: 105.2K Moyen.:21.9K	Min: 0.6K Max: 28.7K Moyen.:...	Min: 0.2K Max: 8.1K Moyen.:1.7K	Min: 0.1K Max: 241.9K Moyen.: 9.7K	Min: 0.5K Max: 44.5K Moyen.: 5.5K			Min: 0.4K Max: 12.2K Moyen.: -
Coûts unitaires (hrs/PF)	Min: 2.5 Max: 66.8 Moyenne:33.8	Min: 2.1 Max: 47.9 Moyen.: 1980: 18.3 1981: 9.4	Min: 0.58 Max: 3.53 Moyen.: 1.77	Min: 0.2 Max: 145.5 Moyen.: 6.2	Min: Max: Moyen.:			Min: 3.8 Max: 34.4 Moyen.: -

6.2 Problématique de l'homogénéité des données

Deux auteurs ont analysé et tenté de quantifier l'impact des différences inter-organisationnelles entre des organisations, à savoir Jones (IFPUG Fall 1988 et IFPUG Spring 1991) et Emrick (IFPUG Spring 1988 et IFPUG Fall 1988).

D'une part, Jones indique que les analyses de productivité doivent non seulement tenir compte des domaines d'applications et des catégories d'industrie (IFPUG 1988) mais aussi des conditions de développement et de leur impact sur la productivité (IFPUG 1991). En plus des causes identifiées aux sections précédentes quant à la fiabilité (répétitivité et exactitude) dans l'application des règles, tant à l'intérieur d'une même compagnie, qu'entre diverses compagnies, Jones note de plus les variations introduites par les différentes façons de mesurer et de comptabiliser le temps affecté aux projets. C'est donc dans ce contexte qu'il propose et utilise une *charte de comptes* comme mécanisme de comptabilisation des ressources humaines affectées aux projets.

D'autre part, la base de données expérimentales qu'utilise Jones n'est pas du domaine public, et de plus la méthodologie statistique qu'il utilise n'est pas disponible. Dans un tel contexte, il n'est donc pas possible d'évaluer la qualité des résultats publiés par Jones; tout au plus peuvent-ils être utilisés comme guides pour des ordres de grandeur.

Les recherches d'Emrick constitue une recherche d'envergure inter-industries utilisant les métriques des points de fonctions comme base de référence. La base d'analyse d'Emrick comprend 375 projets provenant de 17 entreprises différentes, et 66% de ces projets ont été programmés en COBOL. Les données ont été soumises par les différentes compagnies sur une base volontaire, et sans mécanisme de contrôle, ce qui a constitué une difficulté majeure dans l'analyse des résultats.

La productivité y est définie comme les "points de fonction ajustés" par mois-personne brut. Les meilleures corrélations obtenues sur l'ensemble des données apparaissent au tableau 6.3.

Tableau 6.3 : Les meilleures corrélations de productivité - Emrick 88

Variable Dépendante	Variable indépendante	Corrélation
Points bruts	Mois-personne	66%
Productivité	Ré-utilisation du code	48%
Points ajustés	Mois-personne	66%
Points ajustés	Mois-personne pour coder	93%

Le problème d'homogénéité s'est fait particulièrement sentir dans l'étude inter-industrie d'Emrick: il y avait une trop grande variation dans les données de productivité moyenne entre les compagnies[9]: de 3.5 à 37.6 PF-mois. Il s'agit d'une différence d'un ordre de grandeur, ce qui semble invraisemblable. Emrick en attribue la cause aux facteurs suivants: *The difference in productivity between companies cannot be totally explained by any of the collected data. It is not conceivable that there would be an order of magnitude difference in productivity between these companies. ...we must assume that much of the difference is due to individual variance in the method of counting function points between companies* (IFPUG Spring 1988, p. 9).

Une purification des données a donc été essayée par segmentations successives en sous-ensembles de données. Une première purification s'est faite de la façon suivante: en enlevant de l'échantillon les projets qui comprennent des facteurs de productivité, tels que la ré-utilisation et les modifications de code avec toujours peu de bons résultats. De la même façon, l'analyse de la productivité par rapport à l'expérience du personnel n'a pas donné de bonnes corrélations.

Avec les efforts comme variables indépendantes, mais en sous-divisant la taille en trois groupes, selon des intervalles de taille en points de fonction: 0 à 200, entre 200 et 500, et +500, les R**2 sont beaucoup plus élevés (0.68 à 0.8675) indiquant beaucoup plus d'homogénéité. Le même phénomène se répète lorsque cette purification des données se fait pour chaque compagnie. De la même façon si l'échantillon total est découpé en petites tranches de productivité plus homogène, les corrélations s'améliorent (par exemple, pour des intervalles de productivité de 0 à 9 points de fonction ajustés/mois, des intervalles de 10 à 30, et ensuite pour l'intervalle de plus de 30 points de fonction ajustés/mois).

Dans une analyse subséquente des mêmes données (IFPUG Fall 1988), Emrick a proposé une méthode intéressante pour normaliser les données: comme l'application des règles est plus consistante à l'intérieur d'une même entreprise qu'inter-entreprises, il a proposé de normaliser les diverses moyennes de productivité des compagnies en les ramenant à une moyenne uniforme de productivité, tout en gardant leurs mêmes écarts-types proportionnels initiaux. Ceci amène alors des résultats plus consistants, et les interprétations sont plus faciles.

En termes de méthodologie expérimentale, les commentaires les plus importants d'Emrick sont les suivants: Au moment de l'étude il y avait des différences très marquées entre les compagnies dans leur façon de compter les points de fonction et plusieurs admettaient officiellement ne pas se conformer à ce moment-là aux normes officielles d'IFPUG. Cela rendait donc l'analyse statistique particulièrement difficile pour l'interprétation des résultats: il est délicat, écrit-il, de comparer des mesures prises avec des métriques définies et étalonnées de façon différente. Il mentionne qu'il faudra encore probablement quelques années à l'industrie pour se normaliser dans la façon d'utiliser la métrique des points de fonction.

---

[9]En général la productivité moyenne d'une compagnie est basée sur une moyenne pondérée de tous les projets; toutefois cette moyenne pourra être trop basse s'il y a un projet majeur avec une très basse productivité qui diminue considérablement la moyenne.

6.3 Analyse des relations

Le tableau de la page suivante (Tableau 6.4) donne un aperçu des relations (exprimées par les coefficients de corrélation ou de régression) que divers auteurs ont essayé d'identifier dans leurs ensembles de données. Ces auteurs ont en général considéré soit la taille, soit le coût, comme variable tantôt indépendante, tantôt dépendante en fonction de divers autres paramètres. Les sous-sections qui suivent donnent de l'information sur les relations à deux facteurs de même que sur le phénomène des économies d'échelle.

6.3.1 Relations à deux facteurs

Les relations à deux facteurs les plus courantes sont les suivantes: les efforts (heures) par rapport aux points ajustés, aux points bruts et aux milliers de lignes de code. Il faut noter la concordante des coefficients de régression de Kemerer (1987) et Desharnais (1988), avec seulement une légère amélioration du coefficient lorsque l'on passe des points bruts aux points ajustés. Kemerer note le peu d'amélioration apporté par l'information fournie par les 14 caractéristiques générales d'ajustement du modèle des points de fonction. Le groupe de données Informetric fait figure singulière avec des coefficients respectifs très inférieurs (0.24 et 0.16 respectivement).

Il est à noter que seul Kemerer (1987) utilise les tests statistiques: pour la relation avec les points ajustés, l'hypothèse nulle que son modèle et celui d'Albrecht soient les mêmes ne peut être rejetée (avec un test F comparant les deux régressions individuelles avec une seule régression avec les 39 points, et un intervalle de confiance de 95%). De l'autre côté en ce qui concerne le facteur des lignes de code, l'hypothèse que les modèles soient les mêmes est rejetée avec un test statistique F et un intervalle de confiance de 99%.

6.3.2 Économies d'échelle.

Banker et al (1989) a examiné particulièrement le phénomène d'économie d'échelle dans le cadre d'une étude comparative de huit ensembles de données expérimentales dont certaines indiquaient des économies d'échelle, alors que d'autres les contredisaient. En fait, l'étude de Banker en vient à conclure qu'il y a un point d'inflexion en deça duquel il y a des économies d'échelle, et au-delà duquel le phénomène se renverse. Il appelle ce point de retour marginal décroissant le *Most Productive Scale Factor - MPSS*. Il utilise deux méthodes statistiques pour déterminer ces points d'inflexion et il le calcule pour les autres modèles dont les données sont disponibles: un modèle quadratique et un modèle non-quadratique utilisant la technique du *Data Envelopment Analysis - DEA*. Il est à noter que les points d'inflexion sont différents selon les environnements et que de plus, non seulement leurs valeurs mais aussi leurs directions varient selon le modèle statistique utilisé. En ce qui concerne le modèle de Behrens (1989), il faut noter que ses courbes graphiques semblent fortement influencées par les points extrêmes.

Tableau 6.4 : Analyses de productivité avec les points de fonction - Identification des relations (Coeff. de régression & corrélation)

Caractéristiques	Albrecht 83	Behrens 83	Kemerer 87 Banker 89	Emrick 88	Desharnais 88	Informetric 88
LOC (R**2)	.713 (Cobol)		.66 (Cobol)	.91 (1 cie)		
Bruts(R**2)			.54	Corr.: 0.66	.50	.24
Ajustés (R**2)	.87		.55 (Cobol)	Corrélation: .66	.54	.16
Économies d'échelle	Inégalité	Inégalité prononcée	Économie & inégalité			
Productivité: - + 35 hrs/PF - de 5 à 35 " - - 35 hrs/PF				Corrélation: - 0.92 - 0.97 - 0.84		
New/Mod./Re-used code				Inconclusif		
MPSS: - Quadratique - DEA	- 499.4 PF - 199.1 PF	- 105.7 PF - 170.1 PF	- 734.4 PF - 1209.3 PF		- 344.6 PF	
Expérience: - chef d'équipe - équipe				Aucune	- R**2: .497 - R**2: .498	- xxx - xxx

6.4 Analyse d'impact des environnements et facteurs de productivité

6.4.1 Applications interactives versus différées.

Deux études (Behrens, 1989; Emrick, 1988) ont analysé les même facteurs environnementaux, à savoir les différences de productivité du développement d'application interactive versus des applications en différé. Bien que Behrens conclue qu'il soit plus productif de développer des applications interactives, il y a lieu de se poser des questions sur les techniques statistiques utilisées par cet auteur. Ce dernier utilise la technique qu'il intitule "*line of best fit*" pour obtenir un "*trend line*" représentant des inégalités d'échelle très fortes à mesure que la taille des applications augmente. Bien que Behrens ne donne pas le détail de ses données, le positionnement de ses points sur le graphe semble indiquer qu'un point extrême ("outlier") a une importance prépondérante sur la courbe. Cette application d'environ 300 points de fonction a un coût (8000 heures) beaucoup plus élevé par rapport aux quatre autres applications de même envergure (de 250 à 350 PF) qui elles n'ont coûté qu'entre 1200 et 3600 heures. Excluant ce point extrême du graphique, la pente de la courbe serait alors beaucoup moins forte et la présence d'inégalité d'échelle ne serait peut-être pas visible. Ceci se traduirait alors sur son deuxième graphe (productivité par rapport à la taille) par une droite presque horizontale (donc une simple moyenne), plutôt qu'avec une droite avec une forte pente. Les observations et les conclusions par rapport à l'indice de productivité déterminé par cette droite serait alors très différentes.

Tandis que l'étude expérimentale de Behrens ne portait que sur 25 projets à l'intérieur d'une même organisation, l'étude d'Emrick pour ces mêmes facteurs a porté sur 109 projets avec les caractéristiques illustrées au tableau 6.5.

Tableau 6.5 : Caractéristiques des échantillons de projets
Différés/Interactifs - Emrick 88

Caractéristiques	Différé	Interactif
Nombre de projets	37	72
Points de fonction ajustés (moyenne)	422	647
Coûts (heures)	7200	5760
Productivité brute (Hrs/PF)	9.4	5.7
Productivité normalisé (Hrs/PF)	19.0	13.6

En prenant la technique conservatrice de la productivité normalisée, telle que suggérée par Emrick, les applications interactives représentent un gain de productivité de 31%, ce qui est bien plus petit que le gain de 51% avec les données et les techniques d'interprétation de Behrens. De son côté, Emrick signale que plusieurs autres facteurs auraient pu entrer en jeu pour qualifier

ce différentiel de productivité: le fait que la productivité des applications interactives est meilleure pour des applications de taille moyenne plus grande, contredit le phénomène des inégalités d'échelle de production pour des plus grands projets. Selon lui, cela indique alors que c'est soit vraiment plus productif de développer des applications interactives, soit d'autre part, que la métrique des points de fonction n'est pas bien adaptée pour mesurer de façon identique la taille des applications en différé par rapport aux application interactives. Ces hypothèses, ne sont toutefois pas démontrées.

6.4.2 Développement versus Additions-Modifications

Emrick examine 271 projets pour analyser la productivité du développement de nouvelles applications par rapport à des additions ou modifications à des systèmes existants. Un sommaire de ces données est présenté au tableau 6.6. Avec la productivité brute, il semblerait que le développement soit presque deux fois plus productif que les additions ou les modifications majeures à des applications existantes (7.7 versus 14.5 heures par point de fonction). Toutefois, en utilisant la technique normalisée plus conservatrice, il n'y aurait alors qu'une différence de 7% de productivité (15.7 versus 16.8 heures par point de fonction). Des analyses statistiques plus poussées pour déterminer la fiabilité de ces analyses n'ont pas été effectuées: il est donc délicat de tirer des conclusions fermes de ces analyses.

Tableau 6.6 : Comparaison des échantillons
Nouvelles Applications / Add.-Modif. - Emrick 88

Caractéristiques	Nouvelles applications	Add/Mod.
Nombre de projets	156	115
Points de fonction ajustés (moyenne)	509	284
Coûts moyens (heures)	8320	5920
Productivité brute (Hrs/PF)	7.6	14.5
Productivité normalisée (Hrs/PF)	15.7	16.8

6.4.3 Utilisation d'une méthodologie de développement

Emrick a effectué une analyse similaire avec un sous-ensemble de 261 données afin d'analyser l'impact de l'utilisation ou non d'une méthodologie de développement. Ici encore, les gains de productivité ne sont pas importants et sont du même ordre de grandeur que ci dessus; ceci contredit les conclusions de Saindon (1989) qui affirme que la méthodologie de développement par l'approche de données primaires est beaucoup plus productive. La banque

de données de Saindon comprend 76 applications provenant de huit organismes différents[10]. Cette banque de données est utilisée pour analyser les différences de productivité selon les types de langages de programmation utilisés et la méthodologie de développement utilisée, soit l'approche par traitement, soit l'approche par les données primaires (c'est-à-dire par la modélisation des données).

D'une part, l'analyse des données de Saindon pour les projets développés avec des langages de 3ième génération donnent des résultats concordant avec les autres données de recherches publiées, tant en termes de jours par point de fonction que pour les valeurs des R**2. D'autre part, toutefois, les analyses statistiques effectuées avec les projets utilisant les langages de 4ième génération et l'approche par données primaires, contiennent des projets avec des ordres différents de grandeur (de 115 points à 10,857 points!), ce qui crée des points d'ancrage dans les analyses statistiques et ne permet pas de se fier aux résultats indiquant des gains de productivité de 23.83 fois supérieurs de l'approche des données primaires par rapport aux approches traditionnelles avec des 3GL!

6.4.4 Facteurs de niveaux de langages de programmation.

Emrick (1988) et Desharnais (1988) ont abordé ce type de facteurs de productivité. Desharnais, d'une part, a utilisé le niveau de langage de programmation pour segmenter ses données en trois types d'environnement. Son étude a porté non pas sur la productivité elle-même, mais plutôt sur les relations avec les coûts des projets. Quant aux résultats obtenus par Emrick, les résultats sont présentés mais non analysés; les relations intuitives n'y sont pas nécessairement confirmées, telle cette productivité identique entre le macro Assembler et les langages 4GL pour les programmeurs (ADS/O et NATURAL), et ce quelle que soit la technique de normalisation statistique utilisée par Emrick.

Dans un article pour analyser l'impact de l'utilisation de langage de troisième génération (Fortran, Jovial et Ada) par rapport aux langages de niveau de l'Assembler, Gaffney (1986) présente une méthode basée sur les lignes de code pour essayer de déterminer un niveau de fonctionnalité en fonction des efforts dépensés par phases de temps en design, en codification et en test, puis il compare les résultats de ses algorithmes avec la méthode des points de fonction. En se basant sur les données d'Albrecht (1983) il en arrive à la conclusion que la dispersion de la productivité par ligne de code est beaucoup plus grande que celle par point de fonction. De plus, le test statistique sur les moyennes pairées des médianes rejette l'hypothèse de l'égalité: l'hypothèse qu'il n'y a pas d'égalité est rejetée pour celle des lignes de code, mais pas pour celle des points de fonction (statistique "t" au niveau de 6%).

---

[10]Toutefois 46 des 76 projets proviennent du même organisme.

6.4.5 Impact des outils d'aide au développement

L'étude d'Emrick semble, à première vue, indiquer un gain important de productivité, mais le chercheur note que cela est presque toujours sur de plus petits projets. Les gains sont marginaux, et non pas selon un ordre de grandeur (*order of magnitude*) comme le prétendent les vendeurs.

6.5 Analyse par regroupement de facteurs

6.5.1 Concept des caractéristiques dominantes

Johnson (1991, p. 168-170) indique que pour son ensemble de données plusieurs des caractéristiques considérées n'offraient pas de bonnes corrélations: expérience du personnel informatique, l'expérience des usagers, la taille du projet, le pourcentage du nouveau code et la complexité du système. Il mentionne que ces observations contredisent à la fois l'intuition et les progiciels commerciaux d'estimation qui se servent de ces caractéristiques des projets comme base d'estimation.

L'analyse de ses données et des caractéristiques des projets l'a amené à observer que les caractéristiques affectent la productivité, mais pas nécessairement les mêmes pour chaque projet; en d'autres mots, il y a un ou deux facteurs principaux, ou caractéristiques dominantes, qui influencent les projets. Selon lui, le poids d'une caractéristique varie alors avec le projet, et les moyennes numériques n'ont pas de corrélation avec la productivité (Voir le Tableau 6.7).

Tableau 6.7 : Caractéristiques dominantes - Johnson 91

Caractéristique	Productivité élevée	Productivité basse
Écrans	Plusieurs	Peu
Clonage	Extensif	Limité
Traitement	Simple	Complexe
Intégration	Limité	Extensif

Johnson indique qu'en utilisant ces quatre caractéristiques, en même temps que la théorie de la caractéristique dominante, toutes les différences de productivité peuvent être expliquées. Il est à noter qu'il s'agit ici d'une approche à l'analyse de la productivité qui constitue une voie nouvelle par rapport à la majorité des analyses de productivité faite selon une approche dérivée du modèle COCOMO où chacun des facteurs de productivité est étudié séparément et simultanément.

6.5.2 Analyse par composantes principales

Une autre technique est utilisée par Informetric (1988) pour analyser l'impact des facteurs de productivité, soit l'analyse factorielle des composantes principales. Toutefois cette analyse porte à la fois simultanément sur 10 variables dites d'envergure spécifiques à l'application (c'est-à-dire au produit développé) et 10 autres variables de productivité qui caractérisent l'environnement de développement. Cette technique permet de trouver que les six premières composantes (ou axes) expliquent delà 70% de la variance. Le premier axe semble exprimé la "dispersion" (les variables de télécommunications, distribution des données et des utilisateurs multiples - tous des facteurs d'envergure), le deuxième axe, "fiabilité et sécurité" (fiabilité & sécurité, disponibilité du chargé de projet, expérience technique de l'équipe), et le troisième axe, la convivialité du système. Il est noté toutefois que la banque de données de 35 projets est peu peuplée pour le nombre des 20 variables étudiées.

6.6 Analyses d'estimation

6.6.1 Modèles publics

Quatre recherches ont utilisé la métrique des points de fonction pour des études expérimentales sur l'estimation: Albrecht (1983), Desharnais (1988), Informetric 88 et Kemerer (1987). Kemerer indique que deux techniques principales ont été suggérées pour évaluer les modèles d'estimation: l'erreur relative absolue et le test de la régression. Le test de l'erreur relative moyenne (Magnitude of Relative Error - MRE) est suggéré par Conte & al. (1986):

MRE = Valeur absolue ((Estimé - Actuel) / Actuel)

Cette technique MRE est utilisée pour évaluer les estimations basées sur la variable indépendante (temps) et les variables dépendantes (points de fonction et lignes de code). Albrecht a suggéré une autre technique, soit la comparaison de l'estimé par rapport à la ligne de régression linéaire. Dans le cas du MRE, une estimation parfaite aurait une erreur MRE de zéro, tandis que la courbe de régression aurait un R**2 = 1, exprimant une corrélation parfaite.

Il est possible de comparer au tableau 6.8 les résultats du test de régression sur les quatre ensembles de données expérimentales (Albrecht, 1987; Kemerer, 1987; Desharnais, 1988 et Informetric, 1988). Dans les tableaux 6.8 et 6.9, "a" représente la constante de la régression et le "b", le paramètre de la variable dépendante (le nombre des points de fonction) dans l'équation de régression.

Tableau 6.8 : Comparaison des variables d'estimation

Auteurs	Variables	a	b	R**2	Test *t*
Albrecht 83	Points ajustés	-88	0.355	.869	12.37
Kemerer 87	Mois estimés	-37	0.96	.553	4.26
	Points ajustés	-122	0.341	.553	4.28
	Points bruts	-111	0.333	.538	4.16
Desharnais	Points ajustés			.54	-.05
	Points bruts	-220	17.3	.498	-.032
	Modèle brut quadratique			.45	0.09
	Modèle brut cubique			.52	-1.27
Informetric	Points ajustés			0.0157	
	Points bruts			0.0238	

Il faut noter la très grande similarité des R**2 pour Kemerer et Desharnais dans un contexte non homogène de données provenant de plusieurs compagnies. Quant aux données d'Albrecht, elles proviennent de la même organisation et le R**2 y est considérablement plus élevé. Il est à remarquer que Desharnais a testé le modèle avec une forme quadratique et une forme cubique appliquée au pointage brut, avec des améliorations très mineures au test du R**2 qui passe de 0.498 à 0.52 seulement. Il en est de même pour les deux modèles non-linéaires appliqués aux points ajustés où le R**2 ne passe alors que de 0.54 à 0.56.

Il est à noter au tableau 6.9 que Kemerer (1987) démontre que pour son ensemble de données expérimentales le modèle basé sur les points de fonction est beaucoup plus précis que les quatre modèles utilisés comme base de comparaison (SLIM, COCOMO - base, intermédiaire et détaillé) mais moins précis que le modèle ESTIMACS (il faut toute fois noter que le modèle ESTIMACS n'a été testé que pour un sous-ensemble de 9 des 15 projets de l'échantillon et que la fiabilité du test *t* est moins grande).

Tableau 6.9 : Comparaison des modèles d'estimation - Kemerer 87

Modèles	MRE	a	b	R**2	Test *t*
SLIM	772%	49.9	0.082	.878	10.11
COCOMO de base	610%	27.7	0.156	.68	5.54
COCOMO intermédiaire	584%	62.1	0.123	.599	4.68
COCOMO détaillé	608%	66.8	0.118	.525	4.06
Points de Fonction	167%	-122	0.341	.553	4.28
ESTIMACS	85.8%	54.1	0.084	.928	10.17

Les conclusions les plus intéressantes de l'étude de Kemerer sont les suivantes:

1- Les modèles basés sur les lignes de code donnent de meilleurs résultats, en termes de régressions
2- Mais les différences relatives absolues (MRE) sont beaucoup plus grandes, et il faut calibrer ces modèles
3- Les modèles qui ne tiennent pas compte des facteurs de productivité donnent de meilleures prévisions que ceux qui en tiennent compte:
- COCOMO de base versus COCOMO intermédiaire
- Points de fonction Bruts versus Ajustés

6.6.2 Analyses de sensibilité statistique

Kemerer (1987) est un des rares auteurs qui abordent la question de la validité des échantillons utilisés pour les analyses de productivité et d'estimation et qui analyse la sensibilité des résultats statistiques vis-à-vis les cas d'exception (*outliers*) susceptibles de créer des points d'ancrage pour les régressions. Il souligne entre autres qu'il n'est pas rare dans les échantillons dans ce domaine d'avoir des écarts-types de 4.0 (et plus) entre les plus grands pointages et la moyenne de l'échantillon, ce qui est loin d'être désirable du point de vue de la recherche. Knaff et al. (1986) signale également ce problème et mentionne comme exemple que trois projets particulièrement large dans un des échantillons de Albrecht et al. (1983) influencent considérablement les résultats de la corrélation et que s'ils étaient enlevés de l'échantillon, les résultats de la corrélation diminueraient de 44%.

Kemerer (1987) note également que tous les modèles d'estimation sont en général validés avec des informations *a posteriori*, par rapport à des validations avec l'information *a priori*. Ceci limite donc l'utilité de ces modèles en situation *a priori*, ce qui représente les conditions normales d'estimation. Tout comme pour tous les autres modèles d'estimation utilisant d'autres métriques, les modèles basés sur les points de fonction n'ont donc pas fait leur preuve comme outil d'estimation *a priori*.

6.6.3 Modèles privés

Le modèle d'estimation du gouvernement du Québec a été développé à contrat par une firme de consultation. Une recherche a été effectuée par Informétric (1988) afin d'évaluer la fiabilité de ce modèle. La base de données pour cette analyse comprenaient 35 projets mesurés dans six organismes différents et représentant 28 340 jours personne. Six modèles sont analysés dans ce rapport en utilisant les efforts comme variables indépendantes (Tableau 6.10).

Tableau 6.10 : Comparaison des modèles d'estimation - Informétric 88

	Modèle	Caractéristiques	R**2	EA	Prod (10%)	EQR
1	BRUT	Pointage brut	2.38%	40.4%	11.4%	46.1%
2	CIA	Pointage ajusté	1.57%	37.9%	14.3%	44.1%
3	GOUV	Pointage ajusté et modifié	10.38%	24.9%	22.9%	29.7%
4	BRUTP	Pointage brut + facteur de productivité	19.54%	29.0%	20.0%	30.3%
5	CIAP	Pointage ajusté (CIA) + facteurs de productivité	22.75%	21.3%	31.4%	25.4%
6	GOUVP	Pointage ajusté modifié (GOUV) + facteurs de productivité	85.99%	4.7%	94.3%	5.8%

Il s'agit donc de six modèles avec des raffinements successifs, dans lesquels le R**2 passe successivement de 2.38% pour le modèle BRUT à 85.99% pour le modèle GOUVP. L'addition successive des divers niveaux de facteurs permet d'atteindre un R**2 de 85.99% au sixième modèle; simultanément tous les autres indices de fiabilité s'améliorent, soit: l'erreur relative moyenne, l'erreur absolue relative moyenne (EA), le coefficient de prédiction au niveau x (Prod (x))et l'erreur quadratique relative (EQR). Il y a toutefois des discordances majeures pour les modèles BRUT (R**2: 2.38%) et CIA (R**2: 1.57%) par rapport aux autres ensembles de données expérimentales disponibles ailleurs (Voir le Tableau 6.11).

Tableau 6.11 : Comparaison des divers ensembles de données expérimentales

Auteurs	Nombre	R**2 Brut	CIA
Informetric	35	2.38%	1.57%
Desharnais	81	50%	54%
Kemerer 87	15	53%	55%
Albrecht 83	24		86.9%
Montréal Trust	20	81%	76%
RAAQ	25	36%	61%

Il est à noter que les données de Desharnais (1988) et Kemerer (1987) proviennent de plusieurs organisations et que celles d'Albrecht proviennent respectivement d'une seule organisation. Les données de base du modèle gouvernemental n'ont donc pas la même signification, ou le même niveau d'homogénéité, que ce qui est en général dans la littérature, tant pour des groupes de données non homogènes qu'homogènes. Il ne faut toutefois pas oublier

que ces modèles n'ont été vérifiés que par rapport aux données initiales qui ont servi à sa construction! Il n'a donc pas été validé avec d'autres données provenant d'organismes similaires, ni de données additionnelles provenant des mêmes organismes de départ[11].

6.7 Sommaire

6.7.1 Analyses de productivité

Plusieurs problèmes de design expérimental ont été identifiés qui limitent la portée des analyses de productivité avec les métriques des points de fonction. En résumé, à l'intérieur d'une même organisation, il n'y a pas suffisamment de données pour des analyses statistiques portant sur plusieurs variables de productivité, tandis que pour les recherches inter-compagnies, il n'y pas suffisamment d'homogénéité dans les environnements et dans le processus de collecte des données. Particulièrement pour les données soumises sur une base volontaire par différentes organisations, il n'y a en général très peu de mécanismes de contrôle, ce qui constitue une difficulté majeure dans la pertinence et la fiabilité des analyses.

6.7.2 Analyses d'estimation

Enfin, il y a également une faiblesse importante dans l'utilisation des techniques d'analyses statistiques utilisées, particulièrement dans l'absence d'analyses de sensibilité et de l'analyse des cas d'exception susceptibles de constituer des points d'ancrage pour les régressions. En l'absence de telles analyses de sensibilité statistique, il faut être prudent avant d'utiliser tels quels les résultats des recherches publiées à date.

---

[11]Seulement un rapport d'analyse de fiabilité a été accessible pour cette recherche, et non pas le modèle lui-même.

Chapitre 7: RECHERCHES EXPÉRIMENTALES - MÉTRIQUES DÉRIVÉES

7.1 Positionnement

En termes de recherches sur la structure de la métrique elle-même, peu d'auteurs ont examiné le sujet, s'attardant plutôt, comme il a été vu au chapitre précédent, à analyser soit l'impact de la métrique au niveau de la fiabilité inter-pointeurs des comptages, soit de son utilisation comme outils d'analyse de productivité et d'estimation. Quelques auteurs ont néanmoins analysé la structure elle-même du modèle d'Albrecht et y ont proposé des améliorations et des modifications, dont Bock (1989), Desharnais (1988 et IFPUG - Spring 1990), Jones (IFPUG - Fall 1988), Reifer (1991) et Symons (1998). Avant de regarder le détail de ces modèles, il est utile d'utiliser le cadre de référence suggéré par Porter (IFPUG - Fall 1988) pour positionner ces modèles par rapport aux versions successives de la méthodologie officielle. Le tableau 7.1 indique la version de référence de chacun des auteurs pré-cités.

Tableau 7.1: Positionnement des métriques dérivées
(Selon le modèle de Porter)

	Version	Bock	Desharnais	Jones	Reifer	Symons
1	Albrecht 79			X	X	
2	Albrecht 84	X				X
3	IFPUG 86		X			
4	IFPUG 88		X			
5	IFPUG 90		X			

Il est à noter que Jones, Reifer et Symons se sont basés sur les premières versions de cette métrique, puis en ont proposé des modifications majeures, ce qui les a amenés à un développement de métriques dérivées, mais non synchronisées avec les versions officielles de l'industrie. En effet, ils n'ont pas participé activement aux comités de coordination qui avaient été mis sur pied pour améliorer la méthodologie et qui a conduit à la publication de versions successives et améliorées de 1986 à 1990: leurs modèles n'ont donc pas bénéficié de ces améliorations. A ces auteurs, s'ajoutent Laranjeira (1990) et Pfleeger (1990) qui ont discuté de son application au processus de développement par objet, mais qui n'ont pas proposé de modèle détaillé et qui n'ont pas procédé à des expérimentations.

Ce chapitre n'aborde que les modèles de Jones, Reifer et Symons qui ont divergé considérablement des versions officielles de la métrique et qui se qualifient alors de métriques dérivées. Quant aux analyses de Bock et Desharnais, elles se situent à l'intérieur soit des versions officielles, soit des propositions de futures versions officielles: il n'en sera donc question qu'à l'intérieur du prochain chapitre.

7.2 Le modèle de Symons

7.2.1 La structure

Le modèle MARK II de Symons remet en question la structure de base du modèle d'Albrecht. Le modèle de Symons est basé sur le concept de *transaction logique* selon la méthodologie de Jackson dans laquelle chaque transaction logique est une séquence de données à l'entrée, de traitements et de données à la sortie. Une hypothèse fondamentale de son modèle est la suivante:

> "It seems a reasonable hypothesis that a measure of processing complexity is to count *the number of entity-types referenced by the transaction-type* - referenced means created, updated, read, or deleted."[12]

Dans ce modèle, les données en entrée et en sortie sont quantifiées en nombre d'éléments présents et le traitement, en nombre d'entités référencées en cours de traitement. La taille de l'application est alors entièrement déterminée par l'addition simple de tous les éléments en entrée, de tous les entités référencées et de tous les éléments en sortie. Les concepts suivants n'interviennent pas alors dans la quantification le calcul de la taille de l'application dans ce modèle: le type de transactions, une dimension calculée pour chaque transaction, le type de fichiers logiques (internes ou interfaces), une dimension attribuée à chaque fichier logique. Les étapes de sa méthode sont les suivantes:

1- Les entités sont tout d'abord identifiées
2- Les transactions logiques sont ensuite identifiées
3- Chaque transaction logique est alors décomposée pour identifier et compter séparément: les éléments en entrées, les entités référencées et les éléments en sortie
4- Trois totaux sont effectués pour les éléments en entrées, les entités référencées et les éléments en sortie de l'ensemble des transactions logiques.

A partir de là, la formule suivante est utilisée:

$$UFP's = N_I W_I + N_E W_E + N_O W_O$$

où les symboles représentent:

UFP : le nombre de points MARK II non ajustés
N_I : le nombre de types d'éléments de données en entrée
W_I : le poids d'un type d'élément de données en entrée
N_E : le nombre de références à un type entité

---

[12]Note: les italiques sont de Symons.

W_E : le poids d'une référence à un type entité
N_O : le nombre de types d'éléments de données en sortie
W_O : le poids d'un type d'élément de données en sortie

L'étape suivante suggérée par Symons est d'essayer de déterminer les poids en calibrant la formule en fonction des données réelles.

Une différence importante entre la méthode de Symons et celle d'Albrecht vient de la façon de voir les données. La méthode de Symons attribue des points aux données en fonction seulement de leur usage dans les transactions qui font partie de l'application mesurée. Albrecht en tient compte également, avec le comptage des fonctions d'entrée, de sortie et de requête. Mais Albrecht attribue de plus aux données elles-mêmes une valeur d'existence en comptant les entités logiques et les interfaces, indépendamment de leur utilisation. En résumé, la méthode de Symons ne tient compte que des occurences dans les transactions, tandis que celle d'Albrecht tient également compte du vocabulaire (valeur d'existence dans les fichiers logiques et les interfaces).

La métrique de Symons avec seulement un type de fonction (transactions logiques) est en superficialité plus simple. Par contre ses hypothèses sont plus insidieuses parce qu'elles sont basées sur une correspondance entre dimension et facteurs de productivité (sommarisée par le temps requis pour effectuer le travail) et il ne fait pas référence à des fondements théoriques pour cette approche. De plus, par rapport à la méthode proposée par Albrecht, l'agrégation de l'information dans celle de Symons fait perdre de l'information importante sur le contenu des applications développées. Elle fait perdre des visions des applications que comprennent bien les usagers des applications, et avec lesquelles il n'a pas été démontré qu'il y avait un problème conceptuel, ni d'application de la méthodologie. Le modèle de Symons n'est pas un modèle pur d'un modèle de données ou d'un modèle de traitement: en fait, il est un mélange de modèle de données (entité-relation) et d'un modèle de traitement (process diagram). Symons semble de plus avoir de la difficulté à identifier correctement et à séparer les concepts de dimensionalité de l'application, des facteurs d'envergure et des facteurs de productivité.

7.2.2 Complexité

Pour la mesure de la complexité du processus, Symons veut se baser sur les travaux de McCabe, et de la complexité. De là, il passe au concept suivant (l'argumentation est loin d'être rigoureuse, et il l'admet lui-même): la complexité structurelle de McCabe est basée sur le comptage et le poids[13] des branchements et des boucles. A partir de la méthode de Jackson, une fonction bien structurée devrait correspondre à la structure logique, représentée par le chemin d'accès de la transaction à travers le modèle entité-relation de toute l'application. Chaque étape le long du chemin d'accès à l'intérieur du modèle d'entité implique une sélection,

---

[13] *Poids à partir d'un modèle implicite, non défini de façon précise.*

ou un branchement ou une boucle: il semblerait alors raisonnable à Symons que la mesure de la complexité du processus corresponde au comptage du nombre de types d'entités référencées par le type de transactions (Référencé signifie: créé, mis-à-jour, lu ou détruit).

Symons écrit que c'est une approximation rudimentaire de premier niveau. Il n'apporte comme seule preuve que sa propre affirmation que l'expérience de compter les entités référencées par transaction dans des systèmes réels supporte la plausibilité de l'hypothèse. Il affirme que l'effort est du premier degré d'ordre de grandeur avec le nombre d'éléments de données pour chacune des composantes respectives.

7.2.3 Facteurs de productivité

Une autre hypothèse de base de Symons est que la dimension de l'application est reliée explicitement à l'effort pour analyser, pour concevoir le design et pour développer les fonctions de l'application. Symons prétend que l'hypothèse d'Albrecht est basée plus sur un critère subjectif, la "valeur de la fonction livrée à l'usager", et par conséquent moins facile à vérifier ou à calibrer en pratique. L'hypothèse de Symons par contre est basée sur des critères à la fois beaucoup plus nombreux (i.e. les facteurs de productivité) et sur lesquels il n'y a pas de recherches conclusives sur l'influence réelle (et mesurée de façon objective et scientifique) de ces facteurs de productivité. La métrique de Symons n'est donc pas indépendante de la technologie: elle est intrinsèquement basée sur le temps pour effectuer un travail, donc intrinsèquement de la technologie utilisée, et de plus ses calculs ont été basés sur plusieurs technologies pour un très petit échantillonnage de 9 projets.

De plus dans sa discussion sur l'estimation, Symons entremêle les concepts de biens livrables et les facteurs de productivité. Lorsque l'on change de technologie et d'outils de production, il est tout-à-fait normal de calibrer et de changer les facteurs de productivité. Par contre les points de fonction eux-mêmes devraient demeurer constants. Sinon, il est impossible d'évaluer les augmentations de productivité car tout change en même temps et il ne reste plus de variables indépendantes sur lesquelles baser les analyses.

Parmi les autres contraintes à l'utilisation du modèle de Symons il faut noter d'une part que le détail de celui-ci n'est pas du domaine public, et d'autre part qu'il ne s'est pas coordonné avec le groupe international de normalisation et ne bénéficie pas des améliorations apportées constamment au modèle officiel IFPUG.

7.3 Les domaines d'applications autres que M.I.S.

Plusieurs auteurs ont noté la difficulté d'appliquer cette métrique à d'autres domaines à cause de la façon dont elle avait été définie initialement (Grady et al., 1987) et à l'intérieur du groupe IFPUG il existe un sous-groupe d'intérêt qui examine ce problème. Cette section passe en revue les travaux de deux auteurs qui ont proposé des extensions de la métrique à des

domaines autres que celui des applications M.I.S.: Jones (IFPUG Fall 1988) et Reifer (1991).

7.3.1 Jones

Jones (1988) a proposé une variation (*Feature Points*) au modèle initial afin de l'adapter au contexte des systèmes en temps réel et aux systèmes embarqués ("embedded systems"). En résumé:

- il ajoute un sixième type de fonctions pour tenir compte des algorithmes,
- il ne retient que les poids moyens des cinq fonctions d'Albrecht
- il redéfinit le concept des caractéristique générales à l'aide d'une définition de la complexité définie à trois niveaux: problème, code et complexité (Voir Tableau 7.2) dont les poids varient de 1 à cinq, et dont l'influence totale est combinée en un facteur d'ajustement global variant de 0.16 à 1.5.

Tableau 7.2 : Critères et poids de complexité - Jones

Weight	Problem Complexity	Code Complexity	Data Complexity
1	Simple algorithms & simple calculations	Nonprocedural (Generated, spreadsheets, etc)	Simple data with few variables and little complexity
2	Majority of simple algorithms & algorithms	Built with program skeletons and reused code	Several data elements, but simple data relationships
3	Algorithms and calculations of average complexity	Well structured (small modules & simple paths)	Multiple files, switches, and data interactions
4	Some difficult or complex algoritms or calculations	Fair structure, but some complex modules and paths	Complex data elements and comples data interactions
5	Many difficult and complex algorithms & calculations	Poor structure, with many complex modules & paths	Very complex data elements and data interactions

Le modèle de Jones, il est probablement le plus articulé en ce qui concerne les facteurs de productivité. Il est certainement celui avec le plus de données disponibles en terme d'impact par facteurs de productivité et en termes des définitions de ce qui est inclus, ou exclus, à l'intérieur du système de comptabilisation du temps. Cette approche est certainement dans la ligne de pensée du modèle COCOMO de Boehm, mais comme sa base de données n'est pas du domaine public, il s'agit là d'informations du type privé (*proprietory*) qu'il n'est pas possible d'utiliser comme base de recherches, puisque ni les bases de données de Jones ne sont connues, ni ses méthodes statistiques et ni ses méthodes d'analyse.

7.3.2 Reifer

L'autre modèle (ASSET-R) a été proposé par Reifer (1991) pour mesurer la taille des applications scientifiques et en temps réel. Le modèle de Reifer a été construit en vue de tenir compte des caractéristiques (Tableau 7.3) d'autres domaines de logiciel, tel le domaine scientifique et le domaine des applications en temps réel.

Tableau 7.3 : Caractéristiques des domaines d'applications logicielles - Reifer

Data Processing Domain	Scientific Domain	Real-time Domain
I/O Oriented: - Many files - Many screens - Many reports - Many transactions	Processing Oriented: - Many algorithms - Few screens - Few reports - Few transactions	Sequencing and Control Oriented: - Concurrency - Interrupts - Synchronization
Query/response timeliness	Run to completion	Stimulus/response relationships
Issues: - Data model - Distribution - Logical/physical mappings - Structure	Issues: - Control flow - Equation/algorithm performance - Modes - Parallelism	Issues: - Deadlock - Fault tolerance - Queues/controls - Timing and sizing
Interactive	Man-in-the-loop	Interrupt-driven

Reifer propose la mesure suivante pour mesurer la taille d'une application:

$$SIZE = ARCH * EXPF ((FP_A) (LANG) + MVOL)^{RF}$$

Les descriptions des variables de son équation sont indiquées au tableau 7.4.

Tableau 7.4 : Description des variables de l'équation - Reifer

Variable	Description	Comments
SIZE	Application size	Physical size in lines of code
ARCH	Architectural constant	
EXPF	Expansion factor	
FP_A	Function Points	Non standard
LANG	Language expansion constant	
MVOL	Math volume	
RF	Reuse factor	Not yet implemented

Plusieurs tables de critères et de pointage sont fournies afin de déterminer une valeur pour chaque variable de la formule. Il est à noter que même si le terme *Function Point* est utilisé par Reifer, ce n'est pas dans le contexte généralement accepté dans la littérature ou dans l'industrie. Le tableau 7.5 présente deux exemples qui illustrent que les termes utilisés dans modèle de Reifer ne correspondent pas aux définitions officielles: la colonne de gauche indique que huit fonctions sont prises en considération par rapport aux cinq de la version IFPUG 90 et la colonne de droite indique de plus que la définition de ces fonctions (en prenant les "interfaces" comme exemple) ne correspond pas nécessairement aux définition officielles.

En termes de validation du modèle, Reifer présente un certain nombre de cas avec une comparaison entre la taille actuelle et l'erreur relative d'estimation qui, à première vue, parait très bonne. Il est toutefois important de noter qu'il ne s'agit pas de statistiques, mais uniquement d'exemples et d'illustrations. Il n'y a pas de présentation de faits sur la démographie de l'ensemble de l'échantillonnage ni sur la qualité statistique des erreurs d'estimation. Une comparaison par rapport à cinq autres modèles d'estimation est présentée, mais là encore il n'y a pas d'information sur la méthodologie statistique utilisée.

Tableau 7.5 : Listes des estimateurs - Reifer

Function Points Size Estimators (Reifer)	External Interfaces Size Estimators (Reifer)
Number of inputs	Common utilities
Number of outputs	Math librairies
Number of inquiries	Program libraries
Number of files	Stored databases
Number of interfaces	
Number of modes	
Number of rendez-vous	
Number of stimulus/response relationships	

7.4 Sommaire

Les métriques dérivées qui se réclament des concepts initiaux des points de fonction ont été analysés en fonction des versions officielles comme base de référence et de comparaison. Ces métriques dérivées visent à palier à certaines des faiblesses majeures des points de fonction, en particulier à sa transmutabilité à des domaines d'application autres que M.I.S. Il faut toutefois noter que ces métriques dérivées se sont éloignées de la métrique officielle au point de ne pas être facilement conciliables et de ne pas pouvoir bénéficier des améliorations apportées chaque année à la métrique initiale. Quant à leurs méthodologies de mesure, ces métriques dérivées n'ont pas apporté d'amélioration au domaine des métriques en génie logiciel. D'une part, il n'y a eu aucune diffusion de la base d'expérimentation ni de validation expérimentales indépendantes de ces métriques. De plus il a été également mentionné que ces métriques nécessitaient un plus grand nombre d'hypothèses que la métrique officielle, et qu'aucune de ces hypothèse n'était basée sur des théories prouvées et démontrées expérimentalement.

Chapitre 8: RECHERCHES EXPÉRIMENTALES - PROCESSUS DE MESURE

8.1 Design expérimental

En plus d'Albrecht lui-même, deux autres auteurs ont procédé à des analyses et à des expérimentations afin d'analyser de différentes perspectives la structure même de la métrique des points de fonction: Bock (1989) et Desharnais (1988, 1990). Le tableau 8.1 présente un sommaire des caractéristiques de ces recherches expérimentales qui ont analysé la structure de la métrique des points de fonction: on y retrouve des informations comparatives sur la méthodologie de pointage utilisée, le nombre de pointages, les mécanismes de validation, la taille des applications mesurées et les méthodes statistiques utilisées.

Tableau 8.1 : Caractéristiques des recherches expérimentales - Processus de mesure

Caractéristiques	Albrecht 84	Bock	Desharnais
Méthodologie utilisée	Albrecht 84	Albrecht 84	IFPUG 88 et Modèle E/R
Nombre de pointages	24 applications 1 organisation	93 applications 1 organisation	286 applications 18 organisations 74% COBOL
Validation des pointages	Albrecht	6 analystes expérimentés avec contre-vérification pour un taux d'exactitude vérifié de +/- 5%	50% effectué par le même pointeur
Taille des applications	Min: 199 Max: 1902 Moyenne: 648	Min: 15 Max: 7436 Moyenne:1113	Min: 73 Max: 793 Moyenne: -
Méthodes statistiques	Régressions Erreurs relatives moyennes Corrélations	Analyse par composantes principales H_0 et H_1 Test de Wilcoxon Régressions	Régressions

La structure suivante en quatre points est utilisée pour l'analyse de la structure de cette métrique:

1- Le domaine des facteurs de productivité
2- Le domaine des caractéristiques générales de l'application
3- Le domaine des algorithmes d'Albrecht
4- L'identification des composantes primaires

8.2 Les facteurs de productivité.

Dans la métrique des points de fonction les facteurs de productivité sont exclus, de par la définition même de la métrique qui se veut un modèle de la taille de l'application. En fait, la variable *effort* n'est jamais considérée comme une variable indépendante à l'intérieur du modèle, mais toujours comme une variable dépendante: c'est-à-dire que l'effort est toujours analysé en fonction de la taille de l'application mesurée, et non comme une composante intrinsèque de taille. Tous les autres modèles corroborent avec cette hypothèse, à l'exclusion de celui de Symons, tel que mentionné au chapitre précédent, section 7.2.3.

8.3 Le processus de mesure de l'ajustement.

Les facteurs d'ajustements de l'application, au nombre de 14 dans le modèle Albrecht 84, sont également appelées les caractéristiques générales de l'application. Il est à noter que ces caractéristiques générales sont dérivées des modèles de productivité qui ont précédé celui d'Albrecht, et qui se retrouvent pour la plupart dans COCOMO. L'originalité d'Albrecht a été de les isoler spécifiquement dans sa méthode et de les incorporer à la notion de taille de l'application elle-même, les séparant ainsi de toutes les classes de facteurs de productivité. Dans sa métrique, Albrecht est celui qui sépare de la façon la plus claire les concepts du produit logiciel de celui du processus de développement.

Toutefois dans le modèle d'Albrecht, et contrairement à celui de Boehm, tous les facteurs ont la même importance (influence statistique) dans le même continuum de 0 à 5, avec un pas égal à 1 chaque, et pour chacun des 14 facteurs. En réalité cela n'indique aucunement leur importance (influence respective) par rapport aux points bruts. Cette approche d'Albrecht est basée uniquement sur une approche intuitive (expérience personnelle) dont le but était de calibrer son modèle; elle ne bénéficie pas de fondement théorique et il n'y a pas mention d'un fondement statistique à cette partie de son modèle. Quand à Symons, il ajoute 6 autres facteurs aux 14 facteurs d'influence et, dans son expérimentation, la répartition du temps selon ces 20 facteurs, a été faite *a posteriori*, et de façon très subjective. Avec seulement 9 projets, il n'est toutefois pas prudent statistiquement de faire des analyses pour 20 facteurs simultanés.

Desharnais (1990) est celui qui a le mieux posé le problème du point de vue statistique: les 14 facteurs d'envergure ne peuvent être additionnés en statistiques, mais doivent plutôt être multipliés parce qu'ils s'influencent l'un l'autre. Les résultats de ses analyses statistiques en se basant sur son modèle indiquent que seulement 7 des 14 facteurs sont significatifs pour son échantillon. Toutefois même avec les 286 projets de sa base de données historiques, Desharnais observe qu'il est très difficile de distinguer statistiquement les 14 facteurs: ou il n'y avait pas assez de données avec les autres facteurs, ou il n'y avait pas assez de dispersions dans leurs valeurs. Tout comme le suggérait Symons (1988), il y aurait alors lieu d'émettre l'hypothèse que certains sont répétitifs et représentent peut-être le même phénomène.

8.4 La structure du modèle

8.4.1 Une structure inspirée d'Halstead

La structure de base de la métrique des points de fonction se rapproche des concepts exprimés initialement par Halstead, et ceci a été noté par Albrecht lui-même en 1984. La théorie de Halstead indique que le vocabulaire d'un programme est une fonction du nombre d'opérateurs ("n_1 = *operators*") et d'opérandes ("n_2 *operands*") uniques ($n = n_1 + n_2$), que la taille d'un programme est une fonction du nombre d'occurrences de ces opérateurs et opérandes ($N = N_1 + N_2$) et que la dimension ou Volume d'un programme ("*software length equation*") est représentée par l'équation de la forme suivante:

$$V = N_1 \log_2 n_1 + N_2 \log_2 n_2$$

Albrecht note que certaines recherches expérimentales ont démontré des relations étroites entre le comptage réel des instructions et l'estimé de ce comptage à l'aide de l'équation d'Halstead. Albrecht et al. (1984) se base ensuite sur les hypothèses suivantes pour analyser les fondements de son modèle:

> 1- n_1 n'a pas besoin d'être connu pour estimer le nombre de tokens N, ou le nombre d'instructions I; une moyenne pour n_1 peut être utilisée, ou encore le facteur $N_1 \log_2 n_1$ peut être omis, introduisant évidemment un certaine degré d'erreur.
>
> 2- partant de l'observation de Chistiansen que *la taille d'un programme est déterminée par les données qui doivent être traitées par le programme* Albrecht suggère que différentes approches peuvent être utilisées pour estimer la taille et le volume du logiciel et qu'en autres la taille n_2 du vocabulaire des données ("data label") peut être estimée, ou encore qu'un équivalent n_{2*} soit utilisé comme le nombre d'entrées et sorties conceptuellement uniques.

Il démontre alors à l'aide d'une recherche expérimentale avec son échantillon que la formule initiale d'Halstead développée pour de petits algorithmes peut être appliquée à de larges applications[14] où n_{2*} est utilisé pour exprimer l'ensemble des entrées et sorties externes à une application (ou programme). Dans ce contexte les points de fonction, exprimés par n_{2*}, peuvent être interprétés comme la somme avec poids de pondération des items d'entrée/sorties à haut niveau, tels que les panoramas d'écrans, les rapports et les fichiers.

Albrecht explore ensuite différentes hypothèses de formules par rapport au trois variables

---

[14]Albrecht mentionne l'équivalent de programmes pour applications, mais en fait il s'agit de l'ensemble des lignes de code de tous les programmes qui composent les applications de son échantillon. Il note également les types de langages de programmation de chacun de ses échantillons.

suivantes:

F: le pointage des points de fonction (avec les poids et l'algorithme)
V: le total des fonctions[15] (avant les poids et l'algorithme)
SLOC: le nombre de lignes de code

Il est à noter que plusieurs des hypothèses analysées sont elles-mêmes des fonctions soit de F, soit de V.

Le tableau 8.2 donne la liste des variables indépendantes, les formules de bases de même que les variables dépendantes explorées dans les régressions sur son échantillonnage.

Tableau 8.2 : Variables d'estimation explorées - Albrecht 83

No.	Variable indépendante	Formule	PL/1 SLOC	COBOL SLOC	Heures
1	Function Points	F	X	X	X
2	Function Sort Content	$(F/2) \log_2 (F/2)$	X		X
3	Function Potential Volume	$(F+2) \log_2 (F+2)$	X		
4	Function Information Content	$F \log_2 F$	X		X
5	I/O Count	V			X
6	Sort Count	$(V/2) \log_2 (V/2)$			X
7	Count Information Content	$V \log_2 V$			X
8	Sources Lines of COBOL	SLOC			X
9	Source Lines of PL/1	SLOC			X

[15]Il s'agit ici du même concept que celui proposé dans la méthode simplifiée de Bock.

Les résultats des régressions (Tableau 8.3) donnent d'excellents résultats avec des erreurs relatives moyennes relativement basses et des corrélations simples très élevées entre les estimés et les actuels (SLOC ou Work-Hours).

Tableau 8.3 : Résultats des régressions - Albrecht 83

NO.	Variables used were:	Relative Error	Relative Error	Sample Correlation - Variables and:
	Estimators of SLOC	Average	Standard Deviation	Actual SLOC
1	Function Points (COBOL)	.229	.736	.854
2	Function Points (PL/1)	.003	.058	.997
3	Function Sort Content (PL/1)	.007	.057	.997
4	Function Potential Volume (PL/1)	-.002	.057	.997
5	Function Information Content (PL/1)	-.002	.057	.997
	Estimators of Work hours			Actual Work-Hours
1	Function Points	.242	.848	.935
2	Function Sort Content	.192	.759	.945
3	Function Information Content	.194	.763	.944
4	I/O Count	.206	.703	.945
5	I/O Count Sort Content	.195	.630	.954
6	I/O Count Information Content	.195	.637	.954
7	SLOC (PL/1)	.023	.213	.988
8	SLOC (COBOL)	.323	.669	.864

Albrecht observe que les modèles basés sur les comptages bruts donnent des résultats légèrement meilleurs, mais pas de façon significative, que le modèle basé sur les points de fonction. Ceci démontre pour cet échantillonnage qu'il y a des correspondances intéressantes par rapport aux livrables exprimés en lignes de code de même que par rapport aux ressources pour accomplir le travail, mais également que le modèle n'ajoute pas beaucoup d'informations

par rapport à des concepts plus simples qui ne font pas appel aux transformations de l'algorithme officiel et des poids attribués par la méthode. Il ne faut toutefois pas oublier que l'échantillon d'Albrecht contient trois applications de très grande taille qui à elles seules contribuent à 44% de la corrélation (Knaff et al., 1986) et que les formules mathématiques utilisent les logarithmes.

8.4.2 L'algorithme d'Albrecht

La question se pose à la fois sur la validité de ces poids à l'intérieur du domaine initial des applications M.I.S. d'Albrecht, et à la fois sur leur validité pour d'autres domaines d'applications. A l'intérieur du domaine M.I.S, Bock et al. (1990) a exploré un aspect de cette question; toutefois à l'extérieur du domaine M.I.S., aucune recherche expérimentale n'a été recensée sur cet aspect du problème.

Bock a effectué des recherches pour vérifier l'hypothèse suivante qui avait été suggérée par des analystes en points de fonction chez McDonnell Douglas: ... *if the mix of function types is stable between systems and over time, then the process for classifying functions as simple, average, or complex can be replaced by a procedure that relies upon a single multiplicative adjustment factor for each of the five function types.*

L'objectif initial était de simplifier le processus de pointage en examinant si, à partir d'un analyse des données historiques de ce site industriel, il pouvait construire un modèle simplifié et calibré à ce site qui leur permettrait d'accélérer leur comptage en n'ayant pas à identifier et à compter le détail des éléments et des fichiers et enregistrements requis pour la classification selon la structure de l'algorithme d'Albrecht. S'ils pouvaient obtenir ainsi un modèle simplifié et calibré à leur environnement, il ne leur resterait alors plus à identifier que le niveau le plus élevé du processus, soit le nombre de fonctions dans les applications, et leur classification à l'intérieur des cinq types de fonction.

Une hypothèse supplémentaire a été testée, à savoir que les cinq types de fonction telles que définies par Albrecht pouvaient être reliées entre elles. Pour vérifier cette hypothèse, une analyse statistique de facteurs par composantes principales avec rotation varimax a été effectuée. Cette analyse s'est faite simultanément sur les pointages des cinq types de fonction et sur le nombre de fonctions à l'intérieur des cinq types[16]. Cette analyse a révélé de façon non équivoque, avec des facteurs de charge supérieurs à 0.8, qu'il existait un modèle simplifié à deux facteurs, dont le premier comprenait simultanément quatre fonctions (Fichiers internes, entrées, sorties et requêtes) et le deuxième, uniquement la fonction interface. Bock a ensuite utilisé ces deux facteurs de charge pour identifier séparément par régressions linéaires des facteurs d'ajustement pour chacun des types de fonction.

---

[16] *Function points*: calculés avec l'algorithme d'Albrecht; *Function count*: calculés sans utiliser l'algorithme d'Albrecht.

Le modèle simplifié et calibré prend alors la forme suivante:

Nombre de fonctions * Facteur multiplicatif (par type de fonction) = Total des points

Les facteurs multiplicatifs obtenus pour le modèle dérivé sont présentés au tableau 8.4.

Tableau 8.4 : Facteurs du modèle dérivé - Bock

Type de fonction	Facteurs multiplicatifs
Fichiers internes	8.75
Interfaces	5.66
Entrées	3.19
Sorties	5.29
Requêtes	3.69

Les deux coefficients de régression des facteurs de charge pour ce modèle sont très élevés (R**2 = 0.99 et 0.975). Afin de vérifier la validité du modèle simplifié, trois méthodes statistiques ont été utilisées: une analyse graphique, un test de Wilcoxon et une analyse de corrélation pairée des pointages selon les deux méthodes. Les résultats ont été très bons pour les trois tests statistiques (Tableau 8.5).

Tableau 8.5 : Test statistiques - Bock

Analyse graphique	Wilcoxon signed ranked test	Analyse de corrélation pairée
% erreur = (pointage simplifié - pointage détaillé) / pointage simplifié	Vérification de l'hypothèse nulle	Le test de corrélation statistique de Pearson est très élevé: (0.994, p <.001)
79 des 93 observations sont à l'intérieur de +/- 10% d'erreur	H_0: $Médiane_{simplifiée}$ - $Médiane_{détaillée}$ = 0	
Le pourcentage moyen d'erreur est presque zéro (-0.06)	H_1: $Médiane_{simplifiée}$ - $Médiane_{détaillée}$ not = 0	
la distribution des pourcentages d'erreur n'est presque pas biaisée (Skewness = 0.48)	L'hypothèse nulle qu'il n'y a pas de différence n'est pas rejetée (valeur p = 0.97)	

Les résultats de ces divers test statistiques indiquent un support statistique très élevé pour affirmer qu'il n'y a pas de différence statistique entre les points de fonction bruts obtenus par

les deux méthodes (Albrecht et la méthode simplifiée calibrée pour l'échantillonnage étudié). En fait, si les poids dérivés par Bock étaient arrondies au nombre entier près, cela se rapprocherait considérablement du modèle Albrecht 79 ou aux poids moyens de Albrecht 84 (Tableau 8.6).

Tableau 8.6 : Comparaison des poids (Bock versus Albrecht)

Type de fonction	Albrecht 79	Bock & al.	Albrecht 84
Fichiers internes	10	8.75	10
Interfaces		5.66	7
Entrées	4	3.19	4
Sorties	5	5.29	5
Requêtes	4	3.69	4

Il est à noter que le test de Wilcoxon est l'équivalent non paramétrique du test-t pairée[17]. Bock indique que le test-t n'est pas approprié dû au problème associé à l'hétéroskedasticité des données. Le test de Wilcoxon évalue des observations pairées dépendantes. Les deux échantillons sont alors réduits à un seul échantillon par l'analyse des différences de comptage pour les observations pairées. Les tests des hypothèses nulles et non nulles sont alors effectués sur les médianes des deux échantillons.

Pour des questions d'homogénéité des données, il est à noter que l'échantillon des 93 données analysées par Bock et al. ne comprenait que des applications de types M.I.S., excluant ainsi les pointages des applications des types scientifiques et d'ingénierie qui, chez ce site industriel, avaient des distributions de fonction différentes[18].

Bock mentionne les limites et contraintes suivantes à cette approche:

1- Les résultats obtenus sont en fonction de la distribution des fonctionnalités dans cet échantillon; donc les résultats ne sont pas nécessairement généralisables à d'autres entreprises; une entreprise devrait donc effectuer une vérification sur ses données historiques avant d'utiliser le modèle simplifié.

2- L'hypothèse de stabilité dans le temps pour le mixte de complexité simple,

---

[17] *Non-parametric equivalent of the matched pairs t-test.*

[18] "...Only business application systems were selected for this study because scientific and engineering applications exhibit quite different patterns in the overall mix of functions delivered to end-users."

moyenne et complexe dans les cinq types de fonctionnalités constitue une limitation potentielle forte à la méthode simplifiée. Il y aurait plus de recherche à faire pour valider cette hypothèse.

8.4.3 L'identification des composantes primaires

Il a été noté dans la revue des versions officielles de la métrique des points de fonction (Chapitre 3) que la méthodologie IFPUG E/R* basée sur le modèle Entité/Relation était à l'étape de l'étude par le comité de normalisation. Symons (1988) mentionne que le concept de fichiers logiques est très ambigu et qu'il devrait être remplacé par celui d'entité. Il ne définit pas précisément ce qu'est une entité et ne fait pas référence à des normes quelconques. Par contre Desharnais (1988) se réfère de façon explicite au domaine de la modélisation des données comme structure pour l'identification des composantes primaires à la base du processus du pointage. En fait, il y aurait potentiellement un avantage à utiliser le concept d'entité/relation, parce qu'il y a un champ de théorie relié au domaine de la modélisation, avec des règles très strictes et très précises (se référer par exemple aux règles des 5 formes normales des modèles des données)[19].

Même si Symons (1988) a proposé une métrique dérivée basée sur l'approche Entité/Relation, il a simultanément effectué des modifications majeures à la structure de la métrique d'Albrecht, contrairement à Desharnais qui a proposé une méthode beaucoup plus consistante avec le modèle d'Albrecht 1984. C'est d'ailleurs pour cette raison que le modèle Desharnais a été retenu par IFPUG et le M.I.T. pour leur études comparatives de fiabilité inter-compteurs et inter-méthodologies[20] et comme base de discussion pour élaborer les futures normes.

Un autre type de modèle basé sur l'approche orientée objet a été proposé par Laranjeira (1990) pour identifier et comptabiliser les données primaires dans les besoins fonctionnels. Dans le modèle objet proposé, chaque entité du domaine du problème est représentée par un objet. Chaque objet est composé d'un état qui le caractérise, et d'un ensemble de fonctions, appelées méthodes, qui manipulent les données correspondant à son état. Un objet est donc considéré comme ayant 2 projections:

1- la structure des données qui représente son état.
2- les fonctions qui déterminent son comportement.

Un objet agit sur un autre en demandant au second objet de réaliser une certaine fonction sur ses données (état) et de retourner des résultats à l'objet demandeur. Le processus

---

[19]Cette approche a été inspirée initialement par les travaux de K. Zwanzig (1985) de Bell Canada.

[20]Selon Kemerer (1990), les variations inter-méthodologies sont de 8.48% pour son design expérimental (Voir section 5.2.2).

d'estimation de la taille d'un système, dans un modèle fonctionnel caractérisé par un certain nombre de niveaux de décomposition (partitioning), peut être sommarisé par les étapes suivantes:

1- En commençant par le plus bas niveau de décomposition, évaluer la taille de chaque objet. Cette évaluation doit considérer chaque fonction exécutée par l'objet, de même que le code correspondant aux structures de données qui contiennent son état interne.[21]

2- Remonter aux autres niveaux pour tenir compte que les objets de niveau supérieur peuvent recevoir des contributions des (*component objects*) aussi bien que de ses propres données et fonctions[22].

3- Il peut être nécessaire d'inclure des "objets utilitaires" (*utility objects*) pour tenir compte des certaines fonctions "Housekeeping".[23]

4- La taille estimée de tout le système sera la somme des tailles estimées de tous les objets dans le niveau supérieur de décomposition plus la taille de tous les "utility objects" potentiellement existants.

Le détail et les règles du modèle de Laranjeira ne sont pas présentés, et les analyses ne sont pas supportées par des expérimentations complètes. Il en est de même pour une autre discussion d'un modèle conçu par Pfleeger (1990) à partir de l'approche orientée objet. Il est à observer que ces auteurs ne font pas référence à l'approche des modèles de données, qui contiennent plusieurs des concepts proposés.

---

[21]Note: ceci semble correspondre d'une part au niveau des entités logiques internes du modèle de données des Points de fonction.

[22]*Quelle est l'équivalence au niveau des fonctions? Ceci semble correspondre à un mélange du modèle de données et du modèle des traitements (Process model).*

[23]Note: les utilitaires: il s'agit d'un concept qui n'est pas présent dans les points de fonction mais qui devrait peut-être y être.

8.5 Sommaire

En résumé, les recherches sur la structure elle-même de la métrique sont peu nombreuses. Seul Desharnais a analysé un des aspects structurels de la métrique, à savoir le processus de mesure de l'ajustement et en a identifié certaines faiblesses, tant dans la structure de l'algorithme utilisé que dans la validité des poids des 14 caractéristiques. Son expérimentation illustre également la difficulté à valider autant de facteurs simultanés. De son côté l'analyse d'Albrecht et al. est intéressante à cause du point de vue utilisée qui relie les concepts des points de fonction aux concepts de base des métriques d'Halstead. Toutefois il n'a a pas eu d'analyse de sensibilité statistique par rapport aux données de l'échantillon et les observations de corrélation ne sont pas aussi fiables qu'il n'y parait à première vue.

L'analyse de Bock et al. est intéressante parce que, jusqu'à un certain point, elle permet de vérifier l'équivalence entre les versions Albrecht 84 (matrice à 3 dimensions) par rapport à la version simple initiale Albrecht 79 à 1 dimension. Cette analyse illustre également que dans un environnement stable et similaire à celui de l'expérimentation, la procédure simplifiée est valide. Cette étude ne valide pas la métrique de façon extrinsèque, mais uniquement la relation entre deux étapes de celles-ci.

Enfin il y a une absence quasi totale de recherche en ce qui concerne la base l'un processus de mesure, à savoir l'identification de façon exacte des composantes les plus élémentaires du processus de mesure, de même que de l'analyse de leurs relations dans l'ensemble du processus. Tout au plus y a t'il des propositions de nouvelles approches, basées soit sur le modèle Entité/Relations ou sur le modèle orienté objet. L'approche Entité/relation a fait l'objet d'analyses comparatives intéressantes au niveau des résultats globaux des pointages (Kemerer 90) mais l'approche basée sur les objets n'est pas supportée par des expérimentations ni d'analyse d'impact à l'intérieur même du processus de mesure. Quant aux bases des composantes primaires des métriques dérivées, elles n'ont pas fait non plus l'objet d'analyses et, à ce stade-ci, leurs structures relèvent uniquement de l'opinion d'experts, et non pas d'une base scientifique.

Chapitre 9: IDENTIFICATION DES PROBLÉMATIQUES

INTRODUCTION

Ce chapitre établit un inventaire des problématiques spécifiques à la métrique des points de fonction, lesquelles problématiques ont été regroupées sous les thèmes suivants:

1- Les biais structurels
2- Les problèmes organisationnels
3- La structure de l'algorithme
4- La pertinence et les domaines des modèles implicites

L'objectif visé est de compléter le survol de la littérature effectuée dans les chapitres précédents en se concentrant sur les faiblesses structurelles des points de fonction en tant que systèmes de métriques en génie logiciel. Ceci permettra d'identifier une série de limites et contraintes qui n'ont été soit ni identifiées, soit analysées que partiellement dans le domaine des recherches expérimentales. Ce tour d'horizon permettra au chapitre suivant de positionner correctement le domaine de recherche qui y sera choisi.

Il ne s'agit donc pas dans le présent chapitre d'analyser en profondeur, ni d'y apporter des solutions, mais plutôt d'établir les difficultés qui demeurent à résoudre avant d'asseoir une crédibilité scientifique à cette métrique: la majorité des problématiques mentionnées requierront beaucoup d'énergies, de créativité et de recherches expérimentales afin de les régler. L'objectif de ce travail sera donc nécessairement modeste, mais devrait apporter néanmoins une contribution significative à l'analyse de la métrique des points de fonction en tant que système de mesure dans le domaine du génie logiciel.

9.1 Les biais structurels historiques

9.1.1 Biais environnementaux du contexte initial

La métrique a pris naissance dans un milieu IBM. Pour le personnel qui ne connaît que cet environnement, la formulation des concepts, règles et interprétations peut sembler être d'application générique. Or il y a des biais culturels corporatifs de l'environnement d'IBM et d'utilisateurs d'équipement IBM et cela est d'ailleurs un des objectifs du groupe d'utilisateurs IFPUG que de généraliser le vocabulaire et les concepts de la métrique à un contexte plus générique.

Il s'est aussi révélé que dans sa formulation initiale cette métrique a été formulée avec de très forts biais, d'une part d'environnement de développement sans référence à des méthodes structurées et méthodologie formelle, d'autre part d'un environnement de développement avec des bases de données de type hiérarchique (IMS). Il faudra donc au cours des prochaines années

généraliser les règles et en rehausser le niveau d'abstraction en faisant possiblement appel aux méthodes formelles en cours d'élaboration.

9.1.2 Biais M.I.S.

Dans la métrique des points de fonction le point de vue de l'usager est présenté comme le niveau d'abstraction le plus élevé, et la segmentation des fonctionnalités en cinq types comme le deuxième niveau d'abstraction. Toutefois le vocabulaire pour décrire ces niveaux est plutôt d'un niveau spécifique aux logiciels de systèmes d'information M.I.S. Ce biais vers ce domaine d'applications est encore très présent dans la formulation des règles et ne facilite pas l'application de cette métrique dans d'autres domaines d'applications.

Il y aurait peut-être lieu de reformuler cette métrique pour qu'elle soit applicable à un plus large éventail d'applications, scientifiques par exemple ou encore d'applications en temps réel. Il s'agit soit tout simplement d'un problème de vocabulaire, soit au contraire d'un problème plus fondamental de l'absence d'un niveau d'abstraction plus élevé qu'il y aurait lieu de bien identifier, de bien structurer et de bien formuler.

9.1.3 Séparation des concepts: taille, complexité et productivité

Certains praticiens (Knight, 1989; Martin, 1991) ont mentionné que les points de fonctio pouvaient mesurer la complexité des logiciels et des progiciels. Or il y a ici une confusion dans les concepts due à certains termes utilisés dans la méthodologie des points de fonctions. En effet les termes de fonctions "*simple, moyen et complexe*" sont utilisés pour quantifier, à l'aide des tables de décisions, les types de fonction. Or il s'agit plutôt de qualificatifs pour identifier les positions de classement dans les tables de décision et pour leur attribuer des poids différents déterminés à partir de certains critères définis par des règles empiriques. Les points de fonction bruts sont considérés actuellement comme strictement une métrique de taille, et non de complexité. La notion de complexité elle-même n'est pas le concept le plus précis en génie logiciel comme le rapporte Coupal et al. (1990).

De plus, le processus d'ajustement par l'intermédiaire des quatorze (14) caractéristiques générales de l'application ne constitue pas d'un processus rigoureux pour déterminer la complexité d'une application. L'examen attentif de ce processus (voir les sections 4.4 et 8.3) révèle plutôt un essai de quantifier (ou de qualifier!) une application en fonction de ses attributs visibles par les usagers, plutôt qu'une démarche scientifique appuyée par de solides recherches théoriques et expérimentales.

9.2 Les problèmes organisationnels d'implantation

Bien que la métrique ait été déjà introduite dans plus de 400 organisations, son taux de pénétration à l'intérieur même est encore très lent et plusieurs problèmes organisationnels ont été rencontrés, dont en voici trois qui sont détaillés dans les sous-sections qui suivent.

1- La maturité organisationnelle
2- La simplicité apparente et la courbe d'apprentissage
3- Les exigences de pré-requis: la documentation

9.2.1 La maturité organisationnelle

Parmi les problèmes d'ordre organisationnel, il faut noter le faible niveau de maturité de l'organisation informatique, tant dans les méthodologies de développement, que dans les méthodes de gestion. Selon Humphrey (1988), du Software Engineering Institute (SEI), la majorité des organisations informatiques se situent encore au seuil minimum dans les niveaux de maturité de génie logiciel (au niveau 1 (stade initial) sur une échelle de 5 niveaux où le niveau 5 est celui de l'optimisation). Quoique les méthodologies structurées de développement soient connues depuis quelques dizaines d'années, leur taux de pénétration est encore infime dans les organisations informatiques. La majorité de ces organisations travaille encore sans méthodes structurées, sans documentation fiable et de qualité, et sans accumuler d'informations historiques. Non seulement on ne prend pas le temps d'analyser, mais il n'y a pas d'informations fiables à analyser. Il s'agit d'ailleurs d'un des problèmes majeurs de la recherche en génie logiciel, à savoir la difficulté d'obtenir de l'information fiable dans les milieux réels de travail. Ceci conduit à l'impossibilité pour les organisations de répéter systématiquement leurs succès et à démontrer objectivement leur amélioration de productivité.

La très grande majorité des groupes informatiques n'ont pas actuellement en place les outils élémentaires de mesure qui leur permettent de gérer de façon professionnelle: c'est ainsi que plus beaucoup n'ont même pas un système de mesure pour enregistrer de façon fiable le temps passé sur les divers projets et activités professionnelles ou administratives. Les taux d'échecs dans l'implantation des programmes de mesures sont de plus de 80% (Rubin, 1990). Les pré-requis ne sont donc pas en place pour produire des bonnes analyses de productivité:

- Les mécanismes de collecte de données ne sont en général pas en place;
- Lorsque les mécanismes sont en place, ils sont en général peu fiables;
- Il n'y a pas de données historique en nombre suffisant pour des analyses statistiques.

9.2.2 Simplicité apparente et courbe d'apprentissage

Les concepts de base des points de fonction étant présemément simples, il a été assumé que la courbe d'apprentissage était très simple et qu'il suffisait de quelques heures de cours pour acquérir la maîtrise de la méthodologie. Or il en est tout autrement puisque les applications à mesurer sont souvent très complexes et qu'il est très difficile d'exprimer aisément cette complexité. De plus le personnel qui mesure est encore très largement le personnel qui réalise le design et l'implantation de l'application. Leur point de vue est donc normalement celui des réalisateurs, et non pas celui de l'utilisateur. Cela introduit potentiellement des biais structurels à la fois dans l'interprétations des règles.

Or la méthode des points de fonction, en soi n'examine que le *produit* logiciel lui-même, en faisant abstraction des facteurs de production, et selon le seul point de vue de l'usager. Il s'agit là d'un nouveau paradigme pour les informaticiens, et peu d'entre eux ont la formation requise pour mesurer à partir de cette perspective. Il faut donc revenir constamment à l'intention originale de mesurer l'application selon le point de vue des usagers. A la difficulté d'utilisation, Grady et al. (1987) ajoute la difficulté d'enseigner cette métrique.

9.2.3 Exigences de pré-requis: la documentation

L'identification des points de fonction n'est pas basée sur les lignes de code pour lesquelles il existe souvent des outils automatisés de comptage. L'identification des fonctions et des composantes primaires à mesurer dans la métrique des points de fonction repose actuellement sur la partie généralement non automatisée d'une application, c'est-à-dire sa documentation. Or il s'agit certainement du talon d'Achille de tout le développement du logiciel et ce phénomène est reconnu comme un des problèmes majeurs de l'industrie. De plus, dans le cas de très grandes et complexes applications, celles-ci ne sont pas connues dans leur totalité par une seule personne. Il faut alors faire appel à plusieurs personnes pour trouver toutes les composantes de la documentation des cinq types de fonctions et pour leur comptage. Dans le cas des lignes de code, il suffisait en général d'une seule personne pour obtenir les lignes de codes par un examen automatisé des librairies de code. La fiabilité de la documentation à toutes les étapes est un problème sérieux tant dans le processus du développement que dans le processus de la maintenance des applications. Bien souvent la vérité n'est que dans le code!

La quantité de temps requis pour la collecte des données est également un problème majeur. Actuellement le processus de collecte des données est entièrement manuel et demande des ressources humaines importantes en efforts et en temps de collecte des données, sans compter les possibilités certaines d'introduction d'erreurs par un processus entièrement manuel (Whisehunt, 1990). Dans ce contexte, l'utilisation de référentiel à l'intérieur des ateliers intégrés de génie logiciel est présentée par certains comme une avenue intéressantes d'automatisation du processus de collecte des données (Banker et al, 1991; Porter, 1990; Mazzuco, 1990). Il faut toutefois signaler que ceci n'en est encore qu'au stage de prototypes.

9.3 Les structures des algorithmes de mesure

Voici maintenant quelques unes des problématiques spécifiques à la structure de l'algorithme de cette métrique:

1- Les valeurs discontinues des intervalles
2- Les poids
3- L'additivité des fonctions
4- Un pot-pourri d'échelles de mesure

9.3.1 Les valeurs discontinues des intervalles

L'algorithme fonctionne avec des intervalles de valeurs, et l'identification précise du compte exact des éléments n'est pas requis, mais seulement son positionnement pour un intervalle de la matrice 3 X 3. L'importance prise par un seul élément lorsque celui-ci fait changer d'intervalle (de complexité), si lorsque le total des éléments se trouve aux limites d'un intervalle, est une source de biais structurels dans la mesure car ceci introduit des facteurs de discontinuité.

9.3.2 Les poids

Il faut se rappeler qu'Albrecht lui-même n'a pas la prétention d'avoir élaboré une métrique universelle. Son objectif initial n'était que de résoudre le défi précis qui lui avait été posé et les poids ont été déterminés par discussion et essais dans un seul environnement de dévelopement (une division IBM en 1979). Ceci devait faire l'objet de plus d'objectivité et aurait dû être vérifié et analysé dans d'autres environnements. Il serait étonnant qu'un seul ensemble de poids soit valide en toute circonstances aussi bien à l'intérieur d'un même domaine qu'entre divers domaines d'applications. De plus, la classification de la complexité *(simple, moyenne et complexe)* des types de fonctions des applications est sur-simplifiée selon Symons (1988). Les poids attribués créent également des distorsions (Pfleeger, 1989): 100 éléments ont seulement au maximum trois (3) fois le poids d'une fonction avec un seul élément.

9.3.3 L'additivité des fonctions

Le problème de l'additivité des fonctions a été souligné par Symons (1988) qui relève que la somme des parties est plus grande que la somme du tout: cet auteur indique qu'une des faiblesses de la méthode vient du fait du pointage des interfaces au niveau des parties d'une application qui fait que la sommation des parties d'une application donne un résultat plus grand que le pointage de l'application considérée comme un tout.

Ce problème de la non-additivité des interfaces se manifeste par exemple lorsqu'un projet est développé par phase et comptabilisé par sous-ensembles. Cette critique de Symons n'est pas justifiée. Il s'agit tout simplement du phénomène comptable bien connu de la comptabilisation des états financiers consolidés d'un groupe de corporations, opération au cours de laquelle il faut tout naturellement éliminer les opérations de double comptabilisation pour les transactions entre filiales et la société de contrôle. Il en est de même pour le comptage des applications développées en phases: il s'agit alors d'une question de point de vue, et comme en comptabilité il faut utiliser les régles appropriées à la circonstance. L'additivité des fonctions se pose cependant à un tout autre niveau, à savoir la pertinence dans la métrique des points de fonction, comme dans bien d'autres métriques de génie logiciel, d'additionner des éléments de nature et sémantiques différentes. Cet aspect de la question est abordé plus en déail dans la section qui suit.

9.3.4 Pot-pourri d'échelles de mesure

Le résultat du calcul des points bruts et des points ajustés est habituellement interprété comme une valeur qui permet de compter les applications, de les mesurer et de les comparer avec des ratios: par exemple une application de 600 points est considérée comme trois (3) fois plus grande qu'une application de 200 points. Il est toutefois douteux qu'une telle comparaison soit valide. Le point de départ est valide, à savoir que l'on identifie et que l'on compte de façon précise, par exemple, les éléments de données et les fonctions. On obtient alors des unités de mesure, processus par lequel on peut dire par exemple qu'un rapport ayant 21 éléments contient trois fois plus d'éléments qu'un autre rapport avec 7 éléments. Par la suite, ce nombre calculé permet de positionner à l'intérieur d'un intervalle qu'il n'est plus possible d'additionner en tant qu'unité de mesure.

En fait, l'on retrouve simultanément les quatre types d'échelles de mesure dans le processus du calcul des points de fonction, tel qu'illustré dans les exemples qui suivent:

1- L'identification des cinq types de fonctions (Entrées, sorties, requêtes, fichiers logiques internes et interfaces): Échelle nominale

2- La classification dans une échelle de 0 à 5 les 14 caractéristiques générales de l'application mesurée: Échelle ordinale

3- L'utilisation de tableaux de poids pour classer les fonctions selon les niveaux de "complexité": Échelle d'intervalle

4- Addition des éléments de données à l'intérieur d'une fonction: Échelle de ratio

Comme on peut le voir ci-dessus, tous les types d'échelle de mesure sont utilisés, ce qui engendre une confusion générale sur les genres et rend très difficile une interprétation véritable des résultats (eg. mesures) obtenus avec la métrique des points de fonction. Cette problématique spécifique sera analysée plus en profondeur au chapitre suivant.

9.4 La pertinence et les domaines des modèles implicites

9.4.1 L'identification des modèles implicites

Cette problématique a déjà été mentionnée au chapitre 4: le processus global de la métrique n'a pas été modélisé, et par conséquent les modèles implicites n'ont pas été mis en relief pour les scruter à la loupe pour les valider ou les infirmer. Sans prétendre en dresser une liste exhaustive à ce stade ci, quelques uns de ces modèles implicites sont identifiés afin de souligner l'existence des problématiques inhérentes à chacun:

1- Le modèle de la perspective spécifique à cette métrique (la vue de l'usager)
2- Le modèle à cinq fonctions, et la rigueur des définitions de ces fonctions

3- Le modèle des fichiers (ou groupe de données) logiques, et leurs définitions
4- Le modèle des transactions, et leurs définitions
5- Les modèles sous-jacents aux transactions (structure *Add/Modify/Delete*)

L'absence de modèles explicites n'a pas facilité l'examen des principes fondamentaux de la structure de la métrique et a également nuit à la crédibilité de cette métrique. Cette absence de modèles explicites validés a rendu très difficile, comme illustré par les travaux de Jones et Reifer, la transmutabilité tant des algorithmes que de la structure de cette métrique aux autres domaines d'application.

9.4.2 Couverture incomplète des fonctions dans le domaine M.I.S.

Même à l'intérieur du domaine M.I.S., le modèle actuel de la métrique n'est pas nécessairement complet. Ainsi les cinq types de fonctions, telles qu'identifiées et définies actuellement dans la méthodologie, ne couvrent nécessairement l'ensemble des fonctionnalités demandées et livrées à l'usager dans un logiciel du domaine M.I.S. Ceci est illustré entre autres par quelques uns des problèmes considérés depuis plusieurs années par le comité de normalisation du groupe IFPUG, lesquels problèmes n'ont pas encore obtenu de consensus:

1- Les rapports sur demande "ad hoc";
2- Les gi phiques;
3- Les sous-systèmes d'aide (Help).

Il existe également à l'intérieur du domaine M.I.S. d'autres types de fonctionnalités qui sont demandées par les usagers, comme par exemple les fonctions ergonomiques, et dont la métrique ne tient pas compte. De même il y a certaines fonctionnalités spécifiquement exclues par le comité de normalisation parce qu'elles ne rentrent pas dans les cinq catégories actuelles de fonctionnalités:

1- Les menus de navigation demandés par l'usager qui comprennent uniquement des sélections de navigation;
2- Les générateurs de rapports qui permettent par exemple aux usagers de définir eux-mêmes leurs propres rapports.

9.4.3 L'identification des composantes primaires des fonctions

La version GUIDE 85 (Chapitre 3) avait introduit le concept d'éléments de données et d'enregistrements logiques comme composantes de base de tous les types de fonction et en avait ainsi établi des critères mesurables. Comme ce niveau de détail s'était révélé très sujet à interprétation, les définitions ont été clarifiées par le groupe IFPUG qui publia les versions 88 et 90 de la méthodologie de la métrique. Toutefois ce nouveau niveau de détail a certains des mêmes défauts que le niveau supérieur: il ne s'agit pas d'un niveau normalisé et il ne bénéficie pas non plus de base théorique. Ce fait a été reconnu par IFPUG qui considère l'approche par le modèle Entités/Relations comme alternative pour obtenir une base théorique plus solide et

donner un cadre plus cohérent au processus d'identification des fonctions et de leur composantes primaires. Il faut toutefois reconnaître les limites de cette approche E/R* qui n'apportera des solutions qu'à ce niveau de la métrique et n'affectera pas les autres types de faiblesses. De plus cette approche E/R* est spécifique au domaine M.I.S. et non pas à l'ensemble des domaines. Il faudra éventuellement la généraliser à un autre niveau d'abstraction. Il faut également reconnaître que cette approche n'améliorera que cette seule problématique.

9.4.4 Les limites du domaine de la métrique

Cette métrique permet actuellement d'illustrer des différences de productivité, mais ne permet pas en soi de déterminer des liens de causalité. En fait, l'objet de la métrique des points de fonction n'est que la taille des applications, exprimées et quantifiées en termes de fonctionnalités. Cette métrique ne se penche donc pas en soi sur les facteurs du processus de production: il s'agit uniquement, et strictement d'une mesure de la taille du produit logiciel. Dans un contexte de mesure de productivité, le problème n'est pas uniquement avec les points de fonction, mais également avec les autres dimensions, c'est-à-dire les facteurs de productivité, et la méconnaissance de leurs caractéristiques et de leurs impacts.

Les points de fonction peuvent être utilisés en combinaison avec les autres métriques du processus de développement pour faire des analyses de productivité et d'estimation: ils peuvent être utilisés pour construire des modèles de productivité, mais ne garantissent pas la précision des modèles de productivité car ce n'est pas leur domaine. Le domaine des points de fonction est strictement la taille des applications logiciel, et non pas la productivité dans son ensemble. Il s'agit donc d'un champ relativement restreint, et c'est à l'intérieur de ce domaine que les points de fonction doivent être utilisés, analysés et critiqués.

9.5 Sommaire

En résumé, la métrique des points de fonction est une métrique algorithmique (ou synthétique) calculée à partir de formules et, comme tel, elle a les défauts des métriques algorithmiques:

1- Les métriques algorithmiques sont difficiles à interpréter parce que les raisons pour lesquelles sont assignées des valeurs particulières (poids) ne sont pas précisées.

2- La valeur en sortie de la formule est utile seulement si la formule est basée sur une théorie sous-jacente solide, telle l'économétrie ou la physique. Or ce n'est pas le cas actuellement pour les métriques des points de fonction.

3- Il est nécessaire d'emmagasiner les valeurs de base des mesures qui constituent la métrique algorithmique, pour garder le contrôle du processus de mesure, et pour en permettre des analyses plus poussées.

En fait à date, la métrique des points de fonction a été analysée et utilisée principalement dans sa dimension macro, comme cela avait été illustré dans la revue de la littérature des chapitres quatre (4) à huit (8): variation inter-pointeurs des totaux des points et la fiabilité des résultats dans les modèles de productivité et d'estimation au niveau macro uniquement dans des environnements en général non contrôlés, et sujet à beaucoup de bruit statistiques. Afin de renforcer la crédibilité de la métrique il faudra examiner de près la dimension micro, tant au point de vue de la méthode que dans l'analyse des mesures accumulées, et ce aussi bien pour mieux comprendre les applications mesurées et leurs caractéristiques, que pour les analyses de productivité dans lesquelles on met en relation le produit, les ressources et le processus.

La solution aux principaux problèmes soulevés dans ce chapitre au niveau des biais structurels historiques, des problèmes organisationnels et de la structure des algorithmes de mesure viendra potentiellement d'une résolution de la quatrième problématique et des outils disponibles pour l'analyser et l'améliorer, à savoir de l'utilisation de la démarche scientifique.

Chapitre 10: LA PROBLÉMATIQUE DE LA RECHERCHE

INTRODUCTION

La première section de ce chapitre positionne le thème choisi pour cette recherche en analysant une seule des problématiques identifiées aux chapitres précédents: l'utilisation des échelles de mesure à l'intérieur de la métrique des points de fonction. La section 10.2 présente les hypothèses de recherche élaborée dans ce contexte. Quant à la section 10.3, elle définit la méthodologie de recherche qui est proposée pour amorcer une analyse fondamentale sur certains aspects de la structure elle-même de la métrique, dans une perspective strictement d'éléments de mesure.

10.1 Analyse des échelles de mesure

10.1.1 Hypothèse de base

Il a été vu au chapitre trois que les travaux des groupes GUIDE et IFPUG s'étaient concentrés sur la clarification des règles et interprétations pour identifier et classifier de la façon la plus objective possible les fonctions (Fichiers logiques et Transactions). Pour cette analyse, l'hypothèse suivante est faite: les règles et interprétations courantes, les étapes *Identifier* et *Classer* produisent des fonctions de données, des fonctions de transactions et des facteurs d'ajustement *GSC* à partir de critères objectifs. Ceci permet alors de se concentrer uniquement sur la signification mathématique des diverses étapes du modèle et des sous-modèles de mesure à l'intérieur de la métrique.

10.1.2 Propriétés des échelles de mesure

Cette analyse se concentre sur les deux aspects suivants de tout processus de mesure: l'identification et l'analyse de l'utilisation des différentes échelles de mesure à l'intérieur des diverses étapes du modèle de mesure des points de fonction. Les descriptions des échelles de mesure sont présentées ci-dessous et les opérations mathématiques permises (eg. égalité, addition, multiplication, division) pour chaque type d'échelle sont illustrées au tableau 10.1.

1- Échelle nominale: cette échelle est utilisée pour nommer des objets ou des événements; cette échelle est utilisée pour des besoins d'identification seulement, et à part la notion d'égalité entre les objets, elle n'a pas d'autre notion quantitative associée à celle-ci.

2- Échelle ordinale: cette échelle est utilisée pour ordonner des objets, basé sur un critère qui peut être subjectif ou, de préférence, objectif.

3- Échelle d'intervalle: cette échelle (aussi appelée échelle cardinale) est utilisée pour déterminer la différence entre les rangs. C'est un continuum entre deux points, lesquels ne sont pas nécessairement fixe.

4- L'échelle ratio: avec cette échelle, les objets peuvent être distingués et ordonnés, et de plus les différences entre les rangs peuvent être mesurées.

5- L'échelle absolue: en plus des propriétés de l'échelle ratio, l'échelle absolue possède une origine unique à partir de laquelle débute le processus de mesure.

Tableau 10.1 : Échelles de mesure - Transformations admissibles

Types d'échelle	**Représentation**	**Transformations admissibles**	**Description des opérations**	**Exemples**
Nominale	(R,=)	f unique	nommer, identifier	couleurs, formes
Ordinale	(R,>=)	f fonction monotonique strictement croissante	ranger, ordonner	premier, deuxième, ...
Intervalle	(R,>=,+)	f(x)= ax + b, a>0	additionner	age, distance
Ratio	(R,>=,+)	f(x)= ax, a>0	multiplier, diviser	prix
Absolue	(R,>=,+)	f(x)= x	multiplier, diviser	Température absolue: degré Kelvin

10.2 Le processus de mesure des données

L'utilisation des échelles de mesure est analysée à l'intérieur du modèle du processus de mesure des données (Figure 4.2). Dans la présentation des règles de la métrique, l'algorithme pour le calcul des points est décrit comme un processus de calcul à trois étapes: un fichier logique est tout d'abord classifié comme *simple*, *moyen* ou *complexe*, puis un poids est assigné par rapport à son positionnement dans une table de référence et finalement les points sont additionnés.

Ceci est illustré dans l'exemple qui suit: une application a les trois fichiers logiques suivants:

- Le fichier logique 1 possède 1 enregistrement et 5 éléments de données
- Le fichier logique 2 possède 3 enregistrements et 21 éléments de données
- Le fichier logique 3 possède 6 enregistrements et 26 éléments de données

Dans la méthodologie, le fichier logique 1 serait classifié comme *simple* avec un poids de 7, le fichier logique 2 comme *moyen* avec un poids de 10 et le fichier logique 3 comme *complexe* avec un poids de 15. L'addition de leurs poids donne alors un total de 32 points pour ces trois fichiers logiques.

Ceci est toutefois une sur-simplification. Afin d'identifier les types d'échelles et d'analyser leur utilisation dans ce processus de mesure, le modèle du processus de mesure des données illustré à la figure 4.2 est repris pour illustrer ces concepts. Il est décomposé de la façon suivante pour les fins de cette analyse:

1- Les différents types d'objets comptés
2- L'algorithme:
    2a) les intervalles et les rangs
    2b) la fonction à deux paramètres
    2c) le classement des positions dans la matrice
    2d) l'attribution des poids
3- L'addition des poids

1- A l'étape F1 et F3 de la Fig. 4.2, trois différents types d'objets sont comptés: les éléments de données (DET), les enregistrements logiques (RET) et les fichiers logiques (FTR).

Il faut noter que ces trois types d'objets ne sont pas indépendants, mais organisés dans une structure hiérarchique. Un enregistrement est composé d'un ou plusieurs éléments de données (DET), et un fichier logique (FTR) est composé d'un ou plusieurs enregistrements logiques (RET). Ils représentent trois niveaux différents d'abstraction: le RET représente une structure de DET, et le FTR, une structure de RET. Ces trois types d'objets ont donc des sémantiques et des propriétés différentes.

Les processus F1 et F3 comptent les trois différents types d'objets décrits ci-dessus, et les résultats de chaque addition peuvent être utilisés dans une échelle d'intervalle: par exemple, il est possible de dire d'un fichier logique qui a 6 enregistrements logiques qu'il a trois fois plus d'enregistrements logiques qu'un autre fichier logique qui n'a que deux enregistrements logiques.

2- A l'étape F4 (Fig. 2), l'algorithme des données est décomposé en plusieurs sous-étapes:

2a) La première sous-étape de cet algorithme consiste à positionner à l'intérieur d'intervalles les résultats de l'addition des éléments de données et des enregistrements logiques (Tableau 10.2).

Tableau 10.2 : Rangs des fichiers logiques

Objets	Rang 1	Rang 2	Rang 3
Éléments de données (DET)	1-19	20-50	51+
Enregistrements logiques (RET)	1	2-5	6+

Comme ces intervalles ne sont pas égaux, les résultats de l'arrimage à l'intérieur de ces rangs, ne peuvent pas être additionnés, mais uniquement positionnés. Par conséquence, le

résultat ne se qualifie pas comme échelle d'intervalle: il s'agit tout au plus d'un processus de classement ordonné, et par conséquent ceci constitue strictement une échelle ordinale. A partir de ce point, il est uniquement possible de dire que le rang 2 est plus grand que le rang 1, mais pas que le rang 2 a deux ou trois fois plus d'éléments. Il s'agit donc d'une perte importante de flexibilité dans les opérations mathématiques permises.

Tableau 10.3 : Identificateurs des rangs

Objets	Rang 1:	Rang 2:	Rang 3:
Éléments de données (DET)	D1	D2	D3
Enregistrements logiques (RET)	R1	R2	R3

Pour fin de clarification dans les paragraphes qui suivent, les différents rangs de type ordinal des divers objets mesurés (éléments de données et enregistrements logiques) seront identifiés par les notations du tableau 10.3.

2b) La sous-étape suivante illustrée au tableau 10.4 est plus complexe du point de vue d'un processus de mesure: les rangs des deux différents types d'objets (données et enregistrements) sont pris comme paramètres d'une fonction utilisant deux différentes échelles ordinales. Ceci peut être représentée en notation mathématique comme une fonction à deux arguments fn(Di,Rj). Quoique cette technique soit un processus normal pour construire des indices basés sur une variété de facteurs, ceci ne constitue pas un processus normal de mesure.

Tableau 10.4 : Identificateurs de rangs sur 2 axes

Données/ Enregistrements	Rang D1	Rang D2	Rang D3
Rang R1	D1,R1	D2,R1	D3,R1
Rang R2	D1,R2	D2,R2	D3,R2
Rang R3	D1,R3	D2,R3	D3,R3

Cette transformation ne produit pas un résultat de nature ordinale: quoique la cellule (D1,R1) puisse être plus petite que la cellule (D3,R3), il n'est pas possible de déduire si la cellule (D1,R2) est égale, plus petite ou plus grande que la cellule (D3,R1). Il s'agit plutôt d'une échelle nominale, avec une perte d'information de mesure par rapport à l'échelle de mesure ordinale des paramètres de l'algorithme.

2c) La prochaine étape assigne un classement ordonné (*simple*, *moyen*, *complexe*) selon le positionnement dans la matrice:

- Simple : au dessus de la diagonale inversée
- Moyen : sur la diagonale inversée
- Complexe: en dessous de la diagonale inversée

Tableau 10.5 : Matrice de complexité fonctionnelle

Données/ Enregistrements	Rang D1	Rang D2	Rang D3
Rang R1	Simple	Simple	Moyen
Rang R2	Simple	Moyen	Complexe
Rang R3	Moyen	Complexe	Complexe

L'ensemble de ces sous-étapes est parfois appelé l'assignation de la complexité fonctionnelle dans la méthodologie des points de fonction. Toutefois les termes utilisés (*simple, moyen, complexe*) ne sont pas appropriés et obscurent certaines faiblesses de la méthodologie du point de vue d'un processus de mesure: en effet le vocabulaire utilisé tend à faire croire qu'une échelle ordinale est utilisée à cette étape de la mesure. De plus, le terme *Complexité* n'est pas défini et il n'y a pas eu d'analyse d'impact pour vérifier soit les résultats de la classification, soit le processus lui-même de classification (si ce n'est des jugements d'expert).

Il faut noter aussi que la diagonale du tableau 10.5 ne correspond pas à la notation mathématique usuelle d'une matrice. Elle n'exprime pas une symétrie entre les deux axes, mais exprime une augmentation simultanée de rang: sur les deux axes (selon une échelle ordinale, mais avec des rang irréguliers et non similaires sur les deux axes). L'objectif semble plutôt d'exprimer l'équivalence des rangs lorsqu'ils sont combinés sur deux axes! Par exemple, le plus petit rang sur un axe combiné au plus haut rang sur l'autre axe obtient un nouveau type de rang d'équivalence similaire à l'inverse ((D1,R3) = (D3, R1)!

Le résultat ainsi obtenu dans la méthodologie des points de fonction est un nombre de type nominal, mais il est toutefois traité comme un nombre ordinal! Pour des programmeurs expérimentés, cette façon de définir et de quantifier la complexité des fichiers logiques peut sembler intuitivement valide: toutefois, ceci n'est strictement pas une processus de mesure d'équivalent mathématique!

2d) La prochaine sous-étape consiste à assigner un poids au rang, basé sur la nature du fichier logique (interne, ou externe - interface), tel qu'indiqué au tableau 10.6. Puisque tant les échelles nominales et ordinales sont inutiles dans un processus de mesure quantitative, un ensemble de poids est donc nécessaire pour mesurer (et/ou additionner) les différents types de composantes à l'intérieur d'une échelle nominale. Ceci requiert donc une transformation dans une échelle ratio par l'assignation de poids.

Tableau 10.6 : Poids des fichiers logiques

Types de fichiers	Rang 1: Simple	Rang 2: Moyen	Rang 3: Complexe
Interne	7	10	15
Externe	5	7	10

Ces poids en eux-même représentent des nombres absolus relatifs variant de 5 à 15 selon le type de fichiers logiques, et le nombre final obtenu à cette sous-étape semble alors un nombre valide selon une échelle ratio. Le véritable objectif semble d'assigner des nombres aux sous-ensembles de rangs (non pas des intervalles, parce qu'ils sont irréguliers et non égaux) qui puissent permettre leur utilisation dans un processus de ratio. Par exemple un fichier logique externe dont la position sur la diagonale de la matrice aura un poids de 10, soit deux fois autant qu'un autre fichier logique externe situé au dessus de la diagonale à laquelle est assigné un poids de 5 (chacun des trois sous-ensembles a un poids différent).

Finalement, à la condition que les résultats des étapes précédentes eussent été valides, le résultat final de l'addition aurait produit un résultat valide sur une échelle ratio. Ceci est toutefois un artifice pour transformer une échelle ordinale en une échelle ratio. Même si ce processus semble également intuitivement valide à cause des poids assignés par des experts, ceci n'est pas une transformation mathématique permise.

Même si l'hypothèse est faite que les poids fussent valides (ceci sera commenté dans les paragraphes qui suivent) les opérations mathématiques ne le sont pas. Un nombre ordinal (sous l'hypothèse que la transformation d'échelle des étapes précédentes fussent valides) ne peut pas du point de vue mathématique être transformé soit en échelle d'intervalle ou en échelle de ratio.

3) Dans la dernière étape (Figure 4.2, étape F5), les points obtenus par type de fichiers logiques sont additionnés. Sous l'hypothèse que les nombres obtenus précédemment sont des nombres ratios, le résultat de l'addition serait alors un nombre ratio valide.

10.3 Le processus de mesure des transactions

Tous les commentaires ci-dessus sur les utilisations des échelles de mesure s'appliquent aux processus de mesure décrits pour les transactions (Figure 4.3).

Il faut noter de plus que dans les dernières étapes de la méthodologie, tous les points des cinq types de fonction (Fichiers logiques internes, fichiers logiques externes, entrées, sorties et requêtes) sont additionnés ensemble pour obtenir le total des points de fonction (non ajusté ou bruts). Ceci n'est rendu possible que parce que l'assignation des poids aux divers types de fonction a transformé ces cinq différents types d'objets en un seul type d'objet, différent de ses cinq composantes d'origine, et d'une nature non spécifiée, les *points de fonction*. A nouveau, ceci est accompli par l'entremise de transformations sémantiques, et non pas par un processus

rigoureux de mesure avec une utilisation des transformations mathématiques admissibles selon les différents types d'échelle.

Les résultats finaux (points de fonction non ajustés ou ajustés) deviennent alors difficiles d'interprétation: il y a tellement de dimensions impliquées et tellement d'utilisations des différents types d'échelle de mesure que ce qui semblait simple et intuitivement raisonnable, est en fait un pot-pourri qui pourrait ne pas avoir une signification mathématique valide, à moins évidemment d'avoir démontré, théoriquement et expérimentalement, la validité de chacune des transformations.

Il est douteux qu'une telle comparaison, et ratio, soit mathématiquement valide à cause du mélange de pommes et d'oranges qui occurrent à l'intérieur de la méthodologie des points de fonction. Dans un tel contexte il est donc plus approprié, basé strictement sur l'aspect mathématique, d'appeler le résultat final un *indice*, plutôt qu'une *mesure* de la taille d'une de l'application.

10.4 Le processus des ajustements

L'impact des facteurs d'ajustement est égal, quelle que soit le nombre à l'intérieur d'un facteur d'envergure, et quelle que soit la caractéristique d'envergure (Desharnais, 1990): il y a 14 facteurs, avec chacun une valeur potentielle de zéro à cinq, par pas de un (1), avec chacun un poids relatif de un pour-cent (1%). Ceci est en plus de l'erreur des premiers modèles d'estimation, déjà mentionnée à la section 8.3, qui additionnaient les facteurs au lieu de les multiplier (pour tenir compte de l'interaction des facteurs entre eux) et qui a été répétée dans la méthodologie des points de fonction.

Il y a donc une faiblesse méthodologique importante à la détermination des facteurs d'ajustement. En y regardant de près, il s'agit d'un intervalle de mesure strictement ordinale qui permet de classer parmi cinq catégories à l'intérieur de chacun des quatorze facteurs d'envergure. C'est à partir de là que l'approche méthodologique se gâte, car à chaque classement on confond l'identificateur ordonné croissant du classement (de 0 à 5) avec la valeur numérique de l'identificateur. Au point de vue méthodologique on passe de façon non appropriée de variables ordinales qui servent uniquement à classer, à des variables d'intervalles qui servent normalement à compter.

Subséquemment on utilise ces fausses variables d'intervalle comme poids (donc comme une variable ratio), chacune ayant une valeur de 1%, pour les multiplier par les points bruts de façon à obtenir les points ajustés. Il s'agit donc d'une deuxième erreur méthodologique où les variables intermédiaires qui auraient dues être sur une échelle d'intervalle sont ensuite ré-utilisées comme variables de type ratio dans l'intention de calculer les poids (ce qui aurait donné des nombres de type ratio si toutes les transformations avaient été admissibles).

10.5 Définition de la problématique de recherche

Les sections précédentes ont clairement démontré que, sur une base strictement mathématique, il y avait une utilisation inappropriée des propriétés mathématiques des résultats de chacune des étapes ou sous-étapes des processus de mesure dans cette métrique. Cependant il ne faut pas oublier que le passage d'une étape à l'autre dépend potentiellement de modèles implicites, lesquels pourraient valider la transformation mais qui n'ont toutefois été ni identifiés ni encore moins démontrés comme valides mathématiquement. La revue de littérature dans les chapitres 4 à 8 a toutefois illustré d'autre part que des relations avaient été établies entre la taille des applications, telle que mesurée avec les points de fonction, et la variable des efforts de développement, et ce à l'intérieur de contextes de référence. Il est maintenant possible de formuler des hypothèses de recherche.

Hypothèse 1:

Si la métrique est un système de mesure, et non pas seulement une recette pour construire un nombre sans dimension, alors chacune des étapes de la mesure, du point de départ au résultat final, a une signification et une contribution. Dans un tel cas, chacune des étapes ajoute de l'information: la transposition sémantique de l'étape transcende alors l'opération mathématique et rend l'étape valide par l'entremise de cette transformation.

Hypothèse 2:

Si chacune des étapes est valide, il n'est pas nécessaire de se fier uniquement aux résultats de la dernière étape et il est alors plausible que des résultats intermédiaires aient des relations équivalentes. Il se pourrait même que certaines soient plus significatives si, dans la métrique, les étapes subséquentes font perdre de l'information au lieu d'en ajouter.

Hypothèse 3:

Si les deux hypothèses ci-dessus sont vérifiées, il est alors possible d'utiliser les connaissances acquises de la structure de la métrique pour approfondir la notion de contexte de référence et en étudier des relations supplémentaires dans ce contexte. Si cette hypothèse est vérifiée, la métrique des points de fonction pourrait être utilisée non plus uniquement comme une métrique de taille, mais également comme une métrique de la structure fonctionnelle du logiciel.

Hypothèse 4:

Si la métrique des points de fonction est également une mesure de la structure fonctionnelle de l'application, cette information peut alors être utilisée pour améliorer le processus de modélisation pour les analyses de productivité et d'estimation.

10.6 Proposition de la méthodologie de recherche

10.6.1 Site expérimental

Pour les besoins de cette recherche un site expérimental industriel est disponible au chercheur, ce qui permet un accès privilégié à un ensemble de données réelles d'une institution financière canadienne. Dans cette entreprise la métrique des points de fonction a été implantée à l'été 1987 et a consisté initialement dans le comptage de 11 projets complétés au cours des 12 mois précédents. Puis pour la période de 1988 à 1991, 26 projets supplémentaires ont été mesurés avec cette métrique et sont disponibles pour tester les hypothèses de recherche.

10.6.2 Approche

En même temps que l'implantation des points de fonction dans cette entreprise, un modèle traditionnel de productivité et d'estimation a été introduit, lequel modèle s'inspire du modèle de COCOMO. Cette approche traditionnelle comprend un nombre considérable de facteurs de productivité dont la pondération est établie par des "jugements d'experts" (donc de nature subjective et sans base scientifique de validation) et dont la sélection des valeurs par projet (à l'intérieur des échelles de valeurs pré-déterminées) dépend également du jugement de chaque chef de projet (donc également subjectif).

La méthodologie d'expérimentation pour cette recherche tiendra compte simultanément des deux principes suivants:

A- Définition d'un échantillon de projets homogènes à l'intérieur de la population totale des projets disponibles (homogénéité externe - même domaine)

B- Détermination de moyennes, et de profils moyens, pour analyser le comportement des applications mesurées dont les paramètres (mesures) se situent à l'intérieur d'intervalles pré-déterminés de tolérance. Ceci correspond à la définition d'environnements homogènes de mesure et à l'analyse des comportements dans un contexte d'homogénéité (structurelle interne).

C- Élaboration de modèles qui ne fassent pas appel à des poids et des échelles de valeurs pour les critères à l'intérieur même des algorithmes du modèle, lesquels avaient été précédemment pré-déterminés à partir des jugements d'experts.

D- Comparaison de la fiabilité de cette approche par rapport à l'approche traditionnelle dans ce domaine.

10.6.3 Design expérimental

Le premier design expérimental comprendra un modèle très simple qui sera élaboré à partir uniquement de la moyenne de productivité des projets à l'intérieur de la banque de données. Ce modèle ne fera donc pas appel à des pondérations de facteurs avec des poids subjectifs et ne fera pas non plus appel à des évaluations subjectives de facteurs pour chaque projet.

Les résultats de ce modèle très simplifié seront comparés à ceux du modèle traditionnel, à l'intérieur d'un contexte de données très homogènes, c'est-à-dire un sous-ensemble de 7 projets d'additions/modifications majeures se rapportant à la même application M.I.S. au cours de la périodes 1986 à 1988 (donc un environnement homogène de productivité).

A partir de là, une définition de contexte homogène de référence sera élaborée pour établir des prémisses d'expérimentation et de construction de modèles qui ne fassent pas appel à aucun jugement d'expert ni à aucun processus subjectif d'attribution de valeurs pour les critères à l'intérieur des modèles.

A partir de là, les designs expérimentaux 2 et 3 seront mis sur pied: l'un afin d'examiner les hypothèses de recherche à partir la dernière étape de la métrique des points de fonction, et l'autre pour chacune des étape.

Le design expérimental 2 comprendra la construction de modèles (régressions linéaires, non-linéaires et multiples) à partir du pointage des cinq fonctions et de la comparaison des résultats par rapport au modèle traditionnel qui ne tient compte que du pointage final. Les valeurs des coefficients de régression et de l'erreur type serviront comme critères de comparaison de la précision des résultats. Une analyse de sensibilité sera également effectuée.

Le design expérimental 3 reprendra le même processus que le design 2, mais cette-fois avec chacune des étapes du processus de mesure de la métrique des points de fonction. Les mêmes outils d'analyses seront utilisés. Ce design portera sur un ensemble de données couvrant la période 1986 à 1990, mais devra comprendre un processus d'épuration des données afin de ne conserver que les points se référant à un ensemble homogène d'un contexte de référence: en fonction de l'ensemble de données disponibles et de leurs caractéristiques, le contexte d'homogénéité de cet échantillonnage sera défini comme les projets d'addition/modifications majeures à des applications existantes sur un ordinateur centralisé de grande puissance. Tandis que le modèle à 5 variables devra faire appel aux pointages des cinq fonctions, le modèle à dix variables fera appel au calcul des composantes primaires (fichiers/enregistrements/éléments) comptées à l'intérieur de chacune des cinq fonctions.

Comme ces modèles comprennent un plus grand nombre de variables, des critères additionnels de validation statistiques seront ajoutés, tels que le R**2 ajusté et le coefficient de variation.

Quant aux design expérimentaux 4 et 5, ils aborderont la notion d'homogénéité structurelle des applications et en analyseront les impacts à l'intérieur des modèles développés.

Le design expérimental 4 reprendra l'idée de la comparaison par rapport à une moyenne et comprendra des analyses par rapport aux distributions moyennes des points pour chacun des types de fonction. Des intervalles de variations par rapport à ces moyennes seront ensuite prédéterminés et les analyses de régression seront faites sur les données de l'échantillon qui respectent les conditions d'homogénéité structurelle (en excluant les données s'écartant trop de ces moyennes). Des analyses statistiquesà 1, 2, 4, 5 et 10 variables seront effectuées et leur résultats comparés pour déterminer les facteurs structurels qui expliquent le mieux le comportement de l'échantillonnage dans le modèle choisi.

Le design expérimental 5 reprendra toute la méthodologie du design 4, mais cette fois en excluant tout recours aux valeurs des poids de la méthodologie d'Albrecht et en lui substituant des moyennes basées sur la distribution moyenne des données primaires à l'intérieur de chacun des cinq types de fonction.

La comparaison des résultats des divers designs expérimentaux devrait permettre d'une part de valider, ou d'infirmer, chacune des quatre hypothèses de recherche, et d'autre part d'indiquer, pour cet échantillonnage, lequel des modèles est le plus précis et le plus approprié selon les diverses circonstances.

BIBLIOGRAPHIE

Abdel-Hamid, Tarek K., "The Dynamics of Software Project Staffing: System Dynamics Based Simulation Approach", IEEE Trans. Soft. Engineering, Vol 15, no. 2, Feb. 1989.

Abdel-Hamid, Tarek K., "Investigating the cost schedule tradeoff in Software Development", IEEE Software, Jan. 1990.

Abran, Alain, "A Case Study in Function Point Implementation", 1989 Conference on Improving Productivity in System Development, Applied Computer Research, Phoenix, AZ, Jan. 1989.

Abran, Alain, "L'implantation des points de fonction comme outil de gestion", Présentation au Comité des Responsables Informatiques du Secteur Public, Québec, 6 Oct. 1989.

Abran, Alain, "Démonstration de la faisabilité et de l'utilité de l'implantation des métriques de logiciel des points de fonction dans l'entretien des systèmes informatiques", Séminaire 3.6999, École Polytechnique de Montréal, Dépt. génie électrique, 17 Oct. 1989.

Abran Alain, "The State of the Union - Annual Productivity Report", IFPUG Spring Conference, Orlando, Florida, April 4, 1990.

Abran, Alain, Pierre N. Robillard, "Software Management Based on Software Deliverables", Proceedings CIPS Congress 90, Ottawa, May 17, 1990.

Abran, Alain, Pierre N. Robillard, "Identification of the Structural Weakness of the Function Points Metrics", 3rd Annual Oregon Workshop on Software Metrics, Portland, Oregon, March 18, 1991.

Abran, Alain, Pierre N. Robillard, "Identification of the Structural weaknesses of the Function Points Metrics", Montreal Trust, Technical research report 91-04, Feb. 1991.

Albrecht, Allan, J., "Measuring Application Development Productivity", Proceedings IBM Applications Development Symposium, Monterey, CA., Oct 14-17, 1979.

Albrecht, Allan J., "Application Development and Maintenance Measurement and Analysis Guideline", IBM Corporate Information System and Administration, White Plains, N.Y., 1981.

Albrecht, Allan J., Gaffney John E., "Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation", IEEE Trans. on Soft. Eng.,

Vol. SE-9, no. 6, Nov. 1983.

Albrecht, Allan J., "Function Points Fundamentals", IFPUG Fall Conference, Montréal, Oct. 1988.

Albrecht, Allan J., "Measurement and Function Metrics - A Current Perspective", IFPUG Spring Conference, Florida, April 3, 1990.

Applied Information Development Inc., "The Hierarchy of Metrics, Summary of Key MIS Measurement concepts"

Arthur, Lowell Jay, "Softwrae Evolution - The Software Maintenance Challenge", John Wiley & Sons, 1988.

Balzer, Robert, "What We Do and Don't Know about Software Processes", Software Process Symposium, Washington, Sept. 1990.

Banker, Rajiv D., Kauffman R.J., Wright C., Zweig D., "Automating Output Size and Reusability Metrics in an Object-Based Computer Aided Software Engineering (CASE) Environment", Submitted to IEEE Trans. on Software Engineering, Aug. 25, 1991.

Banker, Rajiv D., Kemerer Chris. F., "Scale Economies in New Software Development", IEEE Trans. on Soft. Eng., Vol. 15, no. 10, Oct. 1989.

Barnard, H. Jack, Meta, Robert F., Price, Arthur L., "A Recommended Practice for Describing Software Design: IEEE Standards Project 1016", IEEE Trans. on Soft. Eng., Vol SE-12, no. 2, Feb. 1986.

Basili, Victor, "Viewing Maintenance as Reuse-Oriented Software Development", IEEE Software, January 1990.

Basili, Victor, "Recent Advances in Software Measurement", IEEE conf. on Soft. Eng., 1990.

Basili, Victor R., "Recent Advances in Software Measurement", 1990 IEEE conference in Software Engineering, Voir Polytechnique, 1990.

Basili, Victor R., "Viewing Maintenance as Re-Used-Oriented Software Development", IEEE Software, Jan. 1990.

Basili, Victor R., Freburger Karl, "Programming Measurement and Estimation in the Software Engineering Laboratory", ?Polytechnique, 1981.

Basili, Victor R., Musa John D., "The Future Engineering Software: A Management Perspective", COMPUTER, IEEE Publications, Sept. 1991.

Basili, Victor R., Selby Richard W., Hutchens David D., "Experimentation in Software Engineering", IEEE Trans. on Soft. ENg. Vol, SE-12, no. 7, July 1986.

Basili, Victor, R., Selby Richard W., Phillips Tsai-Yun, "Metrics Analysis and Data Validation Across Fortran Projects", IEEE Trnas. on Soft. Eng., Nov. SE-9, no. 6, Nov. 1983

Bays, Marianne, David Windom, "Matching A Measurement Program to Your Organization", 3rd International Conf. on Improving Software Quality and Productivity, Quality Assurance Institute, Toronto, Sept. 1988.

Beck, Leland L., Perkins Thomas E., "A Survey of Software Engineering Practice: Tools, Methods, and Results", IEEE Tans. on Soft. Eng., Vol. SE-9, no. 5, Sept. 1983.

Behrens, Charles A., "Measuring the Productivity of Computer Systems Development Activities with Function Points", IEEE Trans. on Soft. Eng., Vol. SE-9, no. 6, Nov. 1989.

Betteridge, Roger, David Fisher, Paul Goodman, "Function Points vs Lines of Code", System Development, August 1990.

Beizer, Boris, "Software System Testing and Quality Assurance", Van Nostran Reinhold Co. Publ., NY, 1984.

Belanger, David D., Chappell & Wish, "Evolution of Software Development Environments", AT&T Technical journal, March/April 1990, p. 2-6.

Belden, Andy, McNamara Don, Paulson Rich, "Function Points Analysis - Management Briefing", IFPUG Spring Conference, Feb. 1989.

Bergland, G. David, Krader H., Smith Paul, Zislis Paul M., "Improving the Fornt End of the Software-Development Process for Large-Scale Systems", AT&T Technical Journal, March-April 1990.

Bhide, Sandhiprakash, "Generalized Software Process-Integrated Metric Framework", 2nd Annual Oregon Workshop on Software Metrics, Portland, Or, March 1990.

Biderman, Bev, "Using Function Points", Computing Canada, Feb. 15, 1990.

Binder, L.H., Poore J.H., "Field Experiments With Local Software Quality Metrics", Software Practice And Experience, John Wiley & Sons, vol. 20, no. 7, July 1990.

Binder, Robert R., Phillips Judith E., "What's all this noise about AD/Cycle?", Computerworld, July 2, p. 53-54, 1990.

Bisant, David D., Lyle James R., "A Two-Person Inspection Method to Improve Programming

Productivity", IEEE Trans. on Soft. Eng., Vol. 15, no 10, Oc. 1989.

Blake, Robert, "Méthodes et outils de mesure reliés à la productivité", Centre Canadien de Recherches en Informatisation du Travail, Juin 1987.

Bock, Douglas B., Klepper Robert, "A Simplified Function Point Counting Method", Dept. Management Information Sciences, Southern Illinois University at Edwardsville, IL, Nov. 17, 1989, IFPUG, Spring Confl. Orlando, FL, April 1990.

Boehm, Barry W., "Software Engineering Economics", Prentice-Hall, 1981

Boehm, Barry W., "Software Engineering Economics", IEEE Trans. on Soft. Eng., Vol. SE-10, no. 1, Jan. 1984.

Boehm, Barry W., Paccacio Philin N., "Understanding and Controlling Software Costs", IEEE Trans. on Soft. Eng. Vol. SE-14, no. 10, Oct. 1988.

Brown, Darlene, "Productivity Measurement Using Function Points", Software Engineering, Auerbach Publ., July-August 1990.

Brown, J.C., Shaw Mary, "Toward A Scientific Basic for Software Evaluation", in Perles, A.J., Sayward, F.G. and Shaw Mary, "Software Metrics: An Analysis and Evaluation", MIT Press, Cambridge, Mass., 1981.

Bryan, William, Siegel Stanley, "Making Software Visible, Operational, and Maintainable in a Small Project Environment", IEEE Trans. on Soft. Eng., Vol. SE-10, no 1, Jan. 1984.

Bush, M.W., "Getting Started on Metrics- JET Propulsion Laboratory Productivity and Quality", 1990 IEEE Conf. on Software Engineering, 1990.

Bush, Marilyn, "Improving Software Quality: the Use of Formal Inspections at the JET Propulsion Laboratory", IEEE Conference, 1990.

Butler, Janet, "A Ruler that Works", System Development, Applied Computer Research Publ. Phoenix, AZ, August 1989.

Butler, Janet, "Beyond Customer Satisfaction", System Development, ACR Publ. Phoenix, AZ, Jan. 1990.

Card, David N., Church Victor E., Agresti William W., "An Empirical Study of Software Design Practices", IEEE Trans. on Soft. Eng., Vol. SE-12, no. 2, Feb. 1986.

Card, David N., Glass Robert, "Measuring Software Design Quality", Prentice Hall, New Jersey, 1990.

Chantilly, Virginia, "Software Metrics in a Process Maturity Framework", 2nd Annual Oregon Workshop on Software Metrics, Portland, Oregon, March, 1990.

Chapin, Ned, "Supervisory attitudes toward software maintenance", National Computer Conference, Vol. 55, AFIP Press, 1986.

Chew, W. Bruce, "No-Nonsense Guide to Measuring Productivity", Harvard Business Review, Jan-Feb. 1988.

Christensen, K., Fitsos G.P., Smith C.P., "A Perspective on Software Science", IBM System Journal, vol. 20, no.4 , 1981.

Cleaveland, J. Craig, Kintala C.M., "Tools for building applications generators", AT&T Technical Journal, March-April 1990.

Comptroller General, "Contracting for Computer Development", General Accountinig Office Report, FGMSD-80-4, 9 Sept., 1979.

Conte, S.D., H.E. Dunsmsore, V.Y. Shen, "Software Engineering Metrics and Models", Benjamin/Cummings Publishing, 1986.

Cook, Michael L., "Software Metrics: An Introduction and Annotated Bibliography", ACM SIGSOFT, April 1982.

Côté, Vianney, Bourque P., Oligny S., Rivard N., "Software Metrics: An Overview of Recent Results", The Journal of Systems and Software, Vol. 8, , 1988.

Côté, Vianney, St-Pierre Denis, "Développement d'un modèle d'estimation de maintenance évolutive fondé sur la méthode de pondération des éléments d'un système d'information", Rapport no. 48, Département de mathématiques et d'informatique, Université de Sherbrooke, Hiver 1989.

Coupal, Daniel, Robillard Pierre N., "Preliminary Results of Factor Analysis of Source Code Metrics", 2nd Oregon Workshop on Software metrics, Portland, Oregon, March 1990.

Coupal & Robillard, "Analyses factorielles des métriques de complexité du code source", 1990.

Coupal, Daniel, Robillard Pierre N., "Factor Analysis of Source Code Metrics", J. Systems Software, 12, 1990.

Crouch, Carolyn J., Crouch Donald B., "The Impact of External Factors on Productivity in an Engineering Support Organization", IEEE Trans. on Eng. Management, vol. 35, no. 3, August 1988.

Cusamano, Michael A., Kemerer Chris F., "A Quantitative Analysis of U.S. and Japanese Software Engineering Practice and Performance", MIT Sloan School Research Report no. 3022-89, June 1989.

Davis, "Planning and Estimating Software", 1989 CASE Outlook 89, no. 2, 1989.

Davis, Alan M., Bershoff E.H., Comer E.R., "A Strategy for Comparing Alternative Software Development Life Cycle Models", IEEE Trans. on Soft. Eng., Vol. SE-14, no. 10, Oct. 1988.

De Grace, Peter, Leslie Hulet Stahl, "Wicked Problems: Righteous Solutions - A Catalog of Modern Software Engineering Paradigms", Yourdon Press Computing Series, Prentice-Hall, 1990.

DeMarco, "Controlling Software Projects, Management, Measurement and Estimation", Yourdon Press, 1982.

DeMarco, Tom, Lister Jim, "PEOPLEWARE: Productive Projects and Teams", Dorset House, 1987.

Desharnais, Jean-Marc, "Analyse statistique de la productivité des projets de développement en informatique à partir de la technique des points de fonction", maîtrise en informatique de gestion à l'Université du Québec à Montréal, Décembre 1988.

Desharnais, Jean-Marc, "Adjustment Model for Function Points Scope Factors - A Statistical Study", IFPUG Spring conf., Florida, April 1990.

Development Support Center Inc., "The Who, What, When Where, Why and How of Function Point Counting", Elm Grove, Wisconsin, 1990.

Dijkstra, Edgar W., "Beyond Structured Programming: An Introduction to the Principles of Applied Software Metrics", Journal of Structured Programming, Vol. 11, no. 1, January 1990.

Dreger, J. Brian, "Function Points Analysis", Prentice-Hall, 1989.

Drucker, Peter F., "The Emerging Theory of Manufacturing", Harvard Business Review, May-June 1990.

Duncan, Mark, "What Gets Measured Gets Done", System Development, ACR Publ. Phoenix, AZ, June 1989.

Duncan, Mark, "Back to Basics", System Development, ACR Publ. Phoenix, AZ, Nov. 1990.

Dunn, Robert H., "Software Quality - Concepts and Plans", Prentice Hall, 1990.

Dunsmore, H.E., "Software Metrics: An Overview of an Evolving Methodology", Information Processing & Methodology, Vol;. 20, no. 1-2, Great Britain, 1984.

Egghe, L, "The duality of informetric systems with application to the empirical laws", Journal of Information Sciences, 16, 1990.

Ehrlich, Weilla K., Lee S.K., Molisani, R.H., "Applying Reliability Measurement: A Case Study", IEEE Software, March 1990.

Emrick, Ronald D. "Software Development Productivity - Second Industry Survey", IFPUG Spring Conference, Dallas, May, 1988.

Emrick, Ronald D. "Further Analysis - Software Development Productivity - Second Industry Survey", IFPUG Fall Conference, Montréal, Sept. 1988.

ESPRIT, projet MERMAID, Survey report, Volume 1: Effort and size estimation models, Feb. 1989.

ESPRIT, projet MERMAID, Survey report, Volume 2 - Software metrics, Feb. 1989.

Factor, Robert M., Smith, William B., "A Discipline for Improving Software Productivity", AT&T Technical Journal, July-August 1988.

Fagan, Michael, "Advances in Software Inspections", IEEE Trans. on Software Engineering, Vol. SE-12, no. 7, July 1986.

Filion, Diane, Maîtrise, Gestion de projet, Université du Québec à Montréal, 1988.

Gaffney, John E., "The Impact on Software Development Costs of Using HOL's", IEEE Trans. on Soft. Eng., Vol. SE-12, no. 3, March, 1986.

Gale, J.L., Tirso J.R., Burchfield C.A., "Implementing the Defect Prevention Process in the MVS Interactive Programming Organization", IBM Systems Journal, vol. 29, no. 1, 1990.

Garrett, William A., "Proving Application Development Productivity and Quality", IFPUG Spring Conference, San Diego, April 1989.

Geoffrey, K., Kemerer Chris F., "Productivity Impacts of Software Complexity and Developer Experience", MIT, March 1990.

Gilbert, Kenneth C., Reeve James M., Wannemacher Richard A., "Improving Information System Efficiency Through Statistical Process Control", Journal of Information Systems Management, Spring 1990.

Gill, Geoffrey K., Kemerer Chris F., "Productivity Impacts of Software Complexity and Developer Experience", MIT Sloan School WP no. 3107-90, Jan. 1990.

Glass, Robert L., "Productivity Crisis - Cited at Conference", System Development, Applied Computer Research Publ., Phoenix, AZ, Feb. 1989.

Glass, Robert L., "Quality Measurement: Two Very Different Ways", Systems Development, Applied Computer Research Publ., Phoenix, AZ, June 1990.

Grady, Robert B., "Measuring and Managing Software Maintenance", IEEE Software, Vol. 4, no. 9, Sept. 1987.

Grady, Robert B., "Work-Product Analysis: The Philosopher's Stone of Software?", IEEE Software, March 1990.

Grady, Robert B., Caswell Deborah L., "Software Metrics: Establishing a Company-Wide Program", Prentice-Hall, 1987.

Grossman, Ira, "Managing Software Growth", System Development, ACR Publ, Phoenix, AZ, Jan. 1990.

GUIDE International Corp., "Measurement of Productivity", Guide Publications GPP-65, 1981.

GUIDE International Corp. "Measuring the Effectiveness of Quality Assurance", GUIDE Publications GPP-117, 1984.

GUIDE International Corp., "Maintenance Productivity Improvements Through Matrices and Measurements", GUIDE Publications GPP-130, 1985.

GUIDE International Corp., "Estimating Using Function Points Handbook", GUIDE Publications GPP-134, 1985, Reprint 1989

GUIDE International Inc., "Improving Software Maintenance Productivity Through Measurement", GUIDE Publications GPP-177, 1987.

GUIDE International Corp., "Application Development Productivity Strategy Paper", 2nd edition, GUIDE Publications GRP-234, 1990

Guimaraes, Tor, "Managing Application Program Maintenance Expenditures", Communication of the ACM, VOl. 26, no. 10, Oct. 1983.

Hadlock, Wayne, "Estimation Earlier with Function Points", IFPUG Fall Conf., San Antonio, Tx, Oct. 1990.

Hager, James A. "Software Cost Reduction Methods in Practice", IEEE Trans. on Soft. Eng., vol. 15, no. 12, Dec. 1989.

Henry, Bill, "Measuring IS for Business Value", Datamation, April 1, 1990.

Henry, Sallie, Kafura Dennis, "Software Structure Metrics Based on Information Flow", IEEE Trans. on Soft. Eng. Vol, SE-7, no. 5, Sept. 1981.

Henry, Sallie, Selig Calvin, "Predicting Source-Code Complexity at the Design Stage", IEEE Software, March 1990.

Hiramaya, Masayuki, Sato H, Yamada A. Tsuda J., "Practice of quality modeling and measurement on software life-cycle", IEEE Software Engineering proc., 1990.

Houses, Charles H. and Raymond L. Price, "The Return Map: Tracking Product Teams", Harvard Business Review, Jan.-Feb. 1991.

Howes, Norman R., "Managing Software Development Projects for Maximum Productivity, IEEE Trans. on Soft. Eng, Vol. SE-10, no. 1, Jan. 1984.

Hufschmidt, Bill, "Proving Productivity - Measurement for Managers", IFPUG Conference Workshop, San Diego, April 1989, summary of workshop in System Development, Phoenix, Az, August 1989.

Humphrey, Watts S., Sweet W.L., "A Method for Assessing the Software Engineering Capability of Contractors", Technical Report CMU/SEI-87-TR-23, Carnegie-Mellon UNiversity & Software Engineering Institute, Preliminary version, Sept. 1987.

Humphrey, Watts S.,"Characterizing the Software Process: A Maturity Framework", IEEE Software, March 1988, p. 73-79.

Humphrey, Watts S.,"Statistically Managing the Software Process", Change: Nes Directions for Statistics and Computing, vol. 2, no. 2, 1989.

Humphrey, Watts, S., "Improving the Software Development Process", Datamation, April 1, 1989.

Humphrey, Watts S., "Software Process Management", International Conference on Project Management, Planning and Estimation, Orlando, Florida, May 23-25, 1990

IBM CIS & a Guidelines 313, "AD/M Productivity Measurement and Estimate Validation", Nov. 1984.

IEC/TC 56 (Bureau Central)158, "Gestion de la fiabilité et de la maintenabilité de logiciel", Mai

1990, Proposition de normes internationales.

Informétric Enr., "Évaluation du modèle gouvernemental d'estimation basé sur la pondération des conposantes d'application", Déc. 1988.

International Function Point Users Group (IFPUG), "Function Point Counting Practices Manual, Release 3.0", IFPUG, Westerville, Ohio, April 1990.

International Function Point Users Group (IFPUG), "Function Points as an Asset - Reporting to Management", IFPUG, Westerville, Ohio, April 1990.

Informétric Enr., "Évaluation du modèle gouvernemental d'estimation basé sur la pondération des composantes d'application", Décembre 1988.

ISO/IEC DIS 9126, "Information technology - Software product evaluation - Quality characteristics and guidelines for their use", mai 1990, propositions de normes internationales.

Jain, Aridaman K., "A Guideline to Statistical Approaches in Computer Performance Evaluation Studies", Performance Evaluation Review, Proceedings of the Software Quality and Assurance Workshop, ACM, San Diego, California, Nov. 1978.

Johnson, James R., "Hallmark's Formula for Quality", Datamation, Feb. 15, 1990.

Johnson, James R., "The Software Factory - Managing Software Development and Maintenance", QED Information Sciences Inc., Wellesley, Mass, 2nd edition, 1991.

Jones, Capers, Measuring Programming Productivity and Quality", IBM Systems Journal, vol. 17, no. 1, 1978.

Jones, Capers, "Programming Productivity", McGraw-Hill, 1986.

Jones, Capers, "Measuring the Economic Productivity of Software", Perspective on Technology, Vo. 2, no. 2, Metropolitan Life, Summer 1988.

Jones, Capers, "Building a Better Metric", Computerworld Extra, June 20, 1988.

Jones, Capers, "Feature Points (Function Point Logic for Real Time and System Software)", IFPUG Fall 1988 Conference, Montreal, Québec, Oct. 1988.

Jones, Capers, "Metric With Muscle - Measuring software productivity in economic terms", System Development, Applied Computer Research Publ, Phoenix, AZ, August 1989.

Jones, Capers, " Cost of a Lifetime", Software Maintenance News, vol 7, no 9, p. 14, Sept. 1989.

Jones, Capers, "Using Functional Metrics to Evaluate CASE", IFPUG Spring Conference, Baltimore, Maryland, April 2-5, 1991.

Julien, Germain, "Les facteurs affectant la productivité des professionnels de la fonction publique du Québec", Optimum, Vol. 20, no. 4, 1989-1990.

Kador, John, "Programmer Productivity", Systems Development, Applied Computer Research Publ. Phoenix, AZ, Feb. 1989.

Kearney, Joseph K., Seldlmeyer, Thompson, & All, "Software Complexity Measurement", Communications of the ACM, Nov. 1986.

Kelly, John P.J., Murphy Susan C., "Achieving Dependability Throughout the Development Process: A Distributed Software Experiment", IEEE Trans. on Soft. Eng., vol 16, no. 2, Feb. 1990.

Kemerer, Chris F., "An Empirical Validation of Software Cost Estimation Models", Communications of the ACM, Vol. 30, no. 5, May 1987.

Kemerer, Chris F. "Function Point Measurement Reliability: Interim Report on the Research Study", IFPUG Spring Conference, Florida, April 1990.

Kemerer, Chris F. "Function Point Measurement Reliability: A Field Experiment", IFPUG Fall Conference, San Antonio, Texas, October 1990.

Kemerer, Chris. F., Reliability of the Function Point Measurement: a Field Experiment", to be published in the Communications of the ACM, 1991.

Kenny, Graham K., Dunk Alan S., "The Utility of Performance Measures: Production Managers' Perceptions", IEEE Trans. on Eng. Management, vol. 36, no 1, Feb. 1989.

Knaff, G.J., Sacks J., "Software Development Effort Prediction Based on Function Points", Proceedings, COMPSAC '86, Chicago, Il. 1986.

Knight, Caroline, "Starting an FP Program", System Development, Appied Computer Research Publ. Phoenix, AZ, August 1989.

Koch, Warren B., "Function Points at Bell Canada", System Development, ACR Publ. Phoenix, AZ, August 1989.

Koch, Warren B., "Productivity Results RE Function Points", IFPUG Fall Conference, 1989.

Laranjeira, Luiz A., "Software Size Estimation of Object-Oriented Systems", IEEE, Trans. on

Soft. Eng., Vol. SE-16, no. 5, May 1990.

Le Gall & all, "Atelier de logiciel: bilan et perspectives", 4ième colloque - exposition Génie Logiciel, Paris, Oct. 1988.

Le Groupe DMR Inc., "Points de Fonction - Informations de Base", DMR, Montréal, Juin 1990.

Leclaire, Serge, "Amélioration de la productivité: un mythe ou une réalité?", Journée Informatique du Québec, Fédération Informatique du Québec, Montréal, Avril 1990.

Lehder, Wilfred E., Smith, D. Paul, Weider, D. Yu, "Software Estimation Technology", AT&T Technical Journal, July-August, 1988.

Leonard, Sue, "Selling A Measurement Program", 3rd Internation Conf. on Improving Software Quality and Productivity, Quality Assurance Institute, Toronto, Sept. 1988.

Levendel, Ytzhak, "Reliability Analysis of Large Software Systems: Defect Data Modeling", IEEE Trans. on Soft Eng., Vol 16, no. 2, February 1990.

Lewis, John, Sallie Henry, "A Methodology for Integrating Maintainability Using Software Metrics", Proceedings Conf. on Software Maintenante - 1989, Miami Florida, IEEE, CH2744-1/89, 1989.

Low, Graham C., Jeffery D. Ross, "Function Points in the Estimation and Evaluation of the Software Process", IEEE Trans. on Soft. Eng., Vol. 16, no. 1, Jan. 1990.

MacDonell, S.G., "Rigor in Software Complexity Measurement Experimentation", Journal of Systems Software, Vol. 16, no. 2, Oct. 1991.

Martin, James, "Rapid Application Development", McMillan Publishing, New York, 1991.

Martin, James, McLure Carma, "Software Maintenance - The Problem and Its Solution", Prentice-Hall, 1983.

Mays, R.G., Jones C.L., Holloway G.J., Studinski D.P., "Experiences with Defect Prevention", IBM Systems Journal, vol. 29, no. 1, 1990.

Mazzucco, Frank A., "Automation of Function Point Counting - An Update", IFPUG, Spring Conference, Orlando, Floride, April 1990.

McCabe, Thomas J., Butler, Charles, W., "Design Complexity Measurement and Testing", Communications of the ACM, VOl. 32, No. 12, Dec. 1989.

McCall, J., Markham D., Stosick M., McGindllllly R., "The Automated Measurement of

Software Quality", IEEE Conf. Proceeding, CH1698-0-81, 1981.

McNamara, Don, "IFPUG Survey of Function Point Use for Management Decisions", IFPUG Fall Conference, Montréal, Oct. 1988.

McKee, James R., "Maintenance as a function of design", Proceedings of the 1984 National Computer Conf. Vol. 53, AFIPS Press, 1984.

McQuilken, George, Tomesch Frank, "Metrics at Metropolitan", Perspectives on Technology, Vol. 2, no. 2, Metropolitan Life, Summer 1988.

Meredith, Denis C., "A view form the field", System Development, ACR Publ. Phoenix, AZ, August 1989.

Miller, James C., "Measurement Using Function Point Analysis", IFPUG Spring Conf. April 1989.

Mills, Everald E., "Software Metrics", SEI Curriculum Module SEI-CM-12-1.0, Oct. 1987

Mills, H.D., "Software Productivity", Dorset House Publ., N.Y., 1988.

Mills, Harlan D., Dyson, Peter, B., "Using Metrics to Quantify Development", IEEE Software, March 1990.

Mohanty, Siba N., "Entropy Metrics for Software Design Evaluation", The Journal of Systems and Soft., 2, 1981.

Moineau, Patrick, "Le contrôle qualité des logiciels, un outil de l'assurance produit", Génie logiciel & syst;emes experts, No. 10, janvier 1988.

Moreau, Dennis R., Dominick, Wayne D., "A Programming Environment Evaluation Methodology For Object-Oriented Systems: Part I - The Methodology", JOOP May/June 1990.

Musa, J.D., Iannino, A., Okumoto, K., "Software Reliability, Measurement, Prediction, Application,"McGraw-Hill, 1987.

Navlaka, Jai, "Software productivity metrics: Some candidates and their evaluation", Proc. of National Computer Conference, 1986.

Nejmeh, Brian A., "Open Issues in Process-Based SDEs (PSDEs)", Software Process Symposium, Washington, Sept. 1990.

Odom, J.E., "Using box structures for definitions of requirements specifications", IBM Systems Journal, Vol. 29, no. 1, 1990.

Oligny, Serge, "Estimateurs de la taille d'un logiciel à partir des spécifications de conception", Mémoire M.Sc., Département mathématiques et informatique, Université de Sherbrooke, mars 1988.

Osterweil, Leon, "Application of Software Process", Software Process Symposium, Washington, Sept. 1990.

Parikh, Girish, "Programming Productivity", Reston Publishing, Virginia, 1984.

Parnas, David L., Clements Paul C., "A Rational Design Process: How and Why to Fake It", IEEE Trans. on Soft Eng., Vol. SE-12, no. 2, Feb. 1986.

Penedo, Maria H., "Acquiring Experience with the Modeling and Implementation of the Project Life-cycle Process", Software Process Symposium, Washington, Sept. 1990.

Perry, Deqayne, "Policy and Product Directed Process MOdels", Proceedings Software Process Symposium, Washington, D.C., Sept. 1990.

Pfleeger, Shari Lawrence, "An Investigation of Costs and Productivity for Object-oriented Development", Ph.D. thesis, George Mason University, Ann Arbor, Mi. 1989.

Pfleeger, Shari Lawrence and James D. Palmer, "Software Estimation for Object-Oriented Systems", IFPUG Fall Conference, San Antonio, Texas, October 1-4, 1990.

Pickford, Robert C., "IFPUG: Past, Present, and Future", Systems Development, Applied Computer Research Publ. Phoenix, AZ, August 1989.

Porter, Benjamin, "New Directions", System Development, ACR Publ. Phoenix AZ, August 1989.

Porter, Benjamin, "Function Point Measures - A Critical Comparison", 8th QAI International conf. on Measuring, Orlando Fl, March 1990.

Porter, Benjamin, "Using CASE to Count", IFPUG Spring Conf. Proceedings, Lake Buena Vista, Florida, April 1990.

Porter, Adam A., Selby Richard W., "Empirically Guided Software Development Using Metric-Based Classification Trees", IEEE Software, March 1990

Pressman, Roger S., "Software Engineering, A Practitioner's Approach", 2nd edition, McGraw-Hill, 1987.

Pressman, Roger S. "Making Software Engineering Happen, A Guide for Instituting the Technology", Prentice Hall, 1988.

Putnam, Lawrence H. "Tutorial - Software Cost Estimating and Life-Cycle Control: Getting the Software Numbers", IEEE, NY., 1980.

QED Information Sciences Inc., "CASE' The Potentioal and the Pitfalls", QED Inforamtions Sciences, Wellesley, Mass., 1989.

Quality Assurance Institute, "Quantitative Problem Analysis - Using Key Indicators to Improve Information Systems Quality and Productivity", Video program workbook, QAI, Orlando, F., 1990.

Ramsey, Connie L., Basili Victor R., "An Evaluation of Expert Systems for Software Engineering Management", IEEE Trans. on Soft. Eng., Vol. SE-15, no. 6, June 1989

Reifer, Donald J., "Real-time Function Point Extensions", IFPUG Spring Conference, Baltimore, Maryland, April, 1991.

Roberts, John, W., "Systems Support of Just-In-Time Manufacturing", Journal of Information Systems Management, Winter 1990.

Robillard, Pierre N., "Le logiciel: de sa conception à sa maintenance", Gaëtan Morin éditeur, 1985.

Robillard, Pierre N., "On the evolution of Graphical Notation for Program Design", ACM SIGSOFT Software Engineering Notes 14, no. 1, Jan. 1989.

Rodhani, Adellatif, Robillard Pierre N., "Contrôle et mesure de la qualité des logiciels", CIPS Session 85, 1985.

Rombach, H. Dieter, Ulery, Bradford T., "Establishing a Measurement Based Maintenance Improvement Program: Lessons Learned in the SEL", Proceedings, Conf. on Softwaare Maintenance - 1989, IEEE, CHG2744-1/89, 1989

Rombach, Dieter H., "Design Measurement: Some Lessons Learned", IEEE Software, March 1990.

Rubin, Howard A., "Macro-Estimation of Software Development Parameters: the ESTIMACS System", Proceeding SOFTFAIR 83 Conf. on Soft. Dev. Tools, Techniques and Alternatives, Washington, July, 1983.

Rubin, Howard A., "Software Engineering in the 1990's: Business Oriented Measurement", IFPUG Spring Conf. Proceedings, Lake Buena Vista, Florida, April 1990.

Rudolph, E.E., "Productivity in Computer Application Development", Dep. Management Studies, Univ. Auckland, Australia, 1983.

Rudolph, E.E., "Precision of Function Point Counts", IFPUG Spring Conference, SanDiego, CA, April 1989.

Saindon, Marc, "La producivité et l'estimation de la taille et de la durée des projets réalisés avec une approche de développement de systèmes par les informations primaires", essai MBA, Sciences de l'administration, Université Laval, Février 1989.

Schofield, Joseph R., "Standardizing Complexity Characteristics in Function Point - A Process Improvement", 8th QAI International Conference on Measuring, Orlando, Fa, March 1990.

Schussel, George, "Applications Development: Tools of the Trade", Computerworld, Feb. 3, 1988.

SECOR, The Markets for a Canadian Software Engineering Initiative to Serve the Airospace and Defence Industries, **Draft**", Montréal, August 30, 1990.

Selby, Richard W., Porter, Adam A., "Software Metric Classification Trees Help Guide the Maintenance of Larfge-Scale Systems", Proceedings Conf. on Software Maintenance - 1989, Miami, Florida, IEEE CH2744-1/89, 1989.

Shen, Vincent Y., Conte Samuel D., Dunsmore, H.E., "Software Science Revisited: A Critical Analysis of the Theory and Its Empirical Support", IEEE Trans. on Soft. Eng., Vol. SE-9, no. 2, March 1983.

Shen, Vincent, "Using metrics in quality management", IEEE Software, JUly 1990.

Shinn, Jon, "Measuring the Inspection Process with Regard to Project Size and Life Cycle Phases", QAI Conference, Orlando, FL, April 1990.

Sirkin, Harold, Stalk George Jr., "Fix the Process, Not the Problem", Harvard Business Review, July-August 1990.

Snow, John R., "Management Reporting Using Function Points", System Development, ACR Publ. Phoenix, AZ, August 1989.

Solo, Patricia A., Omachonu, Vincent K., Davis, Elda M., "Computerizing Productivity Measurement for Service Installation Companies", Computers & Industrial Engineering, Vol. 17, no. 1-4, 1989.

Sousa, Eilen, "The Impact of CASE on Traditional Software Development", Handbook of Systems Management - Development and Support, 1990-91 Yearbook, Chapter II-2, Auerbach Publishers, 1990.

Srinivasan, Annanth, Kaiser, Kate M., "Relationships between Selected Organizational Factors

and Systems Development", Communications of the ACM, Vol. 30, no. 6, June 1987.

Statland, Norm, "Meeting Project Deadlines is the Top Priority", Datamation, April 1, 1990.

Swanson, E. Burton, Mathis Beath, Cynthia, "Maintaining Information Systems in Organizations", John Wiley & Sons, 1989.

Symons, Charles R., "Function Points Analysis: Difficulties and Improvements, IEEE Trans. on Soft. Eng., Vol. SE-14, no. 1, January 1988.

Symons, Charles R. "Mark II Function Points For Productivity Measurement & Estimating", Quality Assurance Institute: International Conf. on Project Management, Planning & Estimating, FL, May 1990.

Systemhouse, "Rapport d'intervention sur la méthode PCA au gouvernement du Québec, document de travail, novembre 1989.

Tagg, B.S., "Implementing tool support for box structures", IBM Systems Journal, Novl. 29, no. 1, 1990.

Taguchi, Genichi, Clausing Don, "Robust Quality", Harvard Business Review, Jan.-Feb., 1990.

Tague, Jean, "Ranks and size: some complementarities and contrasts", Journal of Information Sciences, Vol. 16, No. 1, 1990.

Tate, Graham, Verner June, "Software Metrics for CASE Development", COMPSAC 91 Proceedings, Tokyo, Sept., 1991

Thurlow, John, "Effects of Measuring", IFPUG Conference, Spring, 1989, reported in Systems Development, Phoenix, Az, August, 1989.

Timpe, A. Dale, "Productivity, The Art and Science of Business Management", KEND Publishing, N.Y., 1989.

Vo, Kiem-Phong, "IFS: A Tool to Build Application Systems", IEEE Software, July 1990.

Warburton, Roger D.H., "Managing and Predicting the Costs of Real-Time Software", IEEE Trans. on Soft. Eng., Vol. SE-9, no. 5, Sept. 1983.

Watson, C.E., Felix C.P., "A Method of Programming Measurement and Estimation", IBM System Journal, vol. 16, no. 1, 1977.

Weinberg, Gerald M., Freedman Daniel P., "Reviews, Walkthroughs, and Inspections", IEEE Trans. on Soft. Eng., Vol SE-10, no. 1, Jan. 1984.

Weir, Bob, "Implementing Quality and Productivity Measurements", 3rd International Conf. on Improving Software Quality and Productivity", Quality Assurance Institute, Toronto, Sept. 1988.

Whisehunt, Charlie, "How to Implement Function Points Both Ways (Right and Wrong) and still Survive", 8th QAI International Conference on Measuring, Orlando, Fl, March 1990.

Wilkinson, Bryan, "How to Estimate Programming Time", Handbook of Systems Management - Development and Support, Auerbach Publishers, 1989.

Wrigley, Clive Donald, A model and method for measuring information systems size, Ph.D. Dec. 1988, UBC

Wybouv, George, Kanaan Richard, "Office Automation and Productivity Measurements: Some Thoughts!", Conf. Administration Sciences Ass. of Canada, Whistler, B.C., June 1986.

Yau, Stephen S., Jeffery J.P. Tsai, "A Survey of Software Design Techniques", IEEE Trans. on Soft. Eng., Vol. SE-12, no. 6, June 1986.

Zuse, Horst, "Software Metrics - Methods to Investigate and Evaluate Software Complexity Measures", 2nd Annual Oregon Workshop on Software Metrics, Portland, Oregon, March 19, 1990.

Zuse, Horst, "Software Complexity - Measures and Methods", Walter de Gruyter, Berlin, New York, 1991.

Zwanzig, Kenneth, "Estimating Measurement", GUIDE 56 Proceedings, 1985.

ÉCOLE POLYTECHNIQUE DE MONTRÉAL
3 9334 00289715 3

DE LA METRIQUE DES POINTS DE FONCTION

CA
UE
R9
ex