Institutional Repository - Research Portal
Dépôt Institutionnel - Portail de la Recherche
researchportal.unamur.be
University of Namur

THESIS / THÈSE

MASTER EN SCIENCES INFORMATIQUES

Etude de la mesure des points de fonction: application de la méthode de mesure fonctionnelle Cosmic-FFP et analyse de la documentation fonctionnelle

Munezero, Solange Ndagijimana

Award date:
2002

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal ?

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 23. Jun. 2020

Facultés Universitaires Notre-Dame de la Paix, Namur
Institut d'Informatique
Année académique : 2001-2002

ETUDE DE LA MESURE DES POINTS DE FONCTION: APPLICATION DE LA METHODE DE MESURE FONCTIONNELLE COSMIC-FFP ET ANALYSE DE LA DOCUMENTATION FONCTIONNELLE

Solange Ndagijimana Munezero

Mémoire présenté en vue de l'obtention du grade de Maître en Informatique

RESUME

Ce projet parcourt les différents aspects de la mesure fonctionnelle, dans le but de faire ressurgir certains problèmes auxquels le mesureur doit faire face, afin d'améliorer la qualité et la fiabilité des résultats obtenus.

Pour ce faire, une vingtaine de projets d'entreprise, ayant fait l'objet d'une mesure de taille fonctionnelle de type FPA par des experts, a été mesurée à l'aide de la méthode de mesure fonctionnelle COSMIC-FFP.

Les résultats, dont l'analyse est centrée sur l'aspect documentation fonctionnelle, ont fait ressurgir de nombreuses lacunes relatives à l'information nécessaire à la bonne application d'une méthode de mesure.

Il est donc indispensable, lors de la définition d'une méthode de mesure, de détailler et de préciser encore plus le raisonnement du mesureur dans le cas où l'information est incomplète.

ABSTRACT

This project reviews different aspects of the functional measurement in order to highlight the problems confronting the person who measures. The objective is to improve the quality and reliability of the results obtained.

In order to achieve this, the projects of around 20 companies which have been measured by experts with the FPA functional size measurement, have been measured using the COSMIC FFP functional size measurement.

The analysis of the results is centered on the functional documentation aspect which demonstrated a lack of necessary information in order to support the good application of the measurement method.

It is therefore necessary when one defines a measurement method to detail the reasoning process of the person conducting the measurement in case the information is incomplete.

REMERCIEMENTS

Mes premiers remerciements s'adressent à Jean-Marc Desharnais membre du Laboratoire de Recherche en gestion du Logiciel de l'Université du Québec à Montréal (UQAM) et co-auteur de la méthode de mesure COSMIC-FFP. Il m'a relu à plusieurs reprises et les remarques dont il m'a fait part m'ont souvent amené à revoir la pertinence des propos avancés.

Je remercie également les membres du Laboratoire de Recherche en Gestion du Logiciel pour leur accueil et la manière dont ils m'ont intégré durant le stage.

Je tiens ensuite à remercier mon promoteur Mr Naji Habra ainsi que toutes les personnes qui de près ou de loin m'ont aidé par leurs conseils durant la rédaction de ce mémoire (Anne-sophie, Simon, Maryse, Jean-claude, etc.).

Enfin, j'exprime ma gratitude à l'égard de mes parents pour le soutien qu'ils m'ont apporté tout au long de cette dernière année de maîtrise.

TABLE DES MATIERES

LISTE DES TABLEAUX

LISTE DES FIGURES

GLOSSAIRE

Logiciel : Objet abstrait qui a évolué à partir d'un énoncé pour se finaliser par un logiciel. Un logiciel comprend aussi bien le code, objet ou source, que les diverses formes de documentation produites tout au long du processus de développement.[1]

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]

Mesure : Une valeur quantitative de base qui décrit la grandeur d'un élément donné du processus de génie logiciel.[3]

Mesure fonctionnelle : Il s'agit d'une fonction qui prend comme paramètres les données du logiciel et qui fournit comme résultat une valeur numérique qui peut être interprétée comme le degré d'un attribut donné que possède un logiciel.[4]

Point de fonction : Il s'agit d'une méthode de mesure fonctionnelle basée sur une évaluation du logiciel, du processus de développement de celui-ci, ainsi que de sa maintenance. Les points de fonction sont fondés sur la fonction « utilisateur » du traitement de l'information.[5]

Documentation fonctionnelle : Il s'agit de toute information ou documentation concernant un logiciel. La plupart des mesures fonctionnelles se basent sur la documentation fonctionnelle pour évaluer un logiciel.

---

[1] Définition tirée de Naur et al [1969].
[2] Définition tirée de l'ouvrage de Boehm [1981]
[3] Définition donnée dans Meredith [1991].
[4] Cette définition est basée sur la définition donnée par l'IEEE Standards P1045/D2.0 ;1989.
[5] Il s'agit là de la définition de base donnée dans Albrecht [1983].

INTRODUCTION

Le processus d'évaluation de la taille fonctionnelle des logiciels passe par l'analyse des spécifications fonctionnelles afin de dégager toutes les fonctionnalités que le logiciel offre à ses utilisateurs. Ces spécifications se retrouvent le plus souvent sous la forme de documents que le mesureur doit parcourir, ce qui est loin d'être une tâche facile.

Le travail d'analyse effectué se concentre donc sur l'aspect documentation fonctionnelle, dans le but de faire ressurgir les problèmes auxquels le mesureur doit faire face afin d'améliorer la qualité et la fiabilité des résultats obtenus.

Pour ce faire, une vingtaine de projets d'entreprise (26 au total), ayant fait l'objet d'une mesure de taille fonctionnelle de type FPA (*Function Point Analysis*) par des experts, ont été mesurés à l'aide de la méthode COSMIC-FFP (*Common Software Measurement International Consortium - Full Function Points).* Cela a permis parallèlement la comparaison et la validation des résultats obtenus avec les deux méthodes de mesure fonctionnelle ainsi qu'une analyse plus approfondie des projets mesurés.

Les projets mesurés étaient pour la plupart des projets de maintenance d'une grande entreprise canadienne travaillant avec le Laboratoire de Recherche en Gestion du logiciel (*LRGL*) qui fait partie de l'Université du Québec à Montréal (*UQAM*), lieu où l'auteur de ce travail a fait son stage. Pour chacun des projets, il a fallu, outre l'analyse de la documentation fonctionnelle, calculer le nombre de points de fonction et évaluer le type d'activité et de processus.

Ce document présente, dans le premier chapitre, les fondements théoriques de la mesure des logiciels. Le deuxième chapitre aborde le thème des mesures basées sur les points de fonction. La problématique et la structure des mesures fonctionnelles y sont exposées dans une perspective d'évolution historique, afin d'illustrer les améliorations qui ont été apportées et ainsi préciser le processus de mesure.

La méthode de mesure COSMIC-FFP[6] utilisée pour l'analyse des projets est présentée au chapitre 3, les règles ainsi que les différentes étapes de mesure sont passées en revue. Le quatrième chapitre, présente les résultats obtenus pour les 26 projets auxquels les méthodes de mesure COSMIC-FFP et FPA ont été appliquées pour des fins de comparaison.

Cette étude se termine par une proposition d'intégration de la méthode de mesure fonctionnelle COSMIC-FFP, dans le cadre des Petites et Moyennes Entreprises (PME).

---

[6] La méthode de mesure fonctionnelle FPA n'ayant pas été directement appliquée durant le processus d'évaluation des projets (uniquement les résultats obtenus par d'autres mesureurs sure les mêmes projets ont été exploités) est quant à elle présentée à l'annexe 1.

1 La mesure des logiciels

Les logiciels ont connu une expansion considérable ces dernières années. Ils sont devenus des outils de plus en plus indispensables et omniprésents. Il est par conséquent nécessaire de fournir des moyens permettant de gérer de manière optimale la production et la maintenance de ceux-ci.

L'introduction de processus de mesure est apparue comme la panacée pour administrer et gérer le développement vaste du logiciel. Mais trouver une méthode de mesure captant toutes les caractéristiques d'un logiciel est loin d'être facile.

En effet, avant de pouvoir mesurer, il faut avoir un concept clair des attributs et de l'ensemble des entités à mesurer qui possèdent cet attribut [Fenton, 1991]. En général, la mesure d'un logiciel va avec l'évaluation d'une ou de toutes les entités de celui-ci, à savoir les processus de production, les produits et les ressources utilisées, (voir section 1.1.2).

Ce premier chapitre[7] aborde les divers aspects de la mesure des logiciels à travers une mise en évidence des objectifs de la mesure et une classification des entités mesurables en génie logiciel. Un inventaire des différents modèles de mesure sera dressé afin de dévoiler les aspects les plus intéressants à évaluer lors de la gestion des différentes phases de développement du logiciel. La problématique générique aux mesures des logiciels est également accostée.

1.1 Analyse du processus de mesure des logiciels

Toute activité de gestion du logiciel inclut le planning, l'analyse des coûts, la modélisation, la spécification, la mise en place d'architecture, l'implémentation, les tests et la maintenance. Chaque activité doit être comprise et contrôlée, de façon à minimiser les risques (dépassement de budget, crash du systèmes, etc.) et à éviter les mauvaises surprises (inadéquation du logiciel par rapport aux exigences du client, etc.).

Les développeurs de logiciels essaient de faire face aux différents problèmes de gestion des phases d'évolution du logiciel en continuant la recherche de nouvelles technologies et outils, afin d'améliorer les produits et les méthodes de traitement. [Fenton, 1997].

Les outils, ainsi que les méthodologies de mesure de logiciel en général s'inscrivent dans ce cadre. Il convient donc de considérer plus en détail les objectifs de la mesure en génie logiciel ainsi que les entités mesurables du processus de développement du logiciel.

---

[7] Ce premier chapitre est largement inspiré des travaux de Fenton publié dans la seconde édition de l'ouvrage « Software Metrics –Rigorous Approach », Chapman & Hall, London, 1997.

1.1.1 Objectifs des mesures en génie logiciel

Les mesures constituent un élément essentiel pour tout projet. En effet, comment évaluer la santé et la qualité d'un projet si aucune méthode d'estimation ou de mesure n'est disponible?

Les mesures sont donc indispensables ne fût-ce que pour être au courant de l'état d'avancement du projet, de la qualité des produits, des processus de production en application, ainsi que pour évaluer les ressources disponibles et nécessaires pour atteindre l'objectif.

Les objectifs de mesure doivent être bien spécifiés et proches de ce que les programmeurs, les managers et les utilisateurs ont besoin de savoir. Les objectifs peuvent donc être différents suivant le genre de personnes impliquées et suivant le niveau de l'utilisation et du développement du logiciel auquel ils sont générés. Et ce sont ces mêmes objectifs qui renseigneront sur la façon dont sera utilisée l'information collectée [Fenton, 1997].

Le tableau 1.1 [Fenton, 1997] est un exemple du genre d'informations requises pour comprendre et contrôler un projet de développement de logiciel, structuré suivant la perspective du manager et du programmeur ou analyste. Les réponses approximatives à ces questions peuvent être fournies par les mesures.

Manager	**Programmeur**
Quel est le coût de chaque processus (spécification, implémentation, test, temps, effort, etc.)?	Peut-on tester les exigences?
Quelle est la productivité du personnel?	Les fautes ont-elles été toutes trouvées?
Quelle est la qualité du code développé (enregistrement des fautes)?	Les objectifs en terme de produits et de processus de production ont-ils été atteints?
Les utilisateurs seront-ils satisfaits par le produit développé (analyse du produit par rapport aux exigences du client)?	Que peut-on faire comme prédiction (taille du système, maintenance, fiabilité opérationnelle, etc.)?
Comment améliorer les produits?	

Tableau 1-1 : Exemple d'informations utiles aux programmeurs et aux développeurs

Une mesure en génie logiciel doit décrire et spécifier non seulement les algorithmes de recueil et de traitement de données, mais également la façon dont les données seront présentées, interprétées et utilisées.

Une mesure, pour être utile, doit aussi tenir compte des buts à atteindre et du contexte du système dans lequel évolue le logiciel [Fenton, 1997].

1.1.2 La classification des mesures de logiciel

La première étape de toute activité de mesure, selon Fenton [1997], est l'identification des entités et attributs à mesurer. En génie logiciel, il y a trois classes d'entités dont les attributs peuvent faire l'objet d'une mesure :

- **Les processus**, qui sont toutes les activités relatives aux logiciels, ont normalement un facteur temps.
- **Les produits**, qui sont tous les outils, objets livrables ou documents qui ressortent du processus, c'est-à-dire en d'autres termes qu'il s'agit des outputs du processus.
- **Les ressources,** qui constituent l'input des processus.

Tout domaine pouvant faire l'objet d'une mesure ou d'une prévision constitue un attribut d'une de ces trois entités. Une distinction est faite entre les attributs internes et externes [Fenton, 1997] :

- Les attributs internes d'un processus, d'un produit ou d'une ressource sont ceux qui peuvent être mesurés en termes de processus, de produits ou de ressources tout simplement.
- Les attributs externes d'un processus, d'un produit ou d'une ressource sont ceux qui peuvent être mesurés en considérant la manière dont un processus, un produit ou une ressource interagit avec son environnement.

Les attributs externes sont souvent ceux qui intéressent les gestionnaires et les utilisateurs de logiciel au niveau mesure et prévision. Les gestionnaires chercheront par exemple à connaître la productivité de leur personnel ou les coûts effectifs de certains processus, tandis que les utilisateurs voudront connaître le cadre d'utilisation, la fiabilité ou la portabilité des systèmes qu'ils cherchent à acquérir.

Malheureusement, de par leur nature, les attributs externes sont les plus difficiles à mesurer. De plus, la plupart n'ont généralement pas de définition standard agréée. Les attributs comme *la qualité* sont tellement courants qu'ils en perdent leur sens. La plupart du temps, les attributs internes sont utilisés pour mesurer les attributs externes. Les attributs internes peuvent en principe être mesurés directement. [Fenton, 1997].

Le tableau 1.2 fournit des exemples d'entités et d'attributs pouvant faire l'objet de mesure [Fenton, 1997]. Par exemple, en ce qui concerne l'entité Produit, on peut évaluer les spécifications dont les attributs internes sont la taille, les fonctionnalités définies, la correction syntaxiques de ces spécifications, etc. En même temps, on peut évaluer leurs attributs externes qui sont le degré de maintenance et de compréhension.

ENTITES	**ATTRIBUTS**	
PRODUITS	**INTERNES**	**EXTERNES**
Spécification	***Taille, réutilisation, modularité, redondance, fonctionnalité, correction syntaxique,*** …	***Compréhensibilité, maintenabilité,*** …
Design	Taille, réutilisation, couplage, cohérence, héritage, fonctionnalité, …	Qualité, complexité, maintenabilité, …
Code	Taille, réutilisation, couplage, fonctionnalité, complexité algorithmique, …	Fiabilité, utilisation, maintenabilité, réutilisabilité, …
Test des données	Taille, niveau de couverture, …	Qualité, réutilisabilité, …
….	…	…
PROCESSUS		
Construction Spécification	Temps, effort, nombre d'exigences changées, …	Qualité, coût, stabilité, …
Design détaillé	Temps, effort, nombre de défauts de spécification, …	Coût, coût effectif, …
Test	Temps, effort, nombre de défauts de codage trouvés, …	Coût, coût effectif, stabilité, …
…	…	…
RESSOURCES		
Personnel	Âge, prix,…	Productivité, expérience, intelligence
Equipe	Taille, niveau de communication, structure,…	Productivité, qualité, …
Organisation	Taille, certification ISO, niveau CMM, …	Maturité, profitabilité, …
Logiciel	Prix, taille, …	Utilisabilité, fiabilité, …
Matériel	Prix, vitesse, taille mémoire, …	Fiabilité, …
Bureau	Taille, température, lumière, …	Confort, qualité, …
…	…	…

Tableau 1-2 : Classification des activités de mesure en génie logiciel

Ces différentes catégories de domaines mesurables s'expriment à travers des modèles de mesure.

1.2 Les modèles de mesure

Plusieurs modèles de mesure en génie logiciel ont été développés. Ces modèles couvrent plusieurs domaines d'activités [Fenton, 1997]:

- Estimation du coût et de l'effort;
- Mesures et modèles de productivité;
- Collection de données;
- Evaluation et modèle de performance;
- Mesure de structure et de complexité;
- Evaluation de maturité et de capacité;
- Gestion par les méthodes de mesures;
- Evaluation des méthodes et outils.

Les sections suivantes feront une approche des techniques utilisées pour chaque aspect de mesure en génie logiciel.

1.2.1 Les modèles de coût et d'effort

Un des plus grands souhaits des gestionnaires de logiciel a toujours été la capacité de prédire le coût, et ce, durant les toutes premières phases du cycle de vie du logiciel. D'où on retrouve plusieurs modèles d'estimation de coût et de l'effort en génie logiciel. On y retrouve par exemple le modèle COCOMO (The Constructive COst Model) de Boehm [1981], le modèle SLIM (*Software lifecycle Model*) de Putnam [1978], ainsi que le modèle des points de fonction d'Albrecht [Albrecht, 1979].

Tous ces modèles ont un point en commun qui est « l'effort » requis pour produire le logiciel. Cet effort est exprimé comme une fonction (prédéfinie) d'une ou de plusieurs variables (comme la taille du produit, la capacité des développeurs, etc.).

La taille est souvent définie en termes de lignes de code ou du nombre de points de fonction qu'on peut obtenir à partir des spécifications du produit. Ces modèles de coût et d'effort seront rediscutés en détail au chapitre 2.

1.2.2 Les modèles de productivité

Le concept qui permet de relier ensemble diverses facettes multidimensionnelles d'un même phénomène de production est celui de 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 œuvre lors de la production [IEEE Standard P1045, 1989; Filion, 1988] .

Elle s'exprime comme suit :

$$Productivité = \frac{Output(Résultat)}{Input(Ressources)}$$

Dans la définition ci-dessus de la productivité, il existe plusieurs notions qu'il a été difficile de cerner jusqu'ici dans le domaine du génie logiciel. En effet, il faut faire une distinction très nette entre les produits et les processus de production, c'est-à-dire entre les résultats et les activités (voir section 1.1.2).

Afin d'analyser et comprendre la productivité du développement d'une application informatique, il faut approfondir les deux concepts : les produits logiciels et le processus de production du logiciel. Pour faciliter les études sur la productivité, il faut étudier séparément les deux dimensions [Abran, 1994].

« Ainsi on garde une dimension fixe, puis on examine comment des variations dans l'autre dimension vont affecter la relation par rapport à la variable dépendante de l'effort. Par exemple, on garde le processus de production fixe et on examine comment des variations dans la taille du logiciel affectent la relation avec l'effort.» [Abran, 1994, p35].

Le domaine du génie logiciel a produit un nombre considérable de modèles de productivité et d'estimation, mais ils ne sont pas pour autant basés tous sur un contexte de référence rigoureux [Blake, 1987].

Dans un autre contexte plus concret, les modèles de productivité peuvent être utilisés pour évaluer la productivité du personnel à différents niveaux du développement du logiciel et dans des environnements différents. Fenton [1997] propose une vue différente de la productivité (figure 1-1), beaucoup plus compréhensible que la vue traditionnelle qui divise tout simplement la taille en effort, tout simplement.

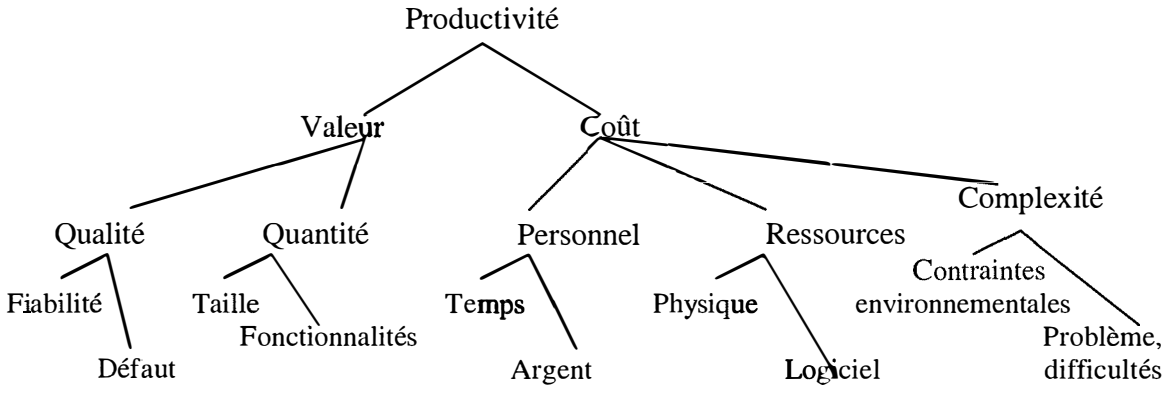


Figure 1-1: Modèle de productivité

1.2.3 Collecte de données

La qualité de tout processus de mesure dépend fortement d'une bonne collecte de données [Fenton, 1997]. La collecte de données utile à la mesure est loin d'être facile, surtout quand il faut aller chercher les données dans différents projets.

Les spécialistes y travaillent afin de s'assurer que les mesures sont définies sans ambiguïté, que ces données sont consistantes et complètes et que l'intégrité des données est garantie. Une analyse de données basée sur l'expérimentation faite lors de cette étude sur des projets réels est présentée au chapitre 4.

1.2.4 Estimation et mesure de performance[8]

Un autre aspect pouvant être évalué est la performance d'un logiciel. Selon les objectifs, il y a trois dimensions à considérer dans le développement d'application et la maintenance des performances :

- **Amélioration de la productivité** qui consiste à pouvoir produire plus à partir de ressources données ou pouvoir produire à moindre coût. Ce qui, en terme de systèmes de développement, signifie moins d'efforts ou moins d'heures de travail.

  La productivité des systèmes de développement est définie comme suit :

  $$Productivité = \frac{Output}{Input} \quad \text{ou} \quad \frac{Taille\ du\ système}{Heures\ de\ travail}$$

- **Délais de livraison :** Il s'agit de la capacité à livrer le système plus rapidement.

  L'élément de performance à mesurer est la livraison :

  $$Livraison = \frac{Taille\ du\ système}{Semaines\ écoulées}$$

- **Amélioration de la qualité** : La qualité est beaucoup plus difficile à définir et il existe plusieurs méthodes de mesure. Les trois plus importantes sont probablement :

  - $Densité\ des\ défauts = \frac{Nombre\ de\ défauts}{Taille\ du\ système}$

[8] Voir Chapitre 1 de l'ouvrage de Symons C.R, "Software sizing and estimating", publié en 1991.

- *Qualité des fonctionnalités :* Il s'agit de la qualité telle que perçue par l'utilisateur, en termes de facilité d'utilisation, de fiabilité, de sécurité, d'exactitude, et autres paramètres de ce genre.
- *Qualité technique :* Il s'agit de la qualité telle que perçue par le département de systèmes d'information, en terme de design, de maintenance et d'opérabilité.

Pour chacune de ces dimensions de performance, il faut choisir vers quel domaine se focaliser (par exemple vers les activités de développement du système ou d'amélioration de performance), et une fois que le système est opérationnel, converger vers la maintenance de celui-ci.

Il n'est généralement pas possible d'obtenir une amélioration de performance du point de vue des trois dimensions en même temps, mais il existe bien entendu des interrelations [Boehm, 1981].

Un bon programme d'amélioration de performance devrait commencer par se fixer des objectifs, ce qui peut amener à faire un choix parmi les trois dimensions de performance à savoir la productivité, la livraison et la qualité [Symons, 1991].

Il faut aussi noter que **la taille du système** est un facteur clé dans la mesure de performance pour chaque dimension.

1.2.5 Mesure de structure et de complexité

La complexité d'un logiciel peut également faire l'objet d'une évaluation d'autant plus que celle-ci peut être faite au tout début du cycle de vie du logiciel. Par exemple, on peut tenter d'évaluer quelle partie du logiciel risque d'être moins fiable, plus difficile à tester, ou qui exigera plus de maintenance que les autres.

Par conséquent, on mesure donc les attributs structurels de la représentation du logiciel qui sont disponibles à l'avance (avant même l'exécution), puis on essaye d'établir des théories de prévision empirique.

Des approches de mesure de complexité et de structure des produits logiciels peuvent être trouvées dans les ouvrages de Symons [1988].

1.2.6 Evaluation de maturité et de capacité ou *Capability Maturity Models (CMM)*

Dans les années 80, la *US Software Engineering Institute (SEI)* a proposé des modèles de maturité et de capacité ou *Capability Maturity Models (CMM)* [Humphrey, 1989] pour mesurer la capacité d'un contractant à développer des logiciels de qualité.

Ce modèle évalue différents attributs, incluant l'utilisation d'outils, les pratiques standards, etc. Il est basé sur des pratiques clés, que tout bon contractant devrait utiliser.

D'autres organisations se sont inspirées des travaux de la SEI pour développer des modèles d'évaluation. L'évaluation de la maturité peut être utilisée pour savoir quand et comment mesurer.

Parmi les techniques d'évaluation de maturité, on peut citer le projet ISO/IEC 15504 (SPICE ou Software Process Improvement and Capability dEtermination) et CMM (*Capability Maturity Models*). D'autres modèles d'évaluation se sont par après inspirés de ces deux ci. On peut citer dans ce cadre, le projet belge OWPL (*Observatoire Wallon des Pratiques Logicielles*).

1.2.7 Gestion par les méthodes de mesures

Il existe également des méthodes de mesure servant à faciliter la comparaison des projets informatiques entre [Fenton, 1997].

Cela est d'autant plus important si les logiciels jouent un rôle de support dans tout le projet, comme c'est parfois le cas quand le logiciel fait partie des produits d'une entreprise dont le principal domaine d'activité n'est pas en rapport avec le développement de produits logiciels.

Souvent, le client ou l'utilisateur final n'est pas familier avec les terminologies logicielles, d'où les mesures peuvent aider à donner une image de l'état d'avancement général. Ce qui fait, par conséquent, que le client n'a pas besoin d'avoir une bonne connaissance des langages de programmation, des compilateurs ou de la structure physique des ordinateurs.

Ces mesures doivent être faites de manière à renseigner à la fois le client et le développeur de l'état d'avancement du projet.

1.2.8 Evaluation des méthodes et outils

Il s'agit ici d'un ensemble de méthodes et d'outils permettant de rendre une organisation ou un projet plus productif et les produits meilleurs et moins chers. Plusieurs organisations expérimentent et font des études de cas sur plusieurs outils et méthodes, afin de savoir lesquels sont plus adaptés à leur situation. Et cela ne peut être fait s'il n'existe pas des mesures contrôlées et un système d'analyse fiable. [Fenton, 1997].

1.3 La problématique des méthodes de mesures en génie logiciel

Comme on vient de le voir il existe un bon nombre de méthodes d'évaluation des logiciels et il n'est pas toujours facile de savoir laquelle convient le mieux. Il y a un peu plus de dix ans, Eijogu [1991] affirmait qu'il n'existait pas encore de méthodes de mesures efficaces en génie logiciel. Cela reste valable de nos jours car on constate toujours que les méthodes de mesures en génie logiciel n'ont pas encore atteint leur niveau de maturité.

Ses principales critiques vis-à-vis des méthodes de mesures 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 suffisamment flexible pour être étendu à d'autres méthodes de mesures;
5. dans les modèles d'estimation, le modèle mathématique viole potentiellement les règles d'analyse de dimensionnalité.

Certains chercheurs [Kitchenham et al., 1989] ont soulevé les problèmes liés aux méthodes de mesure dites indirectes[9]. Celles-ci ont comme objectif, en partant des lignes de code et de l'examen de leur contenu et de leur structure, d'établir des liens et des relations avec diverses caractéristiques des programmes telles que la taille, la complexité, l'exactitude, la fiabilité, etc.

Ces chercheurs jugeaient les méthodes de mesure indirectes difficiles à interpréter et pas assez précises. Le fait qu'elles ne soient pas basées sur une théorie sous-jacente solide était également une des critiques soulevées. Tous cela a tendance à rendre les résultats finaux assez complexes à interpréter.

Comme l'explique Fenton [1991], « il y a tellement de dimensions impliquées et tellement d'utilisation de différents types d'échelle de mesure que le résultat (c'est-à-dire le chiffre final) n'est pas admissible mathématiquement, spécifiquement par rapport aux unités et aux dimensions ».

Quant à Abran [1994], il propose une série de caractéristiques obligatoires que doivent satisfaire les méthodes de mesure de logiciel pour être considérées comme des méthodes de mesure utiles en génie logiciel.

[9] Contrairement aux méthodes de mesure dites directes qui sont effectuées à partir du comptage des unités de base, telles les lignes de code. Généralement, les méthodes de mesures directes sont utilisées pour tenter de mesurer la complexité et la qualité du code.

Il recommande entre autres une identification formelle des dimensions, des attributs, de l'espace de la mesure, des processus de mesure intégrés à l'intérieur de chaque méthode de mesure, du contexte initial de référence, ainsi qu'une identification de la transformation (ou processus de transformation) requise pour retourner au contexte initial de référence.

Il faut donc s'assurer qu'une méthode de mesure vérifie un certain nombre de contraintes avant de l'appliquer.

Les points de fonction peuvent être cités parmi les méthodes de mesures indirectes. Les remarques concernant la précision et la validité des résultats sont également valables pour les méthodes de mesure basées sur les points de fonction.

2 La mesure basée sur l'analyse des points de fonction

Le présent chapitre se concentre sur les mesures de logiciel utilisant les points de fonction comme unité de mesure. Il présente les objectifs ainsi que l'historique de l'évolution des mesures fonctionnelles. Les diverses mesures des points de fonction sont également présentées de même que les domaines dans lesquels ils sont utilisés à travers une analyse de la fiabilité des mesures et des relations entre modèles de mesures.

2.1 Objectifs initiaux des mesures fonctionnelles

Le premier modèle des points de fonction a été proposé en octobre 1979 par Allan J. Albrecht. Il s'agissait d'une nouvelle technique d'analyse de productivité, appelée *Function Points Analysis (FPA)* qui devait aider à mesurer la performance et à améliorer les résultats des estimations.

Le tableau 2.1 présente les définitions de base d'Albrecht et al. [1983].

1	Les points de fonction sont une mesure du produit du processus du développement et d'entretien basée sur la fonction « utilisateur » 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.

Tableau 2-1 : Définitions des points de fonction par Albrecht [1983]

Les points de fonction sont donc basés initialement sur une quantification de la fonctionnalité du logiciel livré à l'utilisateur. L'objectif était d'établir une technique pour mesurer l'augmentation de productivité dans un ensemble de projets réalisés sur une période de 5 ans, de 1974 à 1978, dans le contexte du développement des logiciels chez IBM. Ces projets ont été présentés par Albrecht et Gaffney [1983].

Ces projets avaient été développés avec des langages de programmation et des outils différents. Il s'agissait de trouver une méthode de mesure valide, représentant les services fournis à l'utilisateur et qui soit indépendante du langage de programmation utilisé.

Albrecht a alors voulu concevoir et développer une nouvelle approche plus conforme aux besoins exprimés pour la mesure de l'augmentation de productivité.

2.2 Historique et évolution des points de fonctions

Albrecht a débuté ses travaux sur les points de fonction au milieu des années 70. À la même période, Tom DeMarco [1982], à l'époque consultant en management, entreprend également ses premières recherches, pour les publier[10] trois ans après Albrecht en 1982. Il s'agissait de deux formes indépendantes de méthodes de mesure de points de fonction. Les points de fonction définis par DeMarco et Albrecht étaient similaires dans les concepts mais pas dans la forme.

Les points de fonction de DeMarco sont apparus dans beaucoup d'outils d'estimation de coût, mais ont vite été remplacés par les points de fonction d'Albrecht, qui étaient largement plus faciles à automatiser. De plus ceux-ci avaient l'aura d'IBM derrière eux.

Les points de fonction de DeMarco ont refait leur apparition récemment et se sont inscrits comme fondement d'un bon nombre de points de fonction. On peut citer par exemple FFP *(Full fonction points)* [St-Pierre et al., 1997], ainsi que COSMIC-FFP (*Common Software Measurement International Consortium - Full Function Points)* dont la version 2.1 est sortie en octobre 2001 [Abran et Desharnais, 2001].

En novembre 1983, une version modifiée des points de fonction d'Albrecht/IBM a été publiée à Londres par Charles Symons. Il l'a appelée *Mark II fonction points.* Cette variation de la méthode de mesure des points de fonction basique est devenue largement utilisée en Angleterre, et constituait la première variation d'une trentaine de publications sur les points de fonction [Symons, 1991].

Il est importe de signaler qu'un bon nombre d'outils d'estimation du coût des logiciels supporte à la fois les points de fonction américains tels que définis dans l'*International Function Point Users Group (IFPUG)* et les points de fonction britanniques *Mark II Function points* .

En 1984, (via la conférence Share Guide), IBM a effectué une révision majeure des règles de comptage des points de fonction [Albrecht, 1984]. Les règles revues ajoutaient une procédure d'évaluation de la complexité. Les révisions d'IBM de 1984 sont donc ainsi devenues la méthode basique de comptage des points de fonction quand IFPUG a été formé et a assumé la responsabilité des règles de comptage des points de fonction.

C'est en 1986 que la responsabilité de définir des méthodes de mesures de points de fonction a été prise en charge de façon officielle par IFPUG. Aujourd'hui, la plupart des modèles d'estimation de coût incluent un support explicite de la mesure en points de fonction IFPUG ou Mark II.

Le succès et l'expansion des points de fonction ainsi que l'organisation IFPUG ont amené une normalisation de facto de la méthode de mesure en points de fonction. Il y a eu plusieurs variantes suggérées, la plupart étant des modifications mineures. IFPUG

[10] Le livre publié s'intitulait *Controlling Software Projects* [DeMarco 1982].

considère qu'il est le gardien de la norme des points de fonction des années 80, ce qui n'a pas permis une évolution suffisante de la mesure pour couvrir adéquatement la mesure fonctionnelle des domaines autres que les systèmes de gestion, tel que le temps réel.

Le tableau 2-2 issus d'Abran [1994] présente un récapitulatif des versions officielles:

Versions	
1	Albrecht 79
2	Albrecht 83
3	GUIDE 1985
4	IFPUG 86
5	IFPUG 88
6	IFPUG 90
7	IFPUG E/R
8	IFPUG 1994 (4.0)
9	IFPUG 1999 (4.1)

Tableau 2-2 : Liste des versions officielles

Il est à noter qu'IFPUG n'est pas vraiment une organisation de standardisation au même titre que l'IEEE *(Institute of Electrival and Electronic Engineers)* ou l'ISO[11] *(International Organisation for standardization).* Cela signifie que les règles de comptage des points de fonction telles que définies par IFPUG ne constituent pas des normes. D'où l'émergence d'un bon nombre[12] de mesures fonctionnelles inspirées d'IFPUG. Cette liberté de modification prise par de nombreux organismes dans différents pays vient surtout de la difficulté de bien appliquer la mesure à partir des définitions IFPUG [Abran, 1994]

ISO avec la norme 14143-1 a reconnu l'existence d'une norme fonctionnelle, mais n'a pas voulu reconnaître une norme particulière. La norme 14143-1 sera suivie par d'autres normes (14143-2 à 5) visant à indiquer comment les différentes mesures fonctionnelles pourront être reconnues par ISO comme répondant à la norme 14143-1.

Il y a déjà eu une reconnaissance de Mark II et de IFPUG (FPA) par ISO comme mesures fonctionnelles au sens de 14143-1. (Cette reconnaissance devrait également se faire en 2002 pour la méthode de mesure fonctionnelle COSMIC-FFP[13]).

[11] ISO a proposé un ensemble de définitions standards dans le domaine des mesures de logiciel utilisant les points de fonction, dans « ISO/IEC 14143-1 : 1997- Information technology – Software measurement – Functional size measurement – Définition of concept, October 22 1997 ».

[12] Voir tableau 2.3.

[13] *Common Software Measurement International Consortium –Full Function Points,* désigné sous l'acronyme COSMIC-FFP, est la méthode de mesure fonctionnelle utilise dans le travail d'expérimentation présenté au chapitre 4.

Comme cela a été dit plus haut, un bon nombre de modèles de mesure se sont inspirés des points de fonctions d'Albrecht. Ceux-ci sont présentés dans la section suivante sur les méthodes de mesure dérivées.

2.2.1 Les méthodes de mesure dérivées

D'autres modèles de mesure des points de fonction se sont basés sur les travaux d'Albrecht et ont évolué parallèlement à IFPUG. Mark II peut s'inscrire dans ce cadre.

Le modèle Mark II a été proposé par Symons [1988], il remettait en cause la structure de base du modèle d'Albrecht. Il jugeait que l'hypothèse d'Albrecht était basée 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. Il est basé sur le concept de *transaction logique*, qui définit chaque transaction comme étant une séquence de données en entrée, de traitement et de données en sortie.

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 toutes les entités référencées et de tous les éléments en sortie.

Le modèle de Symons a été principalement critiqué à cause de sa dépendance à la technologie utilisée. Il faut noter également que d'une part, le détail de ce modèle 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 non plus des améliorations apportées constamment au modèle officiel IFPUG.

Depuis 1995, IFPUG possède un équivalent anglais. Il s'agit de l'*United Kingdom Function Point Users Group*. Il existe plusieurs autres organisations des méthodes de mesures à travers le monde. On peut citer par exemple, les organisations australiennes (l'Australian Software Metrics Associations ou *ASMA*), hollandaises (le Netherlands Function Point Users Group (NEFPUG), etc.

Le domaine d'application des méthodes de mesure vu ici (IFPUG, Mark II) a longtemps été les logiciels de gestion comme l'a noté Glady et al [1987]. Depuis, plusieurs auteurs [Reifer, 1991 ; Jones, 1997] se sont penchés sur le problème et ont essayé d'adapter IFPUG au contexte des systèmes en temps réel et aux systèmes embarqués (« embedded systems »). La méthode de mesure fonctionnelle COSMIC-FFP (voir chapitre 3) utilisée dans le travail d'expérimentation (voir chapitre 4) représente un exemple récent d'adaptation aux logiciels en temps réel.

2.3 Variation des points de fonction

En 1997, il existait environ une trentaine de variantes de mesures en points de fonction comme présentée dans le tableau 2-3 [Jones, 1997]. Ces méthodes de mesure n'ont pas tous connus le succès d'IFPUG. En effet, la plupart n'avaient pas fourni assez de détail sur leur applicabilité

1. 1975- La méthode des points de fonction interne à IBM
2. 1979- La méthode IBM des points de fonction publié par Albrecht
3. 1982- La méthode des points de fonction DeMarco bang (qui est une mesure fonctionnelle générique)
4. 1983- La méthode britannique des points de fonction Mark II (Symons)
5. 1984- La méthode IBM révisée des points de fonction
6.1986- La méthode IFPUG version 1
7.1988- La méthode IFPUG version 2
8.1990- La méthode IFPUG version 3
9.1995- La méthode IFPUG version 4
10.1992- Les points de fonction Reifer et les méthodes de mesures Halstead
11.1992- ViaSoft backfire function point method
12.1993-Aperçu de la méthode des points de fonction Gartner Group
13.1994- La méthode des points de fonction non-ajustés
14.1994- La méthode des points de fonction Bachman analyst
15.1995- Aperçu de la méthode des points de fonction Compass Group b
16.1995- La méthode des points de fonction d'Oracle
17.1995- La méthode des points de fonction de l'ingénierie des forces aériennes
18.1996- CRIM (*micro-function point method*)
19.1996- La méthode des points objets (*object point*)
20.1997-La méthode des points de données pour la mesure des bases de données
21.1997-L'approche Nokia des points de fonction (logiciel de télécommunication)
22.1997-*Full Function points* approche pour les logiciels temps - réel
23.1997- Approche proposée des points de fonction Symons-ISMI.

Tableau 2-3 : Variations des points de fonction

La liste présentée dans le tableau 2-3, n'est pas certifiée à 100% complète. En effet, elle ne reprend généralement que les méthodes des points de fonction ayant déjà fait l'objet d'une discussion lors des conférences sur des méthodes de mesure ou qui sont déjà apparues dans la littérature. Bien entendu les méthodes de mesures apparues après 1998 comme COSMIC-FFP [Abran et al, 2001] ne sont pas non plus reprises[14].

[14] Le tableau 2.3 tiré de Jones [1997] a été trié afin de garder uniquement les méthodes de mesure fonctionnelle. Les outils d'estimation, les normes ainsi que les adaptations des méthodes de mesure ont été enlevés.

Jones [1997], dans ses travaux a analysé les raisons pour lesquelles il y avait un aussi grand nombre de variations de méthodes de mesure. Il a été prouvé durant les années 70 que les méthodes de mesure basées sur les lignes de code ne pouvaient mesurer ni le coût ni la productivité dans un sens économique.

Il s'est donc avéré nécessaire de faire la recherche et d'adopter de nouvelles mesures fonctionnelles. A peu près une demi-douzaine de variations ont été principalement créées pour combler les lacunes ou les vides apparus avec les premières versions d'Albrecht, comme par exemple le développement de méthodes de mesure orienté logiciel temps réel[15].

L'existence d'autant de variations dans le comptage des points de fonction constitue un handicap majeur pour l'ensemble des mesures de logiciels.

2.4 Fiabilité de la méthode de mesure des points de fonction

Une bonne méthode de mesure doit posséder parmi ses attributs les caractéristiques suivantes [Navlaka, 1986] :

- **L'exactitude**, c'est-à-dire qu'à partir des mêmes données et règles, on doit pouvoir obtenir les mêmes résultats
- **La répétitivité,** celle-ci implique que quelles que soient les personnes qui utilisent la méthode de mesure, on doit pouvoir obtenir les mêmes résultats et ces résultats doivent demeurer les mêmes dans le temps.

Des travaux ont déjà été menés par des chercheurs [Rudolph, 1989 ; Low et al, 1990 ; Kemerer, 1990 ; Whisehunt, 1990] pour évaluer l'ampleur de la variation des résultats et pour identifier et analyser les sources potentielles de différences.

Il a été observé en industrie [Whisehunt, 1990] que le taux d'exactitude et de vérifiabilité des pointages s'élève à +/- 95% avec la mise sur pied de mécanismes de contrôle et de validation des mesures.

De plus, le fait d'appliquer deux méthodes de mesure différentes aux mêmes projets peut s'avérer utile, bien que pouvant être très coûteux en terme de temps. Comme on va le constater dans le travail d'expérimentation[16] qui a été fait, il arrive souvent que les résultats obtenus par deux méthodes de mesure différentes se rejoignent. Cela peut s'avérer intéressant dans ce sens que cela permet de valider et de vérifier les résultats obtenus.

---

[15] Les points de fonction d'Albrecht ont été principalement conçus pour les logiciels de gestion.
[16] Voir chapitre 4.

Les projets analysés dans le cadre de cette étude ont été mesurés à l'aide de deux méthodes de mesure basées sur la technique des points de fonction. Il s'agit des méthodes de mesure fonctionnelle IFPUG (FPA) et COSMIC-FFP. Sans trop rentrer dans les détails car cela sera revu en long et en large au chapitre 4, on a pu constater que les résultats obtenus sur ces mêmes projets se rejoignaient, alors que ces deux méthodes utilisaient des règles de transformation différentes. IFPUG[17] utilise des tables de complexité, alors que COSMIC-FFP calcule le nombre de sous - processus. [18]

2.5 Analyse de la documentation fonctionnelle à travers le processus de mesure des points de fonction

Le calcul des points de fonction exige souvent que celui-ci soit réalisé par un personnel spécialisé dans le comptage de points de fonction. En effet, pour appliquer la méthode des points de fonction, il faut non seulement une connaissance des règles des points de fonction, mais aussi savoir les appliquer dans différents environnements, puisque le comptage est effectué à partir des spécifications des logiciels et des exigences formelles écrites. Le plus souvent, l'assistance du représentant des utilisateurs du logiciel est requise.

Bien que des outils d'aide au comptage aient été proposés [Desharnais, 2000], la grande majorité des mesures se font manuellement et en se basant sur la documentation sous forme de texte.

Les grandes catégories des documents pouvant être produits en général dans des projets logiciels sont [Jones, 1997]:

1. les documents de planning
2. les exigences *(Requirements)*
3. les spécifications
4. les manuels d'utilisation
5. les matériaux de formation *(training materials)*
6. les Matériaux de Marketing
7. le rapport des défauts
8. les documents financiers
9. les mémos et correspondance
10. les contrats et documents légaux.

La production de ces données peut être très coûteuse suivant la quantité d'information nécessaire. D'ailleurs, la mesure des documents à délivrer est prise en compte par les outils d'estimation du coût des produits logiciels. Entre 20% et un peu plus de 50% du

[17] Une présentation succincte des règles de mesure d'IFPUG sera donnée à l'Annexe 1.
[18] Voir chapitre 4.

budget alloué aux projets logiciels peut aller à la production de papiers et de documents on-line [Jones, 1997].

Un autre problème pouvant être relevé lors de l'application de la mesure est le fait que la grande majorité des méthodes de mesure de points de fonction ne détaille pas de manière concise la façon dont les informations utiles à la mesure seront repérées et extraites de la documentation fonctionnelle [Desharnais, 2001]. Cela est souvent une source d'erreurs de mesure et d'incompatibilité des résultats obtenus. L'analyse de la documentation fonctionnelle est revue au chapitre 4 sur la base de l'expérimentation faite avec la méthode de mesure fonctionnelle COSMIC-FFP.

3 La mesure fonctionnelle COSMIC-FFP[19]

La méthode de mesure COSMIC-FFP (*Common Software Measurement International Consortium - Full Function Points*) retrouve ses origines en 1997 quand la version 1.0 de FFP (*Full Function Point*) a été publiée. FFP a été créée dans le but d'offrir une mesure détaillée de la taille fonctionnelle spécifiquement adaptée aux logiciels en temps réel. Cela ne l'empêche pas d'être tout aussi adaptable aux logiciels de gestion et aux logiciels techniques (comme les systèmes d'exploitation).

La méthode de mesure COSMIC-FFP a été créée par les chercheurs du laboratoire de recherche en gestion du logiciel[20] de l'Université du Québec à Montréal (UQAM) en association avec d'autres chercheurs et experts du groupe COSIMC[21]. COSMIC-FFP a subi plusieurs améliorations depuis 1997, les dernières modifications (COSMIC-FFP version 2.1) datent de mars 2001[22].

3.1 Applicabilité de la méthode de mesure COSMIC-FFP

La méthode de mesure COSMIC-FFP a été conçue pour être appliquée aux domaines suivants :

- Logiciels de gestion, tels que les applications typiques aux banques, aux assurances, à la comptabilité, au personnel, aux achats, à la distribution et la production, entre autres.
- Logiciels en temps réel dont la tâche est de garder le contrôle des événements du monde réel, comme par exemple les logiciels d'échange téléphonique et d'interruption des messages, les logiciels embarqués dans les dispositifs de contrôle de différents appareils tels que les appareils électroménagers ou encore les moteurs d'ascenseur, etc.
- Les logiciels hybrides des précédents tels que les logiciels de réservations d'avions et d'hôtels en temps réel.

MODELES DU PROCESSUS DE MESURE COSMIC-FFP

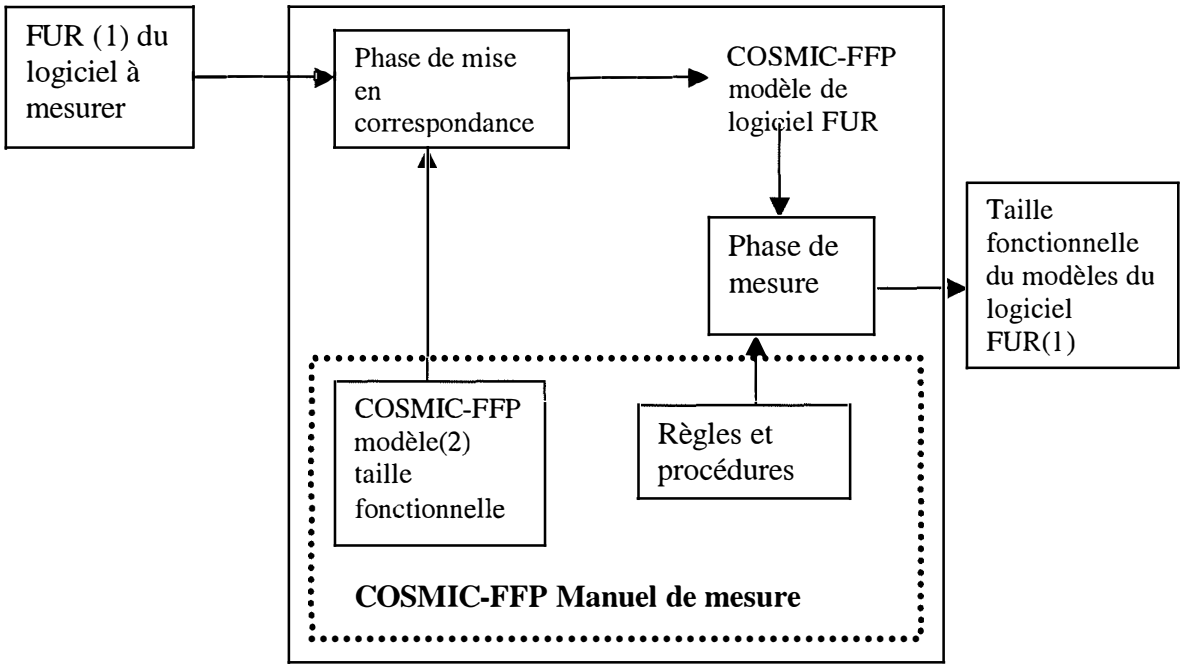
La méthode de mesure COSMIC-FFP consiste à appliquer un ensemble de règles et de procédures sur un logiciel donné, tel qu'il est perçu par ses utilisateurs, c'est-à-dire à

[19] Il s'agit ici de la version 2.1 de COSMIC-FFP.
[20] Voir www.lrgl.uqam.ca pour plus d'information
[21] Voir www.cosmicon.com pour plus d'information
[22] Le présent chapitre est un résumé du manuel de mesure COSMIC-FFP.

travers les besoins fonctionnels. Le résultat de l'application de la mesure donne une valeur numérique qui quantifie la taille fonctionnelle du logiciel.

Dans la perspective de la méthode de mesure COSMIC-FFP, un logiciel est considéré suivant les fonctionnalités livrées aux utilisateurs. La fonctionnalité livrée aux utilisateurs est décrite via les Fonctionnalités Utilisateurs Requises (FUR).

Ces FUR peuvent exister dans la pratique, sous la forme de documents spécifiques, (spécifications des besoins requis, etc.) ou encore ils peuvent être dérivés des autres objets du génie logiciel à savoir les objets de l'architecture, de la modélisation et de la conception. On peut donc aisément imaginer la mesure d'un logiciel avant même son installation. La figure suivante présente le modèle du processus de mesure de COSMIC-FFP.



(1) FUR : Fonctionnalités Utilisateur Requises
(2) Le modèle fonctionnel inclut les concepts, définitions et les structures relationnelles des attributs de la taille fonctionnelle

Figure 3-1 : Modèle du processus de mesure de COSMIC-FFP

Il faut noter que la méthode de mesure COSMIC-FFP n'est pas conçue actuellement pour fournir d'une façon normalisée la taille de certains types de fonctionnalités Utilisateurs

Requises, entre autres, la complexité mathématique des algorithmes ou une séquence complexe des règles telles que trouvées dans les systèmes experts.[23]

3.2 Les composants de la méthode de mesure COSMIC-FFP[24]

Cette section définie les différents composants ainsi que les terminologies de la méthode de mesure COSMIC-FFP.

3.2.1 Couches

Une couche est le résultat du partitionnement fonctionnel de l'environnement du logiciel où tous les processus fonctionnels s'exécutent au même niveau d'abstraction.

Dans un environnement logiciel à plusieurs couches, les logiciels interagissent les uns par rapport aux autres via leurs processus fonctionnels respectifs. Ces interactions sont hiérarchiques par nature. Plus précisément, lorsque considérées par paire, une couche est dite « cliente » d'une autre. Et une couche « cliente » utilise les services fonctionnels fournis par les couches subordonnées.

3.2.2 Frontière

La frontière d'un logiciel est la ligne conceptuelle séparant ce logiciel de l'environnement dans lequel il opère, tel que perçu par ses utilisateurs d'un point de vue externe. La frontière permet à la personne qui mesure de distinguer, sans ambiguïté, ce qui est inclus dans le logiciel de ce qui fait partie de l'environnement dans lequel fonctionne ce logiciel.

Utilisateurs

Les utilisateurs peuvent être soit un être humain, un autre logiciel ou une machine qui interagit avec le logiciel mesuré.

Fonctionnalités Utilisateur Requises (FUR)

C'est une expression ISO [ISO/IEC 14143-1:1997] désignant un sous-ensemble de besoins de l'utilisateur et selon la vue de l'utilisateur. Les FUR représentent les pratiques et procédures de l'utilisateur que le logiciel doit accomplir pour répondre aux besoins de celui-ci. Les FUR excluent les besoins en matière de qualité et les besoins techniques[25]. Les termes « requis fonctionnels » ou « demandes » sont également employés dans l'industrie comme synonymes.

---

[23] Voir manuel de mesure COSMIC-FFP Version 2.1.

[24] Les définitions présentées dans cette section sont tirées du manuel de mesure COSMIC-FFP version 2.1.

[25] Voir « ISO/IEC 14143-1:1997 –Information technology - Software measurement – Functional size measurement – Definition of concepts », October 22 1997, section 3.8.

3.2.3 Les Données

L'ensemble des données considérées dans la méthode de mesure fonctionnelle COSMIC-FFP est constitué d'attributs et de groupes de données pouvant être persistants ou pas.

Attributs

Un attribut est la plus petite parcelle d'information codée portant une signification dans la perspective fonctionnelle des besoins d'un utilisateur.

Groupe de données

Un groupe de données est constitué d'attributs distincts, non vides, non ordonnées et non redondants. Chaque attribut décrit un aspect complémentaire du même objet d'intérêt. Un groupe de données est caractérisé par sa persistance.

Persistance d'un groupe de données

La persistance d'un groupe de donnée caractérise le temps pendant lequel un groupe de données est retenu dans le contexte des Fonctionnalités Utilisateurs Requises (FUR). On identifie trois types de persistance :

- Transitoire : le groupe de données ne survit pas au-delà de la transaction qui l'utilise
- Courte : le groupe de données survit au-delà de la transaction qui l'utilise mais ne survit pas quand le logiciel cesse d'être opérationnel.
- Durable : le groupe de données survit au-delà de la durée opérationnelle du logiciel.

3.2.4 Processus fonctionnels

Processus fonctionnel

Un processus fonctionnel est un ensemble de mouvements de données unique (entrée, sortie, lecture, écriture) réalisant de façon cohérente et logiquement indivisibles des besoins fonctionnels de l'utilisateur.

Il est déclenché directement ou indirectement via un « acteur » par un déclencheur. Il est complet lorsqu'il a réalisé tout ce qui doit être exécuté en réponse à un événement déclencheur.

Événement déclencheur

Un événement déclencheur se produit à l'extérieur de la frontière du logiciel mesuré et initie un ou plusieurs processus fonctionnels. Les horloges et les événements temporels peuvent être un événement déclencheur. Lorsque chaque couche identifiée est entourée par une frontière, des événements déclencheurs peuvent se produire dans une couche et initier un processus fonctionnel appartenant à une autre couche.

3.2.5 Sous-processus fonctionnels

Il s'agit d'un mouvement de données élémentaires fonctionnel se produisant pendant l'exécution d'un processus fonctionnel. Il y a quatre types de mouvements de données : les *Entrées*, les *Sorties*, les *Lectures* et les *Ecritures*. Un mouvement de données élémentaires renvoie à un ensemble d'attributs trouvés dans un et un seul groupe de données.

Un sous-processus au sens de la méthode de mesure fonctionnelle COSMIC-FFP exprime uniquement la fonctionnalité utilisateur requise et exclut les requis de qualité et les requis techniques.

On distingue les sous-processus suivants :

1. *Entrée*

Une *Entrée* déplace un attribut trouvé dans un groupe de données du point de vue de l'utilisateur vers l'intérieur de la frontière du logiciel. Une *Entrée* ne met pas à jour de données lors de leur déplacement. Fonctionnellement, le sous - processus *Entrée* transmet des attributs se trouvant du côté utilisateur à la frontière du logiciel dans les limites du processus fonctionnel auquel il appartient.

2. *Sortie*

Une *Sortie* déplace un attribut trouvé dans un groupe de données de l'intérieur de la frontière du logiciel vers l'utilisateur de celui-ci. Une *Sortie* ne lit pas les attributs qu'il déplace. Fonctionnellement, le sous-processus *Sortie* transmet les attributs se trouvant à l'intérieur du processus fonctionnel auxquels ils appartiennent à l'utilisateur du logiciel dans son environnement opérationnel.

3. *Lecture*

Une lecture réfère aux attributs trouvés dans un groupe de donnés sans en modifier la valeur. Fonctionnellement, un sous - processus de lecture met des attributs se trouvant à l'extérieur de la frontière, du côté stockage, à la portée du processus fonctionnel auquel il appartient.

4. *Écriture*

Une écriture réfère aux attributs trouvés dans un groupe de données et en modifie la valeur. Fonctionnellement, un sous-processus d'écriture transmet des attributs se trouvant à l'intérieur du processus fonctionnel auquel il appartient vers l'extérieur de la frontière, du côté stockage.

3.3 Brève présentation des règles de la méthode de mesure COSMIC-FFP [26]

La méthode de mesure COSMIC-FFP considère la mesure de la taille fonctionnelle du logiciel à travers deux phases distinctes : la mise en correspondance du logiciel à mesurer avec le modèle de logiciel de COSMIC-FFP et la mesure des aspects spécifiques de ce modèle de logiciel, (voir figure 3-1).

3.3.1 Règles et principes d'identification des couches du logiciel

La subdivision fonctionnelle de l'environnement du logiciel suivant la définition donnée des couches permet l'identification de celles-ci. Il s'agit d'un processus itératif, le partitionnement de l'application est affiné tout au long du processus de mise en correspondance.

Après avoir été identifiée, chaque couche potentielle doit répondre aux principes suivants :

1. Les logiciels de toutes les couches livrent des fonctionnalités à ses utilisateurs (un utilisateur peut être une personne, une machine ou un autre logiciel).

2. Un logiciel dans une couche subordonnée fournit des services fonctionnels aux couches clientes.

3. Un logiciel dans une couche subordonnée peut opérer sans l'assistance d'un logiciel dans une couche cliente.

4. Le logiciel dans une couche cliente peut ne pas opérer correctement si le logiciel de la couche subordonnée dont il dépend ne fonctionne pas correctement.

5. Le logiciel d'une couche cliente n'utilise pas nécessairement toute la fonctionnalité fournie par un logiciel d'une couche subordonnée.

[26] L'application faite de la méthode de mesure COSMIC-FFP dans le cadre du travail d'expérimentation est donnée au chapitre 4.

6. Le logiciel d'une couche subordonnée peut être une couche cliente d'une autre couche tierce.

7. Les logiciels des couches clientes et subordonnées peuvent physiquement échanger des données. Cependant le logiciel de chaque couche interprète les données différemment.

8. Des logiciels qui partagent des données ne peuvent être considérés comme appartenant à des couches différentes s'ils interprètent les données partagées de façon identique.

Quelques règles sont également à considérer dans le processus de mise en correspondance au niveau des couches :

1. si un logiciel est conçu sur la base d'un concept architectural connu, alors les paradigmes de cette architecture peuvent être utilisés pour identifier les couches.

2. Le niveau applicatif du logiciel est généralement considéré comme résidant au plus haut niveau de contrôle dans la hiérarchie des couches.

3. Les services fonctionnels des progiciels tels les bases de données, les interfaces graphiques utilisateurs, les systèmes d'exploitation ou les pilotes sont généralement considérés comme des couches distinctes.

3.3.2 Règles et principes d'identification de la frontière du logiciel

L'identification de la frontière est également un processus itératif, elle peut être réajustée au cours du processus de mise en correspondance.

Par définition, il y a une frontière entre des couches adjacentes. Il peut également y avoir des frontières entre les différents morceaux de logiciels de même couche si les échanges de données sont de même niveau.

Les règles suivantes peuvent être utilisées pour identifier la frontière :

1. Identifier un événement déclencheur, ensuite identifier un processus fonctionnel mis en route par cet événement. La frontière se situe entre l'événement déclencheur et le processus fonctionnel identifié.

2. Identifier les dispositifs d'Entrée/Sortie utilisés par le logiciel mesuré. Mettre en relation les dispositifs d'Entrée/Sortie avec les processus fonctionnels du logiciel mesuré. La frontière se situe entre ces processus fonctionnels et les dispositifs d'Entrée/Sortie identifiés.

3. Le concept de couche peut être utilisé pour identifier les frontières des logiciels en temps réel et celles des logiciels techniques.

3.3.3 Règles et principes d'identification des groupes de données

La définition avancée à la section 3.1 sur les groupes de données selon la méthode de mesure fonctionnelle COSMIC-FFP permet de relever les groupes de données potentiels.

Après avoir été identifié, chaque groupe de données doit se conformer aux principes suivants :

1. Le groupe de données doit être implanté dans le système informatique supportant le logiciel.

2. Chaque groupe de données identifié doit être unique quant à l'ensemble de données qu'il contient.

3. Les groupes de données sont directement liés aux objets décrits dans les besoins fonctionnels de l'utilisateur du logiciel.

La définition des groupes de données et ses principes sont intentionnellement élargis en vue d'être applicables à un grand nombre de logiciel, ce qui a parfois comme inconvénient de rendre plus difficile l'application de la mesure pour une pièce de logiciel en particulier[27].

Un groupe de données peut contenir un seul attribut si celui-ci est suffisant pour décrire un objet spécifié par les besoins fonctionnels de l'utilisateur.

Les attributs sont identifiés selon les règles suivantes :

1. Un attribut doit faire partie d'un type d'attribut valide.
2. Un attribut doit représenter une donnée atomique référencée par le logiciel mesuré, du point de vue des besoins fonctionnels de l'utilisateur.

3.3.4 Règles et principes d'identification des processus fonctionnels

L'envergure de la mesure fonctionnelle de la taille correspond à toutes les Fonctionnalités Utilisateurs Requises décrivant les processus fonctionnels identifiés.

L'identification des processus fonctionnels potentiels est effectuée à l'aide de la définition COSMIC-FFP d'un processus fonctionnel (voir section 3.1).

---

[7] Le mesureur peut se référer au manuel de mesure COSMIC-FFP pour des règles d'identification des groupes de données plus détaillées.

Les processus fonctionnels éventuels ayant été identifiés doivent être conformes aux principes suivants :

- Un processus fonctionnel est lié au moins à un besoin fonctionnel de l'utilisateur.
- Un processus fonctionnel est exécuté lorsqu'un événement déclencheur identifié survient.
- Un processus fonctionnel contient au moins un mouvement de données, une entrée et une sortie ou une écriture.
- Un processus fonctionnel ne contient pas plus d'un état d'attente (qui peut apparaître lorsqu'il est complet).
- Un processus fonctionnel appartient à une et une seule couche.

Les règles suivantes peuvent également être utilisées pour identifier un processus fonctionnel avec plus de certitude :

- Des sous ensembles d'événements déclencheurs ne sont pas considérés comme des événements déclencheurs distincts.
- Dans le contexte des logiciels en temps réel, un processus fonctionnel peut également être déclenché par un événement. Il se termine quand un point asynchrone est atteint. Un point asynchrone est atteint quand, dans une séquence de mouvements de données, un des mouvements de données n'est pas synchronisé avec celui qui le précède.

Chaque processus fonctionnel identifié dans le logiciel mesuré peut être subdivisé en sous-processus (entrée, sortie, lecture, écriture).

La figure 3-2 présenté dans le manuel de mesure COSMIC-FFP donne les différentes relations entre les sous-processus, le processus fonctionnel et la frontière du logiciel mesuré.

Comme on peut le constater sur la figure, un utilisateur peut rentrer des données dans la frontière, ces données peuvent être traitées par le processus fonctionnel, qui peut effectuer une lecture ou une écriture de données à l'extérieur de la frontière de traitement (Base de données).

Bien entendu, il peut y avoir également une sortie de données vers l'utilisateur comme des messages d'erreur ou des résultats d'une demande de visualisation suivant le genre le type de processus.

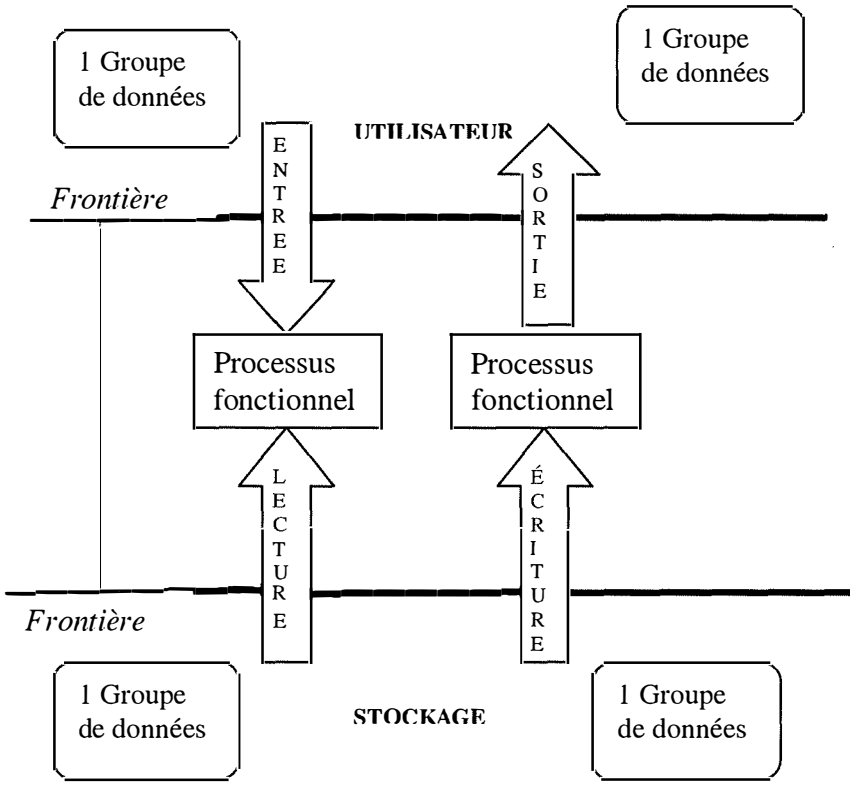


Figure 3-2 : Les quatre types de sous-processus avec leur environnement.

3.3.5 Règles et procédures de mesure du sous - processus Entrée

Une Entrée représente toutes les manipulations de formatage et de présentation requises par l'utilisateur ainsi que toutes les validations associées à l'entrée des données tant que ces manipulations n'impliquent pas d'autres types de sous-processus. Par exemple, une Entrée inclut toutes les manipulations citées à l'exception des lectures que pourraient exiger la validation de certains codes ou l'obtention de descriptions associées.

Les Entrées potentielles sont identifiées conformément à la définition COSMIC-FFP (voir section 3.1). Une fois identifiés, les sous-processus Entrées potentiels doivent s'accorder aux principes suivants :

1. Le groupe de données reçoit des attributs en provenance d'un groupe de données localisé à l'extérieur de la frontière du logiciel, du côté de l'utilisateur.

2. Le sous-processus ne reçoit d'attributs que d'un seul groupe de données. Si plus d'un groupe de données est référencé, une Entrée est identifiée pour chaque groupe de données.

3. Le sous - processus ne sort, ne lit ou n'écrit de données.

4. Au sein du processus fonctionnel où il est identifié, le sous-processus est unique, c'est-à-dire que les traitements et attributs identifiés sont différents des autres Entrées identifiées dans le même processus.

Les règles suivantes peuvent être utilisées pour confirmer le statut de sous-processus Entrée :

- Les évènements déclenchés par l'horloge de l'ordinateur sont considérés comme externes. Par exemple, un événement qui survient toutes les 3 secondes est compté comme une Entrée avec un attribut. Cependant le processus qui génère l'événement périodiquement est ignoré s'il survient en dehors de la frontière du logiciel.

- La lecture de l'horloge n'est pas considérée comme une Entrée à moins qu'un tel processus ne soit explicitement requis au sein du logiciel mesuré, tel que spécifié dans les besoins fonctionnels de l'utilisateur. Par exemple, lorsqu'un processus écrit automatiquement l'heure on n'identifie pas d'Entrée pour l'obtention de la valeur fournie par l'horloge interne.

3.3.6 Règles et procédures de mesure du sous - processus *Sortie*

Une sortie représente toutes les manipulations de formatage et de présentation requises par l'utilisateur, incluant les traitements requis pour acheminer les résultats à l'utilisateur, tant que ces manipulations n'impliquent pas d'autres types de sous-processus.

Les Sorties potentielles sont identifiées conformément à la définition COSMIC-FFP (voir section 3.1). Une fois identifiés, les sous-processus Sorties « candidats » doivent s'accorder aux principes suivants :

1. Le sous-processus envoie des attributs appartenant à un groupe de données à l'extérieur des frontières du logiciel, du côté de l'utilisateur.

2. Le sous-processus envoie des attributs n'appartenant qu'à un seul groupe de données. Si des attributs appartenant à plus d'un groupe de données sont envoyés, il faut alors identifier une Sortie pour chaque groupe de données.

3. Le sous-processus n'entre, ne lit ou n'écrit de données.

4. Au sein du processus fonctionnel où il est identifié, le sous-processus est unique, c'est-à-dire que les traitements et attributs identifiés sont différents des autres Sorties identifiées au sein du même processus.

Tous les messages ne contenant pas de données utilisateur comme par exemple la confirmation et les messages d'erreur, sont traités comme une seule Sortie au sein du processus fonctionnel.

3.3.7 Règles et procédures de mesure des sous - processus *Lecture* et *Écriture*

Une Lecture représente tous les traitements et les calculs associés aux données lues, tant que ces traitements n'impliquent pas d'autres types de sous-processus.

Une Écriture représente quant à elle, tous les traitements et les calculs associés au stockage des attributs pour autant que ces traitements n'impliquent pas d'autres types de sous-processus. Les Lectures et Ecritures potentielles sont identifiées conformément à la définition COSMIC-FFP (voir section 3.1).

Les sous-processus Lecture potentiels doivent être conformes aux principes suivants :

1. Le sous-processus lit des attributs se trouvant à l'extérieur de la frontière du logiciel, du côté stockage, sans modifier leur valeur.

2. Le sous-processus lit des attributs se trouvant dans un seul groupe de données. Si plus d'un groupe de données est lu, il faut alors identifier une Lecture pour chaque groupe de données.

3. Le sous-processus n'entre, ne sort ni n'écrit de données.

4. Au sein du processus fonctionnel où il est identifié, le sous - processus est unique, c'est à dire que les traitements et attributs identifiés sont différents des autres Lectures associées au même processus fonctionnel.

Les sous-processus Ecriture potentiels doivent quant à eux être conformes aux principes suivants :

1. Le sous-processus modifie la valeur des attributs, du côté stockage de la frontière du logiciel.

2. Le sous-processus modifie les valeurs des attributs d'un seul groupe de données. Si on modifie plus d'un groupe de données, alors il faut identifier une Ecriture pour chaque groupe de données.

3. Le sous-processus n'entre, ne sort, ou ne lit de données.

4. Au sein du processus fonctionnel où il est identifié, le sous-processus est unique, c'est à dire que les traitements et attributs identifiés sont différents des autres Écritures associées au même processus fonctionnel.

3.3.8 Application de la fonction de mesure[28].

La fonction de mesure de COSMIC-FFP est une fonction mathématique qui assigne une valeur numérique au sous-processus. Chaque instance d'un sous-processus identifié selon les règles COSMIC-FFP se voit attribuer une unité de taille COSMIC-FFP qui est définie par convention comme un mouvement élémentaire de données.

Après application de la fonction de mesure, les résultats obtenus sont regroupés en un seul nombre représentant ***la taille fonctionnelle***, conformément aux règles et principes suivants :

1. Pour chaque couche identifiée, la taille fonctionnelle des sous-processus est regroupée en un seul nombre par l'intermédiaire d'une addition arithmétique.

$$\text{Taille (couche)}_{FFP} = \Sigma_i \text{ taille (Entrées}_i) + \Sigma_i \text{ taille (Sorties}_i) + \Sigma_i \text{ taille (Lecture}_i) + \Sigma_i \text{ taille (Ecritures}_i)$$

2. Dans chaque couche identifiée, la taille fonctionnelle des changements des Fonctionnalités Utilisateurs Requises (FUR) est totalisée à partir des modifications des sous-processus correspondants, selon la formule suivante :

$$\text{Taille}_{Cfut}(\text{Modification(couche}_i)) = \Sigma_i \text{ taille (sous-processus}_i \text{ ajouté}) + \Sigma_i \text{ taille (sous-processus}_i \text{ modifié}) + \Sigma_i \text{ taille (sous-processus}_i \text{ supprimé})$$

Pour chacune des couches identifiées, la fonction de regroupement est complètement adaptable, ainsi un sous-total peut être généré pour un processus fonctionnel individuel ou pour l'ensemble de la couche, selon le but et l'envergure de chaque exercice de mesure.

En outre, le regroupement des résultats de mesure par type de sous-processus peut être utile pour analyser la contribution de chaque type sur la taille totale d'une couche et peut ainsi aider à caractériser la nature fonctionnelle de la couche mesurée (voir analyse expérimentale au chapitre 4).

[28] Un exemple d'application de la méthode de mesure COSMIC-FFP sur un petit projet est donné à l'annexe 4.

Dans le contexte où la taille fonctionnelle est utilisée comme variable dans un modèle, pour estimer l'effort par exemple, et que le logiciel mesuré possède plus d'une couche, le regroupement sera fait typiquement au niveau de la couche, puisque les couches ne sont généralement pas implantées à l'aide des mêmes technologies.

La méthode de mesure fonctionnelle COSMIC-FFP qui vient d'être présentée ici a été utilisée dans l'évaluation de 26 projets d'entreprise. Sur base des résultats obtenus, outre le fait que l'on a pu évaluer la documentation fonctionnelle, des remarques concernant la méthode de mesure COSMIC-FFP ont pu être formulées. Ces remarques et observations sont présentées au chapitre 4.

4 Analyse de la documentation fonctionnelle à partir de l'expérimentation de la méthode de mesure COSMIC-FFP

Ce projet analyse la méthode de mesure COSMIC-FFP[29] sous l'angle de son application et sa validation principalement à travers une étude expérimentale. Cette expérimentation a été réalisée à partir de la mesure de 26 projets de maintenance des systèmes informatiques d'une grande entreprise[30].

Pour chacun de ces projets, il a été question d'évaluer la qualité de la documentation, de calculer le nombre de points de fonction COSMIC-FFP et de comparer les résultats obtenus avec la méthode de mesure FPA, pour les mêmes projets et dans les mêmes conditions.

4.1 Motivation et conditions de réalisation

Le travail d'analyse réalisé avait pour but l'initiation à l'application de la méthode de mesure des points de fonction COSMIC-FFP. Cette méthode tire son originalité de son applicabilité à plusieurs types de logiciel (logiciel de gestion, logiciel temps réel, etc.). Il s'agit d'une méthode de mesure toute récente qui est toujours en cours d'adaptation afin de faciliter le processus de mesure à travers la conception d'outils d'assistance durant la phase de mesure.

La concrétisation de cette étude a été facilitée par :

- la disponibilité de la documentation nécessaire à la mesure.
- la disponibilité des données sur une méthode de mesure autre que COSMIC-FFP[31].
- la supervision d'un expert et co-auteur de la méthode de mesure COSMIC-FFP.

Avant d'entamer l'analyse proprement dite des résultats de la mesure, il convient de soulever plus en détail la problématique relative à la documentation fonctionnelle des logiciels.

---

[29] Voir Chapitre 3.

[30] Voir Annexe 2 pour le contexte et l'environnement de travail.

[31] Seuls les résultats de la mesure FPA (Function Point Analysis d'IFPUG) étaient disponibles. Pour des raisons administratives, l'accès aux locaux de l'entreprise où l'on pouvait trouver la référence exacte de la documentation utilisée par les mesureurs FPA, n'a pas été possible.

4.2 Problématique de la documentation fonctionnelle

L'identification des processus (ou composants élémentaires) d'un système informatisé repose sur la partie généralement non automatisable d'une application, c'est-à-dire sa documentation fonctionnelle [Desharnais, 2001].
Ceci entraîne un certain nombre de problèmes que nous regroupons de la façon suivante:

- la qualité de la documentation fonctionnelle,
- les coûts de la mesure,
- la cohérence des résultats.

4.2.1 Qualité de la documentation fonctionnelle

L'utilisation de la documentation fonctionnelle pour identifier les processus pose les problèmes suivants [Desharnais, 2001]:

- la qualité de la documentation n'est pas toujours adéquate,
- sa disponibilité est souvent manquante,
- la compréhension de la documentation par une personne extérieure au projet informatique peut parfois être difficile indépendamment de l'expérience du mesureur.

Lorsque le mesureur est expérimenté et que les développeurs sont disponibles pour répondre aux questions en suspens, il est possible de combler en partie ces lacunes. De plus, les mesureurs qui ont une certaine expertise vont, le plus souvent, se documenter à propos de la qualité de l'information reçue si les développeurs ne sont pas disponibles pour répondre aux questions.[32]

4.2.2 Coûts de la mesure

Un autre problème concerne les efforts à déployer pour réaliser les calculs en points de fonction. Actuellement, le processus de collecte des données est entièrement manuel et demande des efforts que les entreprises jugent onéreux, ainsi que des possibilités d'introduction d'erreur par un processus entièrement manuel.

Il existe d'ailleurs un certain nombre de projets de recherche qui visent l'élaboration d'un outil qui permettrait d'automatiser en tout ou en partie l'identification des processus fonctionnels[33]. Ces projets sont encore récents et n'ont pas encore donné des résultats assez satisfaisants.

---

[32] Desharnais, J-M « Application de la mesure fonctionnelle COSMIC-FFP : Une approche cognitive », présentation du projet de recherche, DIC 9410, version 1.05, Avril 2002.
[33] Ibidem.

Les projets les plus prometteurs se basent d'une part sur la précision et la définition d'une spécification standard comme UML (*Unified Modeling Language*) par exemple, qui serait utilisée par tous les concepteurs et développeurs de projets informatiques. Et cela irait avec la mise sur pied d'un modèle général de traduction et d'interprétation de la spécification [Bevo et al, 1999]. D'autre part, l'automatisation du processus de comptage des points de fonction peut être faite à travers l'utilisation de référentiels à l'intérieur des ateliers intégrés de génie logiciel [Mazzicco, 1990].

Il faut signaler enfin que ces projets n'en sont qu'à leurs débuts, leurs aboutissements constitueraient un énorme progrès dans ce sens qu'il simplifierait considérablement l'application des méthodes de mesure, en général, et de COSMIC-FFP, en particulier. Cela pourrait également contribuer à réduire, de manière significative, la subjectivité due à l'interprétation des règles de mesure bien souvent associée à l'application d'une méthode de mesure [Jones, 1997].

4.2.3 Cohérence des résultats

Le problème de la cohérence des résultats entre mesureurs a fait l'objet de plusieurs études[34] citées dans les travaux d'Abran [1994].

« 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 subjective mais plutôt des causes suivantes :

- une méconnaissance des règles par les analystes;
- des organisations qui ne fonctionnent pas à partir des mêmes règles officielles…
- un manque de rigueur dans l'application du processus de mesure;
- un manque de rigueur dans le mécanisme d'enregistrement des données de mesure. » [Abran, 1994, p124]

Il est à remarquer enfin que l'absence de recherches expérimentales sur la fiabilité des mesures prises à chacune des étapes intermédiaires laisse planer une incertitude sur la fiabilité de tout le processus de comptage. En effet pour que le résultat final soit correct il faut que les résultats intermédiaires soient également précis et exacts.

[34] Ces recherches ont porté pour la plupart sur IFPUG 90 mais les résultats sont valables pour toutes les méthodes de mesure en général.

4.3 Méthodologie de travail

4.3.1 Application de la méthode de mesure COSMIC-FFP

La méthode COSMIC-FFP a été appliquée sur 26 projets[35]. Il s'agissait pour la plupart de projets de maintenance d'une grande société canadienne coopérant avec le Laboratoire de Recherche en Gestion du Logiciel *(LRGL)*[36]. Les trois quarts des spécifications relatives à ces projets concernaient des mises à jour ou des suppressions de fonctions.

L'identification et la collecte des données nécessaires à la mesure ont été grandement facilitées par le fait que ces projets avaient déjà fait l'objet d'une mesure FPA (*Fonction Point Analysis*)[37] par des membres du laboratoire. Cela a permis de comparer les résultats des deux méthodes de mesure fonctionnelle à la fin de l'application de COSMIC-FFP.

Chaque projet mesuré a fait l'objet d'une documentation donnant tous les détails sur la mesure effectuée (hypothèses énoncées, interprétation des spécifications et identification des sous-processus, etc.)[38].

Sur base de la mesure des projets et des entrevues avec deux experts de la méthode de mesure COSMIC-FFP ayant déjà mesuré les projets avec une autre méthode, quelques règles et hypothèses locales relatives à l'application de la méthode de mesure COSMIC-FFP, ont été obtenues (voir section 4.4.3).

4.3.2 Analyse de la documentation fonctionnelle relative aux projets mesurés

Les mesures réalisées portaient sur des projets de maintenance. Ces projets ont été documentés sur base de l'information fournie par les concepteurs et développeurs de ceux-ci. La mesure, quant à elle, s'est faite essentiellement sur base des spécifications des processus.

L'identification des processus fonctionnels lors de la mesure des projets a été grandement facilitée par le fait que ces projets avaient déjà fait l'objet d'une mesure FPA (*Function Point Analysis*).

Pour chaque projet, les documents les plus pertinents et les plus utiles pour la mesure étaient donc déjà connus. L'analyse de cette documentation afin d'appliquer la méthode de mesure COSMIC-FFP s'est faite en plusieurs étapes[39] :

---

[35] Les résultats complets de la mesure sont présentés à l'annexe 3.
[36] La société en question n'a pas donné l'autorisation de publier les détails concernant ces projets.
[37] Les principaux concepts de FPA sont présentés à l'Annexe 1.
[38] Cette partie du processus de mesure ne sera pas non plus présentée car l'autorisation de la publier n'a pas été donnée, toutefois les résultats obtenus seront détaillés.

1ère étape : Identification de la disponibilité de la documentation relative à chaque projet en rapport avec les fonctionnalités déjà identifiées avec la méthode de mesure FPA.

2ème étape : Identification de chaque processus fonctionnel COSMIC-FFP à partir de l'information disponible.

3ème étape : Etablissement d'hypothèses pour la mesure des processus et sous processus afin d'assurer la cohérence de la mesure[40].
4ème étape : Utilisation d'une mesure de qualité à chaque processus fonctionnel. La qualité de la documentation était jugée arbitrairement selon 4 critères par le mesureur suivant la disponibilité de la documentation pour chaque processus.

Ces quatre critères sont :

- Le processus est documenté complètement (Fully documented),
- Le processus est seulement mentionné mais pas documenté (Listed),
- La seule information concernant les processus est leur nombre (Quantified),
- Le mesureur identifie un processus implicite (Implied).

5ème étape : Application de la méthode COSMIC-FFP par l'Attribution de valeurs numériques aux différents sous processus identifiés, selon un certain nombre d'hypothèses (voir section 4.4.3).

4.4 Analyse des résultats

Les résultats obtenus avec COSMIC-FFP sur les 26 projets ont été résumés dans des tableaux récapitulatifs afin de faciliter leur analyse.

Le total en points de fonction COSMIC-FFP par projet de maintenance varie de 23 à 248 avec une moyenne de 85 points de fonction FFP (voir figure 4-1).

Dans l'ensemble, les points variaient suivant la dimension du projet de maintenance. Il a été assumé que le nombre de groupes de données pour la plupart de ces projets était de 2 sauf dans les cas où le nombre de groupes de données était explicite dans la documentation.

---

[39] Ces étapes ont été suggérées par J-M Desharnais, expert et co-auteur de la méthode de mesure COSMIC-FFP.
[40] Ces hypothèses, au nombre de 11, sont présentées à la section 4.4.3.

Comme cela a été dit plus haut, 26 projets au total ont été mesurés, cependant les résultats obtenus avec deux projets P 22 et P 24 ont été retirés car la spécification de certains de leurs processus manquait.[41]

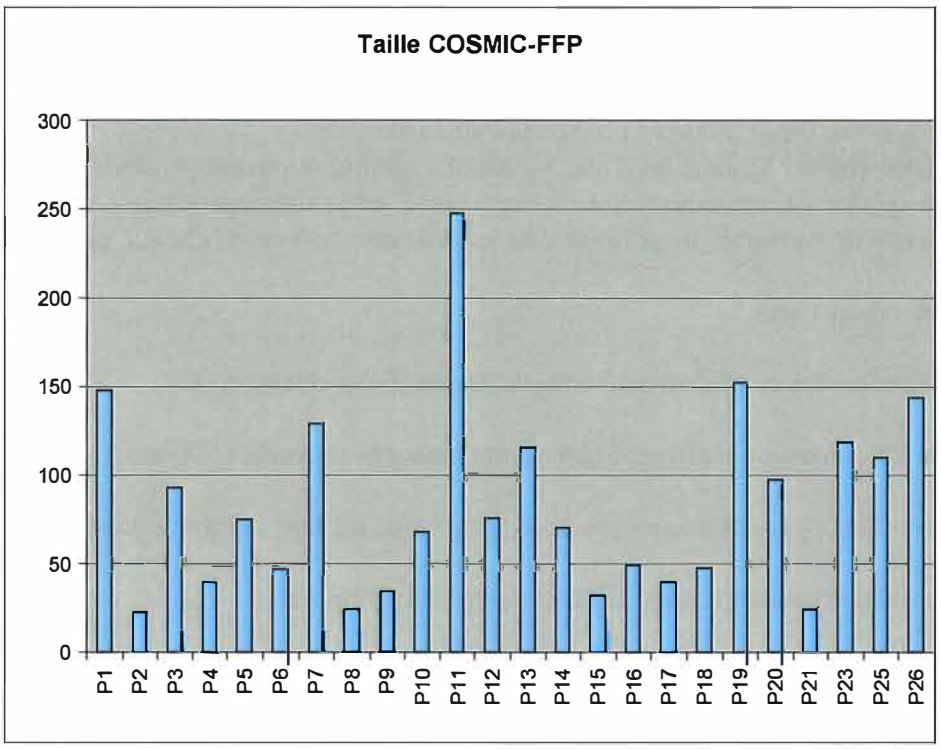


Figure 4-1: Taille COSMIC-FFP de chaque projet de maintenance

4.4.1 Analyse de la qualité de la Documentation fonctionnelle des projets mesurés

La qualité de la documentation analysée dans cette partie est celle nécessaire à l'identification des processus. Pour la reconnaissance des sous processus, la qualité n'était pas suffisante, plusieurs hypothèses ont été faites. [42]

[41] On ne pouvait donc pas considérer les résultats obtenus avec ces projets car cela risquait de fausser l'analyse faite sur les résultats. D'où ces projets ne figurent pas sur le graphique.

[42] Voir section 4.4.3.

La qualité de la documentation a été jugée selon les critères suivants :

NA (Non Applicable) :

Ce critère veut dire que lors de la mesure, il n'a pas été possible de qualifier ce processus compte tenu de la documentation disponible, ou encore il y a eu un «oubli involontaire » de le qualifier.

Documentés :

Ces processus « documentés » sont identifiés clairement et définis en quelques lignes. La documentation ne répond pas nécessairement aux critères de définition des processus d'une méthodologie (ex : le langage de spécification UML), mais est suffisante pour que le mesureur puisse identifier clairement chaque processus fonctionnel.

Listés :

Les processus « listés » sont identifiés par un mot ou une courte phrase sans plus, la documentation n'explique pas en quoi consiste ces processus ou à quel besoin ils répondent.

Quantifiés:

Les processus « quantifiés » sont identifiés par un nombre, c'est-à-dire qu'à part la quantité (ou le nombre) de processus fonctionnels, aucune autre précision n'est donnée sur chacun des processus cités dans la documentation.

Implicites:

Il s'agit de processus non formellement identifiés, mais il y avait suffisamment d'information dans la documentation fonctionnelle pour conclure à l'existence d'un processus.

Si on prend un projet qui traite et sauvegarde des informations à propos d'un ensemble de types d'entité (ex : employé, client), la présence de chaque type d'entité peut automatiquement impliquer l'existence d'un ensemble de fonctions au minimum, par exemple :

- créer un enregistrement dans l'entité Employé.
- modifier un enregistrement dans l'entité Employé
- supprimer un enregistrement dans l'entité Employé

Ceci peut implicitement donner lieu aux transactions suivantes :

- rechercher, afficher un enregistrement de l'entité Employé
- lister les enregistrements de l'entité Employé.

Le tableau ci-dessous donne un récapitulatif de la qualité générale de la documentation

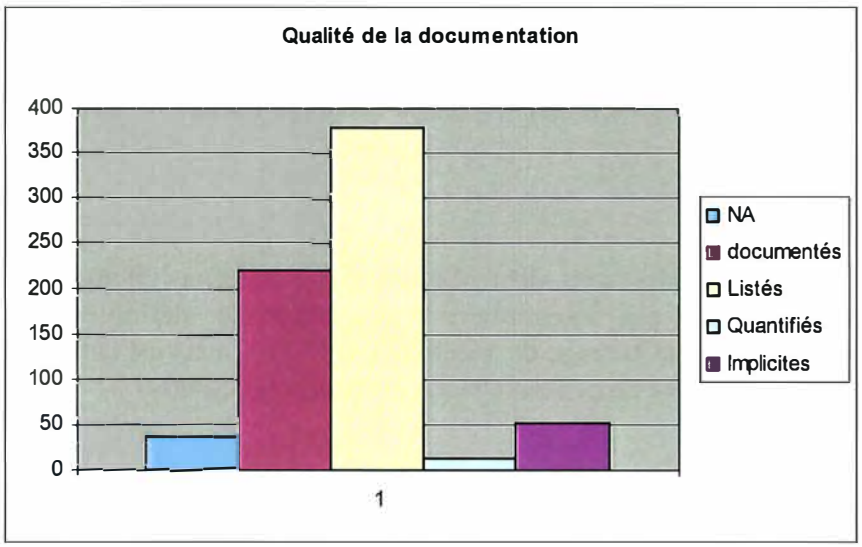


Figure 4-2: Analyse de la qualité générale de la documentation fonctionnelle

On remarque donc que plus de la moitié des processus identifiés étaient tout simplement listés. Seulement 30% des processus identifiés étaient bien documentés. Il y a lieu de souligner également que 7% des processus comptés étaient implicites et qu'ils ne se retrouvaient de ce fait nulle part dans la documentation liée aux projets mesurés, (c'est souvent le cas quand le mesureur n'a pas accès au développeur).

Les processus non définis (NA) qui représentent moins de 5% de l'ensemble n'ont pas été pris en compte lors du comptage des points FFP. Cela peut être expliqué par une quantité d'information et un temps insuffisants pour faire ce travail. Ce n'était pas non plus une priorité par rapport à la mesure elle-même. Les processus quantifiés (dont on connaissait le nombre exact), quant à eux, représentaient 2% du total des processus.

Il est intéressant de considérer la qualité relative de la documentation en isolant les projets les plus représentatifs du point de vue de la qualité de la documentation fonctionnelle. Pour ce faire, les projets ont été divisés en 2 groupes :

- Le premier groupe contenait les 7 premiers projets qui ont été jugés comme étant les mieux documentés par les experts de la mesure FPA.

- Le deuxième groupe contenait les autres projets jugés moins bien documentés.

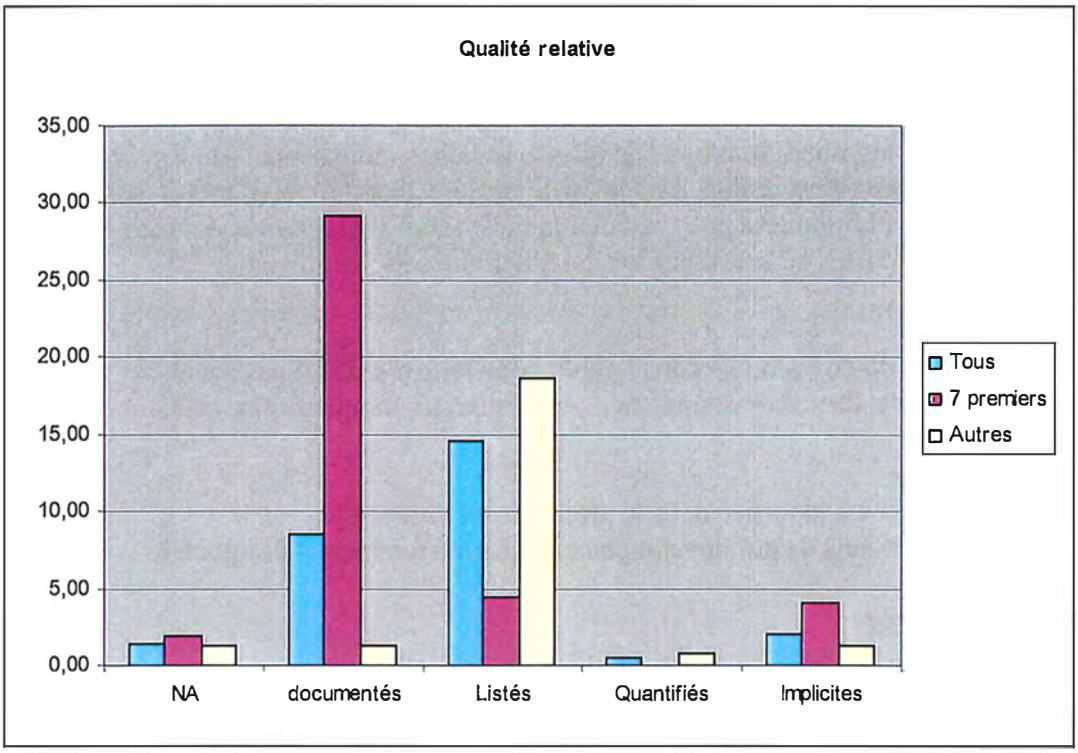


Figure 4-3: Analyse de la qualité relative de la documentation fonctionnelle

On remarque que dans les 7 premiers projets sélectionnés, comme étant les mieux documentés, il y a nettement moins de processus listés par rapport à la distribution générale. On remarque également que c'est dans les projets restants, à savoir 17, que se retrouvent le plus de processus listés, ce qui influence de beaucoup la distribution totale.

La distribution de processus non disponibles et implicites est quasi uniforme pour tous les projets.

On peut relever également que le nombre de processus implicites est plus élevé pour les 7 premiers projets les mieux documentés. Cela est expliqué par la quantité suffisante et complète de l'information qui permet de faire des extrapolations, c'est-à-dire que la documentation fonctionnelle disponible permet de conclure à l'existence d'un processus, même si ce n'est pas dit explicitement.

4.4.2 Analyse des types d'activités

Les activités d'un projet de maintenance se résument en trois types : les ajouts de fonctionnalités, les modifications de fonctionnalités existantes et les retraits de fonctionnalités existantes. Selon les règles de mesure fonctionnelle, toutes ces activités sont mesurées et additionnées pour calculer la taille totale des activités de maintenance.
Les composants utilisés pour évaluer le type d'activité sont les suivants :

Activité ***Ajoutée***
La fonction a dû être ajoutée dans l'application, elle n'existait pas avant.
Dans les projets de pur développement, les fonctions sont toutes ajoutées.

Activité ***Modifiée***
La fonction est modifiée dans le projet en accroissement.
Dans les projets de pur développement, aucune fonction n'est ajoutée.

Activité ***Supprimée***
La fonction est supprimée dans l'accroissement du projet.
Dans les projets de développement uniquement, aucune fonction n'est supprimée.

Les projets qui ont été mesurés étaient des projets de suivi et de mise à jour d'applications, ce qui explique qu'il y a eu beaucoup d'ajouts et de modifications de fonctions. Les processus liés à ces projets représentaient donc une évolution fonctionnelle du système (voir figure 4-4). Les activités étudiées ici sont vues selon la perspective du développeur.

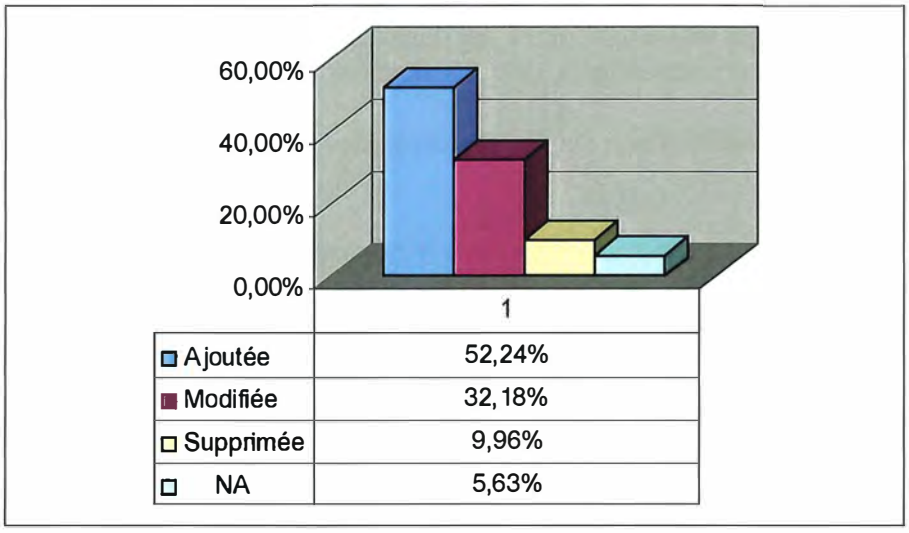


Figure 4-4 : Analyse des types d'activités des processus.

Selon ce graphique, on peut relever le fait que plus de la moitié des activités correspondaient à des ajouts de fonction par le développeur, 30% des activités du développeur correspondaient à des modifications de processus, et à peu près 10% des processus (fonction) ont été supprimés.

Sept projets ont été isolés afin de mesurer les activités relatives du système, ils représentaient à eux seuls plus de 75% des ajouts (voir figure4-5 ci-dessous).

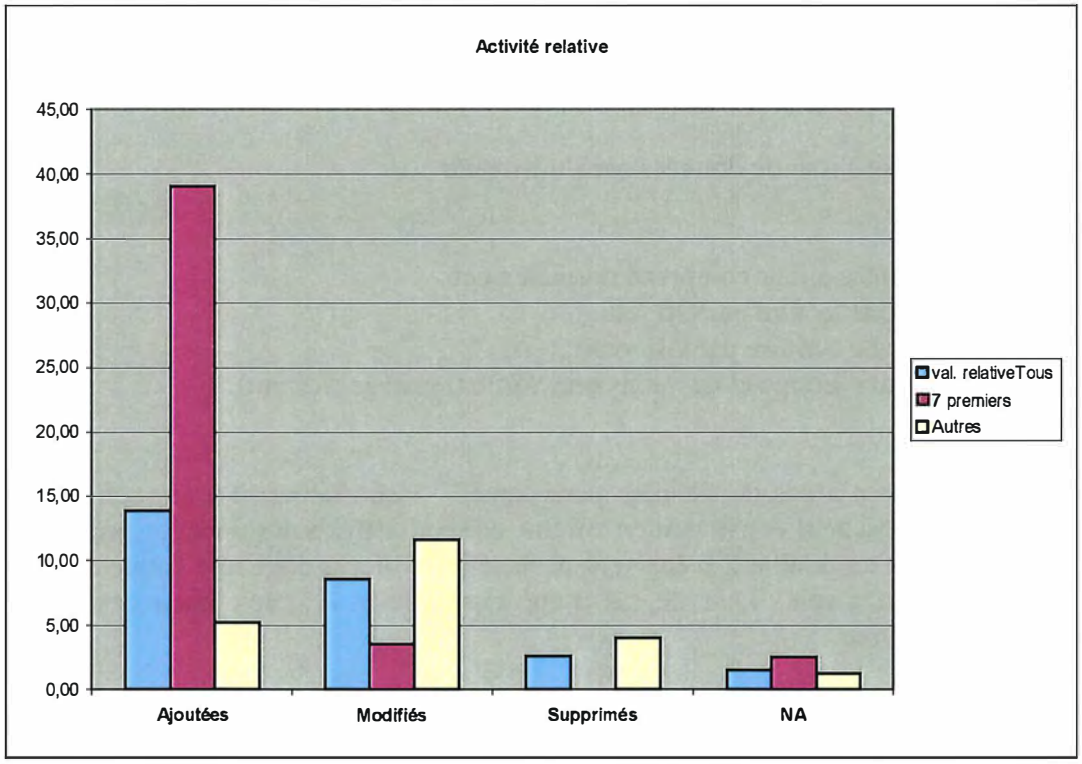


Figure 4-5 : Analyse des activités relatives à chaque projet

On remarque que pour ces projets, aucune suppression de fonction n'a été faite. On relève directement, que pour les 17 autres projets restants, le nombre d'activité de modification, est plus dominant par rapport au nombre d'ajouts.

4.4.3 Analyse des types de processus

Comme cela a été évoqué plus haut, l'information disponible pour mesurer les 26 projets n'était pas toujours suffisante. De plus, il n'y avait pas d'accès possible au développeur. Il a donc fallu formuler des hypothèses pour justifier le processus de comptage d'une part, et d'autre part pour s'assurer que le comptage était effectué de la même façon pour

tous les projets. Les hypothèses formulées sont donc plus spécifiques aux projets mesurés. Elles peuvent toutefois s'appliquer à d'autres types de projets si ceux-ci contiennent le même genre de processus.

Ces hypothèses locales ont été énoncées avec l'aide d'un expert de la mesure fonctionnelle COSMIC-FFP et FPA et vérifiées par d'autres experts.

Dans les hypothèses formulées, il a été supposé qu'il y avait un seul groupe de données en entrée ou en sortie si aucune autre information n'était disponible.

Les hypothèses utilisées lors de l'évaluation des projets sont les suivantes :

Hypothèse 1 : Mise à jour de données

Il s'agit d'une mise à jour de données par l'utilisateur

Énoncé :
Un processus de mise à jour comprend normalement:
- au moins une lecture par GD[43] lu.
- au moins une écriture par GD écrit.
- au moins une Entrée et au moins une Sortie (message d'erreur).

Justification :
Lorsqu'une entrée n'a pas de données persistantes[44] identifiables, il a été convenu que l'événement déclencheur constituait en soi une entrée. S'il y a des données persistantes, on ne calcule pas en double l'événement déclencheur. Il y a donc une entrée dans tout processus quel qu'il soit. Dans le cas d'une mise à jour, il doit y avoir des données persistantes à l'entrée.

En effet, une mise à jour implique qu'il y a une écriture dans un groupe de données à partir de données (attributs) provenant de l'extérieur de la frontière du logiciel.

Lors de la mise à jour de données, on peut supposer qu'il y a une lecture (d'une ou) des bases de données afin de vérifier la présence ou non de la donnée.

On peut aussi supposer une sortie. En effet, il a été convenu de calculer comme une sortie un message d'erreur qui aviserait par exemple l'utilisateur d'un mauvais déroulement du processus. Normalement, lorsque l'utilisateur peut faire une mise à jour, une fonction l'avertit des erreurs possibles.

[43] GD=Groupe de données.
[44] Une donnée persistante est celle qui a une durée de vie plus longue que la durée d'une transaction.

Hypothèse 2 : Modification de fonction (développeur)

Dans les projets mesurés il y a eu beaucoup de spécification concernant des modifications de fonction que l'utilisateur ne voyait pas bien entendu. Ces modifications ne sont en principe pas prises en compte étant donné que la méthode de mesure fonctionnelle COSMIC-FFP n'évalue que les fonctionnalités livrées à l'utilisateur.

Énoncé :
Un processus de modification d'une fonction par le développeur n'engendre aucun processus si on le regarde du point de vue de l'utilisateur.

Le développeur en tant qu'utilisateur :
Dans certains cas, la modification de fonction s'accompagne d'une demande de production de données en sortie (Output) ou d'une demande de suppression de certaines données (Remove). La demande de production de données par le développeur est une fonctionnalité qui peut être calculée à la condition que le développeur soit l'utilisateur. Le développeur peut être un utilisateur, mais la vue du logiciel n'est pas la même que pour un utilisateur final. Il s'agit d'une couche[45] différente. (Voir définition de couche dans COSMIC-FFP; section 3.1).

Justification :
La modification des programmes (écriture de nouveaux programmes) est une fonction de développement et donc technique pour l'utilisateur du logiciel.

Hypothèse 3 : Visualisation de données

Il s'agit des processus qui engendrent un affichage de données à l'écran.

Énoncé :
Dans une demande de visualisation de données, on a :

- au moins une entrée pour la demande de visualisation
- au moins une lecture par groupe de données lues
- au moins une sortie pour la visualisation et probablement une autre sortie pour les messages d'erreur.

Justification :
La justification de l'événement déclencheur donnée pour la première hypothèse est valable ici, sauf que le plus souvent c'est l'événement déclencheur qui est calculé comme entrée, ce qui est différent de ce qui se passe dans une mise à jour.

[45] Une couche est le résultat du partitionnement fonctionnel de l'environnement du logiciel où tous les processus fonctionnels inclus montrent un haut degré de cohésion et s'exécutent au même niveau d'abstraction (COSMIC-FFP, version 2.1, page 7 , voir également le chapitre 3).

De plus, il n'y a pas d'écriture car aucune donnée n'est modifiée, enlevée ou créée. Il y a au moins une lecture, car le logiciel doit accéder à au moins un GD pour permettre que les données de visualisation soient disponibles à l'utilisateur.

Il y a au moins une sortie pour la visualisation des données. On peut aussi supposer une sortie pour un message d'erreur.

Hypothèse 4 : Chargement des données (d'un ancien à un nouveau logiciel)

Il s'agit d'un processus qui extrait des données d'une base de données pour les recharger dans un autre endroit.

Énoncé :
Lors du chargement de fichier il y a :
- au moins une entrée pour la demande de chargement
- au moins une lecture par GD lu, pour acquérir les données à charger
- au moins une écriture pour le GD qui conserve « temporairement » les données
- au moins une sortie pour la production de fichiers.

Justification :
Dans un projet d'amélioration il y a aussi une partie développement, ce qui veut dire que tout comme dans un projet de développement, il est possible que le développeur ait à construire des fonctions « temporaires » de transfert de données, de la base de données de l'ancien logiciel au nouveau logiciel. Ce sont aussi des fonctionnalités, mais sous l'aspect transfert des données.

Ce processus de transfert comprend des entrées (les anciennes données) et la lecture des données à transférer. Il y a aussi l'écriture des données dans le fichier « temporaire » qui créera les fichiers qui seront lus par l'autre application. Le processus s'arrête lorsque les fichiers sont créés. De plus, même si le fichier est « temporaire », les données ont une durée de vie de plus d'une transaction. Les données doivent être conservées tant que le nouveau logiciel n'a pas toutes les données qui sont nécessaires à son fonctionnement et qui proviennent de l'ancien logiciel.

Hypothèse 5 : Traitement de données

Il s'agit d'une demande de traitement d'un composant (ex : remplissage d'un questionnaire avec mise à jour de données).

Énoncé :
Lors du traitement de données, on suppose qu'il y a :
- au moins une lecture pour chaque groupe de données traité
- au moins une écriture de données par groupe de données écrit lors du traitement
- au moins une entrée et au moins une sortie (messages d'erreur).

Justification :
Le traitement de données implique un ensemble d'opérations sur des données, que ce soit des lectures ou des écritures. L'entrée vient au moins de l'événement déclencheur et on peut supposer qu'un traitement peut engendrer des messages d'erreur.

Hypothèse 6 : production de fichier

Il arrive souvent qu'un processus engendre la production d'un fichier suite à une demande exprimée par l'utilisateur.

Énoncé :
Lors d'une demande de production d'un fichier :
- il y a un événement qui déclenche la demande de production d'un fichier.
- il y a une lecture d'au moins un groupe de données pour permettre la recherche de données nécessaires à la production du fichier.
- il y a une sortie pour le fichier produit.
- il y a aussi une sortie pour d'éventuels messages d'erreur.

Justification :
Généralement, le développeur s'occupe de la production des fichiers, dans ce cas, il n'y a pas de fonctionnalités pour l'utilisateur. Mais quand cette production est faite suite à une demande exprimée par l'utilisateur, elle est comptabilisée.

Hypothèse 7 : Envoi de données

Il faut comprendre qu'il s'agit d'un processus ressemblant à la production de fichier. Il peut vouloir dire par exemple qu'un logiciel B doit recevoir des données du logiciel A. Du point de vue du logiciel B le processus est équivalent à une mise à jour (hypothèse 1).

Énoncé :
Un logiciel B reçoit des données du logiciel A, cela engendrera une mise à jour du côté de B. Il faut voir dans ce cas l'hypothèse 1.

Justification :
Les justifications données pour l'hypothèse 1 sont évidemment valables ici.

Hypothèse 8 : Création d'entité

Cette hypothèse illustre l'ajout de données à une entité.

Énoncé :
On considère une création comme étant un ajout d'une donnée du type d'une entité préexistante. Donc, on applique les mêmes hypothèses que pour l'écriture de données (voir l'hypothèse 1).

Justification : Les justifications données pour l'hypothèse 1 sont valables ici également.

Hypothèse 9 : Ecriture de données

L'écriture de données regroupe à la fois les processus de suppression, de création et de mise à jour de données.

Énoncé :
Une Ecriture implique donc:

- au moins une entrée
- une éventuelle lecture de GD pour localiser l'emplacement d'écriture si le GD est trié
- une écriture par GD concernées et une sortie correspondant à d'éventuels messages d'erreur.
- Un éventuel message d'erreur

Justification : les justifications données pour les hypothèses de mise à jour et de création de données sont valables ici également.

Hypothèse 10 : Production de rapport

Cette hypothèse concerne les processus effectuant une demande de production de rapport qui sera soit imprimée ou affichées.

Énoncé :
Lors de la demande de production d'un rapport :

- il y a au moins un événement déclencheur
- il y a une possibilité faible d'écriture de GD (sauf quand c'est explicite)
- il y a au moins une lecture de GD
- il y a 2 sorties : un message d'erreur et un rapport.

Justification :
La demande d'un rapport est faite par un utilisateur. En conformité avec la règle de comptage de l'événement déclencheur, il y a au moins une entrée. Il peut arriver qu'avant la production d'un rapport, il y ait une écriture. C'est le cas par exemple lorsque l'utilisateur veut garder une trace des rapports produits pour consultation ultérieure.

Pour la production d'un rapport, il y a au moins une lecture d'un groupe de données. On suppose aussi une sortie pour le rapport et une autre pour un message d'erreur.

Hypothèse 11 : Recherche de données

Énoncé :
La recherche d'information (sur le WEB ou dans une base de données) est une variante de la visualisation (voir l'hypothèse 3). Elle vise l'obtention d'une information précise et limitée.

On suppose qu'une demande de recherche engendrée par l'utilisateur comprend au moins :

- une entrée
- une lecture d'au moins un groupe de données lors de la recherche
- une sortie pour l'affichage
- une sortie pour les messages d'erreur où l'on peut avoir un message du type "il n'y a pas de données pour le mot xxx" ou une variante.

Justification **:**
La recherche de données est une variante de la visualisation (voir encore l'hypothèse 3). Ici, on compte aussi une sortie automatique pour les messages d'erreur.

Cadre d'utilisation de ces hypothèses

Ces hypothèses ont été écrites conjointement avec un expert et co-auteur[46] de la méthode de mesure COSMIC-FFP. Comme cela a été expliqué plus haut, ces hypothèses ont été formulées durant l'application de la méthode de mesure COSMIC-FFP afin de s'assurer de la cohérence du processus de mesure.

Ces hypothèses sont définies de manière générale et peuvent donc être utilisées et pour mesurer n'importe quel autre projet à l'aide de la méthode de mesure COSMIC-FFP.

Ces hypothèses n'ont pas la prétention d'être complètes, chaque mesureur peut y apporter sa touche personnelle suivant la complexité et la particularité des projets mesurés. Elles peuvent donc servir de référence pour la mesure.

Exemple d'application

Cet exemple est inspiré d'un des 26 projets mesurés et donne un aperçu de l'application des hypothèses pour calculer la taille fonctionnelle des projets. La première colonne reprend le nom du processus, la deuxième donne l'interprétation donnée au processus suivant le contexte, tandis que la troisième colonne fournit l'application de l'hypothèse au processus afin de dégager les sous-processus.

[46] J-M Desharnais membre du laboratoire de recherche en gestion du logiciel

Cela a été fait pour tous les projets et chaque projet possédait en moyenne une trentaine de processus.

Nom	Interprétations	Hypothèses	Entrée	Lecture	Ecriture	Sortie	Total
Service des remises	**Changer les valeurs des remises ...**: il y a mise à jour des valeurs de certaines données	**H1**	1	1	1	1	4
Traitement mensuel de données	Il y a traitement de données régulier	**H5**	1	1	1	1	4
Ecran: 86, 91, 99.	**Les écrans devront montrer uniquement...:** les données à visualiser vont changer.	**H3 :** il y a une mise à jour du côté du développeur de la fonction	1	1		2	4

Tableau 4-1 : Exemple d'application des hypothèses locales de mesure.

La figure ci-dessous donne les proportions obtenues pour les 26 projets pour chaque type de sous-processus COSMIC-FFP.

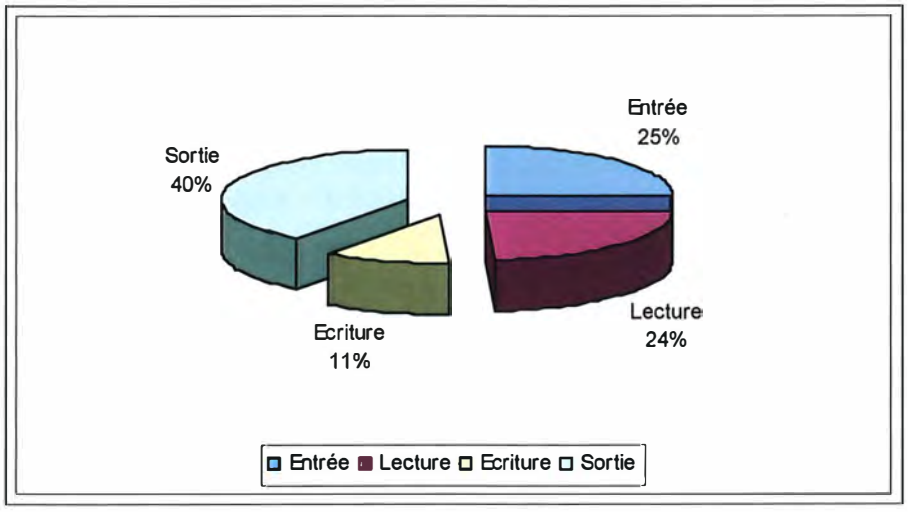


Figure 4-6: Proportion des sous- processus COSMIC-FFP pour l'ensemble des projets mesurés

4.4.4 Présentation et Comparaison des résultats obtenus avec les méthodes de mesure FPA et COSMIC-FFP

Comme on peut le constater dans le tableau 4-2, les résultats FPA de la 2ème colonne sont, au total, supérieurs en point de fonction aux résultats obtenus avec la méthode de mesure COSMIC-FFP.

Les résultats obtenus pour chaque type de mesure[47]

	Résultats FPA		Résultats COSMIC-FFP				
Nom du projet	FPA total	FPA transaction	Entrée	Lecture	Ecriture	Sortie	Total COSMIC-FFP
P1	187,00	163,2	38	37	10	63	148
P2	87,00	47,8	6	3	6	8	23
P3	204,00	144	22	26	18	27	93
P4	118,20	118,2	15	15	13	21	64
P5	124,00	113,4	20	17	14	24	75
P6	53,00	53	12	11	8	16	47
P7	244,00	148,2	32	32	14	51	129
P8	24,40	**24**	6	6	2	10	**24**
P9	41,60	42	7	7	7	13	34
P10	105,00	73,4	17	17	10	24	68
P11	393,00	294	63	59	15	111	248
P12	270,00	262,6	19	16	3	38	76
P13	231,00	168,2	29	29	0	58	116
P14	104,00	93,2	18	18	7	27	70
P15	32,00	**32**	8	8	0	16	**32**
P16	61,60	61,6	12	11	8	18	49
P17	65,00	49,6	10	10	3	17	40
P18	72,60	49,2	12	12	9	15	48
P19	176,00	170,4	38	38	11	65	152
P20	111,20	111,2	26	20	18	34	98
P22	41,00	28,2	6	6	3	9	24
P23	162,20	162,2	30	30	10	49	119
P25	165,00	141,4	29	22	24	35	110
P26	272,00	214,6	39	35	13	57	144
Total	3344,80	2765,6	614	580	345	893	2432

Tableau 4-2 : Résultats bruts obtenus avec les méthodes de mesure FPA et COSMIC-FFP pour les 26 projets évalués.

[47] Les résultats complets de la mesure sont repris à l'Annexe 3.

Mis à part le système de comptage qui n'est pas le même, cette différence peut être expliquée par le fait que la méthode de mesure COSMIC-FFP ne comptabilise pas les fichiers de données.

Si on enlève les fichiers logiques de données au total des points de fonction obtenu avec la méthode de mesure FPA, cela a pour effet de diminuer nettement la taille en points de fonction FPA de certains projets qui comptent beaucoup de groupes de données logique.

La remarque faite plus haut est observée ici également. En effet on peut constater que les valeurs pour FPA transaction (colonne 3), c'est-à-dire sans fichiers de données logique sont beaucoup plus proches des valeurs obtenues avec la méthode de mesure COSMIC-FFP et sont même parfois égales.

Identification des processus avec la méthode de mesure FPA

Lors de la mesure des projets avec FPA, on a supposé que le nombre de groupes de données ne pouvait être déterminé avec exactitude. On a également supposé que la taille des Entrées, Sorties, Interrogations, fichiers internes et fichiers externes, était dans la moyenne[48], donc qu'on n'avait que 2 groupes de données sauf pour les cas explicites.

Les résultats (en processus) obtenus sont résumés dans la figure 4-7.

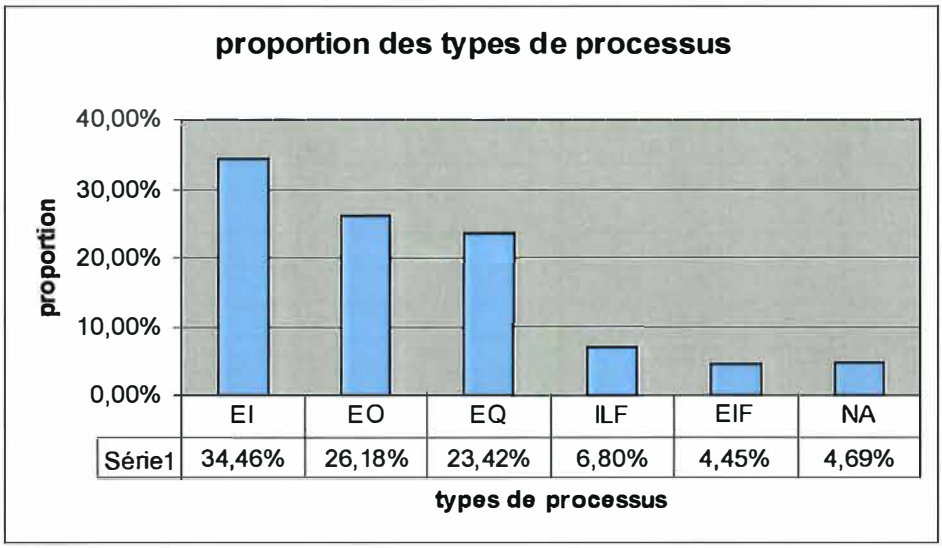


Figure 4-7 : Résumé des types d'activité de la méthode de mesure FPA.

EI représente les Entrées (Input)
EO représente les Sorties (Output)

[48] Voir au Chapitre 3, le tableau de la Pondération des composants par complexité.

EQ représente les Interrogations (Query)
ILF représente les Fichiers de données logiques Internes (Internal Logical Files)
ELF représente les Fichiers de données logiques Externes (External logical Files)
NA représente les processus non disponibles (qui n'ont pas pu être classés).

Formule de comparaison

La formule utilisée pour comparer les résultats obtenus sur les 26 projets avec la méthode de mesure fonctionnelle FPA d'une part, et la méthode de mesure fonctionnelle COSMIC-FFP d'autre part, est celle de l'Erreur Relative Moyenne[49]- ERM (Magnitude of Relative Error – MRE).

MRE est essentiellement utilisée pour évaluer les modèles de productivité (ou d'estimation à posteriori), C'est-à-dire qu'on l'utilise pour comparer les résultats obtenus à l'aide d'une méthode de mesure donnée et les valeurs réelles observées sur le terrain.

La formule est la suivante :

$$MRE = \frac{ValeurAbsolue(ValeurRéelle - ValeurCalculée)}{ValeurRéelle}$$

Dans laquelle :

- **ValeurAbsolue** représente la valeur absolue du résultat
- **ValeurRéelle** représente la taille réelle du logiciel telle que observée sur le terrain
- **ValeurCalculée** représente la taille du logiciel obtenue par application d'une méthode de mesure à celui-ci.

Le but ici étant d'évaluer l'écart entre les résultats de la méthode de mesure fonctionnelle COSMIC-FFP et la méthode de mesure fonctionnelle FPA, la « Valeur réelle » a été remplacée par le résultat obtenu avec la méthode de mesure FPA, et la « valeur calculée » par le résultat obtenu avec la méthode de mesure COSMIC-FFP.

Afin d'être plus précis, on a considéré les résultats de la méthode de mesure FPA avec et sans les fichiers logiques (fichiers internes ou externes).

[49] Conte, S.D., et Dunsmore, H.E., Shen, V.Y., 1986 « Software Engineering Metrics and models", Benjamin/Cummings Publishing.

Terminologie

Les termes utilisés dans la comparaison sont les suivants :

FPA avec Fichiers :

Représente le total des résultats obtenus par l'application de la méthode de mesure fonctionnelle FPA avant la soustraction des fichiers de données (internes ou externes) désigné encore sous le nom de groupe de données logique[50]. Il s'agit donc du résultat global que l'on obtient normalement par l'application de la méthode de mesure FPA.

FPA sans Fichiers (ou FPA transaction) :

Représente les résultats obtenus par l'application de la méthode de mesure fonctionnelle FPA auxquels les fichiers internes et externes ont été soustraits. Les résultats FPA ne sont considérés que du point de vue des transactions engendrées.

GD[51] :

Groupe de données est un 1 GD en abrégé.
Le nombre de groupe de données de chaque projet mesuré à l'aide de FFP a été augmenté afin d'estimer l'écart FFP et FPA au niveau du groupe de données.

Formules utilisées

Les formules suivantes ont été utilisées pour la comparaison.

MRE = (FPA avec fichiers – FPA sans fichiers) / FPA avec Fichiers

MRE 1GD = (FPA *avec* fichiers – COSMIC-FFP 1 GD) / FPA *avec* fichiers

MRE 1GD = (FPA *sans* fichiers – COSMIC-FFP 1 GD) / FPA *sans* fichiers

MRE 2GD = (FPA *avec* fichiers – COSMIC-FFP 2 GD) / FPA *avec* fichiers

MRE 2GD = (FPA *sans* fichiers – COSMIC-FFP 2 GD) / FPA *sans* fichiers

MRE 3GD = (FPA *avec* fichiers – COSMIC-FFP 3 GD) / FPA *avec* fichiers

MRE 3GD = (FPA *sans* fichiers – COSMIC-FFP 3 GD) / FPA *sans* fichiers

Il s'agit d'une adaptation de la mesure de l'Erreur Relative Moyenne ou MRE.

Généralement, les résultats obtenus avec la méthode de mesure FPA sont supérieurs à ceux obtenus avec la méthode de mesure COSMIC-FFP d'ou l'omission du calcul de la

[50] Voir Annexe 1sur la définition des termes et règles de la méthode de mesure FPA.
[51] GD veut dire groupe de données.

valeur absolue. On aurait pu calculer cette valeur absolue pour le MRE à 3 groupes de données, mais cela aurait pu laisser croire que les résultats FPA étaient devenus supérieurs (voir le tableau 4-3).

Les résultats obtenus

Les résultats[52] obtenus par l'application des formules sont présentés dans le tableau suivant :

	FFP - FPA total (avec fichiers)			FFP- FPA sans fichiers		
	Min	Max	Moyenne	Min	Max	Moyenne
MRE	0.00%	45.00%	17.00%			
MRE 1GD	0.00%	74.00%	39.00%	0.00%	71.00%	27.00%
MRE 2GD	2.00%	65.00%	18.00%	2.00%	64.00%	1.00%
MRE 3GD	-66.67%	62.22%	0.30%	-82.93%	61.16%	-21.00%

Tableau 4-3 : Comparaison des résultats COSMIC-FFP et FPA avec et sans les fichiers internes/externes

Les colonnes du tableau sont subdivisées en deux parties, la première partie regroupe les valeurs obtenues en calculant le MRE avec les résultats obtenus avec l'application de la méthode de mesure fonctionnelle FPA, auxquels on n'a pas soustrait les fichiers de données. La seconde partie ne tient compte que les résultats obtenus avec les processus FPA qui sont des transactions[53] (Interrogation, lecture, Ecriture).

Les lignes reprennent les MRE (Erreur relative Moyenne) obtenues au fur et à mesure que variait le nombre de groupe de données (GD). Pour chaque MRE on a repris la plus petite valeur, la plus grande valeur, ainsi que la moyenne des valeurs obtenues pour tous les projets mesurés, afin de donner une idée de la répartition des valeurs.

Les études réalisées par Morris et Desharnais [1998] ont démontré que pour des systèmes de type MIS (gestion), l'écart relatif moyen était entre 0% et 2%, pouvant aller même jusqu'à 3%. Ces résultats sont donc confirmés ici, avec la moyenne du MRE à 2 groupes de données pour le FPA sans fichiers qui est de 1%.

[52] Les resultants détaillés de l'application des formules de calcul du MRE sont donnés à l'Annexe 3.
[53] Voir Annexe 1.

Si on considère la méthode de mesure FPA avec fichiers, on se rend compte que la moyenne atteint sa valeur la plus proche (0.30%) à 3 groupes de données. Cela est expliqué par le fait que lors de la mesure FPA, un tiers des points de fonction représente les fichiers et les deux tiers restants les transactions. D'où le fait de considérer les résultats de la mesure FPA avec les fichiers nous ajoute un groupe de données. Il faut rappeler que les fichiers ne sont pas additionnés aux transactions avec la méthode COSMIC-FFP.

Il y a eu des commentaires à l'effet que l'addition des fichiers aux transactions est une duplication dans le comptage des points de fonction[54].

4.5 CONCLUSION ET REMARQUES

Cette analyse portait sur une étude expérimentale de la méthode de mesure COSMIC-FFP, dans le but de faire ressortir la problématique de la mesure relative à la documentation fonctionnelle. Cela dans le but éventuel de permettre une amélioration de la qualité et la fiabilité des résultats obtenus. Les travaux réalisés dans cette étude constituent une première approche dans l'évaluation de la qualité de l'information et de la spécification fonctionnelle des logiciels.

C'est dans ce cadre que plus d'une vingtaine de projets d'entreprise, ayant fait l'objet d'une mesure de taille fonctionnelle de type FPA par des experts, ont été mesurés à nouveau avec COSMIC-FFP sur base de la documentation fonctionnelle du logiciel.

L'analyse de la documentation fonctionnelle s'est avérée être une étape cruciale et complexe du processus de mesure. En effet, l'application d'une méthode de mesure nécessite en général de parcourir toute la documentation, même celle moins pertinente, pour s'assurer de la saisie de toute l'information nécessaire à l'identification des processus fonctionnels qui rentrent en ligne de compte lors de la mesure.

La qualité de la documentation est une des grandes faiblesses de la pratique du génie logiciel actuel, phénomène reconnu comme un des problèmes majeurs de l'industrie du logiciel [Abran, 1994]. Pour obtenir une mesure cohérente, il est donc indispensable de détailler et de préciser le raisonnement de l'expert de la mesure dans le cas où l'information est incomplète.

À cette fin, la mesure a été documentée et des hypothèses de mesure ont été formulées. Une interprétation a ainsi été donnée à chaque processus de chaque projet, suivant le niveau de détail de la documentation disponible.

---

[54] Desharnais, J-M « application de la mesure fonctionnelle COSMIC-FFP : Une approche cognitive », présentation du projet de recherche, DIC 9410, version 1.05, Avril 2002.

Une étude de la qualité de la documentation du point de vue de la mesure fonctionnelle a également été réalisée. Les résultats de cette analyse ont montré que la plupart des processus fonctionnels identifiés étaient tout simplement cités ou listés dans la documentation fonctionnelle, et que seulement un tiers des projets étaient assez documentés pour les fins de l'identification complète des processus.

En ce qui concerne l'identification des sous–processus, la qualité de la documentation est encore beaucoup plus faible. L'établissement de statistiques pour les sous-processus pourrait être réalisé dans une recherche plus poussée.

Une comparaison des tailles fonctionnelles des projets obtenues à l'aide des deux méthodes de mesure COSMIC-FFP et FPA a également été effectuée. Cela a permis d'évaluer et de valider les résultats obtenus. Les étapes et les règles étant différentes pour chaque mesure, les résultats ne pouvaient être identiques. Mais les proportions obtenues pour chaque projet, à l'aide de ces deux méthodes de mesure, confirmaient des études antérieures.

Sur base de ces constatations faites sur le terrain, il s'est avéré qu'en général, la documentation et les spécifications fonctionnelles relatives au projets informatiques des entreprises (surtout celles n'ayant pas encore atteint un bon niveau de maturité[55]), n'étaient jusqu'à présent pas assez riches en information, ce qui peut rendre difficile l'application de la méthode de mesure fonctionnelle COSMIC-FFP et ainsi nécessiter la présence d'experts ou la disponibilité du développeur.

En ce qui concerne les hypothèses formulées, celles-ci n'ont été utilisées que sur un seule ensemble de projets, elle n'ont pas pu être testées[56] sur plusieurs types de projets différents afin de vérifier leur validité dans des cas hétérogènes.

Il est à préciser enfin que la méthode de mesure fonctionnelle COSMIC-FFP a été expérimentée sur des projets d'une grande entreprise. Cependant, rien n'empêche d'examiner dans quel cadre la méthode de mesure COSMIC-FFP peut être applicable à de petites structures comme les petites et moyennes entreprises (PME).

[55] Voir Chapitre 5.
[56] La durée du stage n'était pas assez longue pour permettre une analyse plus poussée.

5 Intégration des mesures fonctionnelles au processus de développement de logiciels des PME

Avant d'examiner comment on peut intégrer les mesures fonctionnelles dans des PME (*Petites et Moyenne Entreprises)*, on va d'abord survoler quelques problèmes organisationnels qui peuvent être rencontrés dans l'utilisation des points de fonction. Ces problèmes sont les suivants [Abran, 1994] :

- La documentation des spécifications fonctionnelles ; celle-ci a été abordée au chapitre 4.

- La courbe d'apprentissage de la méthode de mesure utilisée. Cela concerne le temps requis pour apprendre et pouvoir appliquer une méthode de mesure.

- Le point de vue des utilisateurs sur lequel les mesureurs doivent se concentrer. La méthode des points de fonction en soi n'examine le produit logiciel que selon le seul point de vue de l'utilisateur.

- La maturité organisationnelle qui se concentre sur le niveau de maturité des organisations informatiques.

Le premier et le dernier point sont ceux sur lesquels se concentre cette dernière étude. En effet, il est intéressant d'analyser dans quelle mesure une entreprise à faible niveau de maturité[57] peut appliquer une méthode de mesure fonctionnelle.[58]

Ce dernier chapitre va donc traiter de l'applicabilité de toute la démarche de mesure dans le contexte des PME (*Petites et Moyenne Entreprises*), plus particulièrement ceux avec un faible niveau de maturité. Une approche pouvant permettre à long terme à une PME de disposer de mesures pouvant l'aider à améliorer ses pratiques est proposée à la fin du chapitre.

5.1 Etude de la maturité organisationnelle des PME

Plusieurs auteurs se sont penchés sur le problème relatif à la maturité organisationnelle des entreprises [Humphrey, 1988 ; MacDonell, 1991].

A la fin des années 80, un grand nombre d'entreprise selon [Humphrey, 1988] du Software Engineering Institut (SEI), se situait encore au seuil minimum dans les niveaux de maturité de génie logiciel, c'est-à-dire au niveau 1 (niveau 1 = initial) sur une échelle de niveaux (niveau 5 = optimisation). Si on se réfère aux résultats publiés par la SEI

[57] Voir section 5.1.
[58] Tout en sachant qu'un faible niveau de maturité implique un niveau de documentation assez bas.

(*Software Engineering Institute*) en mars 2002, on se rend compte que la situation n'a pas énormément évolué[59] car l'évolution d'un niveau de maturité à un autre prend souvent du temps.

Les PME à faible niveau de maturité sont ceux qui se retrouvent dans les premières ou deuxièmes catégories de maturité. En général, on se rend compte que la majorité des concepts de génie logiciel ne sont pas connus ou maîtrisés dans la plus part des PME [Habra et al, 2002].

« La majorité de ces organisations travaille encore sans méthodes structurées, sans documentation fiable et de qualité, et sans accumuler d'informations fiables. Non seulement on ne prend pas le temps d'analyser, mais il n'y a pas d'information fiable à analyser ...ces entreprises en général ne disposent pas non plus de système de mesure pour enregistrer de façon fiable le temps passé sur les divers projets et activités professionnelles ou administratives... » [Abran, 1994]

En d'autres termes, voici les trois grandes caractéristiques de ces organisations qui peuvent être retenues [Abran, 1994] :

- 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 historiques en nombre suffisant pour des analyses statistiques.

5.2 Analyse de l'applicabilité des mesures fonctionnelles aux logiciels produits par les PME

La production des logiciels est devenue de plus en plus complexe. Au fur et à mesure que les technologies évoluent, les clients sont devenus de plus en plus exigeants quant à la qualité du logiciel à recevoir.

Les entreprises ont de plus en plus recours aux mesures pour essayer de contrôler leurs processus de production. Mais l'intégration des mesures dans les pratiques d'une entreprise est loin d'être une chose facile. C'est encore plus compliqué lorsqu'il s'agit d'une petite entreprise ne possédant pas de « bonnes pratiques logicielles » (gestion de cahier des charges, de cycles de vie, de configuration, etc...). C'est ce genre de problème qu'essaye d'analyser cette section.

59 Ces résultats ont été publiés par la Software Engineering Measurement and Analysis Team qui fait partie de la SEI au nom de « Process Maturity Profile of the Software Community 2001 Year End Update » en Mars 2002.

5.2.1 Intégration des mesures fonctionnelles au processus de développement des logiciels

Il existe actuellement un bon nombre de mesures fonctionnelles et il n'est donc pas facile de savoir laquelle utiliser pour estimer telle ou telle autre caractéristique du logiciel. Certaines méthodes de mesures fonctionnelles sont plus adaptées que d'autres pour estimer la productivité du logiciel, comme il en existe d'autres qui conviennent mieux à l'évaluation du processus de production du logiciel.

Le choix de la métrique à utiliser et l'applicabilité de celle-ci à un type de logiciel donné doivent être également bien étudiés pour avoir des résultats de mesure adéquats.

Le coût relatif des mesures de logiciel peut être élevé suivant la complexité du logiciel à mesurer et la disponibilité de la documentation nécessaire à la mesure. Si l'information relative au logiciel est insuffisante à l'application d'une méthode de mesure et qu'en plus, le développeur du logiciel n'est pas disponible, la tâche du mesureur peut s'en trouver alourdie.

De plus, la recherche de l'information nécessaire à la mesure peut être très coûteuse en termes de temps et d'argent.

Pour qu'un logiciel soit facilement mesurable, il faut que les données relatives au développement de celui-ci soient disponibles et qu'elles reflètent fidèlement toute activité concernant le logiciel, c'est-à-dire une bonne spécification des fonctionnalités du logiciel (toutes les fonctions, même celles qui ont été supprimées, doivent se retrouver dans la documentation, l'information concernant le personnel de fabrication et son environnement doivent être également disponibles, etc).

Il est important également d'avoir une idée précise de ce qui doit être mesuré (qualité ou productivité du logiciel, méthodes de travail, etc.) car cela conditionne le choix du modèle de mesure à utiliser.

5.2.2 Applicabilité des mesures fonctionnelles aux logiciels produits par les PME

Les entreprises de production des logiciels peuvent être subdivisées en deux couches distinctes. La première couche regroupe les grandes firmes et organisations comme la Commission Européenne, les banques et assurances, etc..., travaillant directement avec les fabricants d'ordinateurs et les firmes de consultance.

La seconde couche regroupe les petites et moyennes entreprises (PME) et occupe le marché des logiciels dits "à faible niveau", le plus souvent dépourvu d'un département de technologie de l'information. Ces derniers présentent plusieurs faiblesses notamment au niveau des relations entre client/fournisseur et du manque de transparence des méthodologies de travail.

Elles font beaucoup appel aux services de consultance et n'assurent pas un service de qualité en ce qui concerne la fiabilité et la maintenance des produits logiciels qu'ils offrent.

Il est difficile d'imaginer qu'une seule méthode de mesure puisse être utilisable pour évaluer tous les aspects du processus de développement, notamment la conformité du produit final, le respect des délais de livraison, le coût de production, la bonne méthodologie de développement, la bonne gestion, etc.

Il pourrait être plus intéressant de combiner plusieurs méthodes de mesure afin d'estimer l'efficacité des processus utilisés par une entreprise.

5.3 Approche de méthodologie de mesure pour les PME à faible niveau de maturité: illustration avec COSMIC-FFP

Les pratiques logicielles des PME sont très peu structurées car celles-ci privilégient plus les aspects évolution et amélioration par rapport à l'évaluation. Cela a donc comme résultat la qualité relative des produits et services offerts. En d'autres termes, outre la taille réduite de ce genre d'entreprise, on y retrouve également une structure peu complexe, un nombre limité d'acteurs polyvalents et un niveau modeste de maturité de processus.

5.3.1 Approche de méthodologie de mesure en général

Il peut être utile d'utiliser des modèles différents pour dresser le profil de « toutes » les pratiques logicielles d'une entreprise. Le modèle d'évaluation et amélioration mesure OWPL (Observatoire Wallon des Pratiques Logicielles) qui s'inspire du modèle CMM (Capability Maturity Models) et de SPICE (ISO/IEC 15504), peut être utilisé pour évaluer les aspects « processus » et « ressources » vus à la section 1.1.2.

Le modèle OWPL analyse les processus intervenant dans la fabrication du produit logiciel en dressant un bilan de la gestion des ressources, du planning, de la configuration et des exigences émises par le client. Cela permet au bout du compte de mettre en évidence les points faibles et les points forts des entreprises et d'entamer une démarche d'amélioration progressive et guidée.

L'évaluation du « produit » logiciel peut être faite à l'aide des points de fonction afin de déterminer les caractéristiques propres du logiciel produit, comme les coûts de production, la qualité du logiciel, etc...

La méthode de mesure fonctionnelle COSMIC-FFP pourrait être utilisée comme outil (modèle) de mesure.

5.3.2 Applicabilité de la méthode de mesure fonctionnelle COSMIC-FFP dans les PME

Les PME ont la particularité d'avoir une petite structure avec un mécanisme de production floue qui peut être amélioré notamment à l'aide des mesures. Il faut toutefois qu'une entreprise ait une structure minimale pour pouvoir entreprendre des mesures avec des modèles comme COSMC-FFP.

Le modèle de mesure fonctionnel COSMIC-FFP se base essentiellement sur l'analyse de la documentation fonctionnelle relative au produit logiciel. Cela implique le respect d'un certain nombre de procédés, comme la spécification des fonctionnalités du logiciel[60], et un minimum de gestion du cycle de vie afin de permettre de dresser un récapitulatif du processus de production.

La présence du développeur peut grandement faciliter l'application de COSMIC-FFP dans ce sens qu'il répondrait aux questions qui pourraient être soulevées par le manque d'information. Afin d'appliquer la méthode de mesure fonctionnelle COSMIC-FFP aux projets de maintenance, les PME doivent faire un minimum d'archivage afin de reprendre tous les changements qui ont été effectués durant tout le cycle de vie du logiciel.

La méthode de mesure fonctionnelle COSMIC-FFP est donc applicable aussi bien aux grandes qu'aux petites structures. Toutefois, celle-ci ne permet pas d'aborder de manière précise les aspects clés de la méthodologie de production du logiciel, celles-ci pouvant être mieux évaluées à l'aide d'outils de mesure comme OWPL.

[60] Sans nécessairement utiliser les méthodes de spécification détaillées comme UML.

CONCLUSION GENERALE

Le travail d'analyse effectué a porté généralement sur le processus de mesure des points de fonction et s'est concentré sur une analyse de la documentation fonctionnelle sur laquelle on se base pour appliquer une méthode de mesure.

Une vingtaine de projet ont donc été mesurés avec la méthode de mesure COSMIC-FFP, ce qui a permis de juger de la qualité de l'information, surtout dans les cas où il n'y a pas possibilité d'interroger le développeur, comme cela arrive fréquemment dans les grosses entreprises.

Ce travail a entre autres permis de se familiariser avec la documentation des logiciels et des projets dans un contexte d'entreprise (caractéristiques, particularités...) et de s'imprégner des techniques d'analyse du logiciel pour les fins de maintenance, ce qui fournit parallèlement une ouverture à la réalité industrielle.

Les résultats obtenus ont confirmé ce que d'autres chercheurs avaient relevé en ce qui concerne la documentation fonctionnelle. Le processus de mesure étant essentiellement basé sur celle-ci, il pourrait être intéressant de développer des méthodologies ou des outils. Et ceci d'une part pour aider, les développeurs à mettre toute l'information nécessaire à la mesure dans la documentation fonctionnelle. Et d'autre part, pour assister le mesureur dans l'interprétation de la documentation fonctionnelle, afin de s'assurer que celui-ci prend en compte l'information de manière uniforme. En ce qui concerne la méthode de mesure fonctionnelle COSMIC-FFP, on peut s'inspirer des hypothèses formulées utilisées dans le travail d'expérimentation.

Il est donc important pour une entreprise qui veut appliquer les mesures fonctionnelles de bien spécifier et documenter les logiciels produits afin que toute l'information nécessaire soit présente au moment de la mesure et ainsi faciliter l'évaluation des logiciels.

L'applicabilité des méthodes de mesure fonctionnelle en général (et de la méthode de mesure COSMIC-FFP en particulier) aux logiciels produits par les petites et moyennes entreprises a également été appréciée. Et une proposition d'application des mesures fonctionnelle a été suggérée. Cette proposition peut être analysée sur le terrain afin d'étudier sa validité.

Il aurait été possible de pousser plus en profondeur cette étude sur la documentation fonctionnelle des logiciels à travers une analyse de la validité et de la généralité des hypothèses formulées (en les appliquant à un nombre plus élevé de projets), mais malheureusement la durée du stage n'était pas assez longue.

BIBLIOGRAPHIE

Abran A., Desharnais J.-M., Oligny S., St-Pierre D., Symons C., *Measurement Manual*, Version 2.1, April 2001.

Abran A., *Analyse du processus de mesure des points de fonction*, thèse de doctorat, Montréal, École polytechnique, 1994.

Albrecht A.J., Gaffney J.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, pages 639-648, Nov 1983.

Blake R., *Méthodes et outils de mesures relies à la productivité,* Centre Canadien de Recherches en Informatisation du Travail, 1987.

Bevo V., Lévesque G., Abran A., *Application de la méthode FFP à partir d'une spécification selon la notation UML : compte rendu des premiers essais d'application et questions* , International Workshop on Software Measurement- IWSM'99, Lac Supérieur, Canada, 1999.

Boehm B., *Software engineering economics*., Prentice-Hall, 1981.

DeMarco T., *Controlling Software Projects: Management, Measurement, and Estimation*, Yourdon Press, New York, 1982.

Desharnais J.-M., Abran A., *Applying a Functional Measurement Method: Cognitive Issues*, International Workshop on Software Measurement, Montreal, August 2001.

Desharnais J.-M., Description sommaire de Help!CPR et Help FFP, document produit à l'occasion du cours DIC 9200 pour Alain Abran, été 2000.

Desharnais J.-M.; Morris P., *Comparison between FPA and FFP: a field Experience,* in 8th International Workshop on Software Measurement, Magdeburg, Germany, Sept. 1998.

Eijogu L.O., *Software Engineering with Formal Metrics*, QED Technical Publishing Group, Wellesley, Massachusetts, 1991.

Fenton, N., *Software Metrics: a rigorous approach*, Englewood, 1991.

Filion D., *Gestion de la productivité et de la qualité appliquée aux projets informatiques,* Maîtrise, Gestion de projet, Université du Québec à Montréal.

Function Point Counting Practices Manual, International Function Point Users Group, version 4.1, 1999.

Glady R.B, *Measuring and Managing Software Maintenance*, IEEE Software, Vol.4, no. 9, Sept 1987.

Habra N., Niyitugabira E. et Renault A., *Modèle OWPL : Evaluation et Amélioration des Pratiques Logicielles dans les PME Wallonnes*, Technical Report 1/99, OWPL FUNDP, 1999.

Habra N., Renault A., Alexandre S., Lopez M., "*OWPL - Micro-Assessment* ", in Workshop on software quality. Proceedings of the International Conference Software Engineering, Orlando, Florida, USA May, 2002.

Humphrey W.S., Characterizing thee Software Process: A Maturity Framework, IEEE Software, pages 73-79, March 1988.

IEEE standards, P1045/D2.0, *Standard for Software Productivity Metrics,* IEEE Software Productivity Metrics Working Group, No.20, 1989.

ISO, Vocabulaire International des Termes Fondamentaux et Généraux de Métrologie, Deuxième édition, 1993, ISBN 92-67-01075-1.

ISO/IEC 14143-1:1997 - Information technology - Software measurement -Functional size measurement - Definition of concepts.

ISO/IEC 14143-3:1997 – Software engineering – Software measurement – Functional size measurement – Part 3: Verification of functional size measurement methods.

ISO TC JTC1/SC SC7/WG 12, Software Engineering — COSMIC-FFP Functional size measurement method, Document type: International Standard, 2001.

ISO/IEC TR 15504 (SPICE): 1998, Technologies de l'information - Évaluation des procédés du logiciel.

Jones C., *Applied Software Measurement, Assuring Productivity and Quality,* New York, NY, McGraw Hill, 1997.

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

Kitchenham B. et Jaggers C., *State of the Art Survey, Volume 2 – Software metrics*, Esprit project P2046 MERMAID, Feb 1989.

Low G.C. et Jeffery D.R., *Function Points in the Estimation and Evaluation of the Software Process*, IEEE Trans. on Soft. Eng., Vol. 16, no. 1, pages 64-71 Jan. 1990.

MacDonell S.G., *Rigor in Software Complexity Measurement Experimentation, Journal of Systems Software*, Vol. 16, no. 2, oct. 1991.

Mazzicco F.A., *Automation of Function Point Counting – An update*, in IFPUG, Spring Conference, Orlando, Floride, avril 1990.

Meredith D.C., *An Introduction to Software Metrics,* Live Satellite Broadcast, NTU Advanced Technology & Management Program sponsored by the University of Southern California, 30 April 1991.

Naur P et Randel B., *Software Engineering : A Report on a Conference sponsored by the NATO Science Committee,* NATO, 1969.

Navlaka J., *Software productivity metrics: Some candidates and their evaluation,* in Proc. of National Computer Conference, pages 69-75, 1986

Rudolph E.E., *Precision of Function Point Count*, IFPUG Spring Conference, SanDiego, CA, April 1989.

Putnam L., *A general empirical solution to the macro software sizing and estimatics program,* IEEE Transactions on Software Engineering, Vol. SE-4, no. 4, July 1978.

Reifer D.J., *Real-time Function Point Extensions*, in IFPUG Spring Conference, Baltimore, Maryland, April 1991.

SEI, Software Engineering Measurement and Analysis Team, *Process Maturity Profile of the Software Community 2001 Year End Update*, march 2002.

Symons C.R., *Software II FPA, Sizing and Estimating Mark Wiley Series* in Software Engineering Practice, 1991.

Symons C.R., *Function Point Analysis: Difficulties and Improvements*, IEEE Transactions on Software Engineering, Vol. 14, no 1, Jan. 1988.

Whisehunt C., *How to Implement function points Both Ways (Right and Wrong) and still Survive,* dans 8th QAI International Conference on measuring, Orlando, F1, I-119 – I-129, march 1990.

Zuse H.,. *A Framework of Software Measurement*. Berlin, Walter de Gruyter, 1988.

ANNEXE I: La mesure fonctionnelle IFPUG 4.1 ou FPA (Function Point Analysis)[61]

FPA (*Function Point Analysis*) est la méthode de comptage des points de fonction définie dans IFPUG 4.1. Il s'agit d'une méthode standard pour mesurer la taille du développement d'un logiciel du point de vue de l'utilisateur. FPA a été développé essentiellement pour les logiciels de gestion.

L'analyse en Point de Fonction mesure le logiciel en quantifiant la fonctionnalité que le logiciel fournit à l'utilisateur basée principalement sur la conception logique. Les objectifs de l'analyse en points de fonction sont de :

- mesurer les fonctionnalités que l'utilisateur demande et reçoit
- mesurer le développement et la maintenance du logiciel indépendamment des technologies employées pour la mise en œuvre.

Avantages de FPA

La technique des points de fonction peut être utilisée comme :

- un outil servant à déterminer la taille d'un progiciel acheté en comptant toutes les fonctions incluses dans le progiciel
- un outil servant à aider les utilisateurs à déterminer l'intérêt de l'achat d'un progiciel pour leur organisation en comptant les fonctions satisfaisant spécifiquement leurs besoins
- un outil pour mesurer la taille d'un produit logiciel servant à l'analyse de la qualité et de la productivité
- un support pour estimer les coûts et les efforts pour le développement et la maintenance des applications
- un facteur de normalisation pour la comparaison des logiciels

Plus précisément, la méthode FPA mesure les fonctionnalités livrées à l'utilisateur d'un logiciel en quantifiant les processus du logiciel (Entrée, sortie, interrogation) et les fichiers de données (interne et externe).

[61] Les règles et procédures de mesure de FPA présentées ici sont issues de « *IFPUG, Function Point Counting Practices Manual release 4.1, 1999* ».

Le processus de comptage suit les étapes suivantes :

- Déterminer le Type de Comptage
- Identifier l'Etendue du Comptage et la Frontière de l'Application
- Calculer la contribution des composants relatifs aux Transactions
- Déterminer le Facteur d'Ajustement
- Calculer la contribution des composants relatifs aux Données

Le diagramme suivant présente les différentes étapes de la méthode de mesure FPA. Comme on peut le constater, le nombre de points de fonction brut est le résultat des calculs dérivés de la contribution des composants relatifs aux données et aux transactions. Le nombre de points de fonction brut est ajusté à l'aide du facteur d'ajustement pour donner le nombre final de points de fonction net.

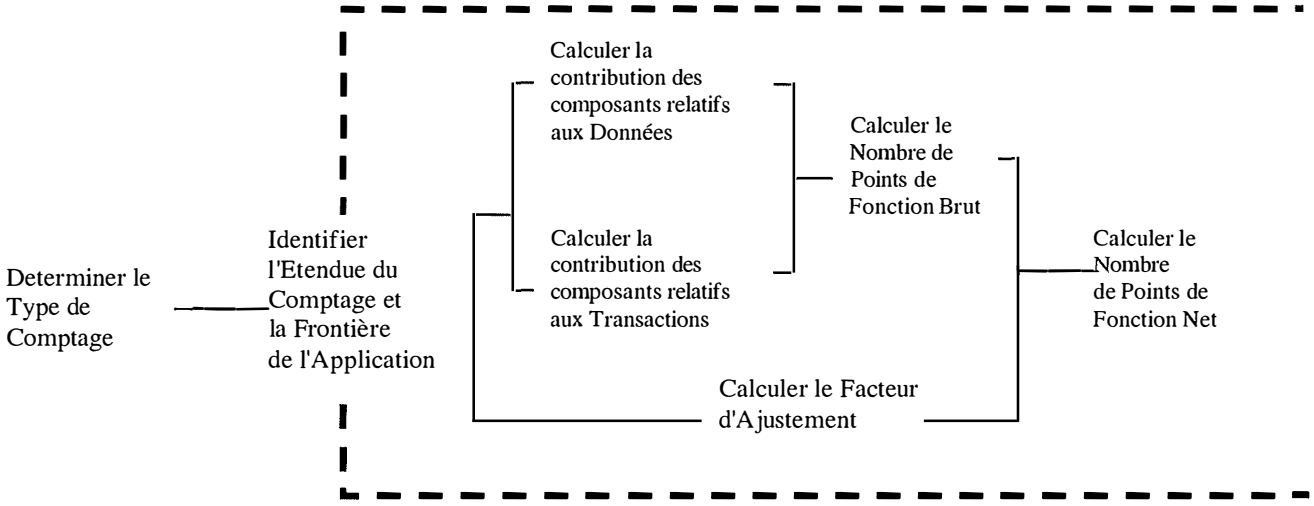


Figure 1 : Etapes de mesure de la méthode de mesure FPA

Les composants de la méthode de mesure FPA

Cette section contient les définitions des termes utilisés dans IFPUG 4.1.

La méthode FPA identifie deux catégories de types de composants : les composants relatifs aux données et les composants relatifs aux transactions, qui sont montrés dans le diagramme suivant :

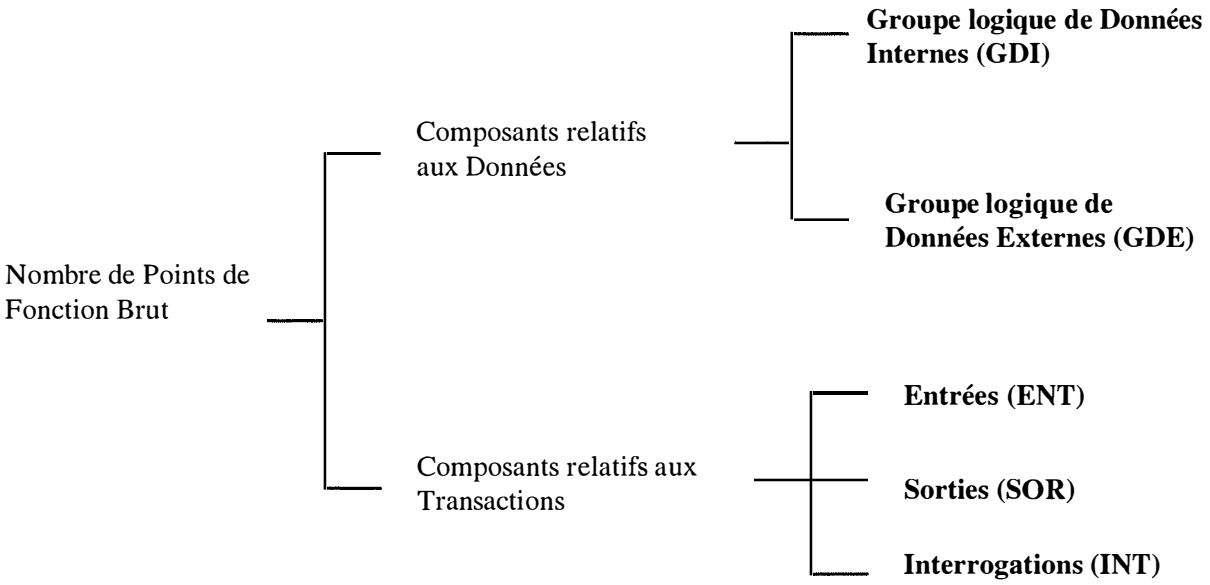


Figure 2 : Les types de composants de la méthode de mesure FPA (IFPUG 4.1)

Les groupes logiques

Les composants relatifs aux données représentent la fonctionnalité fournie à l'utilisateur pour satisfaire aux besoins en données internes et externes. Les composants relatifs aux données sont soit des Groupes logiques de Données Internes (GDI) soit des Groupes logiques de Données Externes (GDE).

Groupes Logiques de Données Internes[62]

Un Groupe logique de Donnes Internes (GDI) est un groupe de données, ou de paramètres de traitement, identifiables par l'utilisateur comme logiquement liés,

[62] Ils sont appelés également Fichiers de données interne ou ILF (Internal Logical Files).

maintenues à l'intérieur de la frontière de l'application. La première intention d'un GDI est de contenir des données maintenues par un ou plusieurs processus élémentaires de l'application mesurée.

Groupes Logiques de Données Externes[63]

Un Groupe logique de Donnes Externes (GDE) est un groupe de données, ou de paramètres de traitement référencés dans l'application, identifiables par l'utilisateur comme logiquement liées, mais maintenues dans la frontière d'une autre application. La première intention d'un GDE est de contenir des données référencées par un ou plusieurs processus élémentaires dans la frontière de l'application mesurée. Cela signifie qu'un GDE compté pour une application doit être un GDI dans une autre application.

Différences entre les GDI et les GDE

La différence essentielle entre un groupe logique interne et un groupe logique externe est qu'un GDE **n'est pas** maintenu par l'application mesurée, alors qu'un GDI l'est.

Les transactions

Par opposition aux données qui sont conservées, les transactions représentent l'aspect dynamique du traitement de l'information dans une application.

Entrée

Une entrée est un processus élémentaire qui traite les données ou les paramètres de traitement qui viennent de l'extérieur de la frontière d'application. La première intention d'une entrée est de maintenir un ou plusieurs GDI et/ou de modifier le comportement du système.

Les entrées doivent provenir de l'usager et apportent certains changements dans les données se trouvant dans l'application. Les changements peuvent provenir d'une entrée directe via un écran, ou indirectement d'un fichier de transaction mis à jour dans une autre application. Les entrées peuvent aussi être des données en lots. Dans tous les cas, il y a changement dans les données, soit par addition, modification ou destruction.

Sortie

Une sortie est un processus élémentaire qui envoie les données ou les paramètres de traitement à l'extérieur de la frontière de l'application. La première intention d'une sortie est de présenter l'information à un utilisateur à travers une logique de traitement autre que, ou en plus, de la récupération des données ou des paramètres de traitement.

La logique de traitement doit contenir au moins une formule ou un calcul mathématique, ou créer des données dérivées. Une sortie peut également maintenir un ou plusieurs GDI et/ou modifier le comportement du système.

---

[63] Ils ont également désignés sou le nom de Fichiers de données externe ou ELF (External Logical Files).

Elle se produit comme une partie régulière du traitement indépendamment des conditions établies dans les données. Par exemple, un rapport est produit mensuellement sans tenir compte de ce qui s'est produit.

Interrogation

Une interrogation est un processus élémentaire qui envoie des données ou des paramètres de traitement à l'extérieur de la frontière de l'application. La première intention d'une interrogation est de présenter l'information à un utilisateur à travers la récupération des données ou des paramètres de traitement d'un GDI ou d'un GDE.

La logique de traitement ne contient pas de formule ou de calcul mathématique, et ne crée aucune donnée dérivée. Aucun GDI n'est maintenu pendant le traitement, le comportement du système n'est pas modifié.

Une interrogation peut être vue comme étant une sortie en réponse à une requête (entrée). Ce qui caractérise cette transaction, c'est qu'elle ne modifie pas les entités, ni n'est le résultat d'une modification. Elle peut être sous la forme d'un écran ou d'un rapport, peu importe la technique. L'interrogation est une réponse directe à une requête, mais pas nécessairement une réponse immédiate.

Groupe de données référencé

Il s'agit soit d'un groupe logique interne lu ou maintenu par un composant relatif aux transactions, soit d'un groupe logique externe lu par un composant relatif aux transactions.

Donnée élémentaire (DE)

Une donnée élémentaire est un champ non - répété, unique et reconnaissable par l'utilisateur

Sous-ensembles logiques de données (SLD)

Il s'agit d'un sous-groupe de données élémentaires reconnaissable par l'utilisateur dans un GDI ou un GDE.

Brève présentation des règles de la méthode de mesure FPA [64]

Cette section explique les règles et les procédures de comptage de la méthode de mesure FPA. Le résultat du processus de comptage est le nombre de Points de Fonction Brut (PFB) relatif à un projet ou une application. Celui-ci reflète le nombre de fonctionnalités spécifiques fournies à l'utilisateur par le projet ou l'application.

[64] Se référer au manuel de mesure de IFPUG 4.1 pour avoir les règles détaillés

Les fonctionnalités spécifiques fournies à l'utilisateur sont évaluées dans des termes de "*ce qui*" est livré dans l'application (QUOI), et non pas *comment* cela est livré. On *ne* comptabilise *que* les besoins de l'utilisateur et les composants qui les définissent.

Règles et principes d'identification du type de comptage

La première étape de la procédure de comptage en points de fonction est d'identifier le type de comptage en point de fonction que l'on va utiliser.

La méthode de comptage en points de fonction peut être appliquée aussi bien sur des projets de développement que sur des applications. Il existe trois types de comptages en points de fonction. Le comptage relatif au :

- **Projet de développement,** dont le comptage consiste à mesurer les fonctionnalités fournies aux utilisateurs après la première installation du logiciel livré quand le projet est terminé.

- **Projet d'évolution fonctionnelle,** dont le comptage consiste à mesurer les modifications devant s'appliquer à une application existante, afin d'ajouter, de modifier, ou de supprimer les fonctionnalités fournies aux utilisateurs quand le projet est terminé.

- **A l'application,** dont le comptage consiste à mesurer les fonctions courantes que l'application fournit à l'utilisateur. Ce nombre est initialisé quand la taille du développement en points de fonction est mesurée. Il est mis à jour après l'installation de tout projet d'évolution fonctionnelle modifiant les fonctionnalités de l'application existante.

Comptages estimés et finalisés

Il est important de comprendre qu'au début d'un projet, tout comptage en points de fonction correspond à une estimation des fonctions livrées. De plus, au fur et à mesure que l'étendue du projet est clarifiée et que les fonctions sont développées, il est normal d'identifier les fonctions supplémentaires qui n'étaient pas précisées dans les besoins originaux. Ce phénomène est quelquefois appelé changement *d'envergure du champ d'étude.*

Il est essentiel de mettre à jour les comptages d'application lors de l'achèvement des projets. Si les fonctions changent durant le développement, le comptage en points de fonction à la fin du cycle de vie devrait refléter exactement les fonctions livrées à l'utilisateur.

Le diagramme suivant illustre les différents types de comptages en points de fonction et les rapports qui existent entre eux (projet A est d'abord terminé, suivi par le projet B).

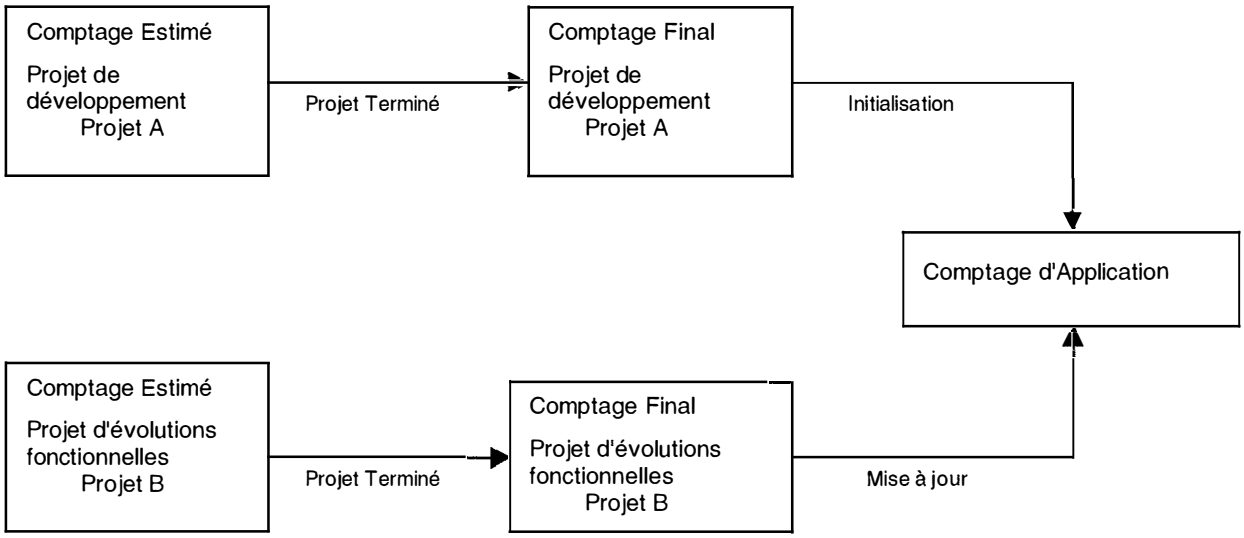


Figure 3: Relation entre les différents types de comptage de la méthode FPA

Règles d'Identification de l'Etendue du Comptage et de la Frontière de l'Application

Cette section définit l'étendue du comptage et la frontière de l'application et explique comment elles sont influencées par l'objectif du comptage.

L'objectif d'un comptage en points de fonction est de fournir une réponse à un problème du business (estimation de l'effort, du coût, comparaison d'application).

La portée du comptage définit la fonctionnalité qui sera incluse dans un comptage particulier en point de fonction, par exemple, les fonctions qui seront prises en compte.

- Un comptage en points de fonction d'une évolution contient toutes les fonctions ajoutées, modifiées et supprimées. La frontière de (des) l'application(s) touchée(s) demeure la même. La fonctionnalité de (des) l'application(s) reflète l'impact des fonctions ajoutées, modifiées ou supprimées.
- Un comptage en points de fonction d'un développement contient toutes les fonctions touchées (construites ou adaptées) par les activités du projet.
- Un comptage en points de fonction d'application peut inclure, selon l'objectif (par exemple, fournir un progiciel comme solution de logiciel) :
  - seulement les fonctions utilisées par l'utilisateur
  - toutes les fonctions livrées

La frontière de l'application indique la limite entre le logiciel mesuré et l'utilisateur. Elle définit ce qui est externe à l'application et elle agit comme une 'membrane' à travers laquelle les données traitées par les transactions (entrée, sortie et entrée) entrent et sortent de l'application. Elle aide à l'identification des données logiques référencées par les applications mais non maintenues par celles-ci (GDE). La frontière d'une application dépend de la vue externe du business de l'utilisateur de l'application. Elle est indépendante des considérations techniques et/ou de mise en œuvre.

Règles et principes de calcul de la contribution des composants relatifs aux Transactions

Les composants relatifs aux transactions représentent la fonctionnalité fournie à l'utilisateur pour le traitement des données par une application. Les composants relatifs aux transactions sont définis en tant que entrées, sorties, et interrogations.

Pour identifier les processus élémentaires (entrée, sortie, interrogation), il faut rechercher les activités de l'utilisateur se produisant dans l'application. Le processus doit être la plus petite unité d'activité autonome qui soit significative pour l'utilisateur. Il doit laisser l'activité de l'application dans un état de cohérence fonctionnelle.

Règles de complexité et de contribution

Le nombre d'entrées, de sorties, et d'interrogations et de leur complexité fonctionnelle relative détermine la contribution des composants relatifs aux transactions au nombre de point de fonction brut.

On attribue à chaque entrée, sortie et interrogation identifiés une complexité fonctionnelle basée sur le nombre de groupes logiques de données référencés (GDR) et sur le nombre de données élémentaires (DE).

La complexité fonctionnelle des Entrées est évaluée en utilisant le tableau de complexité suivant :

.

	1 à 4 DE	**5 à 15 DE**	**16 ou plus DE**
0 à 1 GDR	Faible	Faible	Moyenne
2 GDR	Faible	Moyenne	Elevée
3 ou plus GDR	Moyenne	Elevée	Elevée

Tableau 1 : Matrice de complexité des entrées

La complexité fonctionnelle des Sorties et Interrogations est évaluée en utilisant le tableau de complexité suivant :

	1 à 5 DE	6 à 19 DE	20 ou plus DE
0 à 1 GDR	Faible	Faible	Moyenne
2 à 3 GDR	Faible	Moyenne	Elevée
4 ou plus GDR	Moyenne	Elevée	Elevée

Tableau 2 : matrice de complexité des Sorties et Interrogation.

La traduction de la complexité des entrées ou des interrogations en points de fonction bruts est donnée par le tableau 3.

Niveau de Complexité Fonctionnelle	Points de Fonction Brut
Faible	3
Moyenne	4
Elevée	6

Tableau 3 : Points de fonction brut des Entrées et Interrogations suivant le niveau de complexité

Pour traduire la complexité des Sorties en points de fonction bruts, on utilise le tableau suivant :

Niveau de Complexité Fonctionnelle	Points de Fonction Brut
Faible	4
Moyenne	5
Elevée	7

Tableau 4 : Points de fonction brut des Sortie suivant le niveau de complexité

Principes de détermination du Facteur d'Ajustement

Le Facteur d'Ajustement (FA) est basé sur 14 caractéristiques générales du système (CGS) qui représentent la fonctionnalité générale de l'application mesurée. Chaque caractéristique a des descriptions associées qui aident à la détermination du degré d'influence de cette caractéristique.

Les 14 *caractéristiques générales du système* sont les suivants :

1. **Communications de Données**

   Les Communications de Données décrient le degré selon lequel l'application communique directement avec le processeur.

2. **Système Distribué/Informatique Répartie**

   Le Système Distribué/Informatique répartie décrit le degré selon lequel l'application transfère des données parmi des composants de l'application.

3. **Performance**

   La Performance décrit le degré selon lequel les considérations de temps de réponse et de performance de sortie ont influencé le développement d'application.

4. **Configuration à Utilisation Intensive**

   La Configuration Fortement Utilisée décrit le degré selon lequel les restrictions de ressource de l'informatique ont influencé le développement de l'application.

5. **Taux de Transaction**

   Le taux de transaction décrit le degré selon lequel le taux de transactions a influencé le développement de l'application.

6. **Saisie Interactive**

   La Saisie de Données en Ligne décrit le degré auquel des données sont saisies par des transactions interactives.

7. **Convivialité/Efficacité de l'Utilisateur Final**

   La Convivialité/Efficacité de l'Utilisateur Final décrit le degré de considération sur les facteurs humains et la facilité d'utilisation pour l'utilisateur de l'application mesurée.

8. **Mise à jour en Temps réel**

   La mise à jour en temps réel décrit le degré selon lequel des fichiers logiques internes sont mis à jour en temps réel.

9. **Complexité des Traitements**

La complexité de traitement décrit le degré selon lequel la logique de traitement a influencé le développement de l'application.

10. Réutilisation

La réutilisation décrit le degré selon lequel l'application et le code dans l'application ont été spécifiquement conçus, développés, et maintenus pour être utilisés dans d'autres applications.

11. Facilité d'Installation

La Facilité d'Installation décrit le degré selon lequel la conversion des environnements précédents a influencé le développement de l'application.

12. Facilité d'Exploitation

La facilité opérationnelle décrit le degré selon lequel l'application s'occupe des aspects opérationnels, tels que les processus de démarrage, de sauvegarde, et de restauration.

13. Portabilité de l'application/Multi Sites

Le 'multi sites' décrit le degré selon lequel l'application a été développée pour des sites et des organisations multiples d'utilisateur.

14. Facilité d'Adaptation

La Facilité d'Adaptation décrit le degré selon lequel l'application a été développée pour la modification facile de la logique de traitement ou de la structure des données.

Le degré d'influence pour chaque caractéristique s'étend sur une échelle de zéro à cinq (d'une influence nulle à une influence forte).

Les 14 caractéristiques générales de système sont récapitulées dans la valeur du facteur d'ajustement. Une fois appliquée, la valeur du facteur d'ajustement ajuste le nombre de points de fonction brut de + ou -35 % pour produire le nombre de points de fonction net.

Les étapes suivantes décrivent les procédures pour déterminer la valeur du facteur d'ajustement :

1. Évaluer chacune des 14 caractéristiques générales de système sur une échelle de zéro à cinq pour déterminer le degré d'influence (DI).

2. Ajouter les degrés d'influence de chacune des 14 caractéristiques générales du système afin d'obtenir le degré d'influence total (DIT).

3. Insérer le DIT dans l'équation suivante pour avoir la valeur du facteur d'ajustement.

   FA = (DIT * 0.01) + 0.65

Règles et principes de calcul de la contribution des composants relatifs aux Données

Cette section définit les règles qui s'appliquent lors du comptage des Groupes logiques de Données Internes et des Groupes logiques de Données Externes.

Résumé des Procédures de Comptage

Ce résumé est inclus pour montrer les règles dans le contexte des procédures de comptage des GDI et des GDE.

Les procédures de comptage des GDI et des GDE comprennent les deux activités suivantes :

1. Identifier les GDI et les GDE.
2. Déterminer la complexité des GDI et des GDE, et leur contribution au nombre de points de fonction brut.

	1 à 19 DE	**20 à 50 DE**	**51 ou plus DE**
1 SLD	Faible	Faible	Moyenne
2 à 5 SLD	Faible	Moyenne	Elevée
6 ou + SLD	Moyenne	Elevée	Elevée

Tableau 5: La matrice de complexité

La table suivante est utilisée pour traduire les GDI en un nombre de points de fonction brut.

Niveau de Complexité Fonctionnelle	**Nombre de Points de Fonction Brut**
Faible	7
Moyenne	10
Elevée	15

Tableau 6 : Table de traduction des GDI

Le tableau 7 présente les règles de traduction des GDE en nombre de points de fonction brut.

Niveau de Complexité Fonctionnelle	Nombre de Points de Fonction Brut
Faible	5
Moyenne	7
Elevée	10

Tableau 7 : Table de traduction des GDE

Par exemple, un GDE de niveau de complexité élevée se traduit en 10 points de fonction brut.

Le poids final des groupes de données est la somme des points accumulés.
La taille de l'application sera quant à elle égal au total des points obtenus pour chacune des composantes « transactions » et « données ».

ANNEXE 2 : Contexte de travail et environnement

Le travail d'études des points de fonction a été réalisé au Laboratoire de Recherche en Gestion de Logiciels qui constitue un des départements de Recherche de l'Université du Québec à Montréal (UQÀM).

Ce travail s'est déroulé en 3 phases, la première phase consistait à une étude des différents types de mesures fonctionnelles et leurs domaines d'application. Une étude plus poussée a été faite pour COSMIC-FFP, qui a été utilisé lors de la deuxième phase pour mesurer les points de fonction d'une vingtaine de projets. Dans la dernière phase, une exploitation et une analyse des résultats des mesures ont été effectuées.

Présentation du Laboratoire de Recherche en Gestion des Logiciels[65]

En 1995, Bell Canada s'allie à l'Université du Québec à Montréal (UQAM) pour la création d'un Laboratoire de Recherche en Gestion de Logiciels. L'objectif de ce dernier est de développer les modèles analytiques et les outils de travail qui permettront aux gestionnaires d'ajuster de façon optimale leurs dépenses informatiques à leurs objectifs d'affaires. Ce laboratoire de recherche est dirigé par M. Alain Abran, professeur au département d'informatique et spécialiste du génie logiciel.

A. Projet: Modèles de Productivité et d'Estimation avec les Points de Fonction

Ce premier projet de recherche vise à améliorer les modèles de productivité et d'estimation, qui utilisent la technique des points de fonction comme mesure de la taille fonctionnelle des logiciels.

B. Projet: Modèles de Productivité pour la Maintenance du Logiciel

Ce projet vise à développer des mesures et des modèles d'analyse de productivité spécifiques à la maintenance des logiciels qui pourront être utilisés en entreprise pour la gestion interne des coûts de maintenance, pour la gestion des contrats d'impartition et pour le développement et le contrôle des programmes d'amélioration de la qualité et de la productivité.

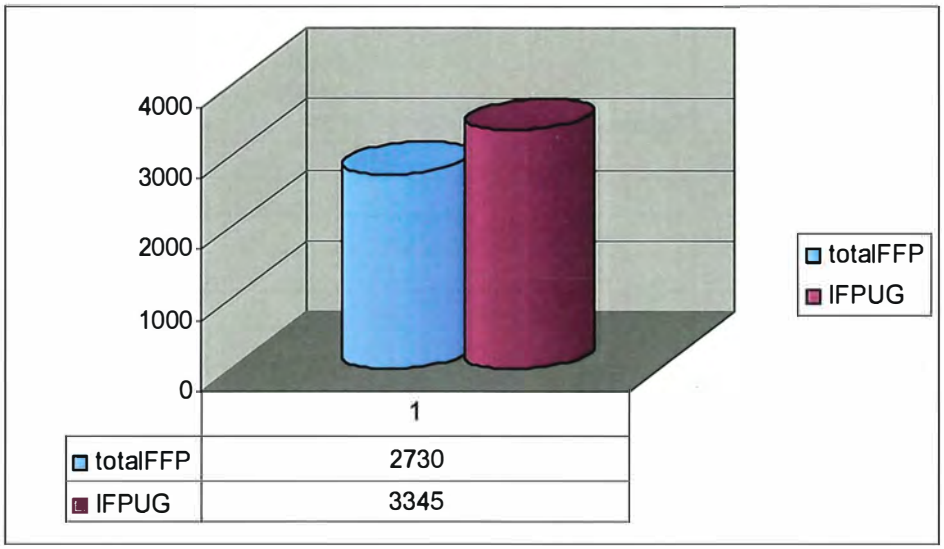
65 ***Alliance UQAM-Bell Canada pour la création d'un Laboratoire de Recherche en Gestion des Logiciels (WWW.lrgl.uqam.ca).***

ANNEXE 3 : RESULTATS COMPLETS DE LA MESURE

Les projets 22 et 24 ont été enlevés car la documentation disponible pour la mesure n'était pas complète.

Nom du projet	Entrée	Lecture	Ecriture	Sortie	Total COSMIC-FFP	Total FPA(transaction +Fichiers)	FPA sans Fichiers logiques
P1	38	37	10	63	148	187,00	163,2
P2	6	3	6	8	23	87,00	47,8
P3	22	26	18	27	93	204,00	144
P4	15	15	13	21	64	118,20	118,2
P5	20	17	14	24	75	124,00	113,4
P6	12	11	8	16	47	53,00	53
P7	32	32	14	51	129	244,00	148,2
P8	6	6	2	10	24	24,40	24
P9	7	7	7	13	34	41,60	42
P1	17	17	10	24	68	105,00	73,4
P1	63	59	15	111	248	393,00	294
P1	19	16	3	38	76	270,00	262,6
P13	29	29	0	58	116	231,00	168,2
P14	18	18	7	27	70	104,00	93,2
P15	8	8	0	16	32	32,00	32
P16	12	11	8	18	49	61,60	61,6
P17	10	10	3	17	40	65,00	49,6
P18	12	12	9	15	48	72,60	49,2
P19	38	38	11	65	152	176,00	170,4
P20	26	20	18	34	98	111,20	111,2
P21	6	6	3	9	24	41,00	28,2
P23	30	30	10	49	119	162,20	162,2
P25	29	22	24	35	110	165,00	141,4
P26	39	35	13	57	144	272,00	214,6
total	614	580	345	893	2432		

Comparaison des résultats COSMIC –FFP et FPA si on suppose deux groupes de données pour la mesure COSMIC-FFP:



Les resultants sont presque equivalents à trois groupes de données pour la méthode de mesure COSMIC-FFP :

Valeurs pour COSMIC-FFP à 3 groupes de données	**Valeurs pour FPA**
3335	**3345**

Le tableau suivant présente les valeurs obtenues pour chaque projet en ce qui concerne la documentation fonctionnelle, il est classé du projet dont les processus sont les mieux documentés aux moins bien documentés.

Nom du projet	NA	Documentés	Listés	quantifiés	implicites
P11	9	71	0	0	4
P26	0	49	8	0	0
P23	0	29	2	0	0
P3	0	14	21	0	4
P16	0	12	0	0	0
P6	0	11	0	0	1
P20	4	8	0	0	18
P4	0	7	14	3	0
P2	0	7	6	0	2
P13	1	5	21	11	0
P7	0	3	45	0	0
P18	3	3	10	0	0
P25	3	1	33	0	0
P12	5	0	46	0	4
P19	6	0	40	0	0
P1	0	0	29	0	9
P10	2	0	21	0	0
P5	0	0	22	0	7
P14	0	0	18	0	2
P17	0	0	12	0	1
P9	4	0	8	0	0
P15	0	0	8	0	0
P21	0	0	8	0	0
P8	0	0	6	0	0

Le tableau suivant fait une interprétation des résultats obtenus au tableau précèdent. La deuxième colonne reprend la somme des valeurs de tous les projets par type de document, la deuxième colonne reprend la somme des valeurs obtenues pour les 7 projets isolés (les mieux documentés). La troisième colonne reprend les résultats des autres produits restants. Les trois colonnes suivantes reprennent respectivement le rapport entre le total (T) pour touts les projets et le nombre de processus mesurés (26), les valeurs obtenus pour les 7 premiers divisées par 7, et enfin le rapport entre les valeurs obtenues pour le reste des projets et le nombre de projet (19).

	Total Tous	Total 7 premiers	Total Autres	Proportion Tous	Proportion 7 premiers	Proportion Autres
NA	37	13	24	1,42	1,86	1,26
Documentés	220	204	25	8,46	29,14	1,32
Listés	378	31	355	14,54	4,43	18,68
Quantifiés	14	0	14	0,54	0,00	0,74
Implicites	52	28	25	2,00	4,00	1,32

Les résultats obtenus dans l'évaluation des types d'activité relatifs à l'évolution de processus fonctionnels sont repris ci-dessous:

Nom du projet	Ajoutés	Modifiés	Supprimés	NA	total
P11	75	0	0	9	84
P26	57	0	0	0	57
P12	43	7	0	5	55
P23	29	2	0	0	31
P20	26	0	0	4	30
P3	23	16	0	0	39
P14	20	0	0	0	20
P25	18	16	0	3	37
P2	14	1	0	0	15
P17	13	0	0	0	13
P21	8	0	0	0	8
P4	8	11	5	0	24
P8	6	0	0	0	6
P1	5	33	0	2	40
P10	5	16	0	2	23
P13	5	1	31	1	38
P19	2	36	2	6	46
P16	2	10	0	0	12
P18	1	11	1	3	16
P6	1	11	0	0	12
P15	1	7	0	0	8
P7	0	26	12	0	38
P5	0	12	17	0	29
P9	0	7	1	4	12

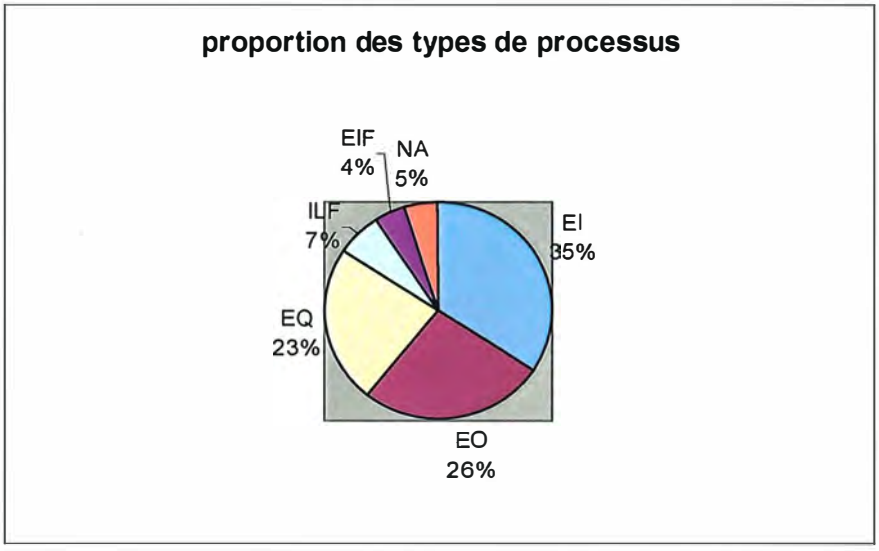
Les résultats du tableau précèdent ont été exploités, dans le but de faire un classement des projets selon le type de modification subi. On a isolé les valeurs pour les projets ayant connu le plus d'ajout de processus (au nombre de 7), ces valeurs se retrouvent à la troisième colonne. La quatrième colonne reprend le total des valeurs pour le restant des projets La cinquième colonne reprend le rapport entre les valeurs totales des projets (par types d'activités) et le nombre total des projets (26). La sixième colonne calcule la proportion de chaque type d'activité pour les 7 projets

	Tous	7 premiers	Autres	Valeurs relatives pour Tous	Valeurs relatives pour les 7 premiers	Autres
Ajoutés	362	273	89	13,92	39,00	5,24
Modifiés	223	25	198	8,58	3,57	11,65
Supprimés	69	0	69	2,65	0,00	4,06
NA	39	18	21	1,50	2,57	1,24

L'analyse des types de processus selon la méthode de mesure FPA ou IFPUG 4.1 est reprise dans le tableau ci-dessous :

Nom du projet	EI	EO	EQ	ILF	EIF	NA	Nombre total de processus
P1	11	5	22	3	0	0	41
P2	6	3	0	5	1	0	15
P3	14	1	14	3	7	0	39
P4	12	1	11	0	0	0	24
P5	27	0	0	0	2	0	29
P6	8	2	2	0	0	0	12
P7	11	10	11	5	11	0	48
P8	2	0	4	0	0	0	6
P9	3	5	0	0	0	4	12
P10	9	2	6	4	0	2	23
P11	14	24	24	12	1	9	84
P12	9	36	4	1	0	2	52
P13	0	29	0	8	0	1	38
P14	7	11	0	0	2	0	20
P15	0	0	8	0	0	0	8
P16	5	7	0	0	0	0	12
P17	3	5	2	0	3	0	13
P18	1	5	4	3	0	3	16
P19	9	7	23	0	1	6	46
P20	18	2	6	0	0	4	30
P21	3	2	1	1	1	0	8
P23	11	20	0	0	0	0	31
P25	24	7	0	3	0	3	37
P26	21	8	20	6	2	0	57
Total	228	192	162	54	31	34	701

Les résultats obtenus pour les types de processus FPA ou IFPUG sont résumés dans le graphique ci-après :



EI représente les entrées (Input)
EO représente les Sorties (Output)
EQ représente les interrogations (Query)
ILF représente les fichiers de données logiques Internes (Internal Logical Files)
ELF représente les fichiers de données logiques Externes (External logical Files)
NA représente les processus non disponibles (qui n'ont pas pu être classés)

Les valeurs utilisées dans la mesure des MRE

Le tableau suivant regroupe les valeurs utilisées dans le calcul de l'Erreur Relative Moyenne (MRE). Les formules présentées à la section 4.4.4 ont été utilisées pour obtenir ces valeurs.

Nom du projet	MRE 1GD-FPA total	MRE 1GD-FPA sans fichiers	MRE 2GD-FPA total	MRE 2GD FPA sans fichiers	MRE 3GD-FPA total[66]	MRE 3GD –FPA sans Fichier[67]
P1	20,86%	9,3%	4,28%	19,49%	-29,41%	-48,28%
P2	74%	51,9%	63,22%	33,05%	52,87%	14,23%
P3	54,41%	35,4%	38,73%	13,19%	25,98%	-4,86%
P4	45,85%	45,9%	22,03%	22,17%	-1,69%	-1,52%
P5	39,52%	33,9%	14,52%	6,53%	-10,48%	-20,81%
P6	11,32%	11,3%	24,53%	24,53%	-60,38%	-60,38%
P7	47,13%	13,0%	28,28%	18,08%	9,43%	-49,12%
P8	1,64%	1,6%	33,33%	31,15%	-66,67%	-63,93%
P9	18,27%	18,3%	14,29%	15,38%	-47,62%	-49,04%
P10	35,24%	7,4%	9,52%	29,43%	-16,19%	-66,21%
P11	36,90%	15,6%	18,07%	9,52%	-0,76%	-34,69%
P12	71,85%	71,1%	65%	64%	62,22%	61,16%
P13	49,78%	31,0%	37,23%	13,79%	24,68%	-3,45%
P14	32,69%	24,9%	8,65%	2%	-15,38%	-28,76%
P15	0,00%	0,0%	25,00%	25,00%	-50,00%	-50,00%
P16	20,45%	20,5%	9,68%	10,39%	-40,32%	-41,23%
P17	38,46%	19,4%	18,46%	6,85%	-1,54%	-33,06%
P18	33,88%	2,4%	5,48%	40,24%	-23,29%	-82,93%
P19	13,64%	10,8%	14,20%	17,96%	-42,05%	-46,71%
P20	11,87%	11,9%	22,52%	22,30%	11,71%	11,87%
P21	41,46%	14,9%	19,51%	17,02%	-2,44%	-48,94%
P23	26,63%	26,6%	2%	1,97%	-22,84%	-22,69%
P25	33,33%	22,2%	5,45%	10,33%	-22,42%	-42,86%
P26	47,06%	32,9%	29,41%	10,53%	11,76%	-11,84%

FPA total représente le total des points obtenus avec la méthode de mesure FPA. La première colonne reprend donc le MRE calculé avec les résultats FPA total et COSMIC-FFP. La deuxième colonne représente le MRE obtenu avec COSMIC-FFP et les résultats FPA auxquels on a soustrait les fichiers de données logiques.

[66] La valeur absolue n'a pas été calculé dans ce tableau afin de bien montrer la diminution des valeurs.
[67] Idem.

ANNEXE 4 : EXEMPLE D'APPLICATION DE LA METHODE DE MESURE COSMIC-FFP.

Durant le stage à l'université du Québec à Montréal, la méthode COSMIC-FFP a été également appliquée sur un petit projet réel. Encore une fois la documentation n'était pas complète et il a fallu faire également des hypothèses pour pouvoir appliquer la méthode de mesure. L'énoncé et les résultats sont présentés ci-après.

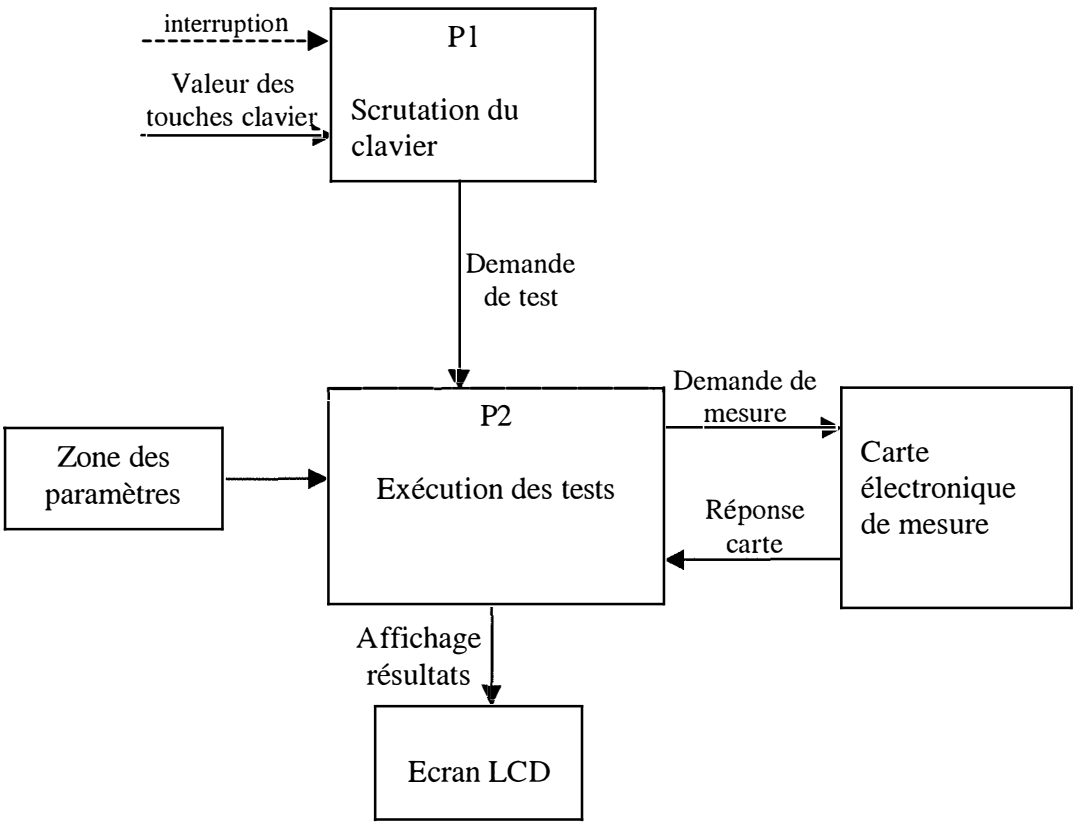
ENONCE

L'exemple applicatif est constitué de 2 processus qui fonctionnent en parallèle (vu de l'utilisateur).

Le premier processus P1 est chargé de la scrutation sur interruption d'un clavier 16 touches.

Le second P2 est chargé d'effectuer en boucle le test demandé par l'opérateur tant que ce dernier n'en demande pas un autre. Pour effectuer un test le processus déclenche une ou plusieurs mesures sur une carte électronique. Il effectue ensuite une lecture des résultats qu'il traite pour déterminer si le test est OK ou NOK.

Le schéma ci-dessous représente le dispositif.



Processus 1 : Scrutation du clavier

Le processus se met en attente d'une interruption. Il en reçoit une par touche de clavier appuyée.

L'utilisateur doit saisir un nombre à 3 chiffres représentatif du numéro du test à faire (de 000 à 017). Le processus acquière ce nombre et vérifie que le numéro demandé est correct.

Ensuite il transmet ce nombre au processus 2 et se remet en attente d'une interruption.

Processus 2 : Exécution des tests

Ce processus est en attente d'une demande de test.

S'il reçoit le numéro 0 :

- il affiche « Autotest en cours »
- il enchaîne tous les tests ci-dessous les uns après les autres.
- si tous les tests sont corrects, à la fin du dernier test, le processus affiche sur l'écran LCD « AUTOTEST OK » puis se met en attente d'une nouvelle demande.
- si un test est mauvais, le processus s'arrête aussitôt sans faire les suivants et affiche « AUTOTEST KO » et n'accepte plus de demande.

S'il reçoit un numéro autre que 0, le processus exécute en boucle le test correspondant. A la fin de chaque boucle, il regarde si une nouvelle demande n'est pas présente. Si une nouvelle demande est présente, il change de test et commence à l'exécuter en boucle.

17 tests sont possibles. Pour la plupart des tests, le processus fait une ou plusieurs demandes de mesure à la carte électronique (sauf pour les tests 1 et 13). Chaque demande est suivie d'une lecture du résultat constitué de 1 à 512 échantillons. Certains tests nécessitent d'obtenir plus de 512 échantillons, la demande de mesure est alors renouvelée. La demande se fait par changement d'un ou plusieurs bits dans un registre. Ce registre n'est pas forcément le même pour tous les tests.

Pour chaque test, un calcul algorithmique est effectué (valeur moyenne, valeur efficace, période, etc..). Le résultat du calcul est comparé à une borne maximum et une borne minimum lues dans la zone de paramètres.

A la fin de chaque test (de chaque boucle), le processus affiche le résultat sur l'écran LCD. L'affichage est constitué du numéro du test, du nom du paramètre, de la valeur calculée, de l'unité du paramètre, et d'un effet clignotant si la valeur est en dehors des bornes minimum et maximum Le nom et l'unité du paramètre sont lus dans la zone des paramètres.

Description des actions lors des tests :

- Test 1 : lecture dans la zone paramètre et comparaison de la version logiciel
- Test 2 à 6 : demande de conversion de tension dans le registre A
  Lecture de 512 échantillons
  Calcul de valeur moyenne
- Test 7 : lecture du registre B pour lire l'état d'entrée tout ou rien
- Test 8 et 9 : 8 demandes de conversion de tension dans le registre A
  Lecture de 8 fois 512 échantillons (une demande entre chaque lecture d'échantillons)
  Calcul de valeur moyenne
- Test 10 et 11 : 8 demandes de conversion de tension dans le registre A
  Lecture de 8 fois 512 échantillons (une demande entre chaque lecture d'échantillons)
  Calcul de valeur moyenne et valeur efficace
  Recherche de période de signal
- Test 12 : lecture registre C pour lire une valeur sur 16 bits (température)
- Test 13 : essai d'accès à l'écran (affichage de caractères et relecture)
- Test 14 à 17 : écriture dans le registre D d'une valeur sur 16 bits
  (positionnement de sorties tout ou rien
  Relecture de la valeur dans le registre D

Les registres A, B, C et D ainsi que les échantillons sont écrits ou lus sur la carte électronique.

Zone de paramètres

Elle comprend les données suivantes :

- Version du logiciel
- Pour les 16 tests :
  - Numéro du test
  - Nom du paramètre (15 caractères)
  - Unité du paramètre (5 caractères)
  - Borne maximum du paramètre
  - Borne minimum du paramètre

RESULTATS DE LA MESURE

P1: chargé de la scrutation sur interruption d'un clavier.

P2 : chargé d'effectuer en boucle le test demandé par l'opérateur tant que ce dernier n'a pas demander un autre.

Interprétation du cas

- *La frontière* de l'application est l'environnement des tests.

Les groupes de données :

  - Les registres A, B, C et D ainsi que les échantillons sont considérées comme *des Groupes de Données*; ils sont lus et écrits sur la carte électronique de mesure.
  - La zone des paramètres est un autre *Groupe de données* qui est lu.

- P1 et P2 sont considérés comme étant *un et un seul processus.*

Justification :

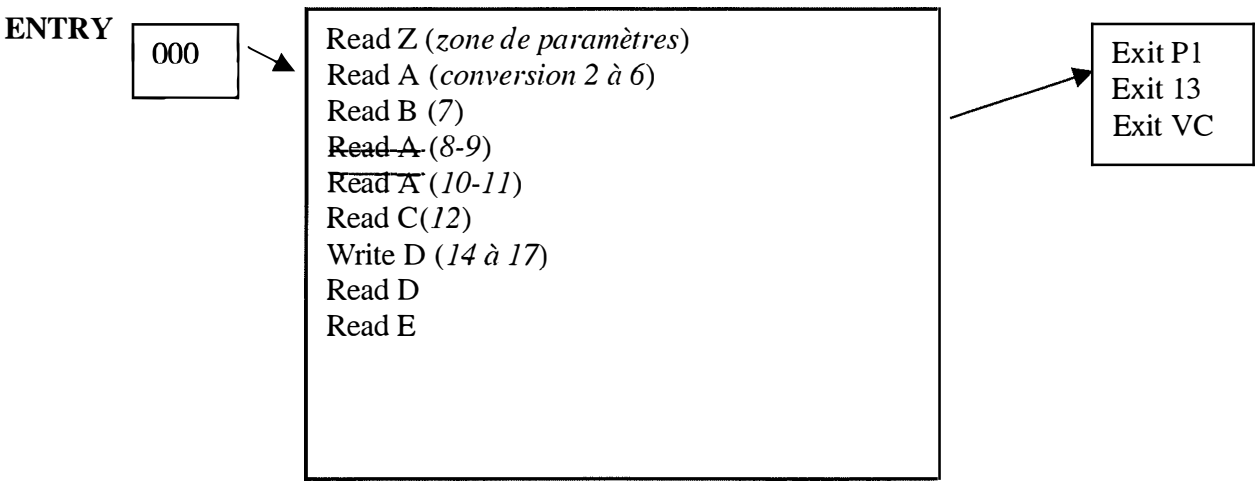
Lorsque l'information est transmise à l'exécution des tests aucune information n'entre à nouveau à l'intérieur de la frontière de l'application.

- L'acquisition du nombre de 3 chiffres entré pas l'utilisateur est *l'événement déclencheur* de l'exécution des tests.

Hypothèses et Raisonnement

L'entrée de 000 : déclenchement de tous les tests, lecture et écriture des registres A, B, C, D et l'échantillon.

Si on a un seul processus voici le calcul.



Z : zone de paramètres.
13 : test de l'écran (affichage de caractères)
VC : Résultat de tous les paramètres du résultat (numéro de test, nom de paramètre, valeur calculée,…).

Après élimination des doublons, la taille du processus est de :
1Entry + 5 Read + 1 write + 3 Exit = 10

Si on considère les autres tests comme étant des processus :

Justification :

- L'utilisateur peut choisir d'avoir un résultat partiel.
- Chaque processus doit fournir des types de résultats différents
- Les tests 2 à 6 ainsi que 8 et 9 fournissent le même type de résultat
- Le nombre d'échantillon ne compte pas puisqu'il s'agit d'une occurrence des mêmes attributs d'une entité (échantillon)
- Le nombre de lecture (8 fois) ne compte pas puisque le traitement est le même à chaque lecture.
- Les tests 10 et 11 sont différents de 2 à 6 et 8 et 9 puisque le traitement et quelque peu différent, on traite la valeur efficace et la recherche de période de signal.

Calcul

Hypothèse : A la fin de chaque test, le processus affiche le résultat sur l'écran.

Entry 001 : -R de Zone de paramètres
Taille= 1entry + 1 Read + 1 Exit= 3

Entry 002 à 006 : -R de A
R de Echantillon
Taille= 1entry + 2 Read + 1 Exit= 4

Entry 007 :- R de B
Taille= 1entry + 1 Read + 1 Exit= 3

Entry 008et 009 : -R de A
R de échantillons
Taille= 1entry + 2 Read + 1 Exit= 4

Entry 010 et 011 : - R de A
R de échantillon
Taille= 1entry + 2 Read + 1 Exit= 4

Entry 012 : - R de C
Taille= 1entry + 1 Read + 1 Exit= 3

Entry de 013 : - Exit de caractères
????????????????????????????

Entry de 014 à 017 : - W de D
- R de D

Taille= 1entry + 1 Read + 1 write + 1 Exit=4

Ajout d'un processus pour erreur d'entrée possible

Entry X : - Read de Z (réponse négative)
-Sortie correspondant au message d'erreur.

Où X correspond à n'importe quel caractère invalide en entrée

Question sur le document :

A quoi correspond la valeur efficace calculée dans le test 10 et 11?
A quoi consiste la Recherche de période de signal?