CHAPITRE 5

LES MESURES DE LA TAILLE FONCTIONNELLE (MTF)

«Une science a l'age de ses instruments de mesure » Louis Pasteur.

Au début de ce chapitre, un aperçu historique sur les classes de mesures des logiciels est présenté. Puis une discussion est engagée sur le concept des points de fonction et de la taille fonctionnelle. Ensuite, une comparaison est discutée entre la méthode de Lignes de Code (LOC) qui permet le comptage du nombre de lignes de code d'un programme et la méthode des points de fonction. Ce chapitre décrit aussi la méta-norme de mesurage de la taille fonctionnelle - ISO 14143 et les MTF qui ont été reconnues comme des normes ISO. Enfin, un rapprochement est aussi réalisé entre deux de ces normes ISO, soit COSMIC-FFP et IFPUG.

5.1 Introduction

McCabe et Butler (1998) proposent trois grandes classes essentielles de mesures des logiciels; la complexité cyclomatique, le nombre de lignes de code et le nombre de points de fonction.

D'autre part, selon Fenton (1997), le logiciel est une entité physique qui peut être mesurée par sa taille, puisque les objets physiques sont facilement mesurables. L'opération de mesurer la taille d'un logiciel devrait être facile. Cependant, la mesure de la taille des logiciels présente beaucoup de difficultés, car il n'y a pas de compréhension profonde des attributs des logiciels pas plus que de disponibilité d'outils sophistiqués de mesure (N. Fenton et S.L. Pfleeger, 1997). Fenton *et al.* suggèrent que la taille des logiciels peut être décrite par trois attributs à savoir : la longueur, la fonctionnalité et la complexité.

La longueur est la taille physique du produit logiciel alors que la fonctionnalité mesure les fonctions fournies par le logiciel à l'utilisateur (N. Fenton et S.L. Pfleeger, 1997). Quant à la complexité, elle peut être interprétée de plusieurs façons. La complexité du problème mesure la complexité du problème à résoudre par le logiciel. La complexité algorithmique reflète la complexité de l'algorithme implanté pour résoudre le problème. La complexité structurelle mesure la structure du logiciel exécutant l'algorithme. La complexité cognitive mesure l'effort exigé pour comprendre le logiciel. D'après Fenton (1997), ces interprétations font de la complexité un attribut difficile à mesurer. De plus, il faudrait faire une distinction entre la complexité du problème et la complexité de la solution.

La méthode des lignes de code (LOC) peut être définie sommairement comme le nombre de lignes source écrites dans un programme; son unité est 1 ligne de code. Jones (1978) souligne que cette mesure de logiciel présente quelques problèmes fondamentaux. Le problème principal consiste à savoir comment déterminer précisément une ligne de code. Les lignes de code des applications sont formées des déclaratifs, des appels, des instructions et des commentaires. De plus, les lignes de code contiennent parfois des instructions complexes qui peuvent prendre plusieurs lignes physiques de code.

La méthode des lignes de code peut être définie de façon différente d'une entreprise à une autre : tantôt, une entreprise ne regarde que les instructions exécutables, dans d'autres situations une entreprise ne compte que les instructions et les déclarations, et quelquefois on considère toutes les lignes non blanches dans le code. À cause de ces disparités dans les façons de mesurer les lignes de code, il est délicat de comparer la taille des logiciels en utilisant uniquement cette méthode de mesure de la taille d'un logiciel. De plus, les différences entre les langages de programmation rendent cette méthode de mesures encore moins pertinente (Fenton *et al.*, 1999).

D'après Zuse, la méthode de lignes de code a été l'objet de critiques très sévères : la critique la plus évidente, c'est que cette technique ne mesure pas les fonctions du

système et ses attributs : *The obvious criticism of the simple Measure LOC is that it does not really measure the systems functions and features* (Horst Zuse, 1998).

5.2 Les points de fonction

Allan Albrecht d'IBM a identifié, au cours des années 1970, la nécessité d'avoir une méthode d'estimation de l'effort déployé pour réaliser un logiciel qui ne dépendrait pas des langages de programmation ni des techniques utilisées pour développer ce logiciel. Comme résultat, Albrecht (1979) a inventé la méthode des points de fonction (*Function Points Analysis – FPA*). C'était une idée originale de proposer, pour la première fois, une mesure des logiciels fondée sur les fonctions des utilisateurs. En résumé, les points de fonctions se basent sur l'analyse et la mesure des fonctionnalités livrées à l'utilisateur. Par design, cette méthode de mesure n'a pas certains des problèmes de la mesure LOC. La définition de la méthode des points de fonction a été discutée par beaucoup d'auteurs tels que (A. Abran, 1994; B. Kitchenham *et al.*, 1993; C. Jones, 1996; C. Symons, 1988; C.F. Kemerer, 1993; D. St-Pierre *et al.*, 1997; D.J. Reifer, 1991; J.B. Dreger, 1989; K. Paton *et al.*, 1995; M. Shepperd, 1994; R. Banker *et al.*, 1994; S. Galea, 1995; S.A. Whitmire, 1995; S.L. Pfleeger *et al.*, 1990).

Les objectifs d'utilisation des points de fonction sont présentés au tableau 15.

Tableau XV

Objectifs d'utilisation des points de fonction (IFPUG, 1994)

Version française (Abran)	Version anglaise IFPUG
1 Mesure ce que l'utilisateur a demandé et reçu	1-Measure what the user requested and received
2 Mesure indépendamment de la technologie implantée	2-Measure independently of technology used for implementation
3 Obtenir une mesure de taille pour permettre l'analyse de la qualité et de la productivité	3-Provide a sizing metric to support quality and productivity analysis

Tableau XVI (Suite)

Objectifs d'utilisation des points de fonction (IFPUG, 1994)

Version française (Abran)	Version anglaise IFPUG
4 Obtenir un véhicule pour l'estimation du logiciel	4-Provide a vehicle for software estimation
5 Obtenir un facteur de normalisation pour comparer les logiciels	5-Provide a normalization factor for software comparison

5.3 Lignes de code ou points de fonction

Dans son article *Function Points or Lines of Code? – An Insight*, Kurmanadham (2004) apporte la comparaison suivante entre les lignes de code et les points de fonction :

5.3.1 Les avantages de la méthode des lignes de code

1. *Scope for Automation of Counting* : puisque la ligne de code est une entité physique, l'effort du comptage manuel peut être éliminé en automatisant le processus de comptage.
2. *An Intuitive Metric* : La ligne de code sert de métrique intuitif pour mesurer la taille du logiciel parce qu'elle est visible et que son effet peut être visualisé.

5.3.2 Les inconvénients de la méthode des lignes de code

1. *Lack of Accountability* : il est incorrect de calculer la productivité d'un projet de développement avec les résultats d'une des phases (phase de codage) qui représente habituellement de 30 à 35 % de l'effort (Kurmanadham, 2004).
2. *Lack of Cohesion with Functionality* : Les développeurs expérimentés peuvent développer la même fonctionnalité avec moins de lignes de code que des programmeurs novices : ainsi un programme avec moins de LOC peut donner autant de fonctionnalité qu'un autre programme avec plus de LOC.
3. *Adverse Impact on Estimation* : Comme corollaire au point(a), les estimations faites basées sur des lignes de code peuvent être biaisées.
4. *Developer's Experience* : Un développeur expérimenté peut mettre en application certaines fonctionnalités en moins de lignes de code qu'un autre développeur qui a relativement moins d'expérience, bien qu'ils utilisent le même langage de programmation.

5. *Difference in Languages* : Considérer deux applications qui fournissent la même fonctionnalité (écrans, rapports, bases de données), une est écrite en C++ et l'autre est écrite en COBOL. Le nombre de points de fonction serait exactement identique, mais les lignes du code requises pour développer la même application ne seraient probablement pas identiques.
6. *Advent of GUI Tools* : Avec l'arrivée des langages et des outils basés sur l'Interface Utilisateur Graphique tel que le Visual Basic, beaucoup de travail de développement est effectué par « cliquer-et-glisser » où le programmeur n'écrit pratiquement aucun morceau de code. Cette différence rend la mesure avec les lignes de code moins pertinente pour les mesures de productivité.
7. *Far from OO Development* : La ligne de code ne fait aucune signification dans le cas du développement Orienté Objet où tout est traité en termes d'objets et de classes. Puisque l'objet est une représentation des données et des fonctionnalités, les points de fonction sont plus appropriés pour les logiciels développés en environnement Orienté Objet.
8. Problems with Multiple Languages: Maintenant dans le développement des logiciels, plusieurs langage de programmation sont utilisées. Dans ce cas la détermination et la traçabilite des taux de productivité et de défauts sont difficiles. Puisque les défauts ne peuvent pas être attribués à un seul langage ceci est due à l'intégration du système. Les points de fonction s'apprêtent d'être la meilleure mesure de taille dans ce contexte. (Kurmanadham, 2004)
9. *Lack of Counting Standards* : Il n'y a aucune définition standard d'une ligne de code.

5.3.3 Les avantages de la méthode des points de fonction

Les points de fonction fournissent une méthode pour mesurer la taille des logiciels, et cette mesure peut être utilisée pour gérer la taille pendant le développement. « La méthode des points de fonction est largement utilisée dans le domaine des systèmes d'informatique de gestion » (Banker *et al.*, 2002; Desharnais *et al.*, 1998). Elle permet :

1. *Helps Comparison* : La seule variable est la quantité d'effort requise pour fournir un ensemble de points de fonction; cette méthode est, par design, indépendante des technologies et des méthodologies de développement; la méthode peut donc être employée pour déterminer si un outil, un environnement, un langage de programmation est plus productif comparé à d'autres dans une même organisation ou par rapport à d'autres organisations.
2. *Helps Monitor Scope Creep* : Le comptage des points de fonction à la fin de la phase de spécification ou de design peut être comparé aux points de fonction réellement fournis et implantés à la fin d'un projet. Si la taille du projet a grandi,

il y a eu changement de portée et ce changement de taille peut être mesuré avec les points de fonction.

3. *Ease of Contract Negotiations* : Pour le client, les points de fonction peuvent être employés pour aider à indiquer et à quantifier pour un fournisseur, les principaux livrables et ce, sur une base commune et acceptée par l'industrie.
4. Handling Volatility : L'avantage est que les points de fonction fournit des mesures a partir des exigences du logiciel. Donc, les points de fonction permet de montrer l'état actuel des exigences. Quand une nouvelle caractéristique est ajoutée, le total des points de fonction augmente en conséquence. Si l'organisation décide d'enlever des caractéristiques, le total des points de fonction change en reflétant l'état réel des caractéristiques du logiciel. (Kurmanadham, 2004)
5. *Use of Historic Data* : Une fois que la taille de projet a été déterminée en points de fonction, des évaluations pour la durée, l'effort, et d'autres coûts peuvent être calculées en employant des données historiques.
6. *Availability of Empirical Formulae* : Les points de fonction peuvent être employés plus efficacement pour développer beaucoup de formules prédictives telles que le taux de défaut et l'effort de maintenance.
7. *Enables Better Communication* : Les points de fonction peuvent améliorer la communication avec la haute direction puisqu'ils parlent en termes de fonctionnalité et non pas en termes de détails d'exécution, d'aspects techniques ou de code physique.
8. *Offers Better Benchmarking* : Puisque les points de fonction sont indépendants du langage de programmation, de la méthodologie de développement, des pratiques de programmation, et du domaine technologique, les projets utilisant les points de fonction sont de meilleurs candidats pour faire du *benchmarking* à travers les organismes.

5.3.4 Les inconvénients de la méthode des points de fonction

Kurmanadham (2004) souligne que les points de fonction offrent de nombreux avantages en donnant la taille du logiciel du point de vue fonctionnel. Cependant, la méthode a quelques inconvénients :

1. *Requires Manual Work* : En raison de sa nature, les points de fonction doivent être comptés manuellement. Le processus de comptage est difficilement automatisable.
2. *Necessitates Significant Level of Detail* : L'information a un niveau de granularité très fin sur les entrées, les sorties, les écrans, les tables de base de données, et même les enregistrements et les champs sont nécessaires pour faire la mesure.

3. *Requires Experience* : L'analyse de points de fonction exige de l'expérience pour avoir des mesures précises.

De plus, selon des auteurs tels que (Desharnais *et al.*, 1998; Galea, 1995; C. Jones, 1991; Maya *et al.*, 1996; Whitmire, 1992) la méthode est moins adéquate pour les logiciels qui ont plusieurs fonctions de contrôle ou plusieurs opérations de calculs, tels que les logiciels temps réel. Pour ces types de fonctionnalité, cette méthode ne tient pas compte des sous-processus fonctionnels à l'intérieur d'un processus de contrôle, ce qui génère des mesures fonctionnelles trop petites ne reflétant pas de façon précise la taille fonctionnelle des logiciels en temps réel. Pour remédier à certains de ces défauts, le Consortium COSMIC (Common Software International Consortium) a développé la méthode de mesure COSMIC-FFP.

5.4 La méta-norme ISO 14143

À la fin des années 1990, une norme générique ISO en mesure de taille fonctionnelle a été créée pour définir ce qu'est une mesure de taille fonctionnelle - MTF (ISO/IEC 14143-1, 1998) et comment en vérifier certaines caractéristiques de qualité. Le but de cette norme est de fournir un ensemble de documents (internationalement acceptés comme normes ou rapports techniques) qui décrivent le concept et la pratique de la MTF. Sur la base de critères définis dans cette norme générique ISO, plusieurs méthodes spécifiques de mesures fonctionnelles ont par la suite été acceptées comme norme spécifiques ISO reconnues comme rencontrant les critères obligatoires spécifiés dans la norme générique de ISO 14143-1. La norme COSMIC-FFP (ISO/IEC 19761, 2002) est un exemple d'une norme spécifique en MTF.

ISO 14143 est une norme générique (qui pourrait également être appelée une métanorme) dont le titre est « Technologies de l'information - Mesurage du logiciel - Mesurage de la taille fonctionnelle » et qui est composée de six parties, les deux premières ayant un statut de normes ISO et les quatre autres le statut de rapport technique ISO. La publication de ces parties s'est effectuée de 1998 à 2005 :

Partie 1 — Définition des concepts (ISO/IEC 14143-1, 1997)

ISO/IEC 14143- 1 définit les concepts fondamentaux de la MTF des logiciels et décrit les principes généraux pour le design d'une méthode MTF. Cette partie constitue le repère contre lequel toutes les méthodes MTF doivent être examinées pour s'assurer qu'elles se conforment à ses critères.

Partie 2 – Évaluation de la conformité des méthodes MTF des logiciels (ISO/IEC 14143-2, 2001)

ISO/IEC 14143-2 est une norme de conformité qui propose une méthodologie pour examiner si une méthode a les caractéristiques exigées selon la norme de référence 14143-1. Cette partie exige un processus rigoureux d'analyse, un rapport complet et une partie indépendante pour vérifier la conformité de la méthode plutôt que de se contenter des déclarations des fournisseurs. Le résultat de ce processus peut aider les éventuels utilisateurs d'une méthode candidate de MTF pour juger si elle est appropriée à leurs besoins.

Partie 3 – Vérification des méthodes MTF (ISO/IEC 14143-3, 2002)

La partie 3 est un rapport technique ISO développé pour fournir un processus afin d'aider l'utilisateur à choisir une méthode appropriée à ses besoins, en fournissant un processus pour vérifier à quel point les déclarations faites pour certaines propriétés d'une méthode MTF sont vraies. Cette partie définit :

1. un cadre pour vérifier certaines propriétés d'une méthode MTF
2. plusieurs propriétés contre lesquelles une méthode MTF peut être vérifiée
3. les types d'essais qui peuvent être réalisés
4. le processus pour la vérification d'une méthode MTF
5. des exemples pour le rapport de vérification.

La partie 3 d'ISO/IEC 14143 établit un cadre pour vérifier les propriétés d'une méthode MTF par rapport aux critères suivants :

1. répétabilité et reproductibilité;
2. exactitude;
3. convertibilité;
4. seuil de tolérance;
5. applicabilité aux domaines fonctionnels.

La partie 3 vise à ce que le résultat de la vérification soit objectif, impartial, conforme et répétable. Le rapport de vérification, produit en conformité avec cette partie d'ISO/IEC 14143, permettra à un éventuel utilisateur de choisir la méthode MTF qui répond bien à ses besoins.

Partie 4 – Modèle de référence (ISO/IEC 14143-4, 2000)

D'après l'*International Software Benchmarking Standards Group* (ISBSG, 2005), le but de cette partie est de fournir des points de référence aux utilisateurs du processus de validation pouvent évaluer l'efficacité d'une méthode FSM pour différents types de logiciels et dans plusieurs environnements.

Ce rapport technique peut permettre :

- Aux développeurs d'une méthode de FSM de tester les domaines fonctionnels dans lesquels leur méthode peut être utilisée efficacement et de raffiner leur méthode FSM;
- Aux personnes vérifiant une méthode de FSM, d'avoir des points de référence pour appliquer et pour comparer des méthodes FSM;
- La réduction de l'utilisation inadéquate de quelques méthodes courantes de mesure des logiciels;
- Une meilleure comparaison des données utilisées pour le *benchmarking* de qualité et de productivité;
- Aux mesureurs de vérifier qu'une méthode particulière de FSM est une méthode efficace;
- Aux mesureurs d'avoir un ensemble d'exigences d'utilisateurs fonctionnelles dans différents domaines fonctionnels.

Partie 5 – Détermination des domaines fonctionnels pour les utiliser avec les MTF (ISO/IEC 14143-5, 2004)

Selon l'*International Software Benchmarking Standards Group* (*ISBSG*, 2005), le but de cette partie 5 est d'établir une norme pour classifier les fonctionnalités utilisateurs requises pour utilisation dans l'application des FSM.

Ce rapport technique permet aux :

- Utilisateurs des FSM d'évaluer les caractéristiques de leurs fonctionnalités utilisateurs requises et de les classer par catégorie de domaine fonctionnel;
- Développeurs d'une méthode FSM d'énoncer clairement les domaines fonctionnels pour lesquels leur méthode peut être employée efficacement;
- Testeurs de conformité des FSM d'avoir des directives claires pour déterminer les domaines fonctionnels pour lesquels leur méthode est efficace;
- Mesureurs d'évaluer le domaine fonctionnel d'un ensemble particulier de FUR afin de déterminer quelle méthode de FSM est la plus appropriée pour leurs besoins.

Partie 6 – Guide d'utilisation des séries ISO 14143 et les normes internationales qui y sont reliées (ISO/IEC 14143-6, 2003)

La partie 6 présente :

1. une description des relations entre :
    i. la série de documents d'ISO/IEC 14143 qui fournit les définitions d'une MTF et les éléments qui sont exigés par les définitions;
    ii. les normes ISO 19761, 20926, 20968 et 24570 qui sont des normes spécifiques pour la MTF ;
2. un processus de sélection d'une méthode MTF qui répond aux exigences d'un utilisateur.

Cette revue de la littérature va maintenant présenter un aperçu des normes spécifiques en MTF mais ne va pas s'attarder sur les règles et sur les procédures des méthodes IFPUG, NESMA et MKII, car l'objectif ici est différent de leur description détaillée. Tout lecteur

intéressé à connaître plus en détail les concepts et les étapes de ces méthodes peut trouver dans les références les informations nécessaires pour ces trois méthodes de mesure).

5.5 IFPUG

L'*International Function Point Users Group* (*IFPUG*) fut fondé en 1986. Ce groupe d'utilisateurs de la méthode FPA a apporté des raffinements à la technique originale d'Albrecht. Ce groupe a comme objectif d'assurer une plus grande diffusion de cette méthode. La méthode de comptage en points de fonction est décrite en détail dans le document *Counting Practices Manual - CPM* (*IFPUG*, 1994). Le manuel CPM contient aussi des exemples pour illustrer ses procédures, ses règles et ses formules de calcul.

Les règles IFPUG :

La méthode FPA est définie selon le point de vue de l'utilisateur. La vue de l'utilisateur est basée sur cinq types de fonction. Ces derniers sont partagés en deux catégories :

1. Les fonctions données; fichier logique interne (FLI) et fichier interface externe (FIE).
2. Les fonctions transactionnelles; entrée externe (EE), sortie externe (SE) et interrogation externe (IE).

Les relations entre ces cinq types de fonctions sont illustrées à la figure 7.

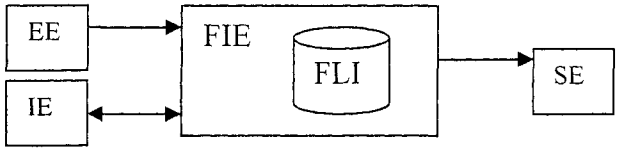

Figure 7 Types de fonction FPA

Les concepts de la méthode à un niveau plus fin de granularité sont :

Data Element Type-DET, *Record Element Type-RET* et *File Type Referenced-FTR*, se traduisant par ; Type d'élément de données-TED, Type d'élément d'enregistrement-TEE et Type de fichier référencé-TFR.

Les processus de comptage comprennent les activités suivantes :

1. Déterminer le type de comptage;
2. Identifier la frontière de l'application;
3. Compter les types de fonctions transactionnelles;
4. Déterminer la valeur du facteur d'ajustement;
5. Calculer le nombre total des PF ajustés.

Il est à noter que bien que les activités 4 et 5 fassent partie de la version initiale de la méthode FPA, ces activités ne font pas partie de la norme ISO correspondante parce que ne rencontrant pas les critères requis par ISO 14143-1.

La frontière entre des applications qui sont liées entre elles ne doit pas être basée sur des considérations technologiques mais sur la séparation des fonctions de gestion telles qu'elles sont vues par l'utilisateur.

5.6 NESMA - ISO/IEC 24570

NESMA (*Netherlands Software Metrics Users Association*) a été fondé en 1989 sous le nom de NEFPUG (groupe hollandais d'utilisateurs de points de fonction). Son objectif est de :

1. Rassembler, maintenir, échanger et développer la connaissance au sujet de la méthode FPA ainsi que son application;
2. Favoriser la standardisation de la méthode FPA;
3. Favoriser l'utilisation de la méthode FPA.

Les similitudes avec la méthode IFPUG sont résumées ainsi :

« Both the NESMA (CPM 2.0) and the IFPUG (CPM 4.1) now use the same philosophy, the same concepts and terms, and the same rules and guidelines within FPA » (NESMA, 2003).

Les utilisateurs des deux méthodes devraient obtenir des résultats comparables car les deux groupes emploient les mêmes philosophies, concepts et limites dans FPA.

Les deux groupes considèrent :

1. le comptage des fonctions que les utilisateurs peuvent identifier;
2. la précision sur le fait que les fonctions soient des processus élémentaires;
3. l'utilisation des mêmes types de fonction : entrée externe, sortie externe; interrogation externe, fichier logique interne, fichier interface externe;
4. le comptage des points de fonction non ajusté;
5. l'utilisation du facteur d'ajustement de valeur;
6. le comptage d'application en points de fonction;
7. le comptage de projet en points de fonction.

Les différences avec IFPUG ont été résumées comme suit :

The NESMA and the IFPUG have resolved almost all their differences in regard to how the complexity of functions should be established (NESMA, 2003). Les différences entre la méthode NESMA (CPM 2.0) et IFPUG (CPM 4.1) sont minimes, étant donné le fait que les deux méthodes partagent les mêmes règles et les mêmes directives dans FPA. Ces différences ont un impact négligeable sur les résultats de comptage des points de fonction.

À l'inverse d'IFPUG, NESMA considère l'option de pouvoir travailler avec les données de sortie qui changent de taille. Cette différence n'a pas une grande influence sur le nombre de points de fonction d'une application ou d'un projet, parce qu'elle affecte seulement le type de fonction mais pas la quantité de fonctions identifiables.

5.7 MKII - ISO/IEC 20968

La méthode Mk II (dans ce document le mot MK II veut dire MarK II FPA *Functional Point Analysis*) est définie par Charles Symons en 1991. Le comité des pratiques de mesures du Royaume-Uni est responsable de la méthode et de son développement.

Les activités de MKII pour le comptage de la taille fonctionnelle des logiciels (C. Symons, 1998) sont :

1. Déterminer le but et le type de mesure. Cette activité consiste à identifier le client qui a demandé ce comptage. S'agit-il de mesurer le rendement d'une équipe de développeurs? Est-ce que le but est de compter toutes les fonctionnalités qui ont été exigées, ou les fonctionnalités fournies à l'utilisateur?
2. Définir la frontière de comptage;
3. Identifier les transactions logiques;
4. Identifier et catégoriser les types d'entités de données;
5. Compter les types élément de données « entrée » (Te), les types référencés d'entités (Tr) et les types élément de données « sortie » (Ts) pour chaque transaction logique;
6. Calculer la taille fonctionnelle. La taille fonctionnelle est la somme de toutes les transactions logiques mesurées en Te, Tr et Ts en multipliant chaque type par son poids correspondant. La taille fonctionnelle de l'application est donnée par l'équation suivante : NFP = Te * Pe + Tr * Pr + Ts * Ps.
7. Déterminer l'effort et le temps passé pour développer le projet;
8. Calculer la productivité et d'autres performances;
9. Marquer les degrés d'influence;
10. Calculer l'ajustement de la complexité technique;
11. Calculer la taille des points de fonction ajustés.

Comparaison entre IFPUG 4.0 et MkII Version 3.0 :

« The MK II method works at a much finer level of granularity than the IFPUG method and this leads to lower sizes for small enhancements » (C. Symons, 1999).

Pour plus de détails sur la comparaison des deux méthodes, consulter le document en ligne de Grant Rule sur le web à: http ://www.gifpa.co.uk/library/Papers/Rule/ MK2IFPUG.html.

Les deux méthodes mesurent la taille de façon sensiblement différente d'un produit logiciel. En termes de taille produite, la différence est que la méthode MKII a une granularité plus fine et un processus de mesure continu alors que la méthode IFPUG limite la taille d'un composant à un seuil.

Les concepts utilisés pour mesurer la taille sont basés sur des entités et des transactions logiques permettant d'exprimer les exigences du logiciel et les spécifications fonctionnelles.

Les ajouts fournis par Charles Symons ont été conçus pour délivrer une mesure pour la taille d'une magnitude similaire pour les deux méthodes. En général, les méthodes donnent les mêmes tailles de logiciels allant jusqu'à 400 points de fonction. Pour des tailles plus grandes, la méthode Mk II FPA tend à produire une taille plus élevée que la méthode IFPUG (C. Symons, 1999).

Selon Symons, pour la méthode MK II, la mesure de la taille a été conçue pour être plus sensible aux petits changements des fonctionnalités que la méthode IFPUG.

5.8 COSMIC-FFP (ISO 19761)

La méthode COSMIC-FFP (*Common Software Measurement International Consortium – Full Function Points*) a été conçue pour mesurer la taille fonctionnelle du logiciel en temps réel et multicouches tel qu'utilisé dans les applications de télécoms, les systèmes d'exploitation, ainsi que pour les logiciels de gestion. La première version de la méthode COSMIC-FFP a été publiée en 1999 et en seulement quatre ans elle a réussi à franchir

toutes les étapes de développement d'un consensus international pour devenir une norme internationale (ISO 19761 : 2003).

L'association COSMIC a été fondée en 1998. C'est une association volontaire d'experts en matière de mesure de logiciels provenant d'Australie, du Canada, de la Finlande, d'Allemagne, d'Irlande, d'Italie, du Japon, de la Hollande et du Royaume Uni.

La méthode de mesure COSMIC-FFP se base sur les exigences fonctionnelles de l'utilisateur (Fonctionnalités Utilisateurs Requises – FUR – en anglais *Functional User Requirement*) pour mesurer un logiciel. Les FUR sont les pratiques et les procédures que le système doit réaliser pour satisfaire les besoins des utilisateurs.

COSMIC-FFP est composée principalement de quatre phases :

A. Appliquer les FUR du logiciel à mesurer au modèle générique COSMIC-FFP ; Cette phase permet d'identifier les éléments de base indispensable à l'opération de mesure, elle est composée de quatre activités :

   i. Identifier les couches du logiciel à mesurer : une couche est le résultat du partitionnement fonctionnel de l'environnement du logiciel. Par exemple le système d'exploitation, les interfaces utilisateurs et le SGBD sont des couches différentes;
   ii. Délimiter la frontière du logiciel : la frontière est une ligne conceptuelle séparant le logiciel de son environnement. Par exemple, la souris, le clavier, l'imprimante, le disque dur, la RAM et la ROM participent à démarquer la frontière du logiciel à mesurer ;
   iii. Définir l'utilisateur du système : un utilisateur peut être un individu, un autre logiciel ou un dispositif qui interagit avec le logiciel à mesurer;
   iv. Identifier des éléments candidats de l'opération de mesure. Ces éléments ne seront pris en compte dans la mesure que s'ils sont validés par le modèle COSMIC-FFP. Ces éléments sont :

      a) les évènements déclencheurs : un évènement déclencheur se passe à l'extérieur de la frontière du logiciel à mesurer

et qui initie un ou plusieurs processus fonctionnels (e.g. entrée des données);

b) les processus fonctionnels : un processus fonctionnel est un ensemble unique et ordonné (e.g. soustraction des opérandes) de mouvements de données (entrée, sortie, lecture et écriture).

B. Faire la mesure de la taille fonctionnelle proprement dite sur les FUR du logiciel à mesurer. Après avoir défini le modèle COSMIC-FFP, comme ci-dessus, on applique un ensemble de règles et de procédures fournies par la méthode COSMIC-FFP. La règle principale est le calcul des mouvements de données. En effet, la méthode permet de valider les processus fonctionnels définis précédemment et d'identifier tous les mouvements de données (un mouvement de données est un déplacement élémentaire d'un groupe de données pendant l'exécution d'un processus fonctionnel).

C. La présentation des résultats détaillés de la mesure :

a) Le rang du processus dans la liste des processus fonctionnels
b) L'identifiant du processus qui est composé de son numéro de couche et son rang dans la couche
c) La description du processus fonctionnel
d) L'événement déclencheur du processus fonctionnel
e) La liste de tous les mouvements de données qui le composent :
    i. leur description
    ii. le groupe de données mis en cause
    iii. le type du mouvement de données (entrée, sortie, lecture et écriture)
    iv. la taille associée
    v. la taille totale du processus fonctionnel.
f) La taille totale du logiciel.

D. La représentation des résultats agrégés de la mesure

Cette phase finale résume les résultats de l'opération de mesure selon :

1. Le nombre de mouvements de données de chaque type par processus fonctionnel;
2. La proportion de la taille de chaque processus fonctionnel dans l'application;

3. La proportion de la taille de chaque type de mouvement de données dans l'application (Entrée, Sortie, Lecture et Écriture);
4. La représentation graphique de la proportion de chaque type de mouvement de données dans l'application.

5.9 Comparaison entre les méthodes IFPUG ET COSMIC-FFP

Dans son article *COSMIC-FFP and IFPUG 4.1 Similarities and Differences*, Pam Morris (2003), présente une comparaison des deux méthodes telles qu'illustrées dans les tableaux suivants.

La comparaison au niveau des concepts est présentée au tableau 16, la comparaison au niveau des processus au tableau 17, la comparaison de l'utilisation des résultats dans le modèle d'estimation au tableau 18 et la comparaison des ressources au tableau 19.

Tableau XVI

Comparaisons entre IFPUG et COSMIC-FFP

Concept	IFPUG	COSMIC-FFP
Méthodes de travail avec les logiciels multicouches	N'est pas explicite dans le guide de comptage	Explicite dans les règles de mesure des architectures multicouches
Vue utilisateur	Mesure selon la vue utilisateur « humaine » uniquement	Mesure à partir de différents points de vue
Exigences techniques et qualité	N'est pas mesuré explicitement	Les fonctionnalités requises par les autres couches utilisatrices sont considérées dans la couche mesurée

Tableau XVII

Comparaisons au niveau des processus

Processus	IFPUG	COSMIC-FFP
Comptage de DET	Oui	Non, la méthode compte les « groupes logiques » des DET
Valeur par défaut pour la complexité des données en industrie	Non	Non
Règles pour déterminer les pondérations	Différentes règles pour chaque type de processus	Les mêmes règles pour tous les processus
Regroupement répétable des groupes de données logiques	Exige une expérience en comptage pour assurer la « répétabilité »	Exige une expérience en modélisation des données pour assurer la « répétabilité »
Niveau de détail demandé dans les spécifications fonctionnelles	Oui	Oui, pour identifier chaque mouvement de données et accès aux fichiers

Tableau XVIII

Comparaison d'utilisation des résultats pour estimation

Résultat	IFPUG	COSMIC-FFP
Corrélation aux efforts à travers tous les domaines fonctionnels	Pour les systèmes informatiques de gestion il existe un appui industriel évident. Les données pour les logiciels scientifiques et les systèmes en temps réel sont limitées	Une recherche de 100 projets est approuvée pour les systèmes en temps réel. Les données pour les systèmes informatiques de gestion sont limitées.
Données industrielles	Domaine public et privé	Domaine de recherche ou privé
Certification internationale	Praticiens à IFPUG CFPS	Laboratoires ISO pour les tests sont planifiés
Sensibilité aux grandes variations dans le processus de complexité	La sensibilité maximale est une variation de deux fois	Pas de seuil ni de limite prédéterminée

Tableau XVIII (Suite)

Comparaison d'utilisation des résultats pour estimation

Résultat	IFPUG	COSMIC-FFP
Sensibilité aux processus qui déplacent beaucoup de données sans qu'ils accèdent aux groupes de données	La mesure de taille maximale exige les mouvements et l'accès aux données	La mesure de taille maximale n'exige pas de processus pour accéder aux groupes de données
Contribution des données dans la taille totale	Les données persistantes contribuent de 30 % de la taille totale, en y ajoutant la contribution des accès aux processus de données	Les données persistantes contribuent seulement via les accès aux processus de données
Compte l'accès multiple aux groupes de données	Oui	Oui

Tableau XIX

Comparaison des ressources

Ressource	IFPUG	COSMIC-FFP
Manuels	À acheter d'IFPUG À acheter d'ISO 390 pages (à 294,00 CHF)	Téléchargement gratuit du guide COSMIC d'implantation de la norme http://www.lrgl.uqam.ca/cosmic-ffp/manual.html À acheter d'ISO 75 pages (à 83,00 CHF)
Formation	Plusieurs vendeurs de cours dans le monde entier Certification	Plusieurs cours disponibles dans le monde entier Pas de certification
Outils	Plusieurs vendeurs d'outils Certification	Limités Pas de certification
Études de cas	Deux études de cas pour les fonctionnalités utilisateurs requises À acheter d'IFPUG	Cinq études de cas pour les fonctionnalités utilisateurs requises Téléchargement gratuit

5.10 Pourquoi COSMIC-FFP ?

Comme vu dans la section précédente, en parallèle à l'adoption de COSMIC-FFP (ISO/IEC 19761) comme norme, trois autres méthodes ont été approuvées par ISO comme normes spécifiques pour les mesures de la taille fonctionnelles des logiciels; IFPUG (ISO/IEC 20926), provenant des États-Unis, MkII FPA (ISO/IEC 20968), qui est proche de la méthode IFPUG, fournie par le Royaume-Uni et la méthode hollandaise NESMA (ISO/IEC 24570) qui est une variante de la méthode IFPUG. Ces trois dernières méthodes datent des années 1970-1980 et ont été conçues pour mesurer la taille fonctionnelle des logiciels de gestion. Elles sont utilisées pour mesurer la performance des logiciels et l'estimation de leur coût.

Seule la méthode COSMIC-FFP a apporté une modification structurelle à la méthode IFPUG, puisqu'elle peut être utilisée pour mesurer les logiciels de types autres que les systèmes d'informatique de gestion, tel que le système en temps réel.

Dans leur article « Mesure de la taille fonctionnelle des logiciels temps réel », Desharnais *et al.* montrent que dans la version 2.0 de COSMIC-FFP, la mesure se fait de façon individuelle pour les sous-processus de lecture et de mise à jour des fichiers logiques ainsi que pour l'envoi de l'information à l'utilisateur. Ainsi la méthode COSMIC-FFP a un niveau de granularité plus fin que la méthode IFPUG, parce qu'elle prend en compte le niveau pour les sous-processus. « Ce niveau de granularité plus fin prend toute son importance dans le cas des logiciels en temps réel, puisque leurs processus contiennent un nombre variable de sous-processus, chacun ayant une contribution individuelle à la mesure de leur taille fonctionnelle » (Desharnais *et al.*, 1998).

Souvent, on reproche à la méthode IFPUG de fournir des mesures fonctionnelles trop petites dans un environnement en temps réel (Galea, 1995; Capers Jones, 1996). Ceci est dû à la granularité de la méthode IFPUG qui n'est pas assez fine pour permettre de tenir

compte des sous-processus, car elle ne dispose pas d'un niveau pour ces derniers. Ainsi, les mesures fournies à la fin du processus ne reflètent pas de façon adéquate, selon ces auteurs, la taille réelle des fonctionnalités données par les logiciels mesurés. Cependant COSMIC-FFP prend en considération les sous-processus fonctionnels qui peuvent exister à l'intérieur d'un processus de contrôle. De plus, la méthode permet de déterminer tous les groupes de données envoyés, lus, écrits et reçus ce qui génère un résultat plus grand en mesure fonctionnelle et reflète le nombre de fonctionnalités réellement fournies à l'utilisateur du logiciel, qu'il soit humain, un autre logiciel ou un appareil qui interagit avec le logiciel.

Enfin, la méthode COSMIC-FFP est fondée sur une théorie solide et des décennies d'expérience internationale. Selon le *Software Measurement Services* (2001), COSMIC-FFP a été conçue dès le début pour être conforme à la norme ISO 14143 pour la MTF et pour être compatible avec les techniques modernes de spécification des exigences comme UML et le prototypage. Cette méthode de mesure est applicable à plusieurs types de logiciels. Elle reconnaît que le développement des logiciels modernes se sert des composants à différents niveaux dans une architecture de logiciel, ce qui permet d'atteindre des niveaux que d'autres méthodes ne tiennent pas compte en termes de fonctionnalité.

5.11 Sommaire

Ce chapitre a présenté un aperçu des normes portant sur la mesure de fonctionnalité des logiciels. Il a aussi situé la méthode lignes de code qui permet de mesurer le nombre de lignes de code dans un programme, par rapport aux MTF. Il a inclus également une description générale de la méta-norme de mesure de la taille fonctionnelle ISO 14143, puis des normes spécifiques qui ont été acceptées par ISO. Une comparaison détaillée entre COSMIC-FFP et IFPUG a été aussi présentée dans ce chapitre. Enfin, le chapitre a présenté les raisons de la sélection de COSMIC-FFP comme la méthode adoptée dans cette thèse pour bâtir les premiers étalons de mesure du logiciel :

1. La méthode COSMIC-FFP se démarque par le fait qu'elle permet de mesurer la taille fonctionnelle des systèmes en temps réel, des systèmes de gestion et des systèmes hybrides, assez tôt dans le cycle de développement, soit dès la phase de spécifications du logiciel.
2. COSMIC-FFP appliquée au début du cycle de développement peut être utilisée, par exemple, pour construire des modèles d'estimation de l'effort requis pour réaliser un projet informatique. Appliquée à la fin du cycle de développement, elle peut servir à bâtir des modèles de productivité ainsi qu'à comparer la productivité d'un projet informatique à un autre.
3. La méthode COSMIC-FFP permet aussi de résoudre quelques imperfections de la méthode des points de fonction pour les systèmes en temps réel et de simplifier la tâche de mesure. Elle est conforme à la norme ISO/IEC 14143-1, qui spécifie un ensemble de conditions obligatoires génériques pour une méthode de mesure pour être une méthode de mesure de taille fonctionnelle. En outre, la méthode est indépendante de toute technologie.
4. La méthode est compatible avec les techniques modernes de spécification des exigences comme UML et le prototypage.

Donc, pour ces raisons la méthode COSMIC-FFP sera utilisée pour la préparation des premiers étalons de mesure des logiciels.