La Méthode de mesurage COSMIC de la taille fonctionnelle

Version 4.0

Manuel de mesurage

(Le Guide COSMIC d'implémentation pour ISO/IEC 19761: 2011)

Avril 2014

Remerciements

Auteurs et Membres fondateurs de la Version 2.0 (ordre alphabétique) :
Alain Abran, École de technologie supérieure – Université du Québec,
Jean-Marc Desharnais, Software Engineering Laboratory in Applied Metrics - SELAM,
Serge Oligny, Bell Canada,
Denis St-Pierre, DSA Consulting Inc.,
Charles Symons, Software Measurement Services Ltd.

Version 4.0 Réviseurs 2014 (ordre alphabétique)		
Alain Abran École de Technologie Supérieure, Université du Québec Canada	Diana Baklizky TI Metricas Brazil	Jean-Marc Desharnais École de Technologie Supérieure – Université du Québec Canada
Peter Fagg Pentad United Kingdom	Cigdem Gencel Free University of Bolzano-Bozen Italy	Charles Symons* United Kingdom
Jayakumar K. R. Amitysoft India	Arlan Lesterhuis* The Netherlands	Bernard Londeix Telmaco United Kingdom
Shin-Ichi Nagano NTT Comware@ Japan	Luca Santillo Agile Metrics Italy	Hassan Soubra Ecole Supérieure des Techniques Aéronautiques et de Construction Automobile France
Sylvie Trudel Université du Québec à Montréal Canada	Monica Villavicencio ESPOL Ecuador	Frank Vogelezang Ordina The Netherlands
Chris Woodward Chris Woodward Associates Ltd. United Kingdom		

* Éditeurs de la version 4.0 de la méthode de mesurage COSMIC

Pour les réviseurs des versions antérieures de la méthode COSMIC, veuillez consulter les documents respectifs.

Contrôle des versions

Le tableau suivant donne l'historique des versions de ce document

DATE	Réviseurs	Modifications / Additions
1999-03-31	Serge Oligny	Première version pour commentaires des réviseurs
1999-07-31	Voir remerciements	Révision, incluant les commentaires des réviseurs
1999-10-06	Voir remerciements	Révision, incluant les commentaires du IWSM '99 workshop[1]
1999-10-29	COSMIC Équipe de base	Révision, derniers commentaires avant la publication des essais sur le terrain, version 2.0.
2001-05-01	COSMIC Équipe de base	Révision pour la conformité à l'ISO/IEC 14143-1: 1998 avec plus de précisions sur les règles du mesurage de la version 2.1.
2003-01-31	COSMIC Comité des pratiques du mesurage	Révision pour la conformité à l'ISO/IEC FDIS 19761: 2002 avec plus de précisions sur les règles du mesurage de la version 2.2.
2007-09-01	COSMIC Comité des pratiques du mesurage	Révision avec de nouvelles précisions et compléments aux règles mesure vers la version 3.0, notamment au niveau de la phase de la stratégie du mesurage. Le nom de la méthode est passé de la « méthode de COSMIC-FFP » à la « méthode COSMIC». Lors de la mise à jour de la version 3.0 à partir de la version 2.2, les parties, de la version 2.2 'Manuel de mesurage', ont été séparées et se trouvent dans divers documents. – Voir l'avant-propos ci-après et l'annexe E
2009-05-01	COSMIC Comité des pratiques du mesurage	Version 3.0 révisée vers la version 3.0.1 avec des précisions et améliorations rédactionnelles mineures en plus de distinguer plus clairement les exemples. Cette version intègre également les changements proposés dans les Bulletins d'amélioration de la méthode 3, 4 et 5. Voir l'annexe E pour plus de détails concernant ces changements.
2014-04-01	COSMIC Comité des pratiques du mesurage et autres	Les révisions de la version 4.0 prennent en compte les Bulletins d'amélioration de la méthode 6 à 11 et incorporent les termes du glossaire. Voir l'annexe E pour plus de détails concernant ces changements.

[1] Proceedings of the International Workshop on Software Measurement IWSM '99, Lac Supérieur, Québec, Canada, September 1999

Préface

La Méthode COSMIC est une méthode normalisée pour mesurer Taille Fonctionnelle d'un logiciel des domaines fonctionnels généralement désignés sous le nom logiciel d'application d'affaires (ou `MIS') et des logiciels temps réel, d'infrastructures ainsi que certains types de logiciel scientifiques et d'ingénierie. La Méthode de mesurage COSMIC a été acceptée par ISO/IEC JTC1 SC7 en décembre 2002 en tant que norme internationale : 'COSMIC-FFP:2003 – A functional size measurement method' [1] (ci-après désignée comme `ISO/IEC 19761').

ISO/IEC 19761 contient seulement les définitions et les règles normatives fondamentales de la méthode. Le but du Manuel de mesurage est non seulement de fournir ces règles et définitions, mais de fournir également davantage d'explications et beaucoup plus d'exemples afin d'aider les mesureurs à comprendre et comment appliquer la méthode. Le Manuel de mesurage est la principale description normalisée de la méthode COSMIC pour une utilisation pratique.

En plus du 'Manuel de mesurage', le document 'Introduction to the COSMIC method of measuring software' [2] donne un sommaire de la version 4.0 de la méthode[2]. Ce document d'Introduction doit être lu en premier par toute personne qui est nouvelle dans le domaine de la taille fonctionnelle ('FSM') ou qui connait une autre méthode de mesurage de la taille fonctionnelle et envisage de convertir certains résultats ou encore qui désire simplement avoir un aperçu de la méthode COSMIC, avant de lire ce Manuel de mesurage. Beaucoup des renseignements généraux sur les FSM et la méthode de mesurage COSMIC, ainsi que soutenir les directives (par exemple sur la façon d'appliquer la méthode dans différentes circonstances particulières, etc.), des études de cas, des rapports de recherche, etc., se trouvent sur www.cosmicon.com.

Principaux changements à la version 4.0 de la méthode COSMIC

Le changement dans la désignation de la version de la méthode COSMIC 3.0.1 à 4.0 indique que cette version vise à améliorer de façon significative l'explication de la méthode par rapport à la version précédente, mais que la nouvelle version ne change en rien les principes de base de la méthode. Les principales modifications apportées au Manuel de mesurage, relativement avec la version précédente, sont les suivantes.

- Les propositions visant à clarifier les règles d'accès au stockage persistant de la méthode de mise à jour du Bulletin 7 v2 ('MUB 7 v2', novembre 2012) ont été incorporées.
- Les limitations sur l'applicabilité de la méthode COSMIC ont été livrées, tel que proposé dans le MUB 8, septembre 2010. On peut considérer que la méthode COSMIC peut être appliquée avec succès à un logiciel de type 'riche en manipulation de données', par exemple, un logiciel scientifique ou d'ingénierie.
- La définition COSMIC d'une «couche» a été simplifiée et met en lumière la conformité avec son utilisation courante dans l'industrie, tel que proposé dans le MUB 9 en novembre 2012.
- Le concept d'un modèle de stratégie du mesurage a été ajouté,
- Une section sur les 'exigences non-fonctionnelles' a été ajoutée en tenant compte du MUB 10 de décembre 2012. La méthode COSMIC peut être utilisée pour mesurer certaines exigences qui apparaissent au premier abord comme étant non fonctionnelles, mais qui évoluent pour devenir des exigences fonctionnelles au fur et à mesure de la progression d'un projet.
- La définition et les règles d'un 'processus fonctionnel' et plusieurs sujets connexes ont été révisés en conformité avec le MUB 11 de décembre 2013.
- De nouvelles règles pour la mesure des messages d'erreurs et de confirmation d'erreurs ont été introduites.

[2] La publication de la version 4.0 du Manuel mesure, incorporant le Glossaire des termes, signifie que le document v3.0.1 'Documentation Overview and Glossary of Terms', est maintenant obsolète. Aussi le document v3.0 'Advanced & Related Topics' sera remplacé par deux lignes directrices par 'Approximate COSMIC Functional Size Measurement' [6] and on 'Convertibility' [13], qui sont actuellement en cours d'élaboration.

- Le glossaire des termes a été mis à jour et inclut dans le Manuel de mesurage.

Les raisons des changements, qui ont été publiés précédemment dans différents MUBs, sont discutées pour chaque MUB. (Les bulletins de mise à jour de la méthode (MUB) sont publiés sur le site www.cosmicon.com entre les versions du Manuel de mesurage pour décrire les importantes modifications proposées à la méthode de mesurage.)

La structure de base de ce Manuel de mesurage est identique à celle de la version 3.0.1, sauf que l'article 4.1 de la version 3.0.1 a été déplacé du Chapitre 4 'Phase du mesurage' à la section 3.5 du chapitre 3 'Cartographie de la phase' où son appartenance est plus logique.

Enfin, de nombreuses améliorations rédactionnelles ont été apportées pour améliorer la compréhension, y compris le recours aux diagrammes. Quelques règles, désormais considérées comme inutiles, ont été supprimées ou leur formulation simplifiée.

Tous ces changements sont résumés à l'annexe E. Les utilisateurs de la méthode COSMIC, familiers avec les versions 3.0.1 ou avec des versions antérieures, qui ont l'intention de passer à la version 4, sont priés de lire en premier l'annexe E pour obtenir un aperçu des changements à la méthode. COSMIC prévoit que les améliorations introduites dans cette version 4.0 de la méthode seront présentées à l'ISO, pour être inclus dans l'ISO/IEC 19761, lors de la prochaine révision de la norme.

Ce Manuel de mesurage pour la version 4.0 de la méthode de mesurage est devenue la définition normalisée de la méthode COSMIC à compter d'avril 2014. Le Comité de pratique des mesures s'est efforcé d'éliminer les défauts et rendre le document plus facile à comprendre. Il reste cependant possible qu'il y ait encore des défauts et/ou certaines parties de ce texte soient toujours difficiles à comprendre. Nous demandons aux lecteurs et traducteurs de communiquer avec le MPC pour signaler tout défaut et/ou tout texte resté obscur, en suivant la procédure de l'annexe F. Si des changements importants sont nécessaires, le MPC produira une version 4.0.1 au début de 2015 avec toutes les corrections (comme ça été fait après la publication de la version 3.0 de la méthode).

Conséquences des principaux changements de la version 4.0 sur les règles du mesurage, etc.

Les principes de base à l'origine de la méthode COSMIC sont restés inchangés depuis leur première publication dans le premier projet du Manuel de mesurage en 1999. Ceci en dépit des diverses améliorations et des ajouts nécessaires pour produire la norme internationale et pour produire toutes les versions de la méthode, y compris cette dernière version (4.0).

La taille fonctionnelle, selon les principes et les règles du mesurage de la version 4.0 du Manuel de mesurage, peut différer des tailles mesurées en utilisant des versions antérieures seulement parce que les nouvelles règles visent à être plus précises et complètes. Ainsi les mesureurs auront moins de latitude pour interpréter de façon personnelle certaines règles, ce qui était possible avec les versions antérieures.

À titre d'indication supplémentaire sur la continuité de la méthode, ceux qui ont passé l'examen de certification niveau intermédiaire pour version 3.0/3.0.1 de la méthode seront toujours considérés comme étant certifiés pour la version 4.0 de la méthode (Foundation level).

Note sur la terminologie

Ce Manuel de mesurage utilise la terminologie normalisée d'ISO, à savoir

• 'doit' ('shall') indique une règle obligatoire; 'devrait' ('should') indique une règle consultative (si aucun terme est présent assumer 'doit')

• 'peut' ('may') signifie 'est autorisé à', 'pourrait' ('can') signifie «est capable de»

Le contenu de ce Manuel

Le chapitre 1 traite les types de logiciels pour lesquels la méthode COSMIC peut être utilisée. Le terme 'fonctionnalité utilisateur requise' (FUR), est défini, ainsi que les principes de base de la méthode COSMIC. Le processus du mesurage de la méthode COSMIC et l'unité de mesurage sont également définis.

Le chapitre 2 décrit la première phase de la stratégie du mesurage du processus de mesurage en utilisant les termes de paramètres clés, tels que le but du mesurage, la portée du mesurage et les utilisateurs fonctionnels d'un morceau du logiciel. Ces paramètres doivent être définis avant de commencer le mesurage afin que les résultats des mesurages puissent être acceptés et compris.

Le chapitre 3 traite de la deuxième phase de la cartographie du processus de mesurage en définissant comment le FUR doit être mappé avec les processus fonctionnels et les mouvements de données. Un mouvement de données déplace un groupe de données, consistant en données d'attributs, décrit comme un 'objet d'intérêt'. Les quatre types de mouvements de données sont: les entrées et sorties qui déplacent des données vers/depuis les utilisateurs fonctionnels et les lectures et écritures qui déplacent des données vers/depuis un stockage persistant.

Le chapitre 4 décrit la phase de mesurage finale du processus de mesurage. Il définit les règles d'affectation de la taille du FUR d'un morceau du logiciel et indique comment calculer la taille des différents morceaux du logiciel. Ce chapitre traite également de la façon du mesurage la taille des changements du logiciel et examine la possibilité d'extensions 'locales' à la méthode COSMIC.

Le chapitre 5 répertorie les paramètres qui devraient être considérés pour l'enregistrement du mesurage.

Le Common Software Measurement International Consortium (COSMIC)

COSMIC est un organisme bénévole sans but lucratif d'experts en génie logiciel provenant de partout dans le monde et fondée en 1998. Toutes ses publications sont «ouvertes» et disponibles pour distributions gratuites et soumises aux droits d'auteur et de reconnaissance des restrictions. Pour en savoir plus sur COSMIC et son organisation voir le site www.cosmicon.com.

Le 'COSMIC Measurement Practices Committee'

Avril 2014

Table des matières

1

INTRODUCTION

1.0 Sommaire du chapitre

Ce chapitre a quatre (4) objectifs:

- Expliquer les types de logiciels (domaines applicables) pour lesquels la méthode COSMIC peut être utilisée, et les limites imposées à leur utilisation.
- Définir les 'Fonctionnalités Utilisateurs Requises' (FURs), c'est à dire les exigences fonctionnelles du logiciel que la méthode COSMIC vise à mesurer. Nous expliquons en termes généraux comment un mesureur peut extraire ou dériver des FURs des artéfacts logiciels disponibles pour un mesurage de la taille fonctionnelle. Les exigences non-fonctionnelles (ENF) sont également définies, car les exigences qui sont d'abord exprimés en non-fonctionnelles évoluent souvent, au fur et à mesure de la progression du projet logiciel, partiellement ou totalement des FURs qui peuvent également être mesurées.
- Définir les principes de base de la méthode COSMIC tel que résumé par les deux modèles. Le 'Modèle de contexte du logiciel' est utilisé pour caractériser un morceau de logiciel à mesurer. Le 'Modèle générique du logiciel' définit les principes clés du modèle COSMIC des FURs dont la taille fonctionnelle doit être mesurée.
- Définir le processus du mesurage de la méthode COSMIC et le principe du mesurage (en lien avec l'unité du mesurage).

1.1 Applicabilité de la méthode COSMIC

La méthode COSMIC a été conçue pour être applicable au mesurage de la fonctionnalité du logiciel à partir des domaines suivants:

- Les logiciels d'application d'affaires sont typiquement utiles pour faciliter l'administration des affaires, tels que les banques, l'assurance, la comptabilité, le personnel, les achats, la distribution, la fabrication, etc. Ce type de logiciel est souvent caractérisé comme ayant des 'données riches' (manipulation). Il est largement dominé par la nécessité de gérer de grandes quantités de données sur les évènements et les objets dans le monde réel.
- Le logiciel en temps réel, dont la tâche est de suivre ou de contrôler des évènements qui se déroulent dans le monde réel. Les logiciels d'échanges téléphoniques et de commutation des messages, le logiciel embarqué dans les appareils à commander des machines telles que les appareils ménagers, les ascenseurs, les voitures et les avions, pour le contrôle des processus et d'acquisition automatique de données ainsi que le logiciel dans le système d'exploitation des ordinateurs.
- Les logiciels d'infrastructure qui supportent les types de logiciels qui précèdent, comme les morceaux réutilisables, les pilotes et les périphériques et ainsi de suite.
- Certains types de logiciels scientifiques et/ou d'ingénierie.

1.2 Fonctionnalités Utilisateurs Requises (FUR)

La Méthode de mesurage COSMIC implique qu'un ensemble de modèles, de principes, de règles et de processus soient appliqués aux Fonctionnalités Utilisateur Requises (FUR) d'un morceau de logiciel donné. Selon la Méthode COSMIC, le résultat est une 'valeur numérique d'une quantité' (tel que défini par ISO) représentant la taille fonctionnelle du morceau de logiciel.

Les FURs sont définis par ISO [1] de la façon suivante

DEFINITION – Fonctionnalités Utilisateur Requises (FUR)
Un sous-ensemble des besoins des utilisateurs. Des exigences qui décrivent ce que le logiciel doit faire, en termes de tâches et de services. Remarque: Les besoins des utilisateurs fonctionnels concernent mais ne sont pas limités: • au transfert de données (par exemple les données client en entrée, envoyer un signal de commande) • à la transformation de données (par exemple le calcul des intérêts de la banque, dériver la température moyenne) • au stockage de données (par exemple commande client stockée, enregistrement de la température ambiante dans le temps) • à l'extraction de données (par exemple la liste des employés actuels, la récupération de la dernière position d'un avion) Les exemples de besoins des utilisateurs qui ne sont pas des FURs incluent, mais ne sont pas limités aux : • contraintes de qualité (par exemple la facilité d'utilisation, la fiabilité, l'efficacité et la portabilité) • contraintes organisationnelles (par exemple les lieux d'exploitation, le matériel cible et la conformité aux normes) • contraintes environnementales (par exemple l'interopérabilité, la sécurité, la confidentialité et la sécurité) • contraintes de mise en œuvre (par exemple la langue de développement et le calendrier de livraison)

Remarque: La méthode COSMIC reconnait que certains types d'exigences (par exemple, la qualité et les contraintes environnementales, telles que mentionnées ci-dessus) peuvent être exprimées au début de la vie d'un projet de logiciel comme 'non-fonctionnels', selon la définition ISO , mais peuvent évoluer au fur et à mesure de l'avancement du projet en FUR (Fonctionnalités Utilisateurs Requises). Voir la section 1.2.3 de ce Manuel de mesurage.

Dans ce Manuel de mesurage, nous n'utiliserons que le terme 'FUR' pour rendre compte des exigences fonctionnelles des utilisateurs qui:

- sont dérivées des artéfacts logiciels disponibles (exigences, dessins, objets physiques et autres)
- sont ajustées, si nécessaire, par des hypothèses pour surmonter les incertitudes dans les objets disponibles,
- contiennent toutes les informations nécessaires pour un mesurage fonctionnel (FSM) COSMIC.

Sinon, nous allons utiliser les termes 'besoins actuels', 'objets physiques' ou autres termes appropriés au contexte.

Les mesures fonctionnelles (FSM) par la méthode COSMIC sont conçues pour dépendre uniquement des fonctionnalités des utilisateurs (FUR) du logiciel à mesurer et être indépendantes de toutes les exigences ou contraintes relatives à la mise en œuvre de la FUR. 'Fonctionnalité' peut être librement défini comme 'le traitement de l'information que le logiciel doit effectuer pour ses utilisateurs'.

1.2.1 Extraire, dans la pratique, les FURs à partir d'artéfacts logiciels

En pratique, lors du développement de logiciels, il est rare de trouver les artéfacts logiciels pour lesquels les FURs se distinguent clairement des autres types d'exigences ou sont exprimés directement dans une forme appropriée à la mesure. Ceci signifie que le mesureur doit habituellement extraire les FURs des exigences existantes ou implicites et des artéfacts logiciels fournis, avant de les arrimer aux concepts de modèle de logiciel COSMIC.

Tels qu'illustrés à la figure 1.1 ci-dessous, les FURs peuvent être dérivées des artéfacts du génie logiciel réalisés avant même l'existence du logiciel (typiquement des artéfacts de l'architecture et de la conception). Ainsi, la taille fonctionnelle d'un logiciel peut être mesurée avant que le logiciel ne soit implanté.

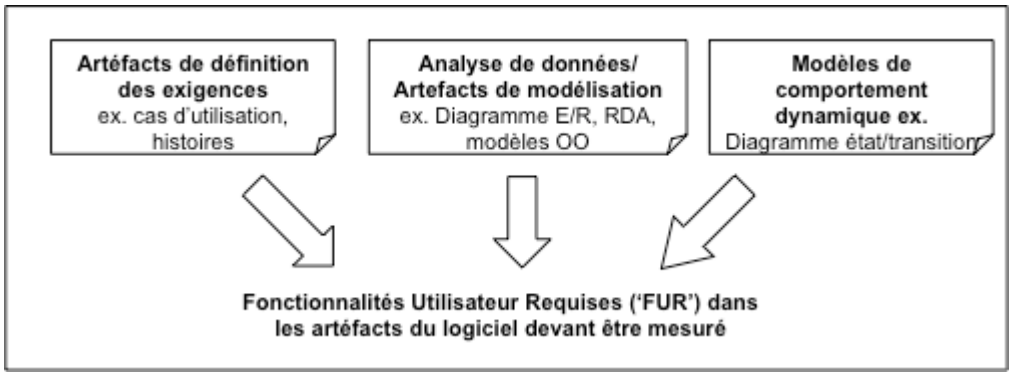


Figure 1.1 – Le modèle COSMIC avant l'implantation des FURs

REMARQUE : Les exigences fonctionnelles des utilisateurs peuvent être produites avant qu'elles soient assignées au matériel ou au logiciel. Puisque la Méthode COSMIC vise à établir la taille de la FUR d'un morceau d'un logiciel, seulement les FUR assignées au logiciel sont mesurées. Cependant, en principe, la Méthode COSMIC peut être appliquée à la FUR avant que les FURs soient assignées au logiciel ou au matériel, et ce, indépendamment de la décision éventuelle d'attribution. Par exemple, il est possible de déterminer assez directement les fonctionnalités d'une calculette avec COSMIC sans qu'il soit nécessaire de connaitre le matériel ou le logiciel (si présent) visé. Cependant, pour que la Méthode COSMIC puisse être utilisée pour évaluer la taille des FURs assignées au matériel et qu'elle soit considérée comme totalement valide, il faut davantage d'essais pratiques, sans qu'il soit nécessaire d'appliquer plus de règles.

Dans d'autres circonstances, une certaine partie du logiciel existant peut devoir être mesurée sans qu'il y ait, ou presque pas, d'artéfacts logiciel disponibles, tant d'architecture que de conception, avec une FUR qui pourrait ne pas être documentée (par exemple, pour des anciennes applications). Dans de telles circonstances, il est encore possible de dériver les FUR des artéfacts installés tel qu'illustré à la figure 1.2 ci-dessous.

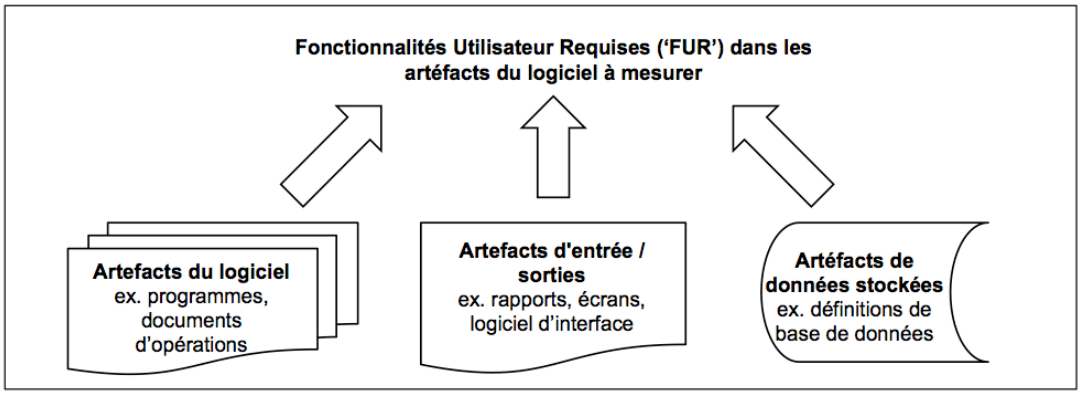


Figure 1.2 – Le modèle COSMIC après l'implantation des FURs

1.2.2 Le processus d'extraction des FURs des artéfacts logiciels

Le processus de fabrication utilisé et l'effort requis pour extraire les FURs de différents types d'artéfacts de génie logiciel, ou encore de les dériver d'un logiciel installé, varieront énormément ; ces processus ne peuvent pas être traités dans ce Manuel. La méthode suppose que les exigences fonctionnelles de l'utilisateur du logiciel à mesurer, sont existantes ou encore qu'elles peuvent être extraites ou dérivées de ses artéfacts, à la lumière de la raison d'être du mesurage.

Le Manuel de mesurage se limite donc à définir et décrire les concepts des modèles COSMIC (le 'modèle de contexte du logiciel' et le 'modèle générique de logiciel' – Voir la section 1.3) et comment les appliquer afin du mesurage la FUR d'un morceau de logiciel[3].

Si le mesureur comprend vraiment ces deux modèles, il sera toujours possible de dériver la FUR d'un morceau de logiciel à mesurer de ses artéfacts présents, bien que le mesureur puisse avoir à faire certaines hypothèses en raison d'informations manquantes ou peu claires.

1.2.3 Exigences non fonctionnelles

La définition normalisée d'ISO pour la FUR répertorie les différents types de 'besoins des utilisateurs' qui ne sont pas des FURs. Par voie de conséquence, ce sont des Exigences Non Fonctionnelles (ENF).

Les ENFs peuvent être très importantes pour un projet de logiciel. Dans les cas extrêmes, un énoncé des exigences pour un système de logiciels peut exiger autant de documentation pour les ENFs que pour la FUR. Mais la distinction entre ENF et FUR n'est pas aussi simple que la définition de FUR telle qu'elle apparait dans ISO. La méthode COSMIC peut servir à mesurer certaines exigences qui sont d'abord définies comme non fonctionnelles. Nous devons donc commencer par définir ENF:

DÉFINITION – Exigence non fonctionnelle (ENF)
Toute exigence ou contrainte sur un système (équipement/logiciel) ou sur un produit logiciel, ou sur un projet pour développer ou maintenir ce système ou produit, sauf une exigence fonctionnelle (FUR) pour un logiciel. Remarque : Les exigences système ou logiciel, initialement exprimés comme non fonctionnelles, peuvent évoluer, en tout ou en partie, au fur et à mesure que le projet progresse, en FUR pour le logiciel.

[3] Diverses directives COSMIC, par exemple pour mesurer la taille des logiciels d'application d'affaires [7] et encore pour le mesurage de la taille des logiciels en temps réel [4], donnent des conseils sur l'arrimage des données d'analyse et des exigences utilisées dans le domaine des applications d'affaires sur les concepts COSMIC.

Plusieurs études [3] ont montré que certaines exigences qui apparaissent initialement comme ENF dans un système évoluent, lorsque le projet progresse, en un mélange d'exigences qui peuvent être implémentées dans les fonctions du logiciel, et d'autres exigences ou contraintes qui ne peuvent être implémentées comme tel et sont vraiment 'non fonctionnelles' (voir la Figure 1.3). Cela est vrai pour beaucoup de contraintes environnementales et de qualité. Une fois identifiées, ces fonctions logicielles, qui ont été « cachées » en tant que ENF au début d'un projet, peuvent être redimensionnées en utilisant la méthode COSMIC comme faisant partie des autres fonctions du logiciel. La non-reconnaissance, au début du projet, de cette taille fonctionnelle « cachée » est une des raisons pour laquelle la taille d'un logiciel peut paraitre augmenter au fur et à mesure de la progression d'un projet.

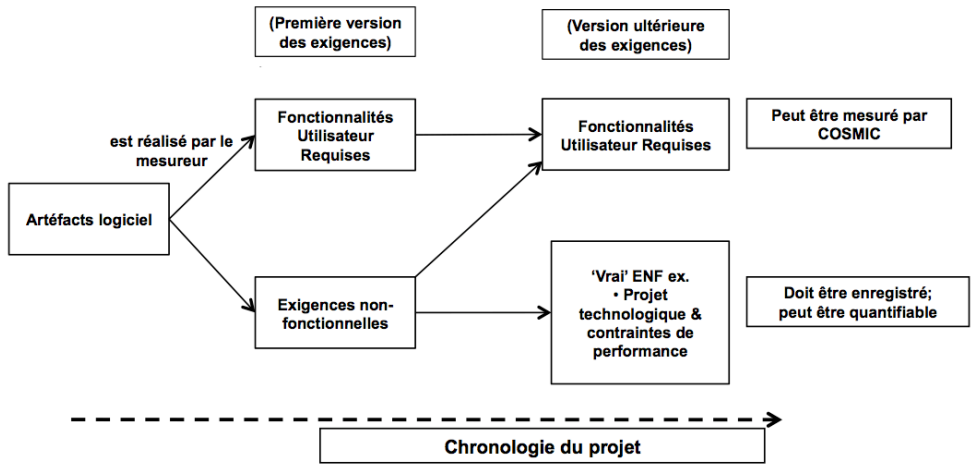


Figure 1.3 – De nombreuses exigences apparaissent comme non fonctionnelles mais évoluent en FUR au fur et à mesure de la progression du projet

EXEMPLE D'AFFAIRES : Les exigences pour un nouveau système logiciel sont déclarées de la façon suivante: 'l'utilisateur a la possibilité de sécuriser les fichiers de cryptage'. Le projet pour développer le système est au stade de l'estimation des efforts et des couts. Deux options sont envisagées :

- *Développer un logiciel de chiffrement propriétaire. Aux fins d'estimation du projet, il peut être nécessaire du mesurage la taille de la FUR pour le logiciel de cryptage.*
- *Acheter un progiciel (COTS) existant. Aux fins d'estimation du projet, il peut être nécessaire du mesurage seulement la taille des fonctions nécessaire pour intégrer le progiciel (COTS) au logiciel désiré. Le cout du progiciel et les efforts pour intégrer et tester le progiciel de chiffrement des fichiers devront également être pris en considération lors de l'estimation de couts du projet.*

EN TEMPS-RÉEL: la tolérance aux pannes sur les systèmes aérospatiaux est obtenue principalement grâce à une combinaison de redondance et de sauvegarde des systèmes physiques. Une fonction comme la surveillance de moteur est copiée sur trois ou quatre ordinateurs embarqués distincts. Cette fonction a une contrainte de calendrier strict énoncée comme ENF: 'chaque ordinateur distinct doit répondre dans un délai précis. Si l'un des ordinateurs répond à plusieurs reprises au-delà du délai imparti ou si ses résultats sont en désaccord avec les autres, l'ordinateur doit être mis hors circuit' (devient alors une exigence fonctionnelle). Une exigence qui est initialement déclarée comme non fonctionnelle évolue donc en FUR mesurable. Le mécanisme de synchronisation peut également être implémenté en partie dans le logiciel et cette fonctionnalité est également mesurable (voir par exemple 'Guideline for sizing real-time software' [4], section 3.2).

Pour plus d'exemples, consultez l'annexe B. Tous ces exemples montrent que, lorsqu'il est nécessaire du mesurage la taille de certains logiciels au début de la vie d'un projet, il est important d'examiner si certaines ENFs pourraient se transformer en FUR et si la taille de ces FURs doit aussi être mesurée.

1.3 Les principes fondamentaux de la méthode COSMIC

La méthode COSMIC est basée sur les principes fondamentaux du génie logiciel. Ces principes sont résumés dans les deux modèles suivants:

De la même manière qu'une maison peut avoir plusieurs tailles selon ce que vous souhaitez mesurer, la taille d'un morceau de logiciel peut être mesurée de plusieurs façons, même en utilisant la même unité du mesurage. Les principes du 'Modèle contextuel de logiciel' COSMIC permettent au mesureur de définir le logiciel à mesurer et le mesurage de la taille. Ceci permet de s'assurer que les résultats peuvent être compris et interprétés de façon cohérente par les futurs utilisateurs.

Les principes du 'Modèle générique de logiciel' COSMIC indiquent comment les FURs du logiciel sont modélisés afin qu'ils puissent être mesurés.

La raison principale pour inclure ces deux modèles à ce stade dans le Manuel de mesurage est de montrer comment la méthode COSMIC est fondamentalement très simple. Nous devrons également faire référence aux deux modèles plus loin dans le manuel. Toutefois, un mesureur novice ne peut s'attendre à être capable, à la simple lecture de ces deux modèles, du mesurage la taille avec précision. Pour appliquer les modèles à une situation du mesurage particulière, le mesureur aura besoin des définitions des différents concepts et principes, règles, explications et exemples supplémentaires donnés dans ce Manuel.

N.B.: les termes qui figurent en caractère gras dans les sections 1.3.1 et 1.3.2 peuvent être spécifiques à la méthode COSMIC. Pour les définitions officielles, consultez le glossaire à la fin de ce Manuel de mesurage. Les références fournies avec chaque principe se retrouvent dans les sections du Manuel où le sujet est traité en détail.

1.3.1 Le modèle contextuel de logiciel COSMIC

PRINCIPES – Le modèle contextuel de logiciel COSMIC	Section
a) Le logiciel est limité par le matériel ;	-
b) Le logiciel est typiquement structuré en **couches**;	2.2.2
c) Une couche peut contenir un ou plusieurs morceaux de logiciel, de **même niveau**;	2.2.2 2.2
d) Tout morceau de logiciel à mesurer doit être défini par un **périmètre de mesurage**, lequel doit aussi être entièrement compris à l'intérieur d'une seule couche de logiciel ;	2.1
e) Le périmètre du morceau de logiciel à mesurer doit dépendre de la **raison d'être** du mesurage ;	2.3
f) Les **utilisateurs fonctionnels** doivent être identifiés à partir des Fonctionnalités Utilisateurs Requises (FUR) du morceau de logiciel à mesurer comme les émetteurs et/ou les destinataires visés pour les données ;	2.4
g) La FUR d'un logiciel peut être exprimée à différents **niveaux de granularité** ;	2.4.3
h) Une mesure précise de la taille d'un morceau du logiciel exige que le niveau de granularité de la FUR soit celui ou le processus ou sous-processus fonctionnel peut être identifié;	
i) Une mesure approximative COSMIC de la taille d'un morceau de logiciel est possible si la FUR est mesurée à un haut niveau de granularité par une approche approximative et mise à l'échelle au niveau de granularité des processus et sous-processus fonctionnels.	2.4.3

1.3.2 Le modèle générique de logiciel COSMIC

Après avoir identifié et défini les FURs du logiciel à mesurer en fonction du modèle contextuel de logiciel, nous allons maintenant appliquer le modèle générique de logiciel aux FURs pour identifier les morceaux de la fonctionnalité à mesurer. Ce Modèle Générique sous-entend que les principes généraux suivants sont valables pour toute mesure du logiciel utilisant cette méthode. (Comme indiqué dans le Glossaire, toute méthode de mesurage de la taille fonctionnelle vise à identifier les 'types' et non les 'occurrences' des données ou des fonctions. Dans le Manuel de mesurage, le suffixe 'type' va donc être omis lorsque mentionnant les concepts de base COSMIC, sauf s'il est essentiel de distinguer 'type' des 'évènements'.)

PRINCIPES – Le modèle générique de logiciel COSMIC	Section
a) Un morceau de logiciel interagit avec ses utilisateurs fonctionnels par-delà la **frontière**, et un **stockage persistant** à l'intérieur de sa frontière.	2.3
b) Les exigences fonctionnelles de l'utilisateur d'un morceau de logiciel à mesurer peuvent être arrimées dans des processus fonctionnels uniques.	3.2
c) Chaque processus fonctionnel comprend des sous-processus.	3.2
d) Un sous-processus peut être soit un **mouvement de données** ou une **manipulation de données**.	3.2 3.3
e) Un mouvement de données déplace un seul groupe de données.	
f) Il y a quatre types de mouvement de données, **entrée**, **sortie**, **écriture** et **lecture**. Une entrée déplace un groupe de données d'un processus fonctionnel à partir d'utilisateur fonctionnel. Une sortie déplace un groupe de données d'un processus fonctionnel vers un utilisateur fonctionnel. Une écriture déplace un groupe de données d'un processus fonctionnel vers un stockage persistant. Une lecture déplace un groupe de données d'un stockage persistant vers un processus fonctionnel.	3.5 3.4
g) Un groupe de données consiste en un ensemble unique **d'attributs de**	3.2

données décrivant un seul **objet d'intérêt.** h) Chaque processus fonctionnel est démarré par le **déclenchement en entrée** d'un mouvement de données. Le groupe de données déplacé par l'entrée de déclenchement est généré par un utilisateur fonctionnel en réponse à un **évènement déclencheur**. i) Un processus fonctionnel doit comprendre au moins un déplacement de données d'entrée et une écriture ou un mouvement de données de sortie, c'est-à-dire qu'il doit comprendre un minimum de deux mouvements de données. Théoriquement, il n'y a pas de limite supérieure pour le nombre de mouvements de données dans un processus fonctionnel. j) En tant qu'approximation pour les objectifs du mesurage, les manipulations de données des sous-processus ne sont pas mesurées séparément ; la fonctionnalité de toute manipulation de données est censée tenir compte du déplacement de données auquel il est associé.	3.5

1.4 Le processus du mesurage COSMIC et l'unité du mesurage

Le processus du mesurage COSMIC comprend trois phases:

- La phase de stratégie du mesurage, ou le but et la portée du mesurage sont définis. Le modèle contextuel de logiciel est ensuite appliqué pour que le logiciel à mesurer et le mesurage requis soient définis de façon unique. (Chapitre 2).
- La phase d'arrimage, dans laquelle le modèle générique de logiciel est appliqué aux FURs du logiciel à mesurer pour produire le modèle de logiciel COSMIC qui peut être mesuré. (Chapitre 3).
- La phase du mesurage, où la taille actuelle est mesurée. (Chapitre 4)

Les règles sur la façon d'enregistrer les mesures sont inscrites au chapitre 5

.

Les relations entre les trois phases de la méthode COSMIC sont montrées à la Figure 1.4.

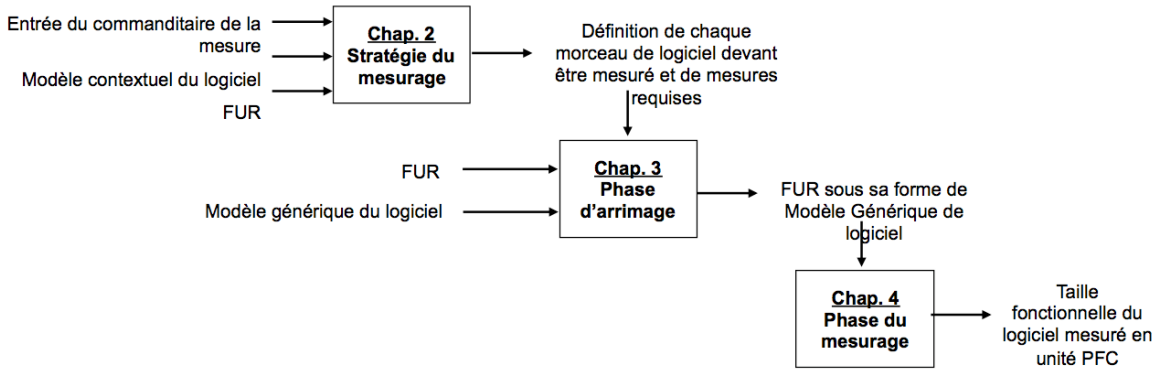


Figure 1.4 – Le processus de la méthode de mesurage COSMIC

L'unité du mesurage COSMIC (PFC) et les principes du mesurage sont définis de la façon suivante.

DÉFINITION – Unité du mesurage COSMIC
1 PFC (Point de Fonction COSMIC), est la taille d'un mouvement de données.

PRINCIPE – Le principe du mesurage COSMIC
a) La taille d'un processus fonctionnel est égale au nombre de ses mouvements de données.
b) La taille fonctionnelle d'un morceau de logiciel, pour une portée définie, est égale à la somme des tailles de ses processus fonctionnels.

Les tailles des modifications nécessaires à un morceau de logiciel sont évaluées comme suit :

- la taille de n'importe quel mouvement de données touchées (qui doivent être ajoutés, modifiés ou supprimés) par le changement requis est par convention 1 PFC.
- La taille des changements nécessaires à un morceau de logiciel est égale au nombre de mouvements de données qui sont touchés par les modifications requises.
- La taille minimale d'un changement à un morceau de logiciel est 1 PFC.

D'autres règles et directives sur le mesurage et l'agrégation des mesures sont données dans les sections 4.1 à 4.4 du présent Manuel de mesurage.

1.5 Limites de l'applicabilité de la méthode COSMIC

Voir la section 4.5 pour d'éventuelles limitations de la méthode et comment il serait possible d'étendre la méthode localement pour surmonter les limitations de COSMIC.

2

PHASE DE STRATÉGIE DU MESURAGE

2.0 Sommaire du chapitre

Ce chapitre aborde les paramètres clefs qui doivent être considérés dans la phase 'Stratégie du mesurage' du processus du mesurage avant de commencer le mesurage. Ce sont (en italique):

- La raison d'être du mesurage, i.e. à quoi servira le résultat. La raison d'être du mesurage détermine les autres paramètres du mesurage.
- Le périmètre du mesurage du logiciel à mesurer et, si le logiciel est constitué de plusieurs parties qui devraient être mesurées séparément (par exemple, les composants d'un système de logiciels distribués), selon le périmètre de chaque composant de logiciel. Il faudra également déterminer dans quelle couche se situe chaque morceau de logiciel et peut-être aussi déterminer le niveau de décomposition des morceaux du logiciel à mesurer.
- Les utilisateurs fonctionnels de chaque composant de logiciel à mesurer. Ce sont les expéditeurs et les destinataires des données depuis/vers le logiciel à mesurer; ils peuvent être humains, matériels ou autres composants du logiciel. Comme différents utilisateurs fonctionnels peuvent 'voir' différents ensembles de la même fonctionnalité, la taille fonctionnelle varie avec le choix des utilisateurs fonctionnels.
- Le niveau de granularité des artéfacts disponibles du logiciel à mesurer. Par exemple, lorsque que l'énoncé des besoins n'est pas défini avec tous les détails nécessaires pour un mesurage COSMIC précise, nous devons décider comment dériver les FUR à mesurer et/ou utiliser une variante de la taille approximative.

La détermination de ces paramètres permet de répondre aux questions: 'quelle taille doit être mesurée', ' quelle précision donner à la mesure', etc. Les paramètres d'enregistrement permettent aux futurs utilisateurs d'un mesurage de décider comment interpréter le mesurage.

Il est important de noter que ces paramètres et les concepts connexes ne sont pas spécifiques à la méthode COSMIC, mais doivent être communs à toutes les méthodes du mesurage fonctionnel du logiciel. Les autres méthodes du mesurage fonctionnel du logiciel peuvent ne pas distinguer différents types d'utilisateurs fonctionnels et peuvent ne pas discuter différents niveaux de granularité, etc. La méthode COSMIC requiert ces paramètres afin de permettre une plus grande applicabilité et souplesse de la méthode, ce qui nécessite une plus grande réflexion que les autres méthodes du mesurage fonctionnel du logiciel.

Il est très important d'enregistrer les données découlant de cette phase de stratégie du mesurage (voir la section 5.2) lorsque vous enregistrez le résultat d'un mesurage, quel qu'il soit. L'omission de définir et d'enregistrer ces paramètres de façon cohérente conduira à des mesures qui ne peuvent pas être interprétées de manière fiable et pouvant être comparées, ou être utilisé avec fiabilité comme entrée pour des processus, tels l'estimation de projet.

Les sections du présent chapitre donnent les définitions officielles, les principes, les règles et les exemples pour chacun des paramètres clés pour aider le mesureur à travers le processus consistant à déterminer une stratégie du mesurage, comme le montre la Figure 2.0 ci-dessous.

Chaque section donne des explications fondamentales sur les raisons pour lesquelles les paramètres clés sont importants. Des analogies sont utilisées pour montrer pourquoi les paramètres sont pris pour acquis dans d'autres domaines du mesurage et doivent donc également être pris en considération dans le domaine du mesurage fonctionnel du logiciel.

Remarque: selon la Figure 2.0 la détermination des paramètres de stratégie du mesurage peut nécessiter quelques itérations.

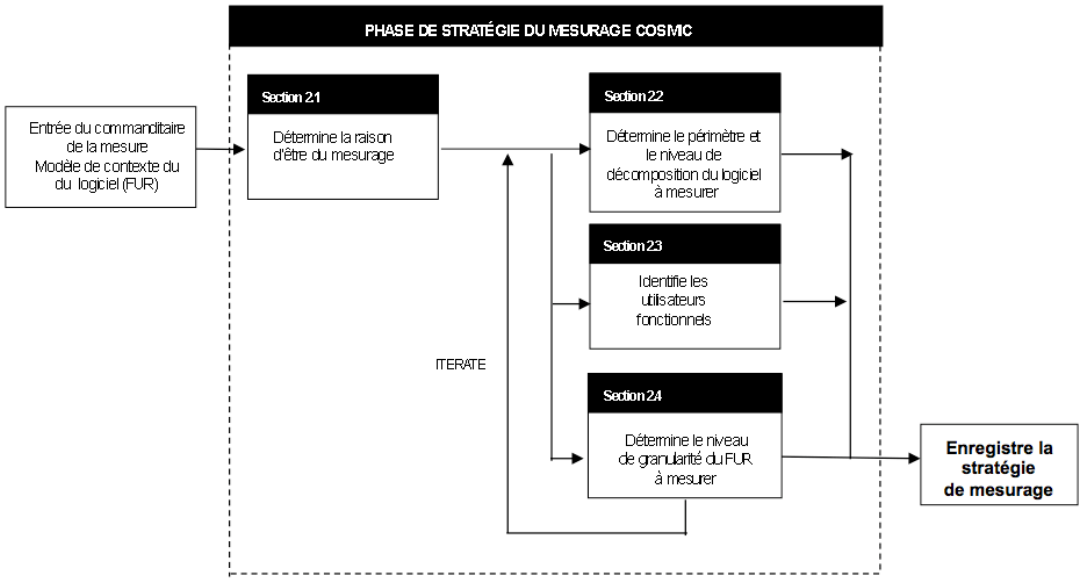


Figure 2.0 – Le processus de détermination de la stratégie du mesurage

Modèles de stratégie du mesurage

Le Guide 'Measurement Strategy Patterns'[5] est une aide à la détermination d'une stratégie du mesurage. Il décrit, pour chacun des différents types de logiciels, un ensemble de paramètres normalisés pour mesurer la taille du logiciel, appelé un "modèle de stratégie du mesurage" (aussi dénommé 'modèle du mesurage'.

DÉFINITION – Modèle (de stratégie) du mesurage
Un modèle normalisé qui peut-être être appliqué lors du mesurage d'un morceau de logiciel provenant d'un domaine fonctionnel du logiciel donné, qui définit les types d'utilisateurs fonctionnels et qui peut interagir avec le logiciel, le niveau de décomposition du logiciel et les types de mouvements de données que le logiciel peut manipuler.

L'utilisation cohérente des mêmes modèles du mesurage devrait aider les mesureurs à s'assurer que les mesures effectuées, pour la même raison d'être, sont faites de manière cohérente, peuvent être comparées en toute sécurité avec les autres mesures effectuées à l'aide du même modèle et seront correctement interprétées pour toutes les utilisations futures. Un autre avantage de l'utilisation du modèle normalisé, c'est la réduction des efforts pour déterminer les paramètres de stratégie du mesurage. Nous recommandons fortement cette démarche aux mesureurs, en plus de maitriser la méthode COSMIC, notamment celles relatives aux paramètres de la stratégie du mesurage, avant d'utiliser les modèles normalisés.

2.1 Définir la raison d'être du mesurage.

DÉFINITION – Raison d'être du mesurage.
Un énoncé qui indique pourquoi une mesure est exigée, et comment le résultat sera employé.

2.1.1 La raison d'être du mesurage – une analogie

Il y a plusieurs raisons de vouloir mesurer la taille fonctionnelle d'un logiciel, tout comme il y a plusieurs raisons de vouloir mesurer par exemple, la superficie d'une maison. Dans les deux cas différentes raisons peuvent donner des résultats différents. Par analogie, la taille d'une maison peut bien varier selon:

- la raison et le moment du mesurage (par exemple en fonction de la nécessité du mesurage on peut utiliser les grandes lignes des spécifications pour fins de budgétisation, ou les plans de l'architecte pour un devis précis, ou la taille réelle pour planifier la réalisation des revêtements de sol),
- les artéfacts mesurés (par exemple, les plans de l'édifice).

Notez toutefois que les mêmes principes du mesurage et d'unité du mesurage sont utilisés pour tous les mesurages. De la même façon, la taille mesurée d'un morceau de logiciel peut bien varier avec :

- la raison et le moment du mesurage (par exemple, en fonction de la nécessité du mesurage avant le développement pour l'estimation des besoins, ou au cours du développement pour suivre la déviation de la raison d'être ou après l'installation pour mesurer le rendement du développeur),
- les artéfacts mesurés (par exemple, un énoncé des exigences ou les artéfacts physiques du logiciel).

Comme pour l'analogie, toutefois, les mêmes principes du mesurage et la même unité du mesurage sont utilisés pour tous les mesurages.

C'est au mesureur d'un morceau de logiciel de décider, selon le but du mesurage, quand mesurer (avant, pendant ou après le développement), ce qu'il faut mesurer (par exemple tous les logiciels livrés pour un projet, ou exclure les logiciels réutilisés) quels artéfacts utiliser pour dériver la FUR à mesurer (par exemple, un énoncé des exigences ou des logiciels installés).

Les exemples suivants sont des énoncés typiques de la raison d'être du mesurage:

- *Mesurer la taille de la FUR au fur et à mesure de son évolution, comme entrée du processus d'estimation du développement de l'effort.*
- *Mesurer la taille des modifications apportées à une FUR après qu'elles aient été initialement acceptées, afin de contrôler l'ampleur des 'déviations incontrôlées' dans un projet.*
- Mesurer la taille d'une FUR d'un logiciel livré pour l'utiliser comme entrée dans la mesure de la performance du développeur.
- Mesurer la taille de la FUR de la totalité du logiciel livré, de même que la taille de la FUR du logiciel qui a été développée, afin obtenir une mesure de réutilisation fonctionnelle.
- Mesurer la taille de la FUR d'un logiciel existant pour l'utiliser comme une entrée dans la mesure de la performance des responsables de maintenance et support de ce logiciel ;
- Mesurer la taille de la FUR de certaines modifications apportées à un logiciel existant comme mesure de la taille du travail d'une équipe de projet en maintenance ;
- Mesurer la taille de la fonctionnalité d'un logiciel existant fourni par des utilisateurs fonctionnels humains.

2.1.2 L'importance de la raison d'être

La raison d'être du mesurage est d'aider le mesureur à déterminer:

- Le périmètre du mesurage, y compris les artéfacts qui seront utiles pour la mesure ;
- L'utilisateur fonctionnel, La taille fonctionnelle change selon la définition de l'utilisateur fonctionnel (personne ou chose) tel qu'indiqué en 2.3 ;
- Le moment dans le cycle de vie du projet quand le mesurage aura lieu ;
- La précision qui sera demandée, si la Méthode COSMIC doit être employée, ou si une version d'approximation locale, tirée de la méthode de COSMIC doit plutôt être employée (par exemple au début d'un cycle de vie d'un projet, avant que la FUR ne soit entièrement élaborée).

Ces deux derniers points détermineront le niveau de granularité à laquelle on mesurera la FUR.

2.2 Définition du périmètre du mesurage

DÉFINITION – Périmètre du mesurage.
C'est l'ensemble des Fonctionnalités Utilisateur Requises (FUR) qui doivent être incluses dans une occurrence spécifique du mesurage de taille fonctionnelle. REMARQUE : une distinction doit être faite entre le 'périmètre global', i.e. tout le logiciel qui doit être mesuré selon la raison d'être du mesurage, et le 'périmètre, de chaque morceau de logiciel à l'intérieur de cette raison d'être globale, et pour lequel la taille doit être mesurée séparément. Dans le Manuel de mesurage, le terme 'périmètre' (ou l'expression 'périmètre du mesurage') sera relié à tout morceau de logiciel dont la taille doit être mesurée séparément.

Règles – Périmètre du mesurage
a) Le périmètre de tout morceau de logiciel à mesurer doit être dérivé de la raison d'être du mesurage. b) Le périmètre de tout mesurage ne doit pas couvrir plus d'une couche du logiciel à mesurer.

Voir la prochaine section pour des exemples d'un périmètre total et des périmètres de mesurages.

2.2.1 Déterminer le composant du logiciel mesuré à partir de sa raison d'être

Le logiciel, circonscrit par un périmètre global, peut être subdivisé en différents morceaux du logiciel chacun avec leur propre périmètre de mesurage défini de plusieurs façons, en fonction de la raison d'être du mesurage. Supposons qu'un périmètre global est défini comme 'le portefeuille de l'application d'organisation X' ou comme 'tous les morceaux de logiciels distribués par le projet Y'. Les subdivisions sont possibles en raison:

- d'un logiciel ayant plusieurs couches (voir règle b) ci-haut),
- des différentes responsabilités organisationnelles, par exemple en équipe client-groupe ou en sous-projet,
- de la nécessité de distinguer les différents livrables: pour la mesure du rendement, l'estimation de l'effort ou encore à des fins contractuelles.

La raison de ce dernier pourrait être due à la nécessité de distinguer différents périmètres de mesure pour les morceaux de logiciels qui :

- sont construits en utilisant des technologies différentes, c'est-à-dire en plate-forme matérielle, langage de programmation etc.,
- fonctionnent en différents modes, c'est-à-dire en ligne versus en différé,
- sont développés par opposition à 'livrés' (ce dernier comprenant l'implantation de produits-programmes ou autres logiciels réutilisables),
- sont à différents stades de décomposition, par exemple pour l'ensemble de l'application ou un composant majeur ou un composant mineur tel un 'objet réutilisable',
- sont les principaux livrables par opposition à un logiciel qui est utilisé une fois, par exemple pour la conversion de données avant d'être éliminés; ce n'est peut-être pas la peine de mesurer ce dernier,
- sont développées versus améliorés,

ainsi que toute autre combinaison de ces facteurs.

En résumé, l'objectif du mesurage doit toujours être utilisé pour déterminer (a) quel logiciel est inclus ou exclus du périmètre global et (b) de quelle façon le logiciel inclus devrait être divisée; en morceaux séparés, chacun avec sa propre portée, à mesurer séparément.

En pratique, l'énoncé de la raison d'être doit être explicite et non générique, par exemple les travaux développés par l'équipe du projet A ou du projet B, ou le portefeuille de l'entreprise C. L'énoncé de la raison d'être peut également, par souci de clarté, avoir besoin d'affirmer ce qui est exclu.

EXEMPLE d'affaires : la Figure 2.1 montre tous les morceaux séparés du logiciel – le périmètre d'ensemble - livré par l'équipe du projet :

- le client et les composants du serveur d'un progiciel mis en œuvre,
- un programme qui fournit une interface entre le composant serveur du nouveau progiciel et les applications existantes,
- un programme qui est utilisé une seule fois pour convertir des données existantes vers le nouveau format requis par le progiciel. Ce programme a été construit à l'aide d'un certain nombre d'objets réutilisables mis au point par l'équipe du projet
- le pilote du logiciel pour le nouveau matériel sur lequel s'exécute le composant du progiciel du client

(Chaque morceau du logiciel pour lequel a été définie un périmètre de mesurage apparait comme une boite rectangulaire solide)

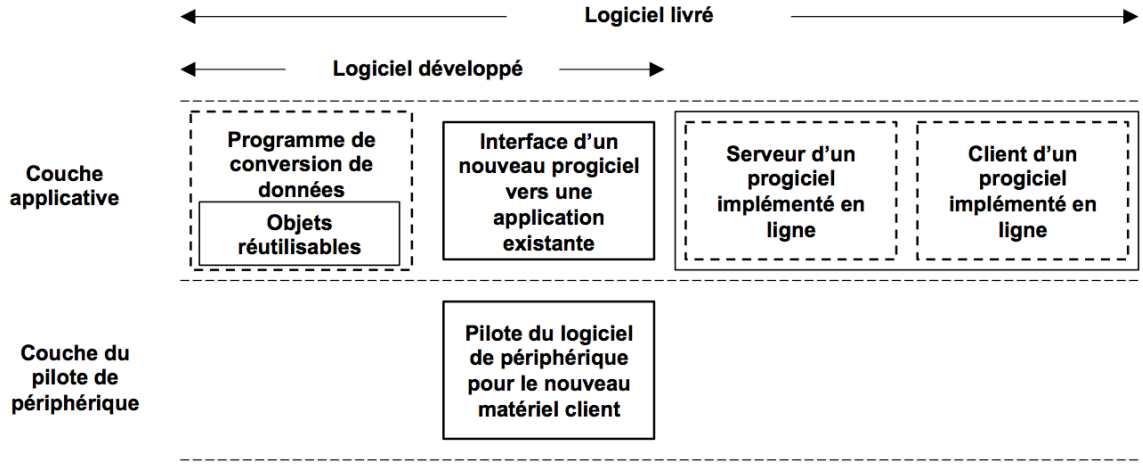


Figure 2.1 – Le périmètre d'ensemble des livrables d'un projet de logiciel et les périmètres du mesurage individuel

Le diagramme montre que les morceaux 'livrés' du logiciel consistent en composants récemment développés et en composants mis en place par l'équipe de projet.

La taille du progiciel mis en œuvre a été mesurée comme un 'tout', c'est-à-dire sans tenir compte de la structure du composant client-serveur. Cette taille a été ajoutée à celui du programme d'interface pour mettre à jour la taille totale du portefeuille d'applications de l'organisation. La taille du programme de conversion de données n'était pas comprise dans la mesure car il a été utilisé une seule fois et jeté. Mais la taille de chacun des objets réutilisables a été enregistrée dans l'inventaire du logiciel des infrastructures de l'organisation, ainsi que le nouveau pilote de périphérique. Encore une fois, elles ont été classées séparément.

En raison de la diversité des produits livrables, l'addition des tailles de tous les logiciels livrés n'est pas significative du mesurage de la performance de l'équipe projet. La performance des équipes qui a livré chaque morceau de logiciel a été mesurée séparément.

2.2.2 Les couches

Puisque le périmètre d'un morceau de logiciel à mesurer doit être confinée à une seule couche de logiciel, le processus de définition du périmètre peut exiger que le mesureur ait d'abord à décider ce que sont les couches de son système logiciel. Dans cette section nous définirons et discuterons des 'couches' du

logiciel, puisque ces termes sont employés dans la Méthode COSMIC. Les raisons pour lesquelles nous avons besoin de ces définitions et de ces règles sont les suivantes:

- Le mesureur peut avoir à mesurer un logiciel dans un environnement 'hérité' ou 'patrimonial'[4] qui a depuis évolué, et ce, sans jamais avoir été conçu selon un design d'architecture prédéfinie (appelé de façon triviale 'architecture spaghetti'). Le mesureur peut donc avoir besoin de conseils sur la manière de distinguer des couches selon la terminologie COSMIC ;
- Les expressions 'couche', 'architecture en couche' et 'composant de même niveau' ne sont pas employées uniformément dans l'industrie du logiciel. Si le mesureur doit mesurer un logiciel décrit comme appartenant à une 'architecture en couche', il est recommandé de vérifier que les 'couches' de cette architecture soient définies de manière compatible à la définition de celle proposée par la Méthode COSMIC. À cette fin, le mesureur devra établir des équivalences entre, d'une part, les objets architecturaux spécifiques de son 'architecture en couche' et le concept de 'couches', tel que défini dans ce manuel.

Les couches peuvent être identifiées selon les définitions et principes suivants :

DÉFINITION – Couche
Une partition fonctionnelle de l'architecture d'un système logiciel.

Dans une architecture de logiciel défini, chaque couche doit respecter les principes suivants :

PRINCIPES – Couches
a) Le logiciel d'une couche fournit un ensemble de services qui est cohérent selon certains critères définis, et le logiciel dans d'autres couches peut être utilisé sans savoir comment ces services sont implémentés. b) La relation, entre le logiciel dans n'importe laquelle des deux couches, est définie par une « règle de correspondance » qui peut-être être soit: • 'hiérarchique', c'est-à-dire que le logiciel de la couche A est autorisé à utiliser les services fournis par le logiciel de la couche B, mais pas vice versa (où la relation hiérarchique peut être vers le haut, vers le bas ou latéral), ou • 'bidirectionnelle', c'est-à-dire que le logiciel de la couche A est autorisé à utiliser le logiciel de la couche B, et vice versa. c) Le logiciel d'une couche échange des groupes de données avec le logiciel dans un autre couche via leurs processus fonctionnels respectifs. d) Le logiciel d'une couche n'utilise pas nécessairement tous les services fonctionnels fournis par le logiciel d'une autre couche. e) Le logiciel, d'une couche d'une architecture logicielle définie, peut être partitionné dans d'autres couches selon une architecture logicielle définie différemment.

Une mesure, qui concerne deux ou plus de morceaux d'un logiciel de même niveau, est définie de la façon suivante:

DÉFINITION – Morceau de logiciel de même niveau
Deux morceaux d'un logiciel sont de même niveau si chacun réside dans la même couche.

EXEMPLE : Les morceaux du logiciel dans la couche d'application de la Figure 2.1 sont tous de même niveau.

Si le logiciel à mesurer existe au sein d'une architecture établie des couches qui peuvent être arrimées aux principes de couche de COSMIC tels que décris ci-dessus, alors l'architecture doit servir à identifier les couches pour les raisons d'être du mesurage.

Si cependant, la raison d'être exige que certains logiciels à mesurer ne soient pas structurés selon les principes de couches de COSMIC, le mesureur doit essayer de partitionner le logiciel en couches en appliquant les principes définis ci-dessus. Par convention, les progiciels d'infrastructure tels que les systèmes de gestion de base de données, les systèmes d'exploitation ou les pilotes de périphériques, qui fournissent des services qui peuvent être utilisés par d'autres logiciels dans d'autres couches, seront situés dans des couches distinctes.

Normalement dans les architectures des logiciels, la couche du 'haut', c'est-à-dire la couche qui n'est pas subordonnée à d'autres couches dans la hiérarchie des couches, est appelée la couche 'd'application'. Le logiciel dans cette couche d'application s'appuie sur les services du logiciel dans toutes les autres couches pour qu'il fonctionne correctement. Le logiciel dans la couche du 'haut' peut lui-même faire partie d'une couche, par exemple comme dans une 'architecture en trois couches': interface utilisateur, règles d'affaires et les services de données (voir exemple 5 plus bas).

Une fois identifiée, chaque couche peut être enregistrée dans la matrice du modèle générique de logiciel (annexe A), avec une étiquette correspondante.

Exemple d'affaires 1: La structure physique d'une couche typique de l'architecture d'un logiciel (en utilisant le terme « couche » tel que défini ici) supportant les logiciels d'application d'affaires est présentée à la Figure 2.2 :

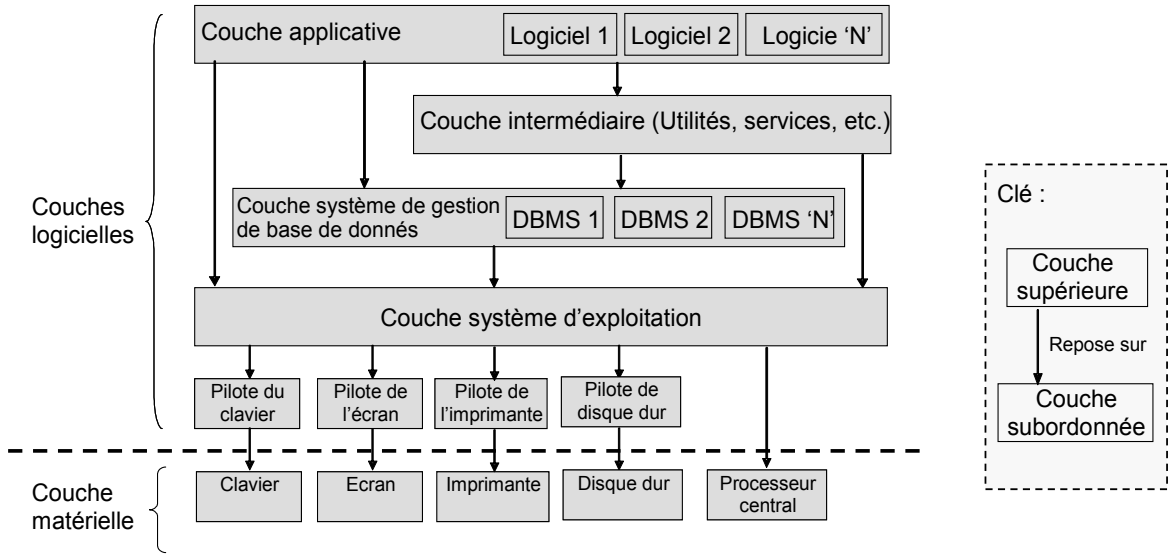


Figure 2.2- Architecture d'un logiciel/matériel en couches, typique des applications d'affaires et de logiciels SIG

Exemple temps réel 2: La structure physique d'une couche typique de l'architecture d'un logiciel (en utilisant le terme « couche » tel que décrit ici) supportant les morceaux du logiciel temps réel embarqué est présentée à la Figure 2.3:

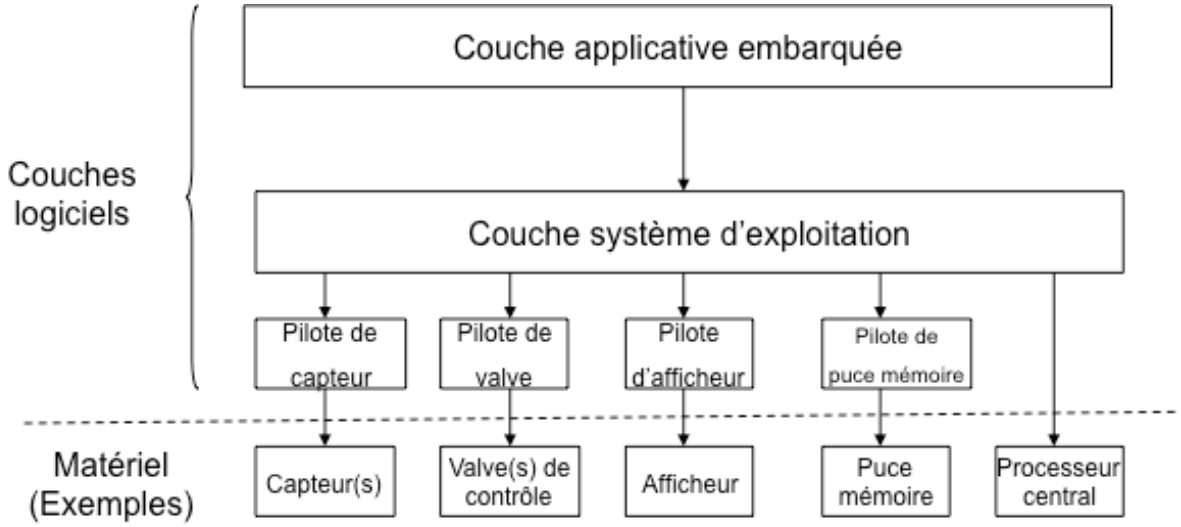


Figure 2.3 - Architecture d'un logiciel/matériel en couches, typique des applications embarquée et de type temps réel

Exemple temps réel 3: Le modèle des couches ISO 7 (OSI) pour les télécommunications. Il définit une architecture en couches pour lesquelles les règles de correspondance hiérarchiques pour les couches du logiciel récepteur de message sont l'inverse des règles pour les couches du logiciel transmettant le message.

Exemple temps réel 4: L'architecture « Autosar » de l'industrie automobile qui présente les différents types de règles de correspondance entre les couches telles que décrites dans les principes d'une couche. Voir www.autosar.org/download/AUTOSAR_LayeredSoftwareArchitecture.pdf.

Une architecture du logiciel peut montrer différentes couches dépendant de la 'vue' de l'architecture.

Exemples d'affaires 5: Envisager une demande A situé dans une architecture de logiciel en couches, comme dans la Figure 2.4 ci-dessous, qui présente trois structures couche possible a), (b) et c) conformément aux 'vues' de différentes architectures.

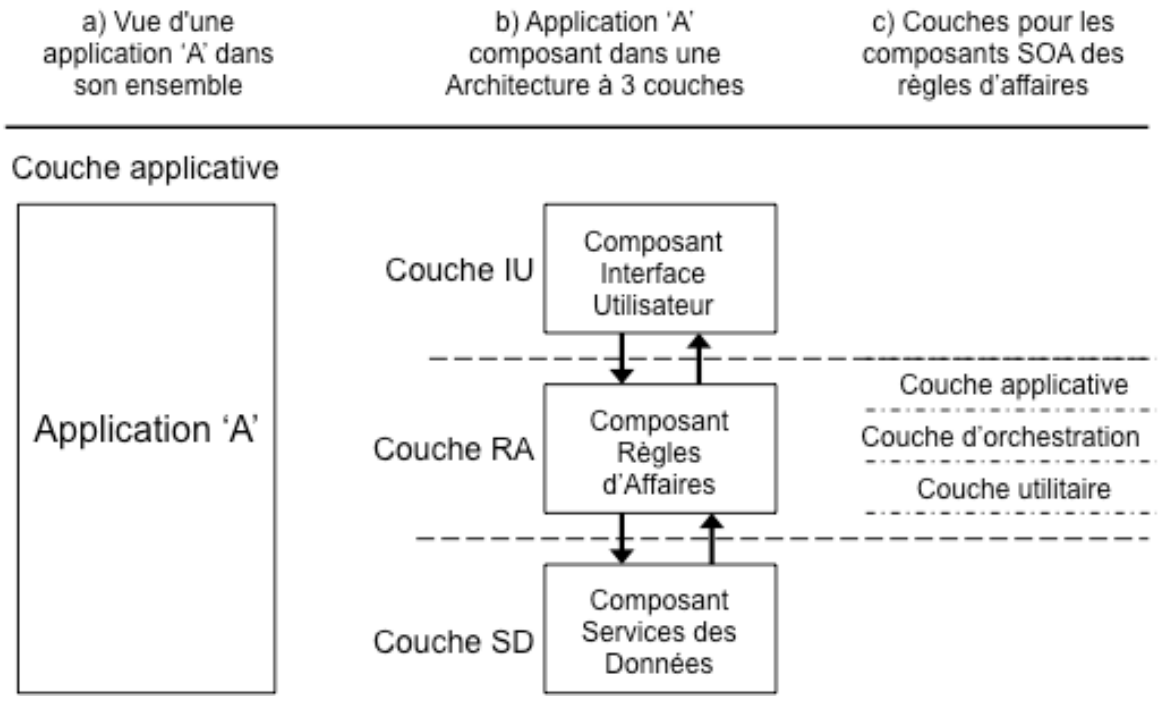


Figure 2.4 - Trois visions des couches d'une application

La raison d'être 1 est de mesurer la taille fonctionnelle de l'application A comme un tout (voir 'vue' (a)) Le périmètre du mesurage est toute l'application A se trouvant dans une seule couche.

La raison d'être 2 est de mesurer la taille fonctionnelle de l'application A en trois composants séparés. Elle a été construite en une 'architecture en trois couches' et comprend une interface utilisateur, des règles d'affaires et des services de données (Voir (b)). Chaque composant est situé dans sa propre couche architecturale et le périmètre du mesurage doit être définie séparément pour chaque composant.

Le composant des règles d'affaires de l'application a été construit à l'aide de composants réutilisables d'une Architecture orientée service (AOS), qui a sa propre structure de couche. La raison d'être de 3 vise à mesurer un composant AOS des règles d'affaires comme selon la vue c). Chaque composant de l'AOS est situé dans une couche de l'architecture AOS et le périmètre du mesurage doit être défini séparément pour chaque composant de l'AOS. (Notez que terminologie AOS utilise aussi « couche d'application » au sein de sa propre architecture.)

2.2.3 Niveaux de décomposition

Le 'niveau de décomposition' d'un morceau de logiciel est défini comme suit :

DÉFINITION – Niveau de décomposition
N'importe quel niveau résultant de la Division un morceau de logiciel en composants (nommé 'niveau 1', par exemple), puis diviser les composants en sous-composantes ('niveau 2'), et ensuite diviser les sous-composants en sous-sous composants ('niveau 3'), etc..
Remarque 1: Ne pas confondre avec 'niveau de granularité'.
Remarque 2: Le mesurage de la taille d'un morceau du logiciel peut être comparé directement seulement aux composants d'un même niveau de décomposition.

EXEMPLE : Le guide 'Measurement Strategy Patterns' [5] reconnait trois niveaux normalisés de décomposition: 'Application dans son ensemble', 'Composent majeur' et 'Composant mineur'. Voir l'exemple d'affaires 5 dans la section 2.2.2 du présent Manuel de mesurage, où les trois niveaux sont montrés (voir Figure 2.4).

Définir le périmètre du mesurage sommaire : déterminer le périmètre du mesurage est plus qu'une simple question de décider quelles fonctionnalités devraient être incluses dans la mesure. La décision peut aussi entrainer des questions sur les couches du logiciel à mesurer et le degré de décomposition du logiciel sur lequel se fera le mesurage, le tout dépendant de la raison d'être du mesurage.

2.3 Identification des utilisateurs fonctionnels et du stockage persistant

2.3.1 La taille fonctionnelle peut varier selon les utilisateurs fonctionnels

Différents types d'utilisateurs d'une 'chose' peuvent 'voir' différentes fonctionnalités et par conséquent peuvent mesurer des tailles différentes de la 'chose'. Dans le cas des logiciels, différents (types) d'utilisateurs fonctionnels peuvent exiger (via leur FUR) différentes fonctionnalités et donc les tailles fonctionnels varieront selon les choix des utilisateurs fonctionnels.

Un utilisateur est défini comme 'toute chose qui interagit avec le logiciel à mesurer'. Cette définition est trop large pour les besoins de la méthode COSMIC. Pour la méthode COSMIC, de l'utilisateur (ou des utilisateurs) est déterminé par la FUR à mesurer. Ce (type) d'utilisateur, connu comme 'l'utilisateur fonctionnel' et définit de la façon suivante.

DÉFINITION – Utilisateur fonctionnel
Un (type) d'utilisateur qui est un expéditeur et/ou un destinataire visé par les données de la FUR d'un morceau de logiciel.

Dans la méthode COSMIC, il est essentiel de distinguer les utilisateurs fonctionnels d'un morceau de logiciel à mesurer de l'ensemble des utilisateurs potentiels.

EXEMPLE 1: Considérons une application d'affaires, ses utilisateurs fonctionnels incluraient normalement les humains et les autres applications de même niveau avec lequel l'application interagit. Pour une application en temps réel, les utilisateurs fonctionnels seraient normalement les morceaux d'équipement ou d'autres logiciels de même niveau avec lequel il interagit. La FUR d'un tel logiciel identifie normalement les utilisateurs fonctionnels que sont les expéditeurs de données et/ou les destinataires des données vers et à partir du logiciel, respectivement.

Toutefois, l'ensemble des 'utilisateurs', c'est-à-dire 'toute chose qui interagit avec le logiciel', doit inclure le système d'exploitation. Mais la FUR de n'importe quelle application ne comprend jamais le système d'exploitation en tant qu'utilisateur. Toutes les contraintes que le système d'exploitation peut imposer à une application seront communes à toutes les applications - normalement traitées par le compilateur ou l'interpréteur - sont invisibles pour les vrais utilisateurs fonctionnels de l'application et n'apparaissent donc pas dans la FUR. Dans la pratique du mesurage de la taille fonctionnelle, un système d'exploitation ne doit normalement jamais être considéré comme un utilisateur fonctionnel d'une application. Mais les utilisateurs fonctionnels ne sont pas toujours évidents.

EXEMPLE 2: Considérons le logiciel d'application d'un téléphone mobile. Après avoir écarté le système d'exploitation de la téléphonie mobile comme utilisateur fonctionnel de l'application, les 'utilisateurs' pourraient encore être soit (a) les humains qui appuie sur les touches, ou (b) l'ensemble des périphériques (par exemple l'écran, clé, etc.) ainsi que les applications de mêmes niveaux qui interagissent directement avec l'application téléphone. L'utilisateur humain, par exemple, verra uniquement un sous-ensemble de toutes les fonctionnalités qui doivent être fournies pour permettre à l'application du téléphone mobile de fonctionner. Si ces deux types d'utilisateurs voient différentes

fonctionnalités, la taille de la FUR pour l'utilisateur humain sera inférieure à la taille de la FUR qui doit être développée pour permettre è l'application téléphone mobile de fonctionner[4].

RÈGLES – Utilisateurs fonctionnel
a) Les utilisateurs fonctionnels d'un morceau de logiciel à mesurer doivent être dérivés de la raison d'être du mesurage. b) Lorsque la raison d'être d'une mesure d'un morceau de logiciel est liée à l'effort de développement ou à l'effort de modification d'un morceau de logiciel, alors les utilisateurs fonctionnels devraient être tous les expéditeurs et/ou destinataires visés des données vers et à partir des fonctionnalités nouvelles ou modifiées, comme requis par les FURS.

Après avoir identifié les utilisateurs fonctionnels, il est alors facile d'identifier la frontière. La frontière se situe entre le logiciel à mesurer et ses utilisateurs fonctionnels. Nous ignorons tout autre matériel ou logiciel dans cette intervention[5].

DÉFINITION – Frontière
Une interface conceptuelle entre les logiciels à mesurer et ses utilisateurs fonctionnels. Remarque: Il découle de la définition, qu'il y a une frontière entre les deux morceaux de logiciel, dans une même ou dans différentes couches qui échangent des données, où un morceau de logiciel est un utilisateur fonctionnel d'un autre utilisateur fonctionnel, et/ou vice versa.

Remarque: Cette définition de 'frontière' est tirée de ISO/IEC14143/1: 2007, modifié par l'ajout du mot 'fonctionnel' à 'utilisateur' pour le qualifier. Pour éviter toute ambigüité, notez que la frontière ne doit pas être confondue avec n'importe quelle ligne qui pourrait être établie autour de certains logiciels à mesurer pour définir le périmètre du mesurage. La frontière n'est pas utilisée pour définir le périmètre du mesurage.

2.3.2 Stockage persistant

DÉFINITION – Stockage persistant
Stockage qui permet à un processus fonctionnel de conserver un groupe de données au-delà de la durée du processus fonctionnel et/ou à partir duquel un processus fonctionnel peut: récupérer un groupe de données conservé par un autre processus fonctionnel, conserver une occurrence antérieure du même processus fonctionnel ou conservé par un autre procédé. Remarque 1: Dans le modèle COSMIC, un stockage persistant est un concept qui existe uniquement à l'intérieur de la frontière du logiciel à mesurer; on ne peut donc le considérer comme un utilisateur fonctionnel du logiciel à mesurer. Remarque 2: Un exemple de 'certains autres processus' serait la 'fabrication' de la mémoire en lecture seule.

[4] Toivonen, par exemple, compare les fonctionnalités des téléphones portables disponibles pour les utilisateurs humains dans 'Defining measures for memory efficiency of the software in mobile terminals', International Workshop on Software Measurement, Magdeburg, Germany, October 2002.

[5] En fait, si le mesureur a dû examiner la FUR afin d'identifier les expéditeurs et les destinataires des données, la limite aura donc déjà été identifiée.

(Pour la définition d'un 'processus fonctionnel', voir la section 3.2)

2.3.3 Diagrammes contextuels

Il peut être très utile, lorsque l'on définit le périmètre du mesurage et les utilisateurs fonctionnels, de dessiner un 'diagramme contextuel' pour le logiciel à mesurer. Ici et dans d'autres directives COSMIC, les diagrammes contextuels sont utilisés pour montrer le périmètre d'un morceau de logiciel à mesurer dans le contexte des utilisateurs fonctionnels (humains, interfaces matérielles ou les autres logiciels) et des mouvements de données entre eux. (Normalement, les diagrammes contextuels montrent aussi le stockage persistant, si pertinent.)

Un diagramme contextuel est en fait une instance d'un modèle du mesurage (voir la section 2.0) appliqué au logiciel à mesurer. L'explication des symboles utilisés dans les modèles contextuels est tel qu'indiqué à la Figure 2.5 :

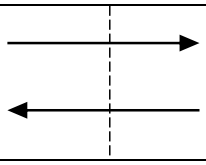
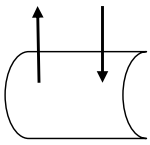
Symboles	**Interprétation**
	Le morceau de logiciel à mesurer (boite avec un contour épais) c'est-à-dire la définition du périmètre du mesurage.
	Tout utilisateur fonctionnel du logiciel à mesurer.
	Les flèches représentent tous les mouvements de données traversant une frontière (ligne pointillée) entre un utilisateur fonctionnel et le logiciel à mesurer.
	Les flèches représentent tous les mouvements de données entre le logiciel à mesurer et le 'stockage persistant'. (Le symbole de diagramme de flux standard pour le 'stockage des données' souligne que le stockage persistant est un concept abstrait. L'utilisation de ce symbole indique que le logiciel n'interagit pas directement avec le stockage du matériel physique).

Figure 2.5 – Symboles clefs des diagrammes contextuels

EXEMPLE d'affaires: La Figure 2.6 montre le diagramme contextuel pour le logiciel client/serveur du progiciel d'application implémenté, comme dans l'exemple illustré à la Figure 2.1 de la section 2.2.1, à mesurer comme un tout.

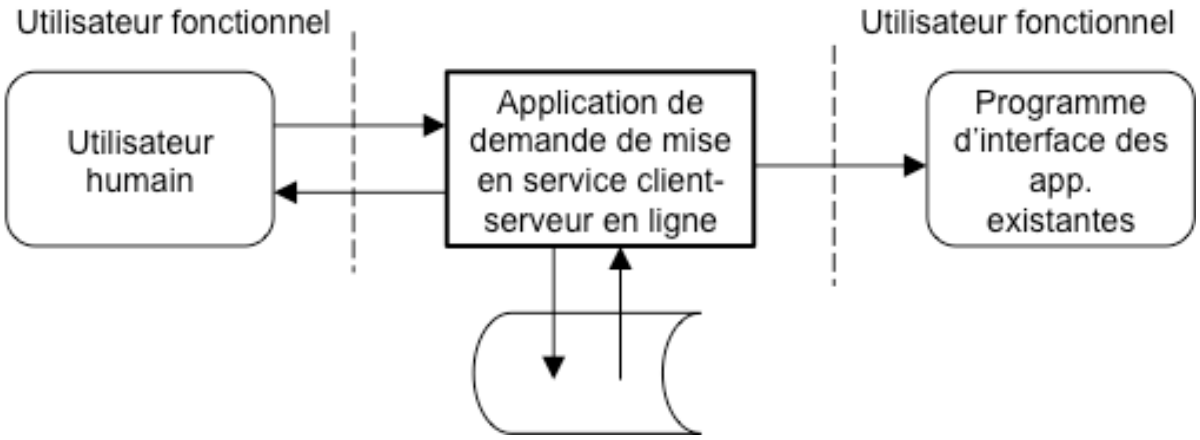


Figure 2.6 - Diagramme de contexte pour l'application client-serveur de la section 2.2.

EXEMPLE temps réel : La Figure 2.7 montre le diagramme contextuel pour un logiciel d'alarme embarqué pour une simple intrusion (tiré de 'COSMIC Guideline for sizing real-time software' [4], version 1.0, section 4.2).

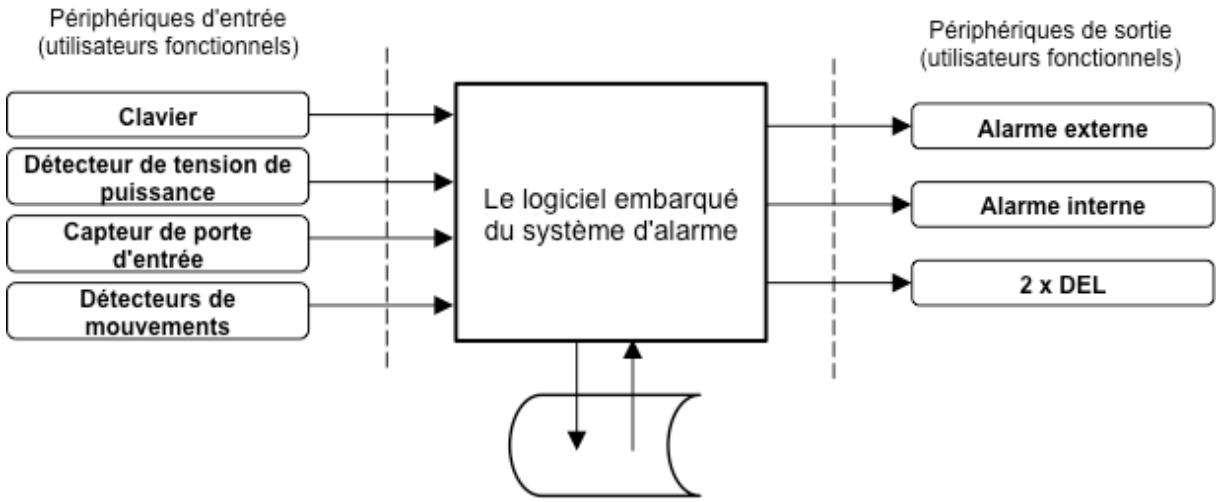


Figure 2.7 - Diagramme contextuel pour un logiciel embarqué d'un système d'alarme d'intrusion

2.4 Identification du niveau de granularité

2.4.1 Le besoin d'un niveau de granularité normalisé

Aux étapes initiales d'un projet de développement d'un logiciel, les exigences connues à ces étapes sont spécifiées 'à un haut niveau', c'est-à-dire dans les grandes lignes et avec peu de détails. Tout au long du projet, les besoins réels sont précisés, (par le biais, entre autre, de différentes versions 1, 2, 3, etc.), révélant plus de détails 'au niveau inférieur'. Ces différents degrés de détail des exigences réelles (et donc de la FUR qui est dérivée) sont les différents 'niveaux de granularité'. (Voir aussi la section 2.4.2 pour les autres termes qui peuvent être confondues avec la notion de 'niveau de granularité' tel que défini ici.).

DÉFINITION – Niveau de granularité
Tout niveau d'expansion de la description d'un seul morceau de logiciel (c.-à-d. un énoncé d'une exigence, ou la description d'une structure d'un morceau de logiciel) tel que chaque niveau supérieur d'expansion, la description de la fonctionnalité du morceau

du logiciel est à un niveau supérieur et uniforme de détails.

REMARQUE: Les mesureurs doivent tenir compte de l'évolution précoce des exigences dans le cycle de vie d'un projet. À tout moment, différentes parties des fonctionnalités requises du logiciel seront typiquement documentées à différents niveaux de granularité.

Dans la plupart des activités de développement du produit, des plans sont dessinés selon des échelles normalisées, et il est facile de traduire les dimensions mesurées sur un dessin à partir d'un autre dessin avec une échelle différente. En revanche, il n'y a aucune échelle normalisée pour les différents niveaux de granularité de spécification d'un logiciel; il est donc difficile d'être certain que les deux énoncés de la FUR sont au même niveau de granularité. Sans ententes pour normaliser le niveau de granularité (quelles mesures devraient être mises à l'échelle), il est impossible d'affirmer avec certitude que les deux mesures de la taille fonctionnelle peuvent être comparées.

Pour illustrer le concept sous un autre angle, considérons cette analogie : un ensemble de cartes routières indique les détails d'un réseau routier national à trois niveaux de granularité :

- La carte A montre les autoroutes et les routes principales ;
- La carte B montre plutôt les autoroutes, les routes principales et les routes secondaires (comme dans un atlas pour automobilistes) ;
- La carte C montre toutes les routes avec le nom de chacune (équivalent des cartes routières locales).

Si nous ne connaissions pas le concept de différents niveaux de granularité, ces trois cartes auraient indiqué différentes tailles du même réseau routier. Habituellement, les cartes routières intègrent le concept et intègrent donc les différents niveaux de détails à différentes échelles normalisées pour interpréter la taille du réseau routier selon le niveau de granularité que l'on veut montrer. Le concept de 'niveau de granularité' se trouve derrière les échelles de ces différentes cartes.
Pour la mesure de logiciel, il y a seulement un niveau normalisé de granularité qu'il est possible de définir clairement. C'est le niveau de granularité auquel les processus fonctionnels ont été identifiés et les mouvements de données décrites. Donc lorsque c'est possible, c'est selon ce niveau normalisé de granularité que les mesures devraient être faites (ou étalonnée)[6].

2.4.2 Clarification du 'niveau de granularité'

Avant de continuer, il est important de s'assurer que la signification du concept de 'niveau de granularité' de la méthode COSMIC est bien assimilée. Tel que décrit plus tôt, approfondir la 'description' de haut niveau d'un logiciel, à un niveau inférieur de granularité, implique qu'il existe une vision plus en profondeur des fonctionnalités et qu'on révèle plus de détails, sans toutefois changer le périmètre. Ce processus ne devrait pas être confondu avec les processus suivants.

- Faire un zoom sur certains logiciels afin de révéler ses composants, sous-composants, etc. (à différents 'niveaux de décomposition' – voir la section 2.2.3 ci-dessus). Un tel zoom peut être requis si la raison d'être du mesurage exige que le périmètre du mesurage soit subdivisé en suivant la structure physique du logiciel.
- L'évolution la description de certains logiciels au fur et à mesure de sa progression tout au long de son cycle de développement, c'est-à-dire des exigences à la conception logique, à la conception physique, etc. Quel que soit le stade de développement de certains logiciels, nous sommes seulement intéressés à sa FUR pour la raison du mesurage.

Le concept de 'niveau de granularité' est donc destiné à être interprété comme s'appliquant seulement aux FURs du logiciel.

[6] Le sujet de la mise à l'échelle du mesurage d'un niveau de granularité à un autre est actuellement traité dans le document 'Advanced and Related Topics' [6].
Le 'Guideline for approximate COSMIC functional size measurement' [12] est en cours d'élaboration remplacera éventuellement le document existant.

2.4.3 Le niveau de granularité normalisé

Des mesures précises, de la taille fonctionnelle COSMIC, exigent que les FURs à mesurer existent à un niveau de granularité ou les processus fonctionnels et leurs mouvements de données peuvent être identifiés. Le 'niveau de granularité du processus fonctionnel' est défini comme suit[7].

DÉFINITION – Niveau de granularité d'un processus fonctionnel

Un niveau de granularité de la description d'un morceau de logiciel pour lequel :

- Les utilisateurs fonctionnels sont des individus humains ou des objets d'ingénierie ou d'autres morceaux de logiciel (et non pas des groupes de ceux-ci) ET ;
- Détectent des occurrences uniques d'évènements auxquels un morceau du logiciel doit répondre (et pas au niveau où les groupes d'évènements sont définis).

REMARQUE 1: En pratique, la documentation du logiciel contenant les FURs décrit souvent les fonctionnalités à divers niveaux de granularité, plus particulièrement quand la documentation est toujours en cours d'élaboration.

REMARQUE 2: Les 'Groupes ciblés (utilisateurs fonctionnels) peuvent être, par exemple, est un 'département' dont les membres peuvent effectuer plusieurs types de processus fonctionnels, ou un 'panneau de contrôle' qui a plusieurs types d'instruments, ou des 'systèmes centraux'.

REMARQUE 3: Les 'Groupes d'évènements' peuvent, par exemple, être exprimés dans l'énoncé d'une FUR à un haut niveau de granularité par un flux d'intrants pour un système comptable nommé 'transactions de vente' ou par un flux d'intrants à un logiciel avionique appelée 'commande de pilotage'.

Avec cette définition, nous pouvons maintenant définir les règles suivantes et faire une recommandation.

RÈGLES - Niveau de granularité d'un processus fonctionnel

a) Le mesurage exact de la taille fonctionnelle d'un morceau de logiciel exige que sa FUR soit connue à un niveau de granularité pour lequel son processus fonctionnel et ses sous-processus de mouvements de données peuvent être identifiés.

b) Si certaines exigences doivent être mesurées avant d'avoir été définies avec suffisamment de détails pour un mesurage précis, les exigences peuvent être mesurées à l'aide d'une approche approximative. Ces approches définissent comment les exigences peuvent être mesurées à un ou des niveau(x) de granularité plus élevés. Les mises à d'échelle sont ensuite appliquées au(x) niveau(x) plus élevé(s) de granularité pour produire une taille approximative des processus fonctionnels et des sous-processus de mouvements de données à leur niveau de granularité. Voir 'Guideline for approximate COSMIC functional size measurement' [6].

En plus de ces règles, COSMIC recommande[8] que le niveau de granularité du processus fonctionnel représente la norme à laquelle les mesures de la taille fonctionnelles doivent se conformer. Cette recommandation s'étend aux fournisseurs de services d'étalonnage, aux fournisseurs d'outils logiciels

[7] Le nom 'niveau de granularité des processus fonctionnels' se justifie parce que c'est le niveau d'identification des processus fonctionnels – voir la section 3.2 pour une discussion plus détaillée sur les processus fonctionnels.

[8] La raison pour laquelle l'utilisation du niveau de granularité des processus fonctionnels est 'recommandé' plutôt que d'être une règle, c'est que cette recommandation ne s'applique pas seulement aux utilisateurs de la méthode COSMIC, mais aussi aux réseaux de fournisseurs de services et d'outils qui utilisent les mesures de la taille. COSMIC peut seulement formuler des recommandations à cette communauté plus large.

conçus pour supporter ou utiliser des mesures de taille fonctionnelles, utile à l'estimation de l'effort par exemple.

EXEMPLE d'affaires : L'exemple, du domaine des logiciels d'affaires, fait partie d'un système bien connu de commande de marchandises sur l'Internet, que nous appellerons "Application de commande d'Everest". La description ci-dessous est très simplifiée aux fins d'illustration des niveaux de granularité.

Pour mesurer cette application, nous pouvons assumer que la raison d'être du mesurage est de déterminer la taille fonctionnelle de la partie de l'application disponible pour les utilisateurs du module 'client humain' (comme 'utilisateurs fonctionnels'). Nous pouvons ensuite définir le périmètre du mesurage comme 'les parties de l'application Everest accessibles aux clients pour la commande de marchandises sur l'Internet'. Notez, toutefois, que le but de cet exemple est d'illustrer les différents niveaux de granularité. Nous explorerons donc seulement certaines parties de la fonctionnalité du système total ce qui sera suffisant pour comprendre le concept des niveaux de granularité.

Au plus 'haut niveau' (niveau 1) de l'application, un artéfact descriptif d'une FUR du système de commande Everest serait représenté par un énoncé simple, tel que :

"Le système d'Everest doit permettre aux clients d'interroger, de sélectionner, de payer et d'obtenir la livraison de n'importe quel article de la gamme de produits offerts par Everest, y compris les produits fournis par les fournisseurs externes."

En réalisant un zoom sur cet énoncé de haut niveau, on peut constater que le prochain niveau (le niveau 2) consisterait en 4 sous-systèmes, tel que montré à la figure 2.8(a).

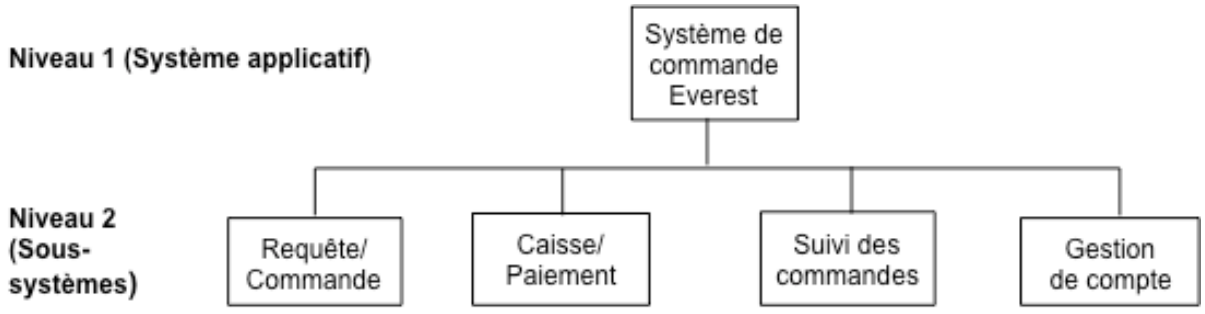


Figure 2.8 (a) - Analyse du système de commande d'Éverest: les deux premiers niveaux de granularité

Les exigences des quatre sous-fonctions sont:

- Le sous-système interrogation/commande permet à un client de trouver n'importe quel produit de la base de données d'Everest, aussi bien que son prix et sa disponibilité. Il permet aussi au client d'ajouter n'importe quel produit sélectionné, à son 'panier' d'achats. Ce sous-ensemble favorise également les ventes, car il présente des offres spéciales, des évaluations pour les articles choisis et permet aussi de répondre à des questions générales sur les conditions de livraison, etc. C'est un sous-ensemble très complexe et son analyse ne dépasse pas le niveau 2 aux fins de cet exemple ;
- Le sous-système vérification/paiement permet au client de confirmer une commande et de payer les marchandises présentes dans sone panier ;
- Le sous-système suivi des commandes permet au client de connaître la progression de la livraison d'une de ses commandes existante, de mettre à jour les informations relatives à sa commande (ex. : modifier l'adresse de livraison) et de retourner les marchandises dont le client n'est pas satisfait ;
- Le sous-système d'entretien des comptes permet à un client existant de mettre à jour les informations détaillées de son compte comme son adresse à domicile ou les modes de paiement préconisés, à titre d'exemple.

Un zoom a aussi été effectué aux figures 2.8 (b) et (c), ce qui permet de montrer plus de détails à travers les niveaux inférieurs, soit vers les sous-ensembles des FUR tels que : sous vérification/paiement, sous de suivi des commandes et sous l'entretien des comptes. Lors du processus de zoom il est important de noter que :

- *nous n'avons pas changé le périmètre des fonctionnalités à mesurer ;*
- *tous les niveaux de la description du système logiciel Everest montrent la fonctionnalité disponible aux clients (en tant qu'utilisateurs fonctionnels). Un client peut donc 'voir' en entier la fonctionnalité du système, à tous ces niveaux de l'analyse.*

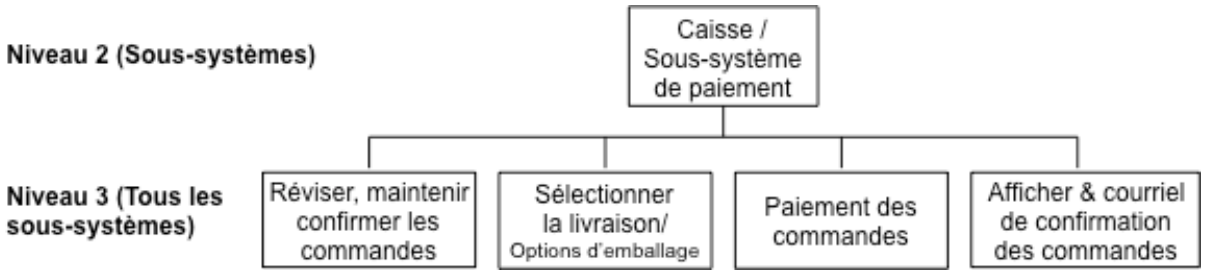


Figure 2.8 (b) - Décomposition de la sous-fonction caisse/paiement

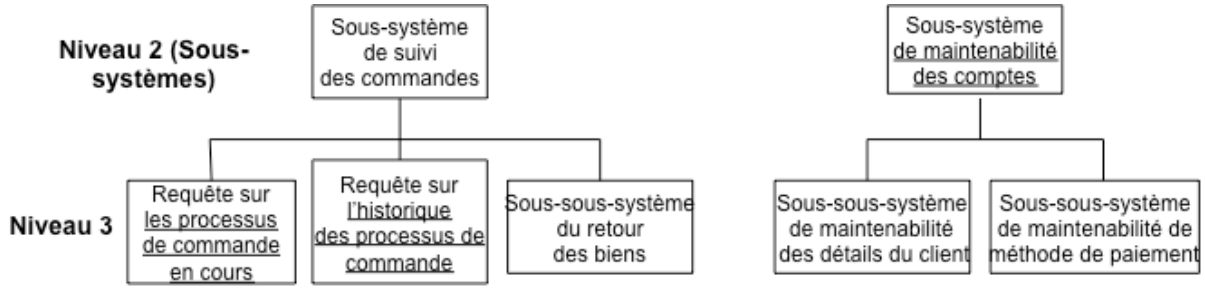


Figure 2.8 (c) - Décomposition du suivi des commandes et du sous-système de maintenabilité des comptes

La figure 2.8 (c) indique également cela quand un zoom est effectué à des niveaux plus bas dans cette analyse particulière. Sous ces trois sous-systèmes, nous trouvons différents processus fonctionnels[9] (au niveau 3), de même qu'au niveau 4 pour deux des sous-systèmes. Cet exemple démontre donc que, quand une certaine fonctionnalité est analysée dans une approche descendante, on ne peut pas supposer que la fonctionnalité d'un niveau particulier correspondra toujours au même 'niveau de granularité', tel que défini dans la Méthode COSMIC. La définition exige plutôt qu'à n'importe quel niveau de granularité, la fonctionnalité soit 'à un niveau comparable de détail'.

De plus, d'autres analystes pourraient dessiner les diagrammes différemment, montrant d'autres regroupements de la même fonctionnalité, à chaque niveau du diagramme. Il n'y a pas une seule façon 'correcte' d'analyser la fonctionnalité d'un système complexe.[10]

Puisque ce type de variation se produira inévitablement dans la pratique, un mesureur doit soigneusement examiner les divers niveaux d'un diagramme d'analyse afin de trouver les processus fonctionnels à mesurer. Là où, dans la pratique, ce n'est pas possible, c'est dans les cas où l'analyse n'a pas encore atteint le niveau où tous les processus fonctionnels ont été définis. Dans ce cas, la

---

[9] *(Remarque: à ce stade, quelqu'un qui est nouveau avec la méthode COSMIC peut ne pas être convaincu, sur la base de la définition et des règles, que le "niveau de granularité des processus fonctionnels " des deux requêtes apparues au niveau 3 sont réellement des processus fonctionnels, par opposition aux deux sous-sous-systèmes qui peuvent encore être décomposés. La section 3.2 "Identification des processus fonctionnels" offrira plus de preuves supportant cette 'analyse'.)*

[10] La figure 2.8 peut ne pas être un exemple de 'meilleure pratique', mais indique bien comment ces diagrammes peuvent être dessinés.

règle (b) doit être appliquée. Pour illustrer ceci, examinons le cas du sous-système 'mise à jour des informations client' de la figure 2.8 (c) ci-dessus, sous la ramification d'entretien des comptes.
Pour un mesureur expérimenté, le terme 'mise à jour' suggère un groupe d'évènements ainsi qu'un groupe de processus fonctionnels qui restent invariables. Nous pouvons donc s'avancer sans trop de risques que 'mettre à jour' comprendra au moins trois processus fonctionnels, à savoir : Lire les informations client, écrire les informations client, et supprimer les informations client. (Le processus 'création des informations client doit aussi exister, mais il se trouve dans une autre ramification du système, soit lorsque le client commande pour la première fois. Il est en dehors de la portée de cet exemple.).

En extrapolant un certain nombre de processus fonctionnels, un mesureur expérimenté devrait être en mesure de connaitre instinctivement (guestimate) la taille d'un de ces sous-système en unité CFP (trois dans ce cas-ci) et en multipliant ce nombre par la taille moyenne d'un processus du système ou dans d'autres systèmes comparables. Des exemples de ce processus de calibration sont donnés dans le document 'Guideline on Approximate size measurement' [6]. Il contient également d'autres exemples et approches pour estimer la taille.

Clairement, de telles méthodes d'approximation ont leurs limites. Si nous appliquons une telle approche à l'énoncé de niveau 1 de la FUR tel que donné ci-dessus ("Le système d'Everest doit permettre aux clients d'interroger, de sélectionner, de payer et d'obtenir la livraison de n'importe quel article de la gamme de produits offerts par Everest, y compris les produits fournis par les fournisseurs externes."), nous pourrions identifier peu de processus fonctionnels. Mais une analyse plus détaillée indiquerait que le vrai nombre de processus fonctionnels dans ce système complexe doit être beaucoup plus grand. C'est pourquoi les tailles fonctionnelles semblent habituellement augmenter au fur et à mesure que plus de détails sur les exigences sont établis, même sans changer de périmètre de mesurage . Ces méthodes d'approximation doivent donc être employées avec précaution aux niveaux élevés de la granularité, quand peu de détails sont disponibles.

Pour un exemple de mesurage à différents niveaux de granularité et de décomposition, voir l'exemple du système de télécommunication dans le 'Guideline for approximate COSMIC functional size measurement' [6].

2.5 Conclusions sur la phase de la stratégie du mesurage

La grande majorité des mesures de la taille fonctionnelle est réalisée dans un but qui est lié à l'effort de développement d'une certaine façon, par exemple pour la mesure du rendement d'un projet de développement ou l'estimation d'un projet. Dans bon nombre de ces situations, un modèle du mesurage normalisé peut-être être utilisé : voir 'Guideline on Measurement Strategy Patterns' [5]).

3

LA PHASE D'ARRIMAGE

3.0 Sommaire du chapitre

Ce chapitre traite de la deuxième phase de la phase 'd'arrimage' du processus du mesurage en définissant les concepts clés du modèle générique de logiciel et le processus à suivre pour l'arrimage de la FUR du logiciel au modèle, afin que la FUR puisse être mesurée. Ces concepts clés du modèle générique du logiciel sont: *(en italique)*

- Un évènement incite un *utilisateur fonctionnel* à demander un service au morceau de logiciel à mesurer. Un tel évènement est appelé un *'évènement déclencheur'* et le service demandé un *'processus fonctionnel'*.
- Les processus fonctionnels sont composés de deux types de sous-processus visant, soit à déplacer des données *("mouvements de données")* ou encore à manipuler des données (*'manipulation de données'*). Les sous-processus de manipulation des données ne sont pas reconnus séparément, mais sont censés être pris en considération par les mouvements de données auxquels ils sont associés.
- Un mouvement de données déplace un *'groupe de données'*. Un groupe de données est constitué *d'attributs de données* qui servent tous à décrire un *'objet d'intérêt'*, c'est-à-dire un objet qui est d'intérêt pour l'utilisateur fonctionnel concerné.
- Il y a quatre types de mouvements de données : *Entrées* et *Sorties* qui déplacent un groupe de données depuis et vers un *processus fonctionnel* à travers la *frontière* de/vers un utilisateur fonctionnel. *Lectures* et *Écritures* qui déplacent chaque groupe de données entre le *processus fonctionnel* et le *stockage persistant*.

Chacun de ces concepts est défini dans le présent chapitre, qui comprend les règles et concepts pour aider à identifier les concepts correctement, ainsi que des exemples détaillés des concepts pour différents types de logiciels.

3.1 Arrimage de la FUR au modèle générique de logiciel

Ce chapitre présente les définitions, les principes, les règles et la méthode du processus d'arrimage. La figure 3.0 montre les étapes du processus pour arrimer les FURs ainsi que les artéfacts disponibles du logiciel sous la forme requise par le modèle générique de logiciel COSMIC.

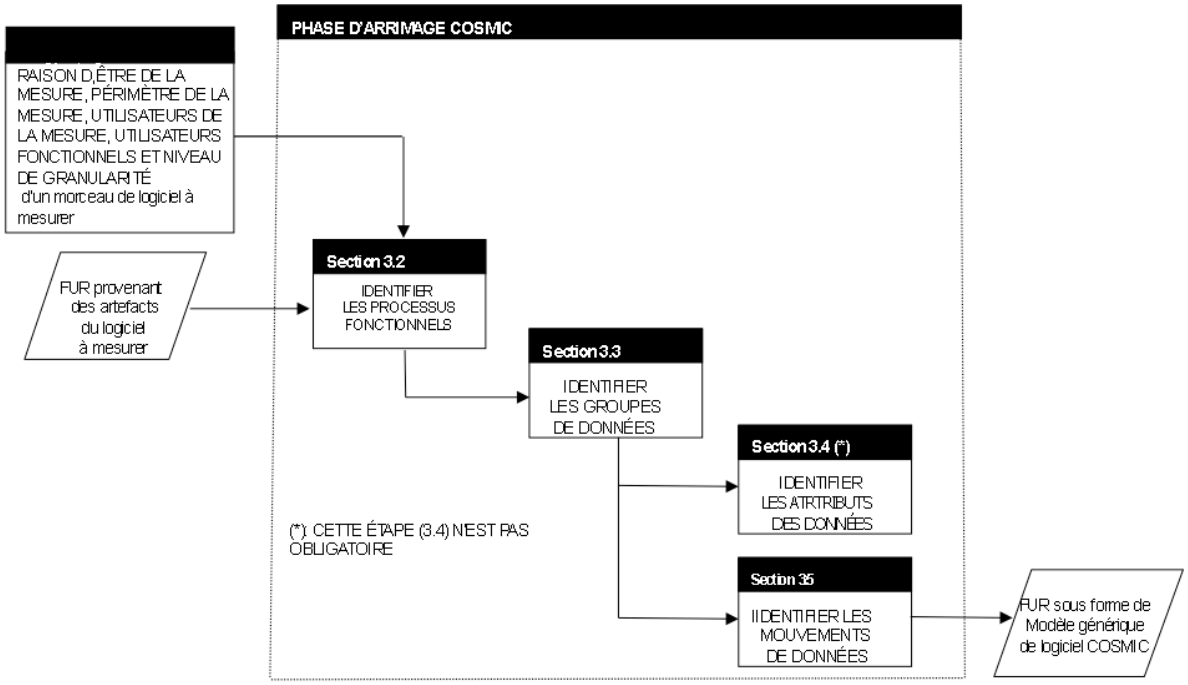


Figure 3.0 – Méthode générale d'arrimage du processus du mesurage COSMIC

Chaque étape de ce processus fait l'objet d'une section déterminée indiquée dans la barre de titre de l'étape à la Figure 3.0.

Le processus est conçu pour s'appliquer à un très large éventail d'artéfacts du logiciel. Nous encourageons les mesureurs à utiliser ce processus général pour dériver des règles plus spécifiques en vue d'une utilisation dans leur environnement local. Ce processus général permet d'arrimer les artéfacts du logiciel produits par la méthode locale de génie logiciel pour le modèle générique de logiciel COSMIC. L'objectif d'un processus local déterminé, illustré par des exemples locaux, devrait être de réduire l'incertitude de l'arrimage et donc d'améliorer la précision et la répétabilité des mesures.

Il existe plusieurs guides fournissant des orientations sur l'arrimage, à partir des différentes analyses de données et des méthodes de détermination des exigences utilisées dans différents domaines, aux concepts de la méthode COSMIC. On peut citer le 'Guideline for Sizing Business Application Software' [7], le 'Guideline for Sizing Data Warehouse Application Software' [8], le 'Guideline for Sizing Service-Oriented Architecture Software' [9] et le 'Guideline for Sizing Real-time Software' [4]. Pour les domaines des affaires [10] et du temps réel [11] il existe également le 'Quick Reference Guides' qui donne un aperçu du processus en quelques pages.

Les Figures 3.1 et 3.2 visent à aider à la transition à partir du Modèle de contexte du logiciel utilisé dans la phase de Stratégie du mesurage du Modèle de logiciel générique. Les figures s'appliquent à un morceau de logiciel d'application d'affaires et à un morceau de logiciel embarqué en temps réel typique. Ils correspondent respectivement aux Figures 2.6 et 2.7.

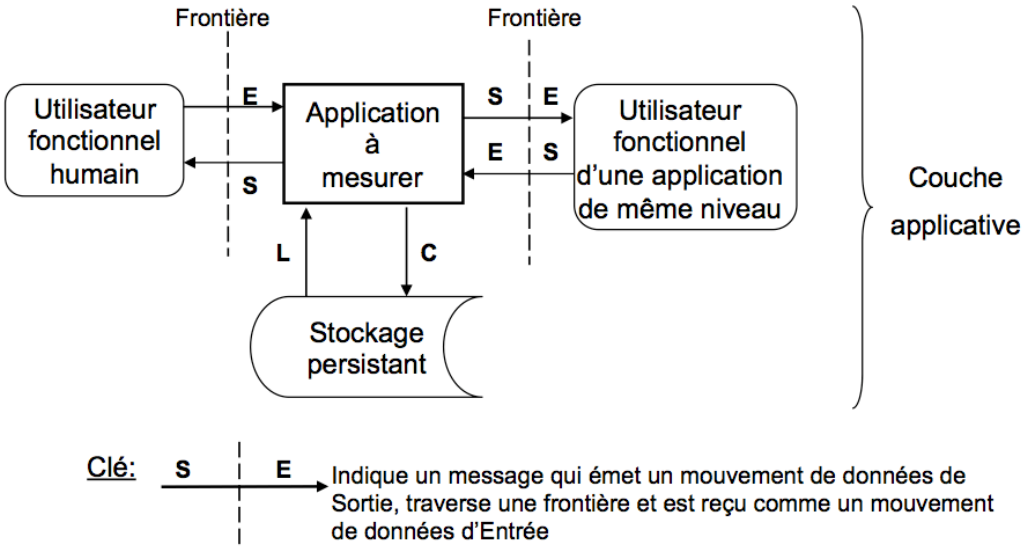


Figure 3.1 - Une application d'affaires avec des personnes humaines et une autre application de même niveau en tant qu'utilisateurs fonctionnels

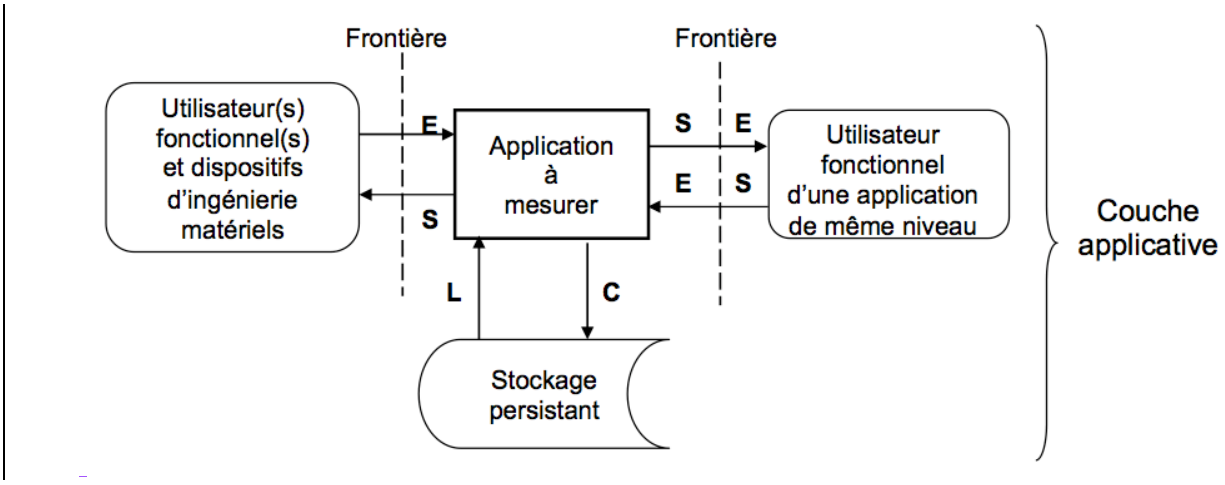


Figure 3.2 - Une application logicielle embarquée en temps réel, avec divers matériels périphériques et un logiciel de même niveau en tant qu'utilisateurs fonctionnels

À partir du principe f) du Modèle de contexte du logiciel, nous avons appris que les utilisateurs fonctionnels du logiciel à mesurer sont respectivement les expéditeurs et/ou ou les destinataires des données depuis/vers le logiciel. Le principe a) du Modèle générique de logiciel nous dit qu'un morceau de logiciel 'interagit avec ses utilisateurs fonctionnels à travers une frontière et un stockage persistant au sein de cette frontière'.

La figure 3.1 montre que le logiciel d'application à mesurer a deux utilisateurs fonctionnels, l'un humain et l'autre une application de même niveau. L'application de logiciel embarqué de la Figure 3.2 comprend uniquement les périphériques matériels comme utilisateur fonctionnel. Les flèches indiquant les déplacements des données sont étiquetées avec les abréviations montrant leurs types (E = Entrée, S = Sortie, L = Lecture, C = éCriture).

Notez que le 'stockage persistant' se réfère à n'importe quel stockage logique que l'application à mesurer doit accéder via les mouvements de données de Lecture ou d'éCriture ; ça n'implique donc pas un type de stockage physique.

3.2 Identifier les processus fonctionnels

Cette étape consiste à identifier l'ensemble des processus fonctionnels du morceau de logiciel à mesurer, à partir de ses FURs.

3.2.1 Définitions

DÉFINITION – Évènement
Évènement Quelque chose qui arrive.

DÉFINITION – Évènement déclencheur
Évènement déclencheur Un évènement, reconnu par la FUR du logiciel à mesurer, qui incite un ou plusieurs utilisateurs fonctionnels du logiciel à générer un ou plusieurs groupes de données, dont chacun sera, par la suite, déplacé par une entrée de déclenchement. Un évènement déclencheur ne peut pas être subdivisé et est soit arrivé ou pas. REMARQUE : Une horloge et ses évènements temporels peuvent être des évènements déclencheurs.

DÉFINITION – Processus fonctionnel
a) Un ensemble de mouvements de données représentant une partie élémentaire des FURs pour le logiciel à mesurer, qui est unique à cette FUR et qui peut être défini indépendamment de tout autre processus fonctionnel dans cette FUR. b) Un processus fonctionnel a seulement un évènement déclencheur. Chaque processus fonctionnel démarre le traitement dès réception d'un groupe de données déplacé par l'Entrée de déclenchement du processus fonctionnel. c) L'ensemble de tous les mouvements de données d'un processus fonctionnel est l'ensemble qui est nécessaire pour satisfaire sa FUR pour toutes les réponses possibles par l'entrée de déclenchement. Remarque 1: Lors de l'implémentation, c'est une occurrence d'un processus fonctionnel qui déclenche l'exécution, et ce à la réception d'une occurrence d'un groupe de données déplacé par une occurrence de l'Entrée de déclenchement. Remarque 2: La FUR pour un processus fonctionnel peut exiger une ou plusieurs autres entrées en plus de l'Entrée de déclenchement. Remarque 3: Si un utilisateur fonctionnel envoie un groupe de données avec des erreurs, par exemple parce qu'un capteur-usager a un mauvais fonctionnement ou une commande d'entrée par un être humain comporte des erreurs, c'est habituellement la tâche du processus fonctionnel de déterminer si l'évènement a vraiment eu lieu et/ou les données entrées sont vraiment valables. Le processus fonctionnel indique aussi comment réagir.

DÉFINITION – Entrée de déclenchement
Le mouvement de donnée 'Entrée' d'un processus fonctionnel qui déplace un groupe de données, généré par un utilisateur fonctionnel, nécessaire au processus fonctionnel pour son démarrage.

La relation entre une entrée de déclenchement, l'utilisateur fonctionnel et le mouvement de données d'entrée qui déclenche un processus fonctionnel à mesurer, est représentée à la Figure 3.3 ci-dessous. L'interprétation de ce diagramme est : un évènement déclencheur incite un utilisateur fonctionnel à générer un groupe de données qui est déplacé par l'entrée de déclenchement d'un processus fonctionnel pour démarrer le processus fonctionnel.

Remarque : Pour faciliter la lecture, nous avons souvent omis la référence au groupe de données et indiquer qu'un utilisateur fonctionnel amorce une entrée de déclenchement qui démarre un processus fonctionnel, ou encore plus simplement qu'un utilisateur fonctionnel enclenche un processus fonctionnel.

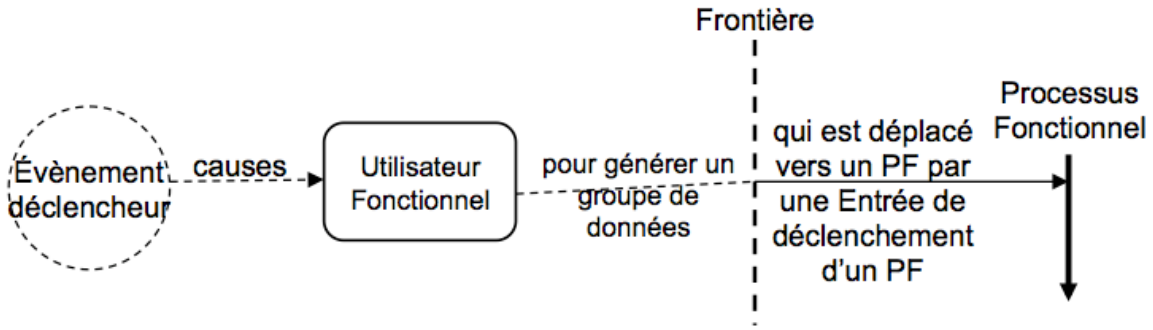


Figure 3.3 –Relations entre un évènement déclencheur, un utilisateur fonctionnel et un processus fonctionnel

Toutes les relations entre les concepts à la Figure 3.3 (l'évènement déclencheur / utilisateur fonctionnel / Entrée de déclenchement / processus fonctionnel) peut être un à plusieurs, plusieurs à un ou plusieurs à plusieurs, à une exception près. L'exception est que le groupe de données proposé par toute Entrée de déclenchement peut initier qu'un seul processus fonctionnel – voir la règle b) pour un processus fonctionnel. Quelques exemples de cardinalités possibles :

- Un évènement déclencheur peut être détecté par de nombreux utilisateurs fonctionnels, par exemple un tremblement de terre est détecté par de nombreux capteurs;
- Certains types d'utilisateurs fonctionnels peuvent détecter de nombreux types d'évènements, par exemple des opérateurs humains d'une application d'affaires qui initialisent une entrée de déclenchement pour chaque évènement considéré comme un évènement déclencheur;
- Un utilisateur fonctionnel 'matériel' peut amorcer plus d'une entrée de déclenchement en détectant un évènement déclencheur dans certains types de logiciels critiques pour la sécurité;
- Certains processus fonctionnels peuvent être initiés par différents utilisateurs fonctionnels, par exemple un composant de logiciel réutilisable qui peut être appelé par un de ses utilisateurs fonctionnels du logiciel.

Dans la pratique, les cardinalités tout au long de l'enchainement du logiciel à mesurer (c'est-à-dire qui décrivent les évènements précis qui emmènent des utilisateurs fonctionnels précis à lancer des processus fonctionnels précis) seront limitées par le logiciel de la FUR. Pour une discussion plus complète sur les cardinalités tout au long de cet enchainement voir la Figure 3.3 et pour obtenir d'autres exemples, voir l'annexe C du présent Manuel de mesurage.

Le(s) groupe(s) de données généré(s) par un utilisateur fonctionnel suite à un évènement déclencheur dépend(ent) de la FUR du processus fonctionnel qui traitera les données, comme le montrent les exemples suivants.

> *Exemple d'affaires 1: Un processus fonctionnel, d'un système de logiciel du personnel, peut être démarré par l'entrée de déclenchement qui se déplace d'un groupe de données décrivant un nouvel employé. Le groupe de données est généré par un utilisateur fonctionnel humain qui entre les données du logiciel du personnel. Normalement, l'évènement d'un nouvel employé qui entreprend son travail n'a pas d'intérêt en soi. Les données saisies fournissent des informations sur l'employé ; ce n'est pas sur l'évènement du début du travail. (Toutefois, la date de début du travail sera très probablement enregistrée en tant qu'attribut dans le groupe de données employé.)*

Exemple temps-réel 2: Un processus fonctionnel d'un système logiciel en temps réel peut démarrer par une entrée de déclenchement pour informer le processus fonctionnel qu'une horloge (utilisateur fonctionnel) a avancé. Le groupe de données déplacé transmet une donnée (le tic tac, peut-être via un seul bit) qui a pour seul but d'informer le détecteur qu'un évènement s'est produit.

Exemple temps-réel 3: Un processus fonctionnel d'un système de logiciel temps-réel industriel de détection d'incendie peut être démarré par son entrée de déclenchement initiée par un détecteur de fumée spécifique (utilisateur fonctionnel). Le groupe de données généré par le détecteur transmet l'information 'fumée détectée" (un évènement s'est produit) et inclut l'ID du détecteur (c'est-à-dire les données qui peuvent être utilisées pour déterminer où l'évènement s'est produit).

Exemple temps-réel 4: Un lecteur de code-barre (un utilisateur fonctionnel) à une caisse de supermarché lance une numérisation lorsqu'un code à barres s'affiche dans sa fenêtre (l'évènement déclencheur). Le lecteur génère un groupe de données, comprenant une image du code à barres qui est entré dans le logiciel à la caisse. L'image du groupe de données est déplacée par une entrée de déclenchement vers son processus fonctionnel. Cette dernière ajoute le cout du produit à la facture du client si le code est valide, un « bip » informe le client que le produit a été accepté et la vente s'enregistre, etc.

3.2.2 Approche à l'identification des processus fonctionnels

L'approche à l'identification des processus fonctionnels dépend des artéfacts logiciels qui sont disponibles au moment du mesurage. Ces derniers dépendent à leur tour de la phase du cycle de vie du logiciel à laquelle on se trouve au moment du mesurage, et dépendent aussi, finalement des méthodes d'analyse, de conception et de développement du logiciel utilisés. Puisque ces méthodes changent souvent ce *Manuel de mesurage* peut seulement fournir un processus général pour identifier des processus fonctionnels.

Le processus d'identification des processus fonctionnels, après l'identification des utilisateurs fonctionnels, et compte tenu de la FUR pour le logiciel à mesurer, suit la séquence de la Figure 3.3, c'est-à-dire:

- Identifier les évènements distincts dans le monde des utilisateurs fonctionnels auxquels le logiciel à mesurer doit répondre pour – les 'évènements déclencheurs' peuvent être identifiés par les diagrammes d'État et les diagrammes d'entité du cycle de vie, puisque certaines transitions d'état et les transitions du cycle de vie correspondent aux évènements déclencheurs auxquels le logiciel doit réagir);
- Identifier quel(s) utilisateur(s) fonctionnel du logiciel peut répondre à chaque évènement déclencheur;
- Identifier l'entrée de déclenchement (ou les entrées) que chaque utilisateur fonctionnel peut amorcer en réponse à l'évènement ;
- Identifier le processus fonctionnel démarré par chaque entrée de déclenchement.

Utilisez les règles suivantes pour vérifier que les processus fonctionnels candidats ont été correctement identifiés.

RÈGLES – Processus fonctionnel
a) Un processus fonctionnel doit appartenir entièrement à un morceau logiciel *dont le périmètre de mesurage a été défini*, et ce, pour une seule et unique couche.
b) Toute Entrée de déclenchement d'un morceau de logiciel devant être mesurer peut initier seulement un processus fonctionnel de ce logiciel.
c) Un processus fonctionnel contient au moins deux mouvements de données, une en Entrée plus une en Sortie et/ou une en éCriture Il n'y a pas de limite supérieure au nombre de mouvements de données dans un processus fonctionnel.
d) l'exécution d'un processus fonctionnel doit être considéré comme terminé lorsqu'il a satisfait aux exigences de la FUR en réponse à une Entrée de déclenchement. Une pause pendant l'exécution pour des raisons techniques ne doit pas être considéré comme étant la terminaison du processus fonctionnel.

3.2.3 Évènements déclencheurs et processus fonctionnels dans le domaine des applications d'affaires

a) Les évènements déclencheurs d'une application de gestion en ligne se produisent habituellement dans le monde réel des utilisateurs fonctionnels humains de l'application. L'utilisateur humain transmet l'occurrence de l'évènement à un processus fonctionnel en entrant les données relatives à l'évènement.

Exemple d'affaires 1: Dans une entreprise, une commande est reçue (évènement déclencheur), incitant un employé (utilisateur fonctionnel) à entrer les données de commande (entrée de déclenchement véhiculant les données relatives à l'objet d'intérêt 'commande'), en tant que premier mouvement de données du processus fonctionnel 'entrer la commande'.

b) Des évènements déclencheurs séparés (-types) donc des processus fonctionnels (-types) séparés doivent être distincts dans les cas suivants:

Lorsqu'un utilisateur fonctionnel humain prend des décisions à l'extérieur du logiciel sur 'ce qui doit être fait ensuite', décisions qui sont indépendantes dans le temps et qui exigent des réponses distinctes du logiciel, chaque décision distincte est un évènement déclencheur séparé pour lequel le logiciel doit fournir un processus fonctionnel distinct.

Exemple d'affaires *2: L'utilisateur fonctionnel entre une commande du client pour un article d'équipements industriels complexes et plus tard confirme l'acceptation de la commande au client. Entre l'inscription de la commande et l'acceptation, l'utilisateur peut se renseigner sur la nouvelle commande à savoir si elle peut être livrée en temps, sur la solvabilité du client, etc. Bien que l'acceptation d'une commande succède à l'entrée d'une commande, dans ce cas l'utilisateur doit prendre une décision distincte avant d'accepter la commande. Cela indique des processus fonctionnels distincts pour la saisie des commandes et pour l'acceptation de la commande.*

Quand les responsabilités pour les activités sont séparées.

Exemple d'affaires *3: Dans un système 'personnel' où la responsabilité du maintien de renseignements de base sur le personnel est séparée de la responsabilité du maintien des données de paie, cette séparation des utilisateurs fonctionnels indique que chacun à ses propres processus fonctionnels.*

c) Parfois, pour une application A il peut y avoir une application de même niveau B qui doit envoyer ou obtenir des données de l'application A. L'application B déclenche un processus fonctionnel dans l'application A lorsqu'il doit envoyer des données ou obtenir des données de l'application A. L'application B est alors un utilisateur fonctionnel de l'application A.

Exemple d'affaires 4: Supposons que à la réception d'une commande dans l'exemple d'affaires 1, l'application 'commande' est tenue de faire parvenir les détails de tout nouveau client à une application 'inscription centrale du client' à mesurer. L'application est donc un utilisateur fonctionnel de l'application centrale. L'application de la commande, ayant reçu des données sur un nouveau client, génère le groupe de données 'client' qui l'envoie à l'application 'inscription centrale du client' ce qui déclenche un processus fonctionnel pour stocker ces données.

d) Il n'y a aucune différence en principe dans l'analyse d'un processus fonctionnel qu'il soit traité en direct ou en lot. Par définition, toutes les données saisies comme entrée lors du traitement en lot doivent être temporairement stockées quelque part avant que le processus puisse commencer. Voir Figure 3.4 (N.B. nous distinguons des données persistantes qui devraient également être lues ou écrites par le traitement en lot des données d'entrée - toutes les entrées -.) Lorsque vous mesurez une application de traitement en lot, toutes les données d'entrée temporaires qui sont stockées doivent être analysées de la même manière, comme si elles étaient entrées directement dans l'application. Ces données d'entrée ne sont pas des 'données persistantes'.

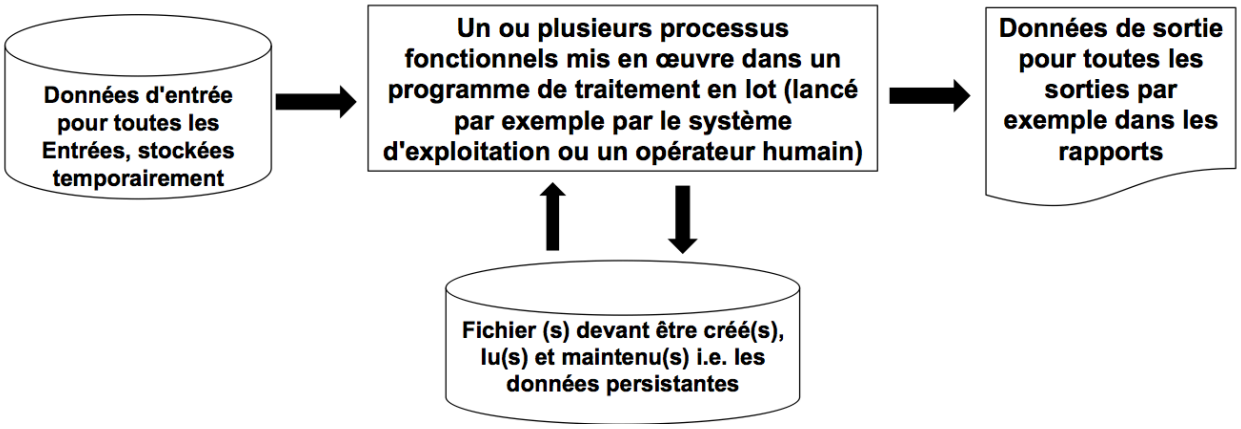


Figure 3.4 – Un travail de traitement en lot qui implémente un ensemble de processus fonctionnels

Remarque: Exiger que certaines données d'entrée soient traitées en lot est une exigence non fonctionnelle. L'effet de cette exigence non fonctionnelle est que les données d'entrée doivent être disponibles (comme un 'lot') pour l'entrée de l'application en lot. Comment cela se produit dans la pratique ne concerne pas l'analyse de la FUR de l'application du traitement des commandes.

Notez également que chaque processus fonctionnel (-type) traité dans un travail en lot doit être analysé dans son intégralité, indépendamment de tout autre processus fonctionnel dans le même travail. Chaque processus fonctionnel dans un travail doit avoir sa propre entrée de déclenchement.

Exemple d'affaires 5: supposons que les commandes de l'exemple d'affaires 1 ci-dessus sont effectuées par un processus 'host ligne' d'une façon quelconque, par exemple par balayage optique de documents papier pour être ensuite stockées temporairement lors d'un traitement automatisé en lot. Comment devrait-on analyser le traitement fonctionnel en lot? L'utilisateur fonctionnel est un humain qui déclenche les données de commande à inscrire hors ligne prêtes à être traitées en lot ; l'entrée de déclenchement du processus fonctionnel qui traitera les commandes en lot est le mouvement de données qui déplace le groupe de données de commande à partir du stockage temporaire vers le processus. (Le processus hors ligne, s'il doit être mesuré, implique un processus fonctionnel distinct. En effet, l'utilisateur fonctionnel a initié deux entrées de déclenchement, l'un démarre le processus hors ligne pour charger les commandes de stockage temporaire et l'autre démarre le traitement en lot des commandes.)

Exemple d'affaires 6: supposons une FUR pour le traitement en lot du compte des résultats annuels de fin d'année de l'entreprise ainsi que la réinitialisation des compteurs pour le début de l'année suivante. Physiquement, le tic tac de l'horloge en fin de l'année, déclenché par le système d'exploitation, est la cause du déclenchement du processus dans l'application. Logiquement, cependant, chaque processus fonctionnel de l'application tire ses données d'entrée du flux de données devant être traité en lot. Cela doit être analysé d'une façon normale (par exemple les données d'entrée pour tout un processus fonctionnel comprennent une ou plusieurs entrées, dont la première est l'entrée de déclenchement du processus).

Supposons encore un processus fonctionnel particulier de traitement en lot de l'application qui ne nécessite pas toutes les données d'entrée pour produire ses rapports. Physiquement, l'utilisateur fonctionnel (humain) a délégué au système d'exploitation la tâche de déclencher ce processus fonctionnel. Étant donné que chaque processus fonctionnel doit avoir une entrée de déclenchement, nous pouvons considérer que le clic de fin d'année de l'horloge, qui a démarré le flux de commandes, a accompli sa tâche pour ce processus. Ce processus fonctionnel peut alors avoir besoin de plusieurs lectures et plusieurs sorties pour produire ses rapports. Logiquement, l'analyse de cet exemple n'est pas différente si l'utilisateur fonctionnel (humain) lance la production de l'un ou plusieurs rapports via une souris en cliquant sur un élément de menu en ligne, plutôt que de déléguer le déclenchement du rapport de production en lot au système d'exploitation.

Pour prendre connaissance de plusieurs exemples distincts de déclenchement d'évènements et de processus fonctionnels dans les traitements en lot, voir 'Guideline for Sizing Business Application Software using COSMIC' [7], section 4.6.3.

3.2.4 Évènements déclencheurs et processus fonctionnels des applications en temps réel

a) Un évènement déclencheur est typiquement détecté par un capteur.

Exemple temps réel 1: Lorsqu'un capteur (utilisateur fonctionnel) détecte que la température atteint une certaine valeur (évènement de déclenchement), le capteur envoie un signal pour initier un mouvement de données d'entrée de déclenchement du processus fonctionnel 'éteindre un appareil de chauffage'.

b) *Exemple temps réel 2: Un avion militaire dispose d'un capteur qui détecte l'évènement "missile approchant". Le capteur est un utilisateur fonctionnel du logiciel qui doit répondre à la menace. Pour ce logiciel, un évènement se produit uniquement lorsque le capteur détecte quelque chose, et c'est le capteur (l'utilisateur fonctionnel) qui génère un groupe de données pour lancer une entrée de déclenchement indiquant, par exemple, que le 'capteur 2 a détecté un missile', en plus, peut-être, de transmettre un flux de données relatives à la vitesse d'approche du missile et ses coordonnées.*

c) Un signal périodique à partir d'une horloge ('tic tac de l'horloge') peut déclencher un processus fonctionnel.

Voir l'exemple 2 de temps réel dans la section 3.2.1. Il n'y a aucune autre donnée d'accompagnement du tic tac de l'horloge. Le processus fonctionnel obtient des données de divers capteurs et prend toutes les mesures nécessaires.

3.2.5 Un peu plus sur les processus fonctionnels distincts

Le logiciel distingue les évènements et fournit les processus fonctionnels correspondants qui dépendent uniquement de sa FUR. Lors du calcul de la taille des logiciels, il peut parfois être difficile de décider entre des évènements distincts de ceux que le logiciel doit reconnaitre. C'est notamment le cas où les exigences originales ne sont plus disponibles et où, par exemple le développeur peut avoir trouvé plus pratique de mettre ensemble, en une seule transaction physique, plusieurs exigences. Il peut être utile pour l'analyse d'examiner l'organisation de la saisie des données (voir ci-dessous), ou d'examiner les menus des logiciels déjà installés, pour aider à distinguer les évènements distincts auxquels le logiciel doit répondre ainsi que les processus fonctionnels correspondants.

Exemple d'affaires 1: Supposons qu'il y a une FUR pour deux types de prestations sociales, le premier pour un autre enfant et le second pour le 'travail crédit d'impôt' pour les personnes à faible revenu. Ces exigences du logiciel répondent à deux évènements séparés pour les utilisateurs fonctionnels (humains). C'est pourquoi il devrait y avoir deux processus fonctionnels, même si un formulaire unique d'impôt peut être utilisé pour fournir les données pour les deux cas.

Conformément à l'alinéa c) de la définition d'un processus fonctionnel, il doit 'rencontrer sa FUR pour toutes les réponses possibles à son entrée de déclenchement'. Cela signifie que le même type de processus fonctionnel doit être en mesure de tenir compte de toutes les occurrences des valeurs des attributs des données du groupe de données proposé par son entrée de déclenchement, y compris les deux valeurs de données valides et non valides et même dans certains cas, des données manquantes. Toutes ces variations des valeurs des données proposées par l'entrée de déclenchement génèrent généralement un traitement avec différents chemins lorsque le processus s'exécute. Mais en dépit de toutes ces variations, il faut toujours identifier uniquement le type de processus fonctionnel démarré par son entrée de déclenchement. (Note: Le mesureur doit seulement trouver tous les mouvements de données de ce processus fonctionnel; les différents chemins de traitement ne sont pas pertinents à la mesure.)

Exemple d'affaires 1: Un processus fonctionnel qui fournit une fonctionnalité de recherche générale sur une base de données peut devoir accepter jusqu'à quatre paramètres de recherche (attributs de son entrée de déclenchement). Mais le même processus fonctionnel fonctionnera seulement si une seule des trois valeurs des paramètres de la recherche est entrée.

Exemple d'affaires 2: Pour un processus fonctionnel visant à enregistrer un nouveau client pour une entreprise de location de voiture, il est obligatoire d'entrer des données pour la plupart des attributs de données, mais certaines (par exemple des données de contacts) sont facultatives et peuvent être laissées en blanc. Peu importe si la totalité ou un sous-ensemble de ces attributs sont entrés, il n'y a qu'un seul processus fonctionnel pour l'enregistrement d'un nouveau client. De même, pour le processus fonctionnel' faire une réservation de location de voiture dans la même entreprise', il y a plusieurs options qui peuvent ou ne peuvent être prises en compte, par exemple l'assurance supplémentaire, de nouveaux pilotes, des demandes de sièges d'enfant, etc.. Ces différentes options conduisent à un traitement différent des chemins dans le processus fonctionnel de réservation location voiture, mais il n'y a toujours qu'un seul processus fonctionnel pour la réservation d'une location de voiture.

Exemple temps réel: Une entrée de déclenchement (information sur l'altitude de l'avion envoyée par le système de positionnement géographique) vers un processus fonctionnel d'un système d'avionique va déclencher un ou deux chemins de traitement différents dans le processus fonctionnel selon la valeur de l'entrée, par exemple si l'altitude est supérieure ou inférieure à une hauteur donnée. Les différents chemins afficheront les différents groupes de données sur la carte du pilote. Si l'altitude est trop basse, des avertissements supplémentaires seront délivrés. Il n'y a qu'un seul processus fonctionnel.

Une fois identifié, chaque processus fonctionnel peut être enregistré sur une ligne individuelle, sous la couche appropriée et du morceau identifié de logiciel, dans la matrice du modèle générique de logiciel (annexe A).

3.2.6 Mesurer les composantes d'un logiciel distribué

Lorsqu'une la raison d'être d'un mesurage est du mesurage séparément la taille de chaque composant d'un logiciel distribué, un périmètre distinct pour le mesurage doit être définie pour chaque composant. Dans ce cas, la taille des processus fonctionnels de chaque composant suit toutes les règles déjà décrites.

À partir du processus de chaque mesurage (... définir le périmètre, les utilisateurs fonctionnels, les frontières, etc.) il s'ensuit que si un morceau de logiciel se compose de deux composants ou plus, il ne peut y avoir de chevauchement entre le périmètre de chaque composant. Le périmètre pour chaque composant doit définir un ensemble de processus fonctionnels complets. Par exemple, il ne peut y avoir un processus fonctionnel compris partiellement dans un périmètre et partiellement dans un autre. De même, les processus fonctionnels, dans le périmètre d'une composante, n'ont aucune connaissance des processus fonctionnels dans le périmètre d'un autre composant, même si les deux composants échangent des messages.

Le(s) utilisateur(s) fonctionnel(s), de chaque composant, est déterminé(s) en observant où les évènements déclencheurs des processus fonctionnels se produisent dans le composant en cours d'examen. (Les évènements déclencheurs ne peuvent apparaitre que dans l'univers d'un utilisateur fonctionnel.)

La figure 3.6 illustre les processus fonctionnels des deux composantes d'un système client-serveur distribuées et les mouvements de données qu'ils échangent.

3.2.7 Indépendance des processus fonctionnels partageant certaines fonctionnalités communes ou similaires

N'importe quel processus fonctionnel de deux ou plusieurs logiciels à mesurer peut avoir certaines fonctionnalités qui sont identiques ou très similaires dans chaque processus. Ce phénomène est dénommé 'partage fonctionnel' ou 'similarité fonctionnelle'.

Toutefois, dans la méthode COSMIC (comme dans toutes les autres méthodes du mesurage fonctionnelles) chaque processus fonctionnel est défini, modélisé et mesuré indépendamment, c'est-à-dire sans mention d'un ou de tout autre processus fonctionnel dans le même logiciel à mesurer (voir l'article 'a' de la définition d'un processus fonctionnel). Toute fonctionnalité requise commune ou similaire par deux ou plusieurs processus fonctionnels, d'un logiciel à mesurer, doit être comptabilisée dans chacun de ces processus fonctionnels. Voici des exemples de similarité fonctionnelle que l'on peut trouver dans la pratique.

Exemple d'affaires: Plusieurs processus fonctionnels du même logiciel à mesurer peuvent avoir besoin de la même fonctionnalité de validation. Par exemple valider la 'date d'ordonnance' ou accéder aux mêmes données ou encore effectuer le même calcul d'intérêt.

Exemple temps réel: Plusieurs processus fonctionnels du même logiciel à mesurer peuvent avoir besoin d'obtenir des données du même capteur (mouvement du même groupe de données commun) ou pourraient devoir effectuer le même calcul de conversion d'échelle, par exemple de Fahrenheit à Celsius (manipulation de données commune).

Une déclaration de la FUR est implémentée dans le logiciel, une 'similarité fonctionnelle' peut ou ne peut pas être développée comme un logiciel réutilisable. L'ampleur de la réutilisation du logiciel réelle ou potentielle découlant de la similarité fonctionnelle peut donc être prise en compte lors de la taille fonctionnelle aux fins d'estimation de la performance du projet.

3.2.8 Évènements qui déclenchent un (système) logiciel pour lancer l'exécution

Quand vous mesurez la taille d'un morceau de logiciel, identifier uniquement les évènements et les entrées de déclenchement qui déclenchent les processus fonctionnels auxquels le logiciel doit répondre tel que défini dans sa FUR. Les fonctionnalités nécessaires à la mise en marche (ou 'lancement') du logiciel, ne font pas partie de ces processus fonctionnels et doivent être ignorées (ou mesurées séparément, le cas échéant). Les exemples suivants décrivent comment le logiciel démarre dans trois domaines.

Exemple d'affaires: Pour les applications d'affaires, l'utilisateur fonctionnel qui démarre l'application peut être un composant du planificateur du système d'opération, un opérateur de l'ordinateur, ou tout autre humain (par exemple lorsqu'un utilisateur PC lance un navigateur ou un logiciel de traitement de texte).

Exemple temps réel: Pour les applications en temps réel, l'utilisateur fonctionnel qui démarre l'application peut être le système d'exploitation ou le gestionnaire du réseau générant un signal d'horloge, ou un opérateur humain (par exemple pour démarrer le processus de contrôle du système à partir du poste d'un opérateur).

Exemple d'infrastructure: Pour un système d'opération, l'utilisateur fonctionnel qui démarre l'application est un programme d'entrainement qui est démarré quand l'ordinateur est allumé.

Voici des exemples de deux domaines sur les relations qui peuvent exister, le cas échéant, entre un évènement/processus qui démarre le logiciel à mesurer et les évènements/processus qui doivent exécuter le logiciel tel que décrit dans sa FUR.

Exemple d'affaires 1: Une application pour traiter les données d'entrée pour une variété de processus fonctionnels de traitement en lot peut être démarrée par un planificateur du système d'exploitation. Si la raison d'être est du mesurage la FUR du traitement en lot, la fonctionnalité 'démarrer-le-système' doit être ignorée. L'entrée de déclenchement des processus fonctionnels de traitement en lot et toutes les autres entrées qui peuvent être requises constitueront les données d'entrée pour l'application de traitement en lot.

Exemple d'affaires 2: Exceptionnellement, on peut démarrer une application de traitement en lot pour produire des rapports de synthèse, à la fin d'une période, sans avoir besoin de toutes les données d'entrée fournies directement à partir de l'utilisateur fonctionnel. Pour l'analyse, voir l'exemple d'affaire 6 au point 3.2.3.

Exemple temps réel: Un véhicule moderne possède un système distribué d'unités de contrôle électronique (UCEs) pour contrôler plusieurs fonctions, par exemple le moteur de direction, les freins, la climatisation, etc.. Dans l'architecture AUTOSAR, dans un système distribué, le module "Gestion de réseau" (GR), qui est toujours en cours d'exécution, est responsable de l'activation des UCEs qui sont reliées entre eux via un réseau ('bus'). Ce module gère également la commutation coordonnée entre les États de marche UCE: de fonctionnement normal, de faible puissance et de sommeil. Par

conséquent, c'est le GR qui réveille ou met en sommeil des UCEs. Lors du mesurage de n'importe quel logiciel d'application UCE, cette fonctionnalité GR doit être ignorée.

3.3 Identification des objets d'intérêts et des groupes de données.

3.3.1 Définitions et principes.

Cette étape consiste à identifier les groupes de données référencés par le morceau de logiciel à mesurer. Pour identifier les groupes de données, particulièrement dans le domaine des logiciels d'application d'affaires, il est habituellement utile d'identifier les objets d'intérêt et probablement aussi leurs attributs. Les groupes de données sont déplacés par des 'mouvements de données' tels que présentés dans le prochain chapitre.

DÉFINITION – Objet d'intérêt.
Toute 'chose' qui est identifiée du point de vue des Fonctionnalités Utilisateurs Requises (FUR). Ce peut être une chose physique, aussi bien qu'un objet conceptuel ou parti d'un objet conceptuel dans le monde de l'utilisateur fonctionnel pour lequel un logiciel est requis pour un traitement et/ou un stockage de données. REMARQUE : Dans la méthode COSMIC, le terme 'objet d'intérêt' est utilisé pour éviter d'utiliser des termes reliés à des méthodologies spécifiques du génie logiciel. Le terme ne correspond pas nécessairement au terme 'objet' selon le sens utilisé dans les méthodes orientées objet.

DÉFINITION – Groupe de données.
Tout ensemble distinct, non vide, non ordonné et non redondant de types d'attributs de données où chaque type d'attribut de donnée décrit un aspect complémentaire du même objet d'intérêt.

DÉFINITION – Stockage persistant.
Un type de stockage qui permet à un processus fonctionnel de conserver des données au-delà de la vie du processus fonctionnel et/ou qui permet à un processus fonctionnel de retrouver une donnée stockée par un autre processus fonctionnel, ou stockée plus tôt au cours d'une occurrence du même processus fonctionnel ou stockée par un autre processus. REMARQUE 1 : Dans le modèle COSMIC, puisque la partie de stockage persistant est du côté logiciel de la frontière, elle n'est pas considérée comme un utilisateur du logiciel mesuré. REMARQUE 2 : Un exemple d'un autre type de stockage persistant est la fabrication de mémoires à lecture seulement.

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

PRINCIPES – Groupe de données.
a. Chaque groupe de données identifié doit être unique quant à l'ensemble des attributs qu'il contient ; b. Chaque groupe de données doit être directement lié à un objet d'intérêt décrit dans les FUR du logiciel à mesurer ; c. Le groupe de données doit être matérialisé dans le système informatique supportant le logiciel.

Après avoir été trouvé, chaque groupe de données est enregistré dans une colonne individuelle de la matrice du modèle générique de logiciel (voir annexe A), sous l'étiquette correspondante.

3.3.2 À propos de la matérialisation des groupes de données.

En pratique, la matérialisation d'un groupe de données peut prendre plusieurs formes, par exemple :

a. Comme structure physique d'un enregistrement sur un dispositif de stockage persistant (c.-à-d. un dossier, une table de base de données, la mémoire morte (ROM), etc.) ;
b. Comme structure physique dans la mémoire vive (RAM), de l'ordinateur (c.-à-d. la structure de données a assigné dynamiquement ou par un bloc préaffecté d'espace mémoire) ;
c. Comme présentation groupée des attributs de données, fonctionnellement reliés sur une interface-personne – machine (IPM) ou autre dispositif d'interfaçage (c.-à-d. écran de visualisation, rapport imprimé, affichage de panneau de commande, etc.) ;
d. Comme message étant transmis entre un dispositif physique quelconque et un ordinateur, via un réseau, etc.

3.3.3 À propos de l'identification des objets d'intérêt et des groupes de données.

La définition et les principes supportant les objets d'intérêt et leurs groupes de données sont intentionnellement génériques en vue d'être applicable à un plus grand nombre possible de types de logiciels. Cette 'généricité' a comme inconvénient de rendre plus difficile l'application du mesurage sur un morceau de logiciel particulier. Les règles suivantes issues de la pratique du mesurage fonctionnel peuvent aider à résoudre certains cas particuliers.

Lorsqu'on est confronté à un besoin d'analyser un groupe d'attributs de données déplacé à partir, ou vers un processus fonctionnel, ou encore déplacé par un processus fonctionnel à partir, ou vers un stockage persistant, il s'avère critique de déterminer si tous ces attributs communiquent des données à un unique 'objet d'intérêt' car cet 'objet d'intérêt' déterminera le nombre de 'groupes de données' distincts tel que défini par la méthode COSMIC. Par exemple, si les attributs de données qui entrent dans un processus fonctionnel sont des attributs de trois objets d'intérêt distincts, alors on devra identifier trois mouvements de données en entrée distincts.

Objets d'intérêts et groupes de données dans le domaine des applications d'affaires

- Exemple d'affaires 1: Dans le domaine des logiciels d'application d'affaires, supposons un logiciel permettant de stocker des données au sujet des employés ou des commandes : un objet d'intérêt pourrait alors être un 'employé d'entreprise' (individu physique) ou une 'commande d'articles' (conceptuel). Dans le cas de la 'commande', il arrive généralement qu'une FUR de commandes 'multi lignes' soit à l'origine, c.-à-d. qu'il y ait deux objets d'intérêt identifiés plutôt qu'une, c.à.-d.: 'commande' et 'ligne de commande'. Les groupes de données correspondants pourraient alors être : 'données de commande', et 'ligne de donnée de commande' ;

Des groupes de données sont constitués toutes les fois qu'il y a une requête ad hoc qui demande des données sur une certaine 'chose'. La particularité est que ces données ne sont pas toujours contenues dans le stockage persistant, mais peuvent toutefois être dérivées des données contenues dans le stockage persistant. Les mouvements de données en entrée d'une requête ad hoc (soient les paramètres de choix pour dériver les données requises) ainsi que les mouvements de données en sortie (contenant les attributs désirés) déplacent tous les deux des groupes de données à propos de cette 'chose'. Ce sont des groupes de données temporaires qui ne survivent pas à l'exécution du processus fonctionnel. Il reste que ces groupes de données sont valides parce qu'ils traversent la frontière entre le logiciel et son (ses) utilisateur(s).

- Exemple d'affaires 2 : Considérons une requête ad hoc pour d'extraire à partir de la base de données 'Employé' les noms de tous les employés de plus de 35 ans. Le mouvement de données en entrée déplace un groupe de données contenant les paramètres choisis. Le mouvement de données Sortie déplace un groupe de données contenant seulement l'attribut 'nom' ; l'objet d'intérêt (ou 'la chose') est représenté par 'tous les employés âgés plus de 35 ans'. Il est important en enregistrant le processus fonctionnel d'identifier clairement un groupe de données temporaire par rapport à son objet d'intérêt, plutôt que de le relier ce groupe de données à l'objet d'intérêt (ou aux objets) lui-même, duquel le résultat de la requête ad hoc est dérivé.

Pour une discussion détaillée sur les méthodes d'analyse des données pour déterminer des objets d'intérêt et des groupes de données distincts, le lecteur doit se référer au '*Guideline for Sizing Business Application Software'*.

Objets d'intérêts et groupes de données dans le domaine du temps réel

Exemples d'affaires 3: Les mouvements de données qui sont en entrée de dispositifs physiques contiennent souvent des données sur l'état d'un objet d'intérêt, comme une valve dont l'état est « ouvert » ou « fermée » par exemple. Ces mouvements de données indiquent souvent, aussi, les délais signalant à court terme si des données temporaires sont valides ou non. Ces mêmes mouvements de données indiquent si un événement critique s'est produit, causant ainsi un blocage. Réciproquement, des mouvements de données en sortie, on obtient souvent des données au sujet d'un objet d'intérêt unique, comme une commande pour allumer ou éteindre voyant lumineux d'avertissement ;

1. Un commutateur de messages peut recevoir un groupe de données en entrée et le transférer en sortie sans le modifier, selon les besoins de la FUR du morceau de logiciel en particulier. Les attributs de message du groupe de données peuvent être, par exemple, 'expéditeur, destinataire, code de route et contenu du message' ; l'objet d'intérêt de ce message est donc 'message' ;
2. Une structure de données similaire, représentant les objets d'intérêt qui sont mentionnés dans les FUR, laquelle peut être mise à jour par des processus fonctionnels, et pouvant être accessible par la plupart des processus fonctionnels trouvés dans le logiciel mesuré ;
3. Une structure de données référencée, représentant des graphiques ou des tables avec des valeurs trouvées dans les FUR, qui sont conservées dans le stockage persistant (mémoire ROM, par exemple) et accessible par la plupart des processus fonctionnels du logiciel mesuré ;
4. Fichiers généralement désignés comme 'Fichiers plats', représentant des objets d'intérêt mentionnés dans les FUR qui sont conservés sur un dispositif de stockage persistant.

3.3.4 Données ou groupes de données non éligibles aux mouvements de données.

Toute donnée apparaissant sur une interface d'entrée ou de sortie (écran, rapports) qui n'est pas liée à un objet d'intérêt d'un utilisateur fonctionnel, ne devrait pas faire part d'un mouvement de données, et ne devrait donc pas être mesurée.

Exemple 1: Des données de type 'application générale' telles que des rubriques et titres de bas de page (nom de compagnie, nom d'application, date de système, etc.) apparaissant sur tous les écrans ;

Exemple 2: Des commandes de contrôle du logiciel (un concept défini seulement dans le domaine des logiciels d'application d'affaires) permettant à un utilisateur fonctionnel de contrôler son logiciel plutôt que de déplacer des données, telles que : des commandes permettant de feuilleter (page haut / page bas), des commandes de type « cliquer 'OK' pour accepter un message d'erreur », etc. - voir aussi la section 4.1.10.

Le modèle générique de logiciel COSMIC suppose que toute manipulation de données dans un processus fonctionnel est associée à un des quatre types de mouvements de données. - voir la section 4.1.6. Par conséquent, aucun mouvement ou manipulation de données dans un processus fonctionnel ne peut être identifié comme étant un mouvement de données autres que : Entrées, Sorties, Lectures ou éCritures. (Pour des exemples manipulation et de mouvements de données qui pourraient être mal interprétées, voir la section 4.1.4, principe (c)) pour la Lecture, et la section 4.1.5 (d)) pour l'eCriture).

3.3.5 L'utilisateur fonctionnel comme objet d'intérêt.

Dans beaucoup de cas simples de systèmes type temps réel, tel que décrit à l'exemple 3 de la section 3.3.3, le dispositif physique - un utilisateur fonctionnel – ne se distingue pas de l'objet d'intérêt du mouvement de données qu'il envoie ou reçoit. Dans ces cas-là, cela ajoute peu d'intérêt de documenter un objet d'intérêt comme si c'était quelque chose de distinct de l'utilisateur fonctionnel. Il est important d'employer ces concepts quand c'est utile, afin de distinguer des groupes de données distincts et par conséquent, de séparer les mouvements de données.

Exemple temps réel: Supposons un capteur de température 'A' qui, une fois interrogé par un processus fonctionnel, envoie la température courante de ce dernier. L'utilisateur fonctionnel est le capteur de température 'A' ; le nom de la donnée en entrée pourrait être 'la température courante A ' et finalement, l'objet d'intérêt de ce message pourrait également être considéré comme étant le capteur de température 'A'. Mais théoriquement, l'objet d'intérêt n'est pas le capteur de température 'A', mais bien 'la chose dont la température est mesurée par le capteur A'. En pratique cependant, effectuer ce type distinction raffinée ajoute peu de valeur, ce qui fait qu'il est peu ou pas intéressant d'identifier l'objet d'intérêt de manière distincte.

3.4 Identification des attributs (optionnel).

Cette section présente comment est réalisée une identification des attributs de données référencés, par le morceau de logiciel à mesurer. Dans cette version de la méthode de mesurage, il n'est pas obligatoire de trouver les attributs de données. Cependant, la compréhension du concept des 'données d'attributs' est nécessaire afin de comprendre la section sur la "mesure des modifications..." où une FUR modifiant un attribut de données peut résulter en un mouvement de données à mesurer. Cependant, il peut être utile analyser et identifier les attributs de données dans le processus de distinction des groupes de données et des objets. Par exemple, des attributs de données pourraient être trouvés si une sous unité du mesurage de taille était exigée, tel que présenté dans la section 4.5 *'Extension de la méthode de mesurage COSMIC*'.

3.4.1 Définition.

DÉFINITION – Attribut de donnée.
La plus petite parcelle d'information codée, dans un groupe de données, possédant une signification dans la perspective des Fonctionnalités Utilisateurs Requises (FUR) du logiciel.

Exemple d'affaires: Les attributs dans le contexte du domaine des logiciels d'applications d'affaires sortent par exemple des éléments de données enregistrés dans un dictionnaire de données et les éléments de données apparaissant dans un modèle conceptuel ou dans un modèle logique de données.

Exemple temps réel: Les attributs dans le contexte du domaine des logiciels en temps réel sont par exemple des éléments de donnés d'un signal reçu d'un capteur et des parties du contenu (d'adresse à adresse) d'un message de transmission.

3.4.2 À propos de l'association entre les attributs et les groupes de données.

En théorie, un groupe de données ne pourrait contenir qu'un seul attribut de donnée, et ce serait suffisant du point de vue de la FUR, afin de décrire l'objet d'intérêt. Dans la pratique, de tels cas se produisent généralement dans des logiciels de type temps réel (par exemple : Un mouvement de données en Entrée véhiculant un signal périodique d'une horloge) et sont moins communs dans les logiciels d'application d'affaires.

3.5 Identification des mouvements de données

Cette étape consiste à trouver les mouvements de données (Entrée, Sortie, Lecture et éCriture) de chaque processus fonctionnel.

3.5.1 Définition des types de mouvements de données

DÉFINITION – Mouvement de données
Un composant fonctionnel de base qui déplace un seul type de groupe de données.

REMARQUE 1 : Il y a quatre sous-types de mouvements de données: entrée (E), sortie (S), lecture (L), écriture (C).

REMARQUE 2 : Pour le besoin du mesurage, chaque sous-type de mouvement de données est considéré comme incluant un certain nombre de manipulations de données qui y sont associées – voir le Manuel de mesurage pour plus de détails.

REMARQUE 3 : Pour préciser, c'est l'occurrence d'un mouvement de données, non son type, qui en pratique déplace les occurrences d'un groupe de données. Cette remarque s'applique aussi aux définitions de: Entrée (E), Sortie (S), Lecture (L), Écriture (C).

DÉFINITION – Entrée (E).

Un type de mouvement de données qui déplace, à travers la frontière, un groupe de données depuis un utilisateur fonctionnel vers le processus fonctionnel où il est requis.

REMARQUE : Un type d'Entrée inclut aussi certaines manipulations de données associées - voir la section 3.5.6 pour plus de détails.

DÉFINITION – Sortie (S).

Un type de mouvement de données qui déplace, à travers la frontière, un groupe de données d'un processus fonctionnel vers l'utilisateur fonctionnel qui le demande.

REMARQUE : Une sortie type inclut aussi les manipulations de données associées - voir la section 3.5.6 pour plus de détails.

DÉFINITION – Lecture (L).

Un type de mouvement de données qui, dans le contexte de son processus fonctionnel, déplace un groupe de données depuis sa partie de stockage persistant pour le mettre à la portée du processus fonctionnel (-type) qui le requiert.

REMARQUE : Une Lecture inclut certaines manipulations associées de données nécessaires - voir la section 3.5.6 pour plus de détails.

DÉFINITION – ÉCriture (C).

Un type de mouvement de données qui déplace un groupe de données depuis un processus fonctionnel vers un stockage persistant.

REMARQUE : Un type d'écriture inclut certaines manipulations associées - voir la section 3.5.6 pour plus de détails.

La figure 3.5 ci-dessous, illustre la relation entre chacun des quatre types de mouvement de données et le processus fonctionnel auquel il appartient de même que la frontière du logiciel mesuré.

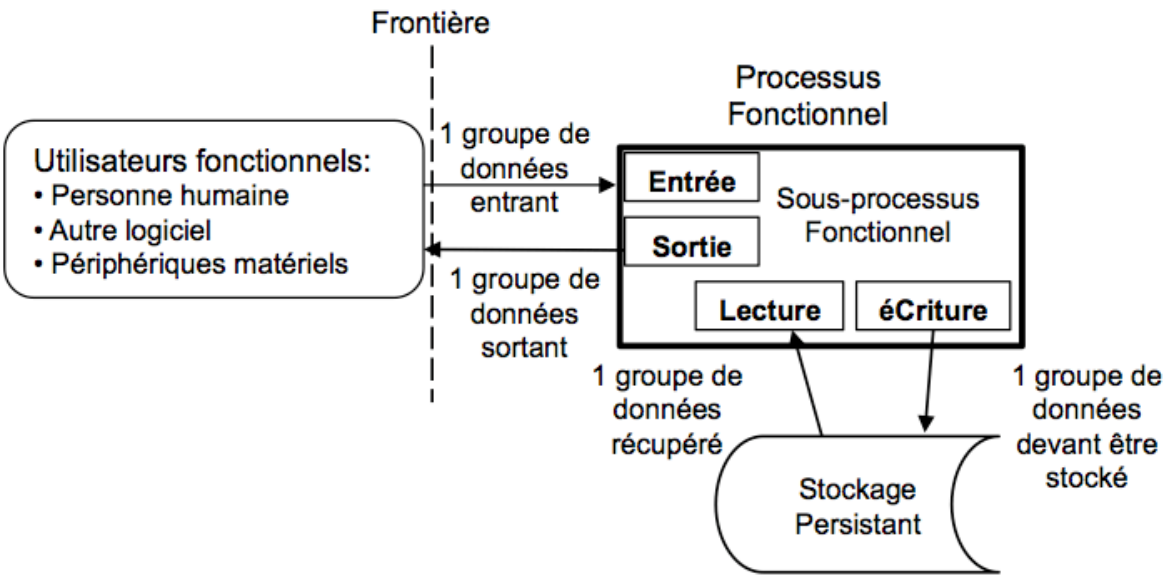


Figure 3.5 - Les quatre types de mouvement de données, chacun déplaçant un groupe de données en relation avec un processus fonctionnel. (Un processus fonctionnel peut, bien sûr, avoir plusieurs mouvements de données de type E, X, R et W.)

3.5.2 Identification des Entrées (E)

Une fois identifié, chaque mouvement de données en Entrée doit répondre aux principes suivants :

PRINCIPES – Entrée(E)
b) Une Entrée déplacera un seul groupe de données décrivant un seul objet d'intérêt d'un utilisateur fonctionnel, à travers la frontière du logiciel vers le processus fonctionnel auquel cette Entrée fait partie. Si l'Entrée d'un processus fonctionnel comporte plus d'un groupe de données, il faudra identifier une seule Entrée pour chaque groupe de données. (voir également la section 3.5.7 sur 'l'unicité des mouvements de données' ;
c) Une Entrée ne doit pas sortir des données au-delà de la frontière, ou lire ou écrire des données depuis/vers un stockage persistant.

Les règles suivantes aident à confirmer le statut 'd'Entrée' pour mouvement de données:

Règles – Entrée (E)
a. Un groupe de données d'une Entrée de déclenchement peut comporter un seul attribut de données informant tout simplement le logiciel qu'un événement 'Y' est survenu. Très souvent, plus particulièrement pour les logiciels d'application d'affaires, le groupe de données de l'Entrée de déclenchement a plusieurs attributs de données qui informent le logiciel qu'un événement 'Y' s'est produit et présente ainsi les données au sujet de cet événement particulier ;
b. Les signaux périodiques d'une l'horloge comme déclencheur d'événements sera toujours 'extérieur' au logiciel mesuré. Par conséquent lorsqu'une horloge émet un signal périodique pour un événement se produisant toutes les 3 secondes, ce signal sera associé à une Entrée déplaçant un groupe de données d'un seul attribut de données. Noter qu'il n'y a aucune différence si l'événement déclencheur est produit périodiquement par du matériel ou par un autre morceau de logiciel en dehors de la frontière du logiciel mesuré ;

c. À moins qu'un processus fonctionnel spécifique soit nécessaire, le fait d'obtenir le temps d'une horloge d'un système ne sera pas considéré comme la cause suffisante de caractérisation d'un mouvement de données en Entrée ;

d. Si une occurrence d'un évènement spécifique déclenche l'Entrée d'un groupe de données comportant 'N' attributs (d'un objet d'intérêt particulier), et que la FUR admet qu'une autre occurrence du même évènement puisse déclencher une Entrée d'un groupe de données ayant les même valeurs pour un sous-ensemble de ces 'N' attributs, alors une seule Entrée sera identifiée : celle comportant tous les 'N' attributs.

Lors de l'identification des Entrées dans un écran qui permet aux utilisateurs fonctionnels (humain) d'entrer des données dans les processus fonctionnels, par exemple dans une application d'affaires en ligne, d'analyser seulement des écrans qui contiennent des données. Ignorer 1) n'importe quel écran qui est formaté normalement 'vide', sauf pour les valeurs possibles par défaut et 2) ignorer tous les champs et autres rubriques permettant aux utilisateurs humains de comprendre les données d'entrée requises. (Pourquoi une remarque dans une boite de règles???) REMARQUE. Il peut être nécessaire d'envisager les champs et autres rubriques lors du mesurage de la FUR pour les modifications apportées aux entrées – voir la section 4.4.1.

Exemple d'affaires explicitant la règle c): Lorsqu'un processus fonctionnel ajoute un horodatage à un résultat qui deviendra persistant ou sortira (du processus), aucune Entrée n'est identifiée pour obtenir la valeur de l'horloge du système

Une fois identifié, chaque mouvement de données d'entrée peut être enregistré en cochant la case correspondante de la matrice du modèle générique de logiciel (annexe A) avec un 'E'.

3.5.3 Identification des Sorties (S))

Une fois identifié, chaque mouvement de données en Sortie doit répondre aux principes suivants :

PRINCIPES – Sortie (S).
a. Une Sortie déplacera un seul groupe de données décrivant un seul objet d'intérêt d'un processus fonctionnel, à travers la frontière du logiciel vers un utilisateur fonctionnel. Si la Sortie d'un processus fonctionnel comprend plus d'un groupe de données, il faudra identifier une seule Sortie pour chaque groupe de données. (voir également la section 4.1.7 sur 'l'unicité des mouvements de données'.) ; b. Une Sortie ne doit pas entrer des données à travers la frontière du logiciel, ni lire, ni écrire des données. à partir/vers un stockage persistent.

Les règles suivantes aident à confirmer le statut de 'Sortie' pour le mouvement de données:

RÈGLES – Sortie (S).
a. Une requête qui affiche le texte fixe (où « fixe » signifie le message ne contient aucune valeur de données variables, par exemple le résultat de la pression sur un bouton pour « Modalités » d'un site commerçant), doit être modélisée comme ayant une Sortie pour la visualisation du texte fixe. Remarque: Pour la sortie de la fonctionnalité 'Aide', voir 'Guideline for Sizing Business Application Software'. Pour la sortie des messages liés aux conditions d'erreurs ou de réussites confirmées, voir la section 3.5.11 de ce Manuel de mesurage. b. Si la Sortie d'un processus fonctionnel déplace un groupe de données comportant 'N' attributs (d'un objet d'intérêt particulier), et que la FUR admet qu'une autre occurrence du même événement puisse déclencher une Sortie d'un groupe de données ayant les mêmes valeurs pour un sous-ensemble de ces 'N' attributs, alors une seule Sortie sera identifiée : celle comportant tous les 'N' attributs. c. Lors de l'identification des Sorties, ignorer tous les champs et autres 'en-têtes' qui sont là

seulement pour permettre à des utilisateurs humains de comprendre les données de sortie.
Remarque: Il peut être nécessaire de considérer le champ et les autres rubriques lors du mesurage de la FUR pour les modifications apportées aux Sorties – voir la section 4.4.1.

Règles – Sortie (S)
a. Une requête qui sort du texte fixe (où « fixe » signifie le message ne contient aucune valeur de données variables, par exemple le résultat de la pression d'un bouton pour les 'Conditions générales' sur un site commerçant), doit être modélisée comme ayant une Sortie pour le texte fixe. Remarque : Pour la fonctionnalité 'AIDE' voir 'Guideline for sizing Business Application Software'. Pour la sortie des messages visés par les conditions d'erreur ou confirmation de réussite voyez la section 3.5.11 de ce Manuel de mesurage.
b) Si la Sortie d'un processus fonctionnel déplace un groupe de données comportant 'N' attributs (d'un objet d'intérêt particulier), et que la FUR admet qu'une autre occurrence du même évènement puisse déclencher une Sortie d'un groupe de données ayant les mêmes valeurs pour un sous-ensemble de ces 'N' attributs, alors une seule Sortie sera identifiée : celle comportant tous les 'N' attributs.
c) Lorsque de l'identification des Sorties, ignorer tout les champs et autres en-têtes qui ne visent qu'à permettre à des utilisateurs humains de comprendre les données de sortie. Remarque : Il peut être nécessaire d'envisager le champ et autres en-têtes lors du mesurage de la FUR pour les modifications apportées aux Sorties – voir la section 4.4.1

Une fois identifié, chaque mouvement de données de Sortie peut être enregistré en cochant la case correspondante de la matrice du modèle générique de logiciel (annexe A) avec un X.

Voir aussi la section 3.5.11 pour l'identification des mouvements de données de Sortie pour les messages d'erreur.

3.5.4 Identification des Lectures (L)

Une fois identifié, chaque mouvement de données en Lecture doit répondre aux principes suivants :

PRINCIPES – Lecture (L)
a) La Lecture déplacera un seul groupe de données décrivant un seul objet d'intérêt d'un stockage persistant, à travers la frontière du logiciel, vers le processus fonctionnel auquel cette Lecture fait partie. Si le processus fonctionnel doit lire plus d'un groupe de données à partir du stockage persistant, il faudra trouver une seule Lecture pour chaque groupe de données à lire. (voir également la section 3.5.7 sur 'l'unicité des mouvements de données') ;
b) La Lecture ne doit pas recevoir de données à travers la frontière du logiciel, ni sortir, ni écrire des données dans un stockage persistant;
c) Les mouvements (ou la manipulation) des donnés qui ne proviennent pas d'une FUR, mais qui sont internes à un processus fonctionnel et qui peuvent être changés seulement par : un programmeur, par un calcul de résultats intermédiaires, ou à partir de données conservées par un autre processus fonctionnel résultant de l'implantation, ne seront pas considérés comme étant un mouvement de données en Lecture ;
d) Un mouvement de données en Lecture inclura toujours toutes fonctionnalités de 'demande de Lecture'. Ainsi un mouvement de données additionnel ne sera jamais nécessaire pour représenter une éventuelle fonctionnalité distincte de 'demande de

Lecture'. Voir également la section 3.5.9

Les règles suivantes aident à confirmer le statut de 'Lecture' pour le mouvement de données:

Règles – Lecture (L)
a) Identifier une Lecture lorsque, selon la FUR, le logiciel à mesurer doit récupérer un groupe de données d'un stockage persistant. b) Ne pas identifier une Lecture lorsque la FUR du logiciel à mesurer spécifie n'importe quel logiciel ou matériel utilisateur fonctionnel comme source, ou comme moyen d'extraction, d'un groupe de données. (Dans ce cas, voir les principes et règles pour les Entrées et les Sorties.)

Une fois identifié, chaque mouvement de données Lecture peut être enregistré en cochant la case correspondante de la matrice du modèle générique de logiciel (annexe A) avec un 'L'.

3.5.5 Identification des éCritures (C)

Une fois identifié, chaque mouvement de données en éCriture doit répondre aux principes suivants:

PRINCIPES – éCriture (C).
a. Une éCriture déplacera un seul groupe de données décrivant un seul objet d'intérêt d'un processus fonctionnel duquel l'éCriture fait partie, à travers la frontière du logiciel, vers le 'stockage persistant'. Si le processus fonctionnel doit écrire plus d'un groupe de données dans le stockage persistant, il faudra identifier une seule éCriture pour chaque groupe de données à écrire. (voir également la section 3.5.7 sur 'l'unicité des mouvements de données'.) ; b. Une éCriture ne doit pas recevoir de données à travers la frontière du logiciel, ni sortir, ni lire des données dans un stockage persistant; c. Une manipulation de données de suppression d'un groupe de données du 'stockage persistant' ne sera mesurée que par un seul mouvement de données en éCriture ; d. Ce qui suit n'est pas est considéré comme des mouvements de données éCriture: • Le déplacement ou la manipulation des données, qui n'existaient pas au début d'un processus fonctionnel, et qui ne sont pas devenues persistantes à la fin du processus fonctionnel; • Création ou mise à jour des variables ou des résultats intermédiaires qui sont internes au processus fonctionnel; • Stockage des données par un processus fonctionnel résultant uniquement de son implémentation, plutôt que de la FUR. (Un exemple serait le stockage temporaire de données lors d'un tri d'un grand nombre d'informations dans un traitement en lot).

Les règles suivantes aident à confirmer le statut de d'éCriture pour le mouvement de données:

Règles – éCriture (C)
a) Identifier une éCriture quand, selon la FUR, le logiciel à mesurer doit déplacer un groupe de données vers un stockage persistant. b) Ne pas identifier une éCriture lorsque la FUR du logiciel à mesurer spécifie n'importe quel logiciel ou matériel utilisateur fonctionnel comme destination, ou comme moyen de déplacement, du groupe de données. (Dans ce cas, voir les principes et règles pour les Entrées et les Sorties.)

Une fois identifié, chaque mouvement de données en éCriture peut être enregistré en marquant un 'C' dans la cellule correspondante de la matrice du modèle générique de logiciel (voir en annexe A).

3.5.6 Sur les manipulations de données associées à des mouvements de données

Les sous-processus sont soit des mouvements de données, soit des manipulations de données, comme définies dans le principe (d) du modèle générique de logiciel (voir la section 1.3). Cependant, actuellement, par une convention COSMIC (voir le principe (j) du générique de logiciel), l'existence de sous-processus distincts de manipulation de données n'est pas reconnue.

DÉFINITION – Manipulation des données
Tout ce qui survient aux données, autres qu'un mouvement de données, vers l'intérieur ou l'extérieur d'un processus fonctionnel, ou entre un processus fonctionnel et un stockage persistant.

Le principe et les règles suivants déterminent comment la méthode COSMIC traite la manipulation des données.

PRINCIPE – Manipulation des données associées aux mouvements de données
Toute manipulation de données d'un processus fonctionnel sera associée aux quatre types de mouvements de données (E, S, L, et C). Par convention, on suppose que les mouvements de données d'un processus fonctionnel représentent des manipulations de données pour le même processus fonctionnel.

Règles – Manipulation des données associée aux mouvements de données
a) Un mouvement de données d'Entrée comprend toutes les manipulations de données pour permettre à un groupe de données d'être entré par un utilisateur fonctionnel (par exemple les manipulations de mise en forme et de présentation) et permettre leur validation, b) Un mouvement de données de Sortie comprend toutes les manipulations de données créant les attributs d'un groupe de données en sortie et/ou permettant au groupe de données en sortie (par exemple les manipulations de mise en forme et les présentations) d'être acheminé à l'utilisateur fonctionnel prévu, c) Un mouvement de données de Lecture comprend tous les calculs et/ou traitements logiques nécessaires visant à récupérer un groupe de données à partir du stockage persistant. d) Un mouvement de données éCriture comprend tous les calculs et/ou traitements logiques pour créer ou mettre à jour un groupe de données à écrire, ou de supprimer un groupe de données. e) La manipulation de données associée à n'importe lesquels de ces mouvements de données n'inclut aucune des manipulations de données nécessaires après que le déplacement des données a été mené à bien, pas plus qu'il ne comprend les manipulations de données associées à tout autre mouvement de données.

Exemple d'affaires 1: Une Entrée comprend toutes les manipulations nécessaires au formatage d'un écran et ainsi permettre à un utilisateur humain l'entrée et la valider des données d'entrées SAUF toute(s) Lecture(s) pouvant être nécessaire(s) pour valider certains codes ou données de saisis, ou pour obtenir certains codes descriptifs associés.

Exemple d'affaires 2: Une Sortie comprend toutes les manipulations pour le format de sortie et de préparation de certains attributs de données pour l'impression (ou la sortie sur un écran), y compris les en-têtes de champ lisible SAUF la/les Lecture(s) qui pourrai(en)t être nécessaire(s) pour fournir les valeurs ou les descriptions de certains des attributs imprimés.

La nécessité de savoir quel genre de manipulations de données est associé à quel type de mouvement de données ne survient que lorsqu'on mesure les modifications apportées à un logiciel (voir la section 4.4). Une demande de modification typique affecte le plus souvent le déplacement des attributs que la manipulation liée à un mouvement de données, mais il peut aussi seulement affecter la manipulation de données, (sans toucher au mouvement de données). Une telle modification doit alors être identifiée et mesurée. Ainsi, quand il y a une exigence visant à modifier la manipulation des données dans un processus fonctionnel, le mesureur doit identifier quel mouvement de données est associé à quel changement de manipulation des données.

3.5.7 Unicité des mouvements de données et exceptions possibles[11]

Le modèle générique de logiciel suppose que généralement, pour tout processus fonctionnel, toutes les données décrivant un objet d'intérêt doit prendre part dans au moins un mouvement de données, soit : être saisi en Entrée et/ou lu par mouvement de données de Lecture et/ou écrit par un mouvement de données en éCriture et/ou sorti au moyen d'un mouvement de données en Sortie. De plus, le modèle suppose aussi que toute manipulation de données (modification, ajout, suppression) exercée sur un groupe de données spécifique ainsi déplacé, est associée à son propre mouvement de données.

EXEMPLE : illustrant ces hypothèses: considérons deux occurrences d'un processus fonctionnel donné (- type). Supposons que dans la première occurrence, les valeurs de quelques attributs de données à déplacer mènent à un sous-processus de manipulation de données (- type) 'A'. Supposons aussi que dans l'autre occurrence du même processus fonctionnel, la valeur des attributs mène à un sous-processus différent de manipulation de données (- type) 'B'. Dans de telles circonstances, les sous-processus 'A' et 'B' de manipulation de données doivent être associés au même mouvement de données et par conséquent, seulement un mouvement de données devra normalement être trouvé et compté dans ce processus fonctionnel.

Il existe cependant des circonstances exceptionnelles dans lesquelles différents types de groupe de données (décrivant un objet d'intérêt), peuvent être demandés (par la FUR) à être déplacés par un mouvement de données de même type (C, L, E, S) dans le même processus fonctionnel. Sinon, et encore exceptionnellement, il peut être demandé que le même groupe de données soit déplacé dans le même type de mouvement de données (C, L, E, S) dans le même processus fonctionnel, mais avec des manipulations de données différentes.

Les règles suivantes et les exemples couvrent les cas normaux (règle a)), quelques exceptions et exemples qui apparaissent comme étant des cas valides, mais qui ne le sont pas (règles d) et e)).

Règles – Unicité des mouvements de données et exceptions possibles.
a) Sauf si la FUR l'indique autrement, tous les groupes de données, décrivant un objet d'intérêt exigé dans un processus fonctionnel, de même que toutes les manipulations de données associées, seront identifiées et comptées comme étant des mouvements de données en Entrée (-type) ;
b) (REMARQUE: Un processus fonctionnel pourrait, en principe, manipuler de multiples Entrées (-type), chacune déplaçant un groupe de données décrivant un objet d'intérêt (-type) distinct)
La même règle correspondante s'applique à n'importe quel mouvement de données de Lecture, d'éCriture ou de Sortie dans tous les processus fonctionnels.
b) Plus d'un mouvement de données d'Entrée (type), chacun déplaçant, un groupe de données décrivant le même objet d'intérêt (type), dans un processus fonctionnel (type) donné, peuvent être identifiés et comptés s'il existe une FUR pour ces

[11] Cette section était intitulée « la déduplication des mouvements de données' dans la version 2.2 du Manuel mesure. Le terme « la déduplication » étant considérée comme inutile, alors la terminologie a été changée.

entrées multiples et qu'au moins une des conditions suivantes s'applique:

- les Entrées diffèrent en déplaçant leurs groupes de données vers un autre utilisateur fonctionnel
- les Entrées déplacent différents groupes de données,
- les Entrées manipulent différentes données associées.

La même règle correspondante s'applique pour les Sorties dans n'importe quel processus fonctionnel donné.

Remarque : Tout processus fonctionnel ne peut avoir qu'une seule entrée de déclenchement.

c) Plus d'un mouvement de données (type), chacun déplaçant un groupe de données décrivant un objet d'intérêt (type) dans un processus fonctionnel (type) donné, peuvent être identifiés et comptés s'il existe une FUR pour ces Lectures multiples et qu'au moins une des conditions suivantes s'applique:

- les Lectures déplacent différents groupes de données,
- les Lectures manipulent différentes données associées.

La même règle correspondante s'applique pour les éCritures dans n'importe quel processus fonctionnel donné.

d) Plusieurs *occurrences* d'un même type de mouvement de données (i.e. déplaçant le même groupe de données avec la même manipulation de données) ne seront pas trouvées ou comptées plus d'une fois dans tout le processus fonctionnel.

e) Dans le cas où des *occurrences* multiples d'un mouvement de données (associés à un processus fonctionnel donné), diffèreraient dans leur manipulation de données en raison des différentes valeurs des attributs du groupe de données, cela résulterait dans le suivi de différents chemins de traitement. Ce type de mouvement de données ne sera pas identifié et compté plus d'une fois dans ce processus.

Les exemples suivants illustrent les règles ci-dessus.

Exemple d'affaires 1 pour la règle a): Toute Lecture de données déplaçant un objet d'intérêt particulier peut être logiquement considéré comme déplaçant tous les attributs de données requis de cet objet d'intérêt (i.e. la totalité du 'vecteur d'état' de cet objet d'intérêt). Par conséquent et habituellement, seulement une Lecture (de n'importe quel attribut, et de n'importe quel objet d'intérêt), est fonctionnellement nécessaire et devrait être identifiée et comptée dans tout processus fonctionnel.

Exemple d'affaires 1 pour les règles a) et c): Suite à l'exemple 1, étant donné toutes les données devant être lues, celles-ci doivent aussi être reconnues comme étant des données persistantes provenant d'une éCriture. Il est donc normal dans ce cas (règle a), qu'une éCriture identifiée comme étant celle qui déplace tous les attributs du groupe de données d'un objet d'intérêt soit nécessairement persistant dans un processus fonctionnel donné (selon la règle (a)). Exceptionnellement cependant, il est possible qu'une FUR d'un processus fonctionnel unique écrive deux groupes de données différents, décrivant le même objet d'intérêt. Un exemple d'application de la règle b) serait quand un unique processus fonctionnel A est requis pour stocker deux groupes de données d'un fichier d'un compte de banque courant pour une utilisation ultérieure dans des programmes séparés. Le premier groupe de données est le détail du 'compte à découvert' (qui comprend les attributs du solde négatif). Le deuxième groupe de données est le détail du 'compte de grande valeur' (qui ne comprend que le nom et l'adresse du titulaire du compte destinés au marketing par courriel). Un processus fonctionnel aura deux éCritures, une pour chaque groupe de données. (Pourrais être en 2 exemples pour simplifier)

Exemple temps réel 3 pour la règle b): Un processus fonctionnel est tenu d'accepter deux Entrées provenant des différents capteurs (utilisateurs fonctionnels) chacun décrivant des données différentes sur le même objet d'intérêt (l'évènement), par exemple une explosion expérimentale. Déterminer deux Entrées.

Exemple d'affaires 4 pour la règle b): Il est possible qu'une FUR exige qu'un processus fonctionnel produise deux Sorties déplaçant des groupes de données différents destinés à différents utilisateurs fonctionnels, mais qui décrivent le même objet d'intérêt. Par exemple, un nouvel employé se joint à une nouvelle compagnie : un rapport est alors produit pour que l'employé révise ses données personnelles afin de les valider. Un message est alors envoyé à la sécurité pour autoriser cet employé à entrer dans les bâtiments de l'organisation. Déterminer deux Sorties.

Exemple d'affaires 5 pour la règle d): Supposons une FUR pour imprimer des enregistrements d'adhésion à un club qui ont des données complètes et incomplètes pour les différents membres. Déterminer une Sortie.

Exemple d'affaires 6 pour la règle d): Supposons qu'une Lecture est requise dans la FUR qui, dans la pratique, nécessite de nombreuses occurrences de récupération, comme dans une recherche dans un fichier. Pour mesurer la taille, ne déterminer qu'une seule Lecture.

Exemple temps réel 7 pour la règle c): Supposons que dans un processus fonctionnel en temps réel, la FUR nécessite que les données soient saisies par utilisateur fonctionnel donné. Par exemple, pour un périphérique matériel les données sont saisies deux fois à un intervalle de temps fixe afin de mesurer un taux de variation au cours du processus. Les données de la première Entrée doivent être stockées temporairement (ils n'ont pas besoin d'être persistants, car ils ne survivront pas au processus fonctionnel). Les données de la deuxième Entrée doivent être comparées avec les premières données d'Entrées pour calculer le taux de variation. Il n'y a aucune manipulation de données associée aux deux Entrées. (Le calcul du taux de change doit être associé à la Sortie indiquant le taux). Par conséquent, les deux Entrées sont identiques. Ils sont donc des occurrences multiples de la même Entrée. Une seule Entrée peut être identifiée pour ces données et pour ce processus fonctionnel. (Voir section 3.5.6 pour les types de manipulation des données qui sont censées être associées à une Entrée).

Exemple d'affaires 8 pour la règle d): Voir la section 3.5.2, règle d) pour les Entrées et l'article 3.5.3, règle b) pour les Sorties.

Exemple d'affaires 9 pour la règle e): Supposons qu'un processus fonctionnel est nécessaire pour offrir diverses options de manipulation de données selon les valeurs des attributs d'une Entrée. Déterminer une seule Entrée.

Exemple d'affaires 10 pour la règle e): Supposons que la FUR d'un processus fonctionnel de demande de renseignements généraux permet à l'utilisateur d'entrer plus de 1, 2 ou 3 critères de sélection pour la requête. Déterminer une seule entrée.

Exemple d'affaires 11 pour la règle e): Supposons que la Lecture d'un groupe de données est requise dans la FUR, mais le développeur décide de la mettre en œuvre avec deux commandes pour récupérer différents sous-ensembles d'attributs du même objet d'intérêt à partir d'un stockage persistant en différents points du processus fonctionnel. Déterminer une seule Lecture.

3.5.8 Quand un processus fonctionnel est requis pour déplacer les données vers ou à partir d'un stockage persistant

Cette section décrit les mouvements de données impliqués quand un processus fonctionnel d'un morceau de logiciel est nécessaire pour obtenir certaines données à partir du stockage persistant, en utilisant quatre exemples:

- L'exemple 1 est typique des logiciels d'application, mais pourrait s'appliquer au logiciel dans n'importe quelle couche, à l'exception de la couche où logiciel interagit directement avec le matériel de stockage physique. Le processus fonctionnel est requis pour extraire des données d'un stockage persistant, mais la FUR ne nécessite pas la participation de tous les autres logiciels;
- L'exemple 2 vise le logiciel avec une architecture 'client-serveur' où un processus fonctionnel d'un client doit accéder au serveur pour certaines données persistantes;
- Exemple 3 ou les différents morceaux de logiciels ont différents droits d'accès aux données persistantes;

- Exemple 4 lorsque les données doivent être collectées directement du matériel de stockage physique, peut-être même par le pilote du logiciel.

Les exemples sont illustrés à l'aide des conventions de diagrammes de séquence de Message. La notation de ces schémas est comme suit:

- Une flèche verticale en "GRAS" pointant vers le bas représente un processus fonctionnel.
- Les flèches horizontales représentent les mouvements de données, étiquetés E, S, L ou C pour Entrée, Sortie, Lecture et éCriture, respectivement. Les éCritures et Lectures sont indiquées par des flèches entrantes vers le processus fonctionnel et les Sorties et les éCriture sont indiquées par des flèches sortantes ; ils apparaissent dans l'ordre requis, de haut en bas, du processus fonctionnel.
- Un ligne verticale avec des pointillés représente la frontière du logiciel.

Exemple 1: Quand un processus fonctionnel est requis pour déplacer un groupe de données vers ou à partir d'un stockage persistant. Cet exemple concerne un morceau de logiciel A ce qui est nécessaire pour récupérer un groupe de données stockées où la FUR du logiciel 'A' n'est pas intéressé sur la façon que ces données sont gérées par un autre logiciel dans des couches identiques ou différentes.

Les utilisateurs fonctionnels du logiciel A pourraient être, par exemple, des utilisateurs humains si le logiciel A est dans la couche applicative 'faire une requête sur un groupe de données stocké'. La figure 3.6 montre le modèle COSMIC de cette requête. La requête est déclenchée par une Entrée, suivie d'une Lecture du groupe de données de stockage persistant, puis d'une Sortie avec le résultat de la requête. Le processus fonctionnel 'FP A' n'est pas concerné par l'origine des données extraites, seulement si les données sont persistantes.

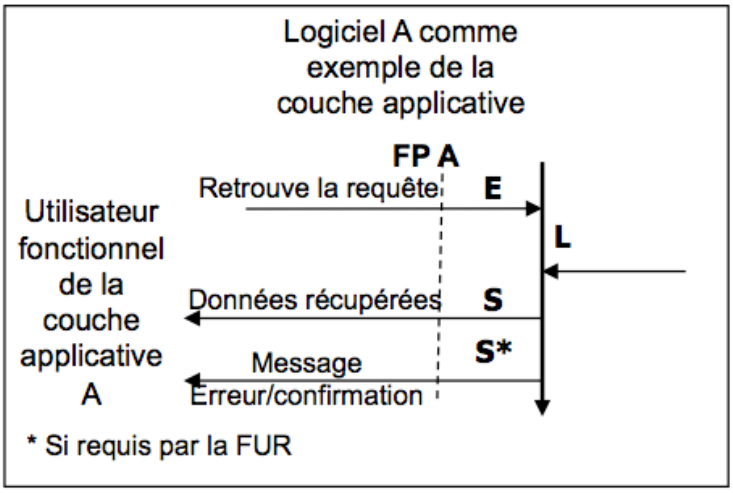


Figure 3.6 - Solution pour une Lecture émise par le logiciel 'A' dans la couche applicative

Un modèle analogue s'appliquerait si le processus fonctionnel 'FP A' était tenu de créer un groupe de données persistant avec un mouvement de données d'éCriture. Par convention, les mouvements de données de Lecture et d'éCriture sont pris en considération pour tenir compte de n'importe quel code de retour ou de déclaration d'une condition d'erreur.

La figure 3.6 affiche un message d'erreur potentielle, spécifique à l'application, qui pourrait provenir de 'FP A' si, par exemple, l'enregistrement demandé est introuvable. Toutefois, une condition d'erreur non spécifique à l'application, par exemple 'la défaillance d'un disque', ne serait pas comptabilisée comme une sortie pour 'FP A'. Voir aussi la section 3.5.11 sur les messages d'erreur/confirmation.

Exemple 2: Lorsqu'un processus fonctionnel est nécessaire pour obtenir des données d'un autre morceau de logiciel.

Dans cet exemple, les morceaux de logiciels à mesurer sont censés avoir une relation 'client/serveur', c'est-à-dire lorsqu'un morceau provenant du client obtient les services et/ou données de l'autre

morceau provenant du 'serveur' dans la même couche. La figure 3.7 montre un exemple d'une telle relation, dans laquelle les deux morceaux sont des éléments majeurs de la même application. Dans toute relation client/serveur, la FUR du composant client C1 permettrait d'identifier le composant serveur C2 comme l'un des ses utilisateurs fonctionnels et vice versa. La même relation existerait et le même schéma s'appliquerait si les deux morceaux étaient des demandes distinctes, ou si un des morceaux était un composant d'une demande distincte.

Physiquement, les deux composantes peuvent s'exécuter sur des processeurs séparés ; dans ce cas, ils échangeraient des données via les systèmes d'exploitation respectifs et avec toutes les autres couches intermédiaires de leurs processeurs dans une architecture logicielle, tel qu'illustré à la Figure 2.2. Cependant logiquement, les deux composants appliquant les modèles COSMIC, échangent des données via un mouvement de données de Sortie suivi d'un mouvement de données d'Entrée. Tous les logiciels intermédiaires et le matériel sont ignorés dans ce modèle. (Voir aussi la partie droite de la Figure 3.1 pour un exemple similaire.).

La figure 3.7 montre qu'un processus fonctionnel 'FP C1' du composant client C1 est déclenché par une Entrée d'un utilisateur fonctionnel (tel un humain) qui est constitué des paramètres d'une requête. La FUR du composant C1 reconnaitra que ce composant doit demander au composant du serveur C2 les données requises et lui indiquer quels sont les groupes de données nécessaires.

Pour obtenir les groupes de données requis, 'FP C1' transmet une Sortie au composant C2 contenant les paramètres de la requête. Ce mouvement de données de Sortie traverse la frontière entre C1 et C2 et devient l'Entrée pour le déclenchement d'un processus fonctionnel 'FP C2' dans le composant C2. Le processus fonctionnel 'FP C2' du composant C2 est censé obtenir le groupe de données requises d'une Lecture de son propre stockage persistant, et renvoie les données au C1 via une Sortie. Le processus fonctionnel 'FP C1' du composant C1 reçoit ces données sous forme d'Entrée. 'FP C1' transmet ensuite le groupe de données comme Sortie pour satisfaire la demande de son utilisateur fonctionnel.

En tenant compte du message d'erreur/confirmation potentiel émis par le client, la requête de l'exemple 2 nécessite donc 5 mouvements de données (c.-à-d. 5 PFC) pour satisfaire la demande de requête pour le composant C1 et 3 PFC pour le composant C2. Cela se compare à 4 PFC (1 x E, 1 x L et 2 x S) qui auraient été requis pour le composant C1 si ce dernier avait été en mesure de récupérer le groupe de données de stockage persistant à l'intérieur de sa propre frontière par une Lecture, tel qu'illustré à la Figure 3.6.

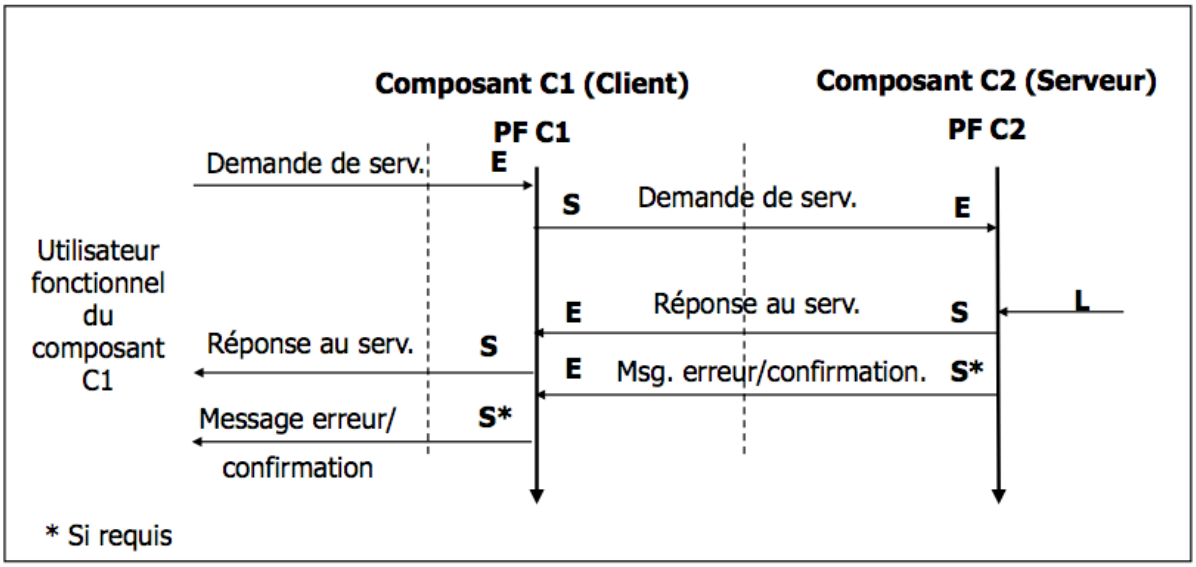


Figure 3.7 - Échanges de données entre les composants du client et du serveur

Le composant C2 utilisera probablement les services d'un pilote de stockage provenant d'une autre couche de l'architecture logicielle pour récupérer les données à partir du matériel, comme dans l'exemple 4, Figure 3.9 (b).

Les exemples 1 et 2 illustrent les mouvements de données lorsqu'il ressort clairement de la FUR que le logiciel à mesurer doit accéder au stockage persistant à l'intérieur de sa frontière, ou qu'il doit transmettre la demande d'accès à un autre morceau de logiciel en dehors de ses frontières, respectivement. Parfois, cependant, le morceau de logiciel à mesurer peut devoir employer différents 'chemins' pour accéder aux données persistantes dépendant soit des attributs spécifiques devant être accédés ou encore du type d'accès (stockage ou extraction). Cela peut survenir lorsque l'accès aux données par le logiciel à mesurer est soumis à diverses règles ou différents 'droits' en raison de problèmes de sécurité ou de protection de la vie privée par exemple, ou encore la nécessité d'assurer l'intégrité des données en limitant l'accès pour tout de qui concerne la création, la mise à jour et la suppression des processus. Si la FUR du logiciel à mesurer n'est pas claire sur ce point, le mesureur doit veiller à déterminer les droits réels d'accès. (Ne pas confondre le 'droit d'accès' aux données avec les données 'propriétaires', ces dernières sont sans rapport avec le modèle COSMIC. Le stockage persistant est un concept abstrait : il n'appartient pas aux morceaux du logiciel)

Exemple 3: Le logiciel possède différents droits d'accès aux données enregistrées avec des raisons d'être différentes

Voir la Figure 3.8. Un morceau de logiciel A à mesurer est autorisé à récupérer certaines données stockées Z (comme dans l'exemple 1, Figure 3.6), mais il n'est pas autorisé à conserver (par exemple, créer, mettre à jour ou supprimer) ces mêmes données Z directement. Lorsque le logiciel A est nécessaire pour maintenir les données Z, logiciel A doit transmettre sa demande à un autre morceau de logiciel B via une Sortie suivie d'une Entrée (comme dans l'exemple 2, la Figure 3.7 du composant C1 transmet sa demande au composant C2). Le morceau de logiciel B est nécessaire pour assurer l'intégrité des données Z en assurant une validation compatible, ainsi il traite toutes les demandes de maintenabilité des données pour Z.

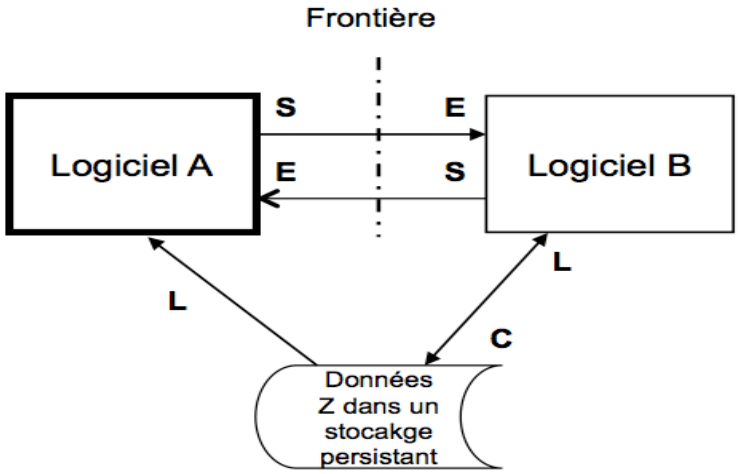


Figure 3.8 - Données persistantes Z à l'intérieur de la frontière des logiciels A et B pour une Lecture

Dans cet exemple 3, les modèles COSMIC montreraient que les données Z se trouvent sur un stockage persistant à l'intérieur de la frontière du morceau de logiciel A, mais uniquement à des fins de récupération où l'on peut y accéder par des mouvements de données de Lecture. Pour le logiciel B, ces mêmes données Z se tiennent dans un stockage persistant, à l'intérieur de la frontière du logiciel et le logiciel B peut à la fois Lire et éCrire ces données Z. Pour la gestion des erreurs dans cet exemple, consultez les exemples 1 et 2 ci-dessus.

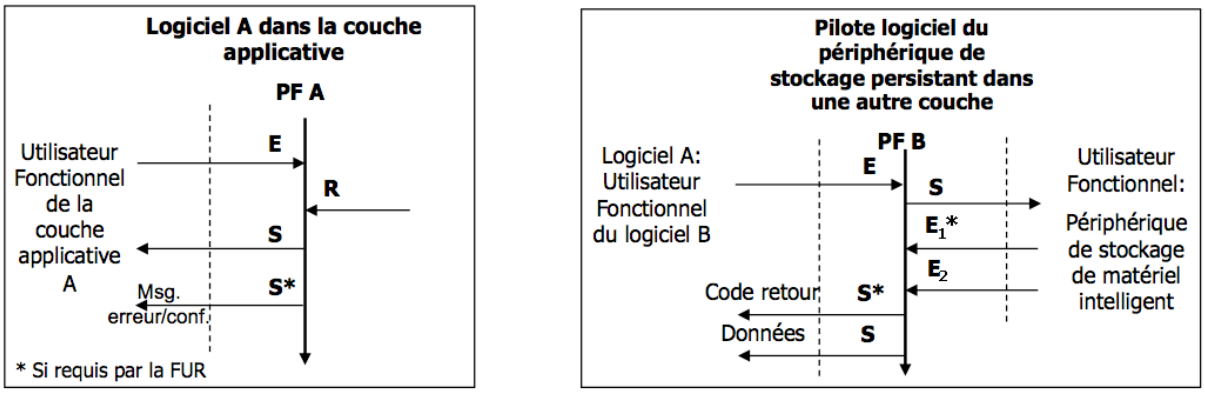
Exemple d'infrastructure 4: Comment les données persistantes sont-elles obtenues par le pilote qui interagit avec le périphérique de stockage physique?

Cet exemple concerne le morceau de logiciel A de l'exemple 1 qui est requis pour récupérer un groupe de données stockées. Nous considérons également un morceau de logiciel B, qui est le pilote du périphérique, pour le stockage intelligent du matériel et qui détient le groupe de données du morceau de logiciel A requis pour y accéder (au matériel). (Nous ignorons la présence probable d'un système d'exploitation pour plus de simplicité; le système d'exploitation transmet les requêtes d'application pour le pilote de périphérique et retourne les résultats des requêtes.)

Les deux morceaux de logiciels sont dans différentes couches dans une architecture telle qu'illustrée à la Figure 2.2. Le logiciel A est, par exemple, dans la couche d'application et le logiciel B est dans une couche du pilote. Physiquement, il y a probablement une relation hiérarchique entre les deux morceaux et (en ignorant le système d'exploitation) une interface physique entre les logiciels dans les deux couches, comme illustrés à la Figure 2.2. Cependant, les modèles des processus fonctionnels des logiciels A et B sont indépendants de la nature de la relation entre les couches, qui peut être hiérarchique ou bidirectionnelle.

Les utilisateurs fonctionnels du logiciel B dans la couche du pilote sont les morceaux du logiciel A (en ignorant le système d'exploitation) et le périphérique de stockage intelligent du matériel qui contient les données requises. ('Intelligent' signifie que le périphérique doit savoir quelles données sont nécessaires.)

Supposons que le processus fonctionnel qui est une requête 'FP A' du logiciel A doit récupérer un groupe de données stockées. La figure 3.9 (a) montre le modèle COSMIC de cette requête. La figure 3.9 (b) montre le processus fonctionnel 'FP B' du logiciel B dans la couche du pilote de périphérique qui gère la récupération physique des données requises d'un périphérique de stockage de matériel (par exemple, un disque ou une clé USB).



Figures 3.9 (a) et (b) - Solution pour une Lecture émise par le logiciel A de la couche applicative vers le logiciel B de la couche du pilote de périphérique

La figure 3.9 (b) montre que la requête de Lecture du logiciel A est reçue comme une Lecture de déclenchement du processus fonctionnel 'FP B', qui transmet la demande comme une Sortie pour le matériel périphérique. La réponse de ce dernier dépend du matériel périphérique particulier. L'appareil peut simplement retourner les données demandées comme Entrée (E_1), comme illustré à la Figure 3.9. L'appareil peut également émettre un message d'erreur distinct décrivant le succès ou la raison de l'échec de la requête, par exemple 'données non trouvées', ou 'erreur de disque', voir Entrée (E_1) à la Figure 3.9 b). 'FP B' retourne les données pour le logiciel A comme Sortie (S). 'FP B' transmet aussi normalement un 'code de retour' décrivant le succès ou la raison de l'échec de la demande. (Bien que le code de retour peut être physiquement attaché aux données retournées, c'est logiquement un groupe de données différent de celui du groupe de données retourné - c'est des données sur les résultats du processus de demande). Pour le 'FP A' aucune Entrée n'est trouvée pour ces messages, comme le déplacement des données de Lecture représente les données et messages d'erreurs, selon la règle d) pour une Entrée. Pour le FP A une sortie est identifiée pour un message d'erreur/confirmation, si nécessaire.

Remarque : en pratique, il peut y avoir plusieurs mouvements de données entre le pilote du périphérique et le périphérique intelligent du matériel tel qu'illustré à la figure 3.8 b). Par exemple, cette

Figure ne montre pas l'effet du pilote de périphérique mesurant le temps d'arrêt pour la non-réponse du matériel.

En comparant les exemples 2 et 4, nous constatons que dans l'exemple 4, les modèles du morceau de logiciel A et le pilote de périphérique B de l'exemple 4 ne peuvent pas ê*tre rassemblés comme* dans l'exemple 2. C'est parce que A et B sont dans différentes couches et une Lecture ne peut traverser la frontière. La figure 3.9 (b) montre que le logiciel A un utilisateur fonctionnel du pilote du périphérique B. De plus, l'inverse n'est pas vrai, car une Lecture ne peut franchir la frontière. En revanche, la Figure 3.7 peut montrer les deux composants dans un seul modèle, parce que le composant C1 est un utilisateur fonctionnel du composant C2 et inversement; ils partagent une frontière commune.

3.5.9 Lorsqu'un processus fonctionnel demande des données à un utilisateur fonctionnel

Si un processus fonctionnel doit obtenir des données d'un utilisateur fonctionnel, il y a deux cas possibles. Si le processus fonctionnel n'a pas besoin de dire à l'utilisateur fonctionnel quelles sont les données à transmettre, une seule Entrée suffit (par objet d'intérêt). Si le processus fonctionnel doit dire à l'utilisateur fonctionnel quelles sont les données à transmettre, une Sortie suivie d'une Entrée sont nécessaires. Les règles suivantes s'appliquent :

RÈGLES – Lorsqu'un processus fonctionnel demande des données à un utilisateur fonctionnel.

a) Un processus fonctionnel doit obtenir un groupe de données via un mouvement de données en Entrée provenant d'un utilisateur fonctionnel, et ce, dans le seul cas où le processus fonctionnel n'a pas besoin de préciser à cet utilisateur, de quelle donnée il s'agit. Voici certains de ces cas, soit :

- Quand un utilisateur fonctionnel envoie un déclenchement en Entrée qui démarre le processus fonctionnel ;
- Quand un processus fonctionnel, ayant déjà commencé, attend une prochaine Entrée de l'utilisateur fonctionnel (typique dans les logiciels d'application d'affaires, où utilisateur fonctionnel est un humain) ;
- Quand un processus fonctionnel, ayant déjà démarré, demande à l'utilisateur fonctionnel d'envoyer des données, l'utilisateur fonctionnel envoie alors ses données (typiquement connues sous les termes 'd'invitation à émettre' ou de 'scrutation' dans les logiciels de type temps réel) ;
- Quand un processus fonctionnel, ayant déjà commencé, inspecte l'état de l'utilisateur fonctionnel et récupère les données dont il a besoin.

REMARQUE : Dans les deux derniers cas (en général, se produisant dans le cas de logiciel de type temps réel), par convention, aucune Sortie du processus fonctionnel ne sera comptée pour obtenir la donnée requise. Le processus fonctionnel enverra plutôt un message à l'utilisateur fonctionnel pour l'obtention de cette donnée et ce message sera considéré comme faisant partie de l'Entrée. Par contre, en aucun cas ce message sera comptabilisé comme étant une Entrée en soit : une seule Entrée sera comptabilisée dans ce cas, soit la donnée qui était attendue.

b) Dans le cas où un processus fonctionnel requiert les services d'un utilisateur fonctionnel (par exemple, pour obtenir des données) et que cet utilisateur fonctionnel a besoin qu'on lui précise la donnée à envoyer (par exemple, quand cet utilisateur fonctionnel provient d'un autre morceau de logiciel ne faisant pas partie du périmètre du mesurage), une paire de mouvements de données en 'Sortie et Entrée' devra être comptée. La Sortie contient la donnée de « demande de données » spécifiques et l'Entrée contient les données renvoyées.

Exemple temps réel 1 de la règle a), troisième et quatrième points : Supposons un processus fonctionnel d'un logiciel de contrôle de type temps réel devant recenser les informations provenant d'une table de capteurs passifs. Ce processus fonctionnel obtient donc ses données via un seul

mouvement de données en Entrée (type) puisque les capteurs sont identiques, et qu'une seule Entrée (type) est trouvée et comptée (bien qu'il y ait de multiples occurrences).

Supposons ensuite que ces données en Entrée doivent être passées à une couche inférieure de l'architecture logicielle, sur un pilote de périphérique, qui lui obtient physiquement ses données des capteurs comme illustrés dans la figure 2.3. Les processus fonctionnels du logiciel de contrôle des pilotes de périphérique et des logiciels des capteurs passifs sont présentés dans les figures. 3.10 (a) et (b) ci-dessous.

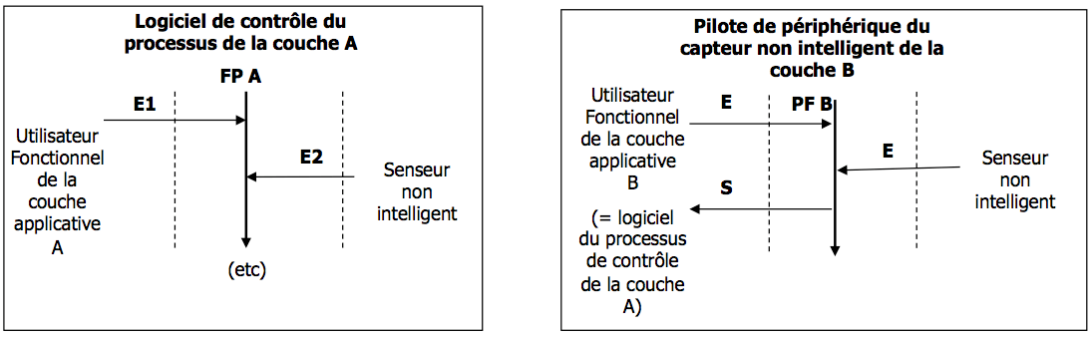


Figure 3.10 (a) and (b) - Solution au problème d'une Entrée transmise par le logiciel 'A' de la couche applicative où logiciel de contrôle de traitements se situe, vers le logiciel 'B' de la couche du pilote du dispositif où le logiciel du capteur non intelligent se situe.

La Figure 3.10 (a) montre que le processus fonctionnel 'PF 'A'' du logiciel de contrôle de traitements est déclenché par une Entrée 'E1', par un signal d'horloge. Le processus fonctionnel reçoit alors des données d'Entrée 'E2' par le réseau de sondes non intelligentes par les occurrences multiples de lectures de sonde. Ces sondes non intelligentes sont également les utilisateurs fonctionnels des traitements du logiciel de contrôle de ce modèle applicatif. (La présence du pilote logiciel de périphérique est inconnue à ce niveau.)

La figure 3.10 (b) montre que le logiciel qui commande le réseau de sondes non intelligentes reçoit un déclenchement en Entrée (probablement par l'intermédiaire d'un système d'exploitation) par le processus fonctionnel 'PF B'. Ce processus fonctionnel obtient les données requises par l'intermédiaire d'une Entrée (E) de son utilisateur fonctionnel, les capteurs non intelligents.

Le groupe de données est transmis vers le logiciel de contrôle de processus par une Sortie. Cette Sortie est reçue comme l'Entrée E2 par le processus fonctionnel de l'application 'FP A'. 'FP A' continue ensuite avec le traitement des données des capteurs. À nouveau, le fait qu'il y ait plusieurs occurrences du cycle de collecte de données de chacun des capteurs (identiques) est peu pertinent pour le modèle COSMIC.

La disparité apparente entre le mouvement de données unique d'Entrée E2 du capteur non intelligent du logiciel de contrôle de processus et le mouvement de données Entrée suivi d'un mouvement de données de Sortie des données du logiciel de pilote du périphérique, est due à la convention qu'un mouvement de données Entrée d'un capteur non intelligent est considéré inclure n'importe quelle fonctionnalité de 'requête pour entrer' puisque l'utilisateur fonctionnel non intelligent n'a aucune capacité de traiter avec les messages d'un processus fonctionnel.

Exemple temps réel 2 de la règle b): Supposons un processus fonctionnel envoyant, à un de ses utilisateurs fonctionnels, un dispositif 'intelligent' ou un morceau de logiciel distinct de même niveau, des paramètres pour une requête ou des paramètres pour un calcul, ou encore quelques données à compresser. La réponse des utilisateurs fonctionnels serait obtenue par l'intermédiaire d'une paire d'Entrée/Sortie, comme décrite dans la section 3.5.8, exemple 2.

3.5.10 Navigation et affichage des commandes de contrôle pour les utilisateurs humains

Une 'commande de contrôle' est une commande qui est reconnue dans toutes les applications utilisées par les utilisateurs fonctionnels humains et qui doit être ignorée lors du mesurage de la taille fonctionnelle. La définition est:

DÉFINITION – Commande de contrôle
Une commande qui permet à un utilisateur fonctionnel de contrôler l'utilisation du logiciel, mais qui n'implique pas de mouvement de données des objets d'intérêt de la FUR du logiciel à mesurer. Remarque: Une commande de contrôle n'est pas un mouvement de données parce que la commande ne déplace pas les données relatives à un objet d'intérêt. On peut citer les commandes « page up/Down » ; frapper une touche de tabulation ou 'Enter' (retour ce charriot), cliquer OK pour confirmer une action précédente, appuyer sur une touche pour continuer, etc.

RÈGLE – Commande de contrôle dans les applications avec une interface humaine
Dans les applications avec une interface humaine, la 'commande de contrôle' doit être ignorée puisqu'elle n'implique pas de mouvement de données des objets d'intérêt.

Exemple: Des exemples de 'commandes de contrôle' dans le domaine des logiciels d'application d'affaires pourraient être les fonctions qui permettent à un utilisateur fonctionnel de commander l'affichage d'un entête ou de totaux partiels ayant été calculés, de passer d'un écran d'affichage à un autre, de cliquer sur 'OK' avec un souris pour reconnaitre un message d'erreur ou pour confirmer des données saisies, etc. Les commandes de contrôle incluent donc également les commandes de menu qui permettent à l'utilisateur fonctionnel de passer d'un ou plusieurs processus fonctionnels spécifiques à d'autres processus, ou des commandes pour afficher un écran vierge pour la saisie de données, mais qui par elles-mêmes, ces commandes n'initient aucun processus fonctionnel.

REMARQUE : En dehors du domaine d'application d'affaires, le concept de 'commande de contrôle' n'a pas de signification spéciale et tout signal ou mouvement des données au sujet d'un objet d'intérêt venant d'un utilisateur fonctionnel doit être considéré, c'est à dire doit être mesuré.

3.5.11 Messages d'erreur/confirmation

.

DÉFINITION – Messages d'erreur/confirmation
C'est une Sortie émise par un processus fonctionnel à l'attention d'un utilisateur humain qui soit, confirme que les données saisies ont été acceptées, soit confirme qu'il y a une erreur dans les données saisies. Remarque : Si un message à un utilisateur humain fournit des données autres que la confirmation que des entrées données ont été acceptées, ou que les données saisies sont erronées, alors ces autres données doivent être déterminées comme un groupe de données proposé par une Sortie et non comme un message d'erreur/confirmation.

RÈGLES – Messages d'erreur/confirmation et autre indication des conditions d'erreur
a) Une Sortie doit être identifiée pour tenir compte de tous les types de messages d'erreur/confirmation émis par un processus fonctionnel du logiciel à mesurer, à partir de toutes les causes possibles selon sa FUR, par exemple les succès ou les échecs de validation des données saisies pour un appel pour récupérer des données ou pour créer des données persistantes ou pour répondre à un service demandé d'un autre morceau de logiciel.

Remarque : Si la FUR du processus fonctionnel ne nécessite pas la présence d'un message d'erreur/confirmation, ne pas déterminer une Sortie correspondante.

Toutes les autres données, émises ou reçues par le logiciel à mesurer, vers ou à partir de son matériel ou du logiciel des utilisateurs fonctionnels, doivent être analysées selon la FUR comme Sorties ou Entrées respectivement, selon les règles normales de COSMIC, indépendamment du fait que les valeurs des données indiquent ou pas une condition d'erreur.

b) On doit prendre en considération seulement les Lectures et éCritures des présentations associées aux conditions d'erreur. Donc aucune Entrée vers le processus fonctionnel à mesurer ne doit être déterminée pour une indication d'erreur reçue suite à une Lecture ou éCriture de données persistantes.

c) Aucune Entrée ou Sortie ne doit être déterminée pour les messages qui indiquent une condition d'erreur, qui pourrait apparaitre lors de l'utilisation du logiciel à mesurer, et qui n'a pas besoin d'être traitée par la FUR de ce logiciel, par exemple un message d'erreur émis par le système d'exploitation.

Exemple d'affaires 1 illustrant la règle a): Un dialogue personne-machine, les exemples de messages d'erreur survenant lors de la validation des données en entrée qui pourraient être: 'erreur de format', 'client introuvable', 'erreur : veuillez cocher la case indiquant que vous avez lu nos 'clauses', 'dépassement de crédit', etc.. Tous ces messages d'erreur doivent être considérés comme des occurrences d'une Sortie pour chaque processus fonctionnel où ces messages se produisent (qui pourraient être appelées 'messages d'erreur').

Exemple d'affaires 2 illustrant la règle a): le processus fonctionnel A peut potentiellement livrer 2 messages de confirmation distincts et 5 messages d'erreur aux utilisateurs fonctionnels. Identifier une Sortie pour tenir compte des (5 + 2 = 7) messages d'erreur/confirmation. Le processus fonctionnel B peut potentiellement livrer 8 messages d'erreur aux utilisateurs fonctionnels. Identifier une Sortie pour tenir compte de ces 8 messages d'erreur.

Exemple d'affaires 3 illustrant la règle b): Dans un dialogue personne-machine, si un message est émis dans un cas d'erreur, mais que ce cas contient des données de l'utilisateur fonctionnel, alors ce cas devrait être considéré comme une Sortie dans le processus fonctionnel où il se produit. Un exemple d'un tel message pourrait être une mise en garde tel 'le montant que vous désirez retirer dépasse votre limite de découvert de $100' (où la somme de 100 $ est une variable calculée). Dans cet exemple, la Sortie contient un groupe de données du compte bancaire client.

Exemple d'affaires 4 illustrant la règle d): Les messages d'erreur provenant des utilisateurs humains, mais pas générés ou traités par le logiciel d'application à mesurer, doivent être ignorées dans la mesure de l'application. Un exemple d'un tel message transmis au système d'exploitation pourrait être 'imprimante X ne répond pas.

Exemple temps réel 1 illustrant la règle b): Dans un système en temps réel, un processus fonctionnel qui vérifie périodiquement le bon fonctionnement de tous les périphériques matériels, pourrait livrer un message indiquant que le 'capteur X a échoué', où X est une variable. Ce message doit être identifié comme une Sortie de ce processus fonctionnel.

Exemple temps réel 2 illustrant la règle b). La FUR de l'exemple temps réel 1 dans la section 3.5.9 peut aussi indiquer que les ' FP A' et 'FP B' doivent gérer une condition d'erreur lorsque le pilote du périphérique ne parvient pas à obtenir les données d'un ou plusieurs capteurs non intelligents. Un capteur non intelligent ne peut pas, par définition, livrer un message d'erreur. Le pilote de périphérique 'FP B' obtiendra, très probablement, une chaine de valeurs de la matrice de capteurs non intelligents, par exemple, état 1, état 2, état 3, aucune réponse, état 5, aucune réponse, état 7 etc. et passe cette chaine au 'FP de A' de l'application. Aucun message d'erreur distinct ne doit être déterminé comme une Sortie de 'FP B' du pilote de l'appareil ni être déterminé comme une Entrée vers le 'FP A' du processus de contrôle de l'application.

4

LA PHASE DU MESURAGE

4.0 Sommaire du chapitre

Le présent chapitre traite de l'étape finale du processus du mesurage. Tout d'abord, l'unité COSMIC du mesurage est définie (à savoir qu'un mouvement de données est mesuré comme un Point de fonction COSMIC ou 'PFC'). Ensuite, les règles d'attribution de la taille de la FUR du logiciel à mesurer sont définies. Les règles définies indiquent comment calculer la taille des différents morceaux du logiciel.

De plus, les règles sont définies pour calculer la taille des changements des morceaux du logiciel (traitées de façon similaire aux projets d'amélioration). Enfin, le chapitre examine la possibilité d'utiliser des 'extensions locales' avec la méthode COSMIC. Par exemple, une règle dans l'environnement local d'une organisation qui désire tenir compte de certains aspects de la fonctionnalité qui un sens au niveau local seulement.

4.1 Le processus de la phase du mesurage

La méthode générale pour mesurer un morceau de logiciel, lorsque ses exigences fonctionnelles de l'utilisateur ont été exprimées dans le cadre du modèle générique de logiciel COSMIC, est résumée à la Figure 4.0 ci-dessous.

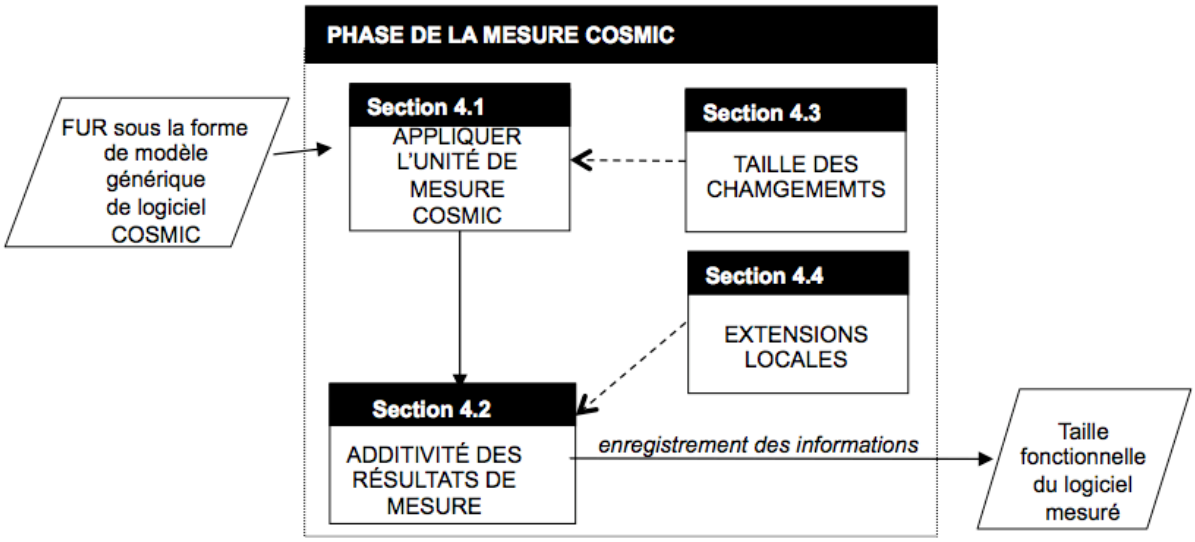


Figure 4.0 – Processus général de la phase du mesurage COSMIC

Chaque étape de cette méthode fait l'objet d'une section dans ce chapitre où les définitions et les principes à appliquer sont présentés, ainsi que des règles et des exemples.

4.2 Application de la fonction du mesurage

DÉFINITION – Étalon du mesurage COSMIC.
L'étalon du mesurage COSMIC, soit 1 PFC (Point de Fonction COSMIC), est défini comme la taille d'un mouvement de données. REMARQUE : L'étalon du mesurage était désigné dans le versions 2.0 de COSMIC comme un 'Cfsu' (COSMIC functional size unit)

Selon cette fonction du mesurage, chaque instance d'un mouvement de données (Entrée, Sortie, Lecture ou éCriture) (identifié selon la section 4.1) où la donnée déplacée doit subir une manipulation de données telle que : un ajout, une modification ou une suppression, doit se voir attribuer une taille fonctionnelle d'un point numérique CFP.

4.3 Additivité des résultats du mesurage.

Cette étape consiste à regrouper les résultats de la fonction du mesurage, telle qu'appliquée à tous les mouvements de données identifiés, en un seul nombre représentant la taille fonctionnelle. Cette étape se réalise selon les règles suivantes.

4.3.1 Règles générales pour l'additivité

RÈGLES – Additivité des résultats du mesurage.

a. Pour chaque processus fonctionnel, la taille fonctionnelle des mouvements de données est regroupée en une seule valeur de taille fonctionnelle par l'intermédiaire d'une addition arithmétique du nombre de mouvements de données ;

```
Taille CFP ( processus fonctionnel ) =
      Σ taille ( Entrée i ) + Σ taille ( Sortie i ) +
      Σ taille ( Lecture i ) + Σ taille ( éCriture i )
```

b. Pour chaque processus fonctionnel, la taille fonctionnelle de chaque modification de FUR est additionnée à la somme des tailles fonctionnelles des mouvements de données, pour qui les données ont subit une manipulation de donnée par l'entremise d'un processus fonctionnel. Voici la formule :

```
Taille CFP ( modification ( processus fonctionnel i ) ) =
   Σ taille ( mouv. de donnée i --> manip. 'ajouté' ) +
   Σ taille ( mouv. de donnée i --> manip. 'modifié' ) +
   Σ taille ( mouv. de donnée i --> manip. 'supprimé' )
```

Pour en savoir plus sur l'additivité de la taille fonctionnelle, voir la section 4.3.2. Pour mesurer la taille d'un morceau de logiciel modifié, voir la section 4.4.2

c. La taille d'un morceau de logiciel pour un périmètre défini doit être obtenue en additionnant les tailles fonctionnelles des processus fonctionnels de ce morceau, selon les règles (e) et (f) ci-dessous ;
d. La taille d'une modification d'un morceau logiciel d'un périmètre donné doit être obtenue en additionnant les tailles fonctionnelles de toutes les modifications de tous les processus fonctionnels de ce morceau, selon les règles (e) et (f) ci-dessous ;
e. La taille de chaque morceau de logiciel à mesurer dans une couche peut être obtenue en les additionnant, mais seulement au même niveau de granularité des processus fonctionnels de leur FUR ;
f. Les tailles des morceaux de logiciel et/ou des modifications apportées à un morceau de logiciel dans leur couche respective ou dans des couches différentes, ne devraient pas être additionnées entre elles, sauf si il est logique de le faire, pour les fins du mesurage ;
g. La taille d'un morceau de logiciel ne peut pas être obtenue en additionnant les tailles de ses composants (quelques soit le mode de décomposition) à moins que la taille des mouvements de données entre ces composants soient éliminée du calcul;
h. Si la méthode COSMIC est étendue localement (par exemple pour mesurer certains aspects de la taille non couverte par la méthode normalisée), alors la taille mesurée en utilisant une extension locale doit être rapportée séparément tel que décrit dans la section 5.1, et ne peut pas être additionné pour obtenir une taille selon l'étalon du mesurage COSMIC (voir la section 4.5).

EXEMPLE 1 pour les règles (b) et (c) : Des modifications apportées à un morceau de logiciel pourraient être : d'ajouter un nouveau processus fonctionnel de taille 6 CFP, d'ajouter un mouvement de données dans un autre processus fonctionnel, de faire des modifications dans trois autres mouvements de données et de supprimer deux mouvements de données. La taille totale de ces modifications serait 6 + 1 +3 +2 = 12 CFP.

EXEMPLE 2 pour la règle (f) : Si plusieurs parties importantes d'un morceau de logiciel étaient développées en utilisant des technologies différentes, développées aussi par des équipes de projets différentes, il pourrait ne pas y avoir de valeur ajoutée significative que d'additionner leur taille respective.

EXEMPLE 3 pour la règle (g) : Si un morceau de logiciel était :

Mesuré comme un tout, i.e. provenant d'un seul périmètre de mesurage ;

Et qu'ensuite, la taille de chacun de ses composants était mesurée séparément, i.e. chacun dans son propre périmètre de mesurage;

La taille fonctionnelle obtenue en additionnant toutes les tailles des composants séparés (au deuxième point) aurait la conséquence suivante : La taille mesurée dans ce cas dépasserait la taille mesurée du logiciel pris comme un tout (premier point). Ceci est due au fait que la taille de tous les mouvements de données entre les composants ont été comptés plus d'une fois. Ces mouvements de données inter-composants ne sont pas visibles quand le morceau de logiciel est mesuré comme un tout et ces derniers doivent donc être éliminés de l'équation du mesurage des composants séparés pour obtenir la vraie taille de l'ensemble. Voir aussi l'exemple de la section traitant du mesurage à différent niveau de granularité d'une architecture logiciel, dans le document *'Advanced and Related Topics' [12].*

Il est à noter que, dans chaque couche identifiée, le processus de l'addition est progressif. De cette façon, un sous-total peut être généré pour chaque processus fonctionnel individuel ou pour un logiciel complet d'une couche en particulier, et ce, en fonction de la raison d'être et du périmètre de chaque exercice du mesurage et aussi suivant les règles (d), (e), et (f)
ci-dessus.

4.3.2 Informations supplémentaires sur l'additivité des tailles fonctionnelles

Dans un contexte où la taille fonctionnelle doit être utilisée comme variable dans un modèle, par exemple pour estimer l'effort d'un logiciel de plus d'une couche ou de composants de même niveau par exemple, une addition sera habituellement pratiquée par couche ou par composant de même niveau, et ce même si souvent, il ne sont pas développés avec la même technologie.

Exemple1: considérons un logiciel où la couche applicative doit être réalisée en utilisant un langage de troisième génération et une bibliothèque de programmes existante, tandis qu'une couche contenant un pilote doit être réalisée en utilisant un langage assembleur. L'unité d'effort associé à la construction de chaque couche sera, très probablement, différente, et par conséquent, l'estimation de l'effort sera calculée distinctement pour chacune des couches en tenant compte de leur taille respective.

EXEMPLE 2: si une équipe de projet doit développer un certain nombre de morceaux de logiciel d'importances et que cette équipe est intéressée par sa performance globale, elle peut additionner les heures de travail nécessaires pour développer chaque morceau. De la même manière, cette équipe peut additionner les tailles des morceaux majeurs de logiciel qu'elle a développé si (et seulement si) ces tailles satisfont les règles ci-dessus.

La raison pour laquelle les tailles des morceaux majeurs de logiciel de différentes couches d'une architecture à couches standard, mesurées au même niveau de granularité de processus fonctionnel, peuvent être additionnées ensemble, est qu'une telle architecture possède un ensemble d'utilisateurs fonctionnels définis de manière cohérente. Chaque couche est un utilisateur fonctionnel des couches inférieures qu'elle utilise, et tout morceau de logiciel d'une couche en particulier peut être un utilisateur fonctionnel de son autre morceau de logiciel de même niveau. La seule chose qu'une telle architecture requiert est que les FUR de tous les morceaux de logiciel échangent des messages. Il est donc logique et raisonnable que les tailles de ces morceaux de logiciels puissent être additionnées ensemble, toujours assujetties aux règles (d), (e) et (f) ci-dessus. Cependant, à l'opposé, la taille d'un morceau de logiciel ne pourrait pas être obtenue en additionnant les tailles de ses composants réutilisables séparément, à moins que les mouvements de données inter-composants soient éliminés, en référence à la règle (f) ci-dessus.

L'additivité des résultats du mesurage par type de mouvement de données pourrait être utile pour analyser la contribution de chaque type (de mouvement de données) à la taille totale d'une couche et pourrait ainsi aider à caractériser la nature fonctionnelle de la couche mesurée.

4.4 Informations supplémentaires sur la taille fonctionnelle de modifications logicielles

Une 'modification fonctionnelle' d'un logiciel existant est interprétée par la méthode COSMIC comme étant 'toute combinaison d'ajouts de nouveaux mouvements de données ou modifications ou suppressions de mouvements de données existants'. Les termes 'amélioration' et 'maintenance'[15] sont le plus souvent utilisés comme synonyme de 'modification fonctionnelle'.

Le besoin de modifier un logiciel peut provenir:

- *D'une nouvelle FUR (i.e. que des ajouts à une fonctionnalité existante), ou ;*
- *D'une modification dans la FUR (pouvant être accompagné d'ajouts, de modifications et/ou de suppressions) ou ;*
- *D'un besoin de 'maintenance' pour éliminer un/des défauts.*
- *Les règles du mesurage liées aux modifications mentionnées ci-dessus sont les mêmes, mais le mesureur doit être vigilant afin de distinguer le contexte dans lequel ces mesures et estimations seront faites.*

Lorsqu'un morceau de logiciel est complètement remplacé par un autre, par exemple, en réécrivant le code avec ou sans extension (ou omission) de fonctionnalité, la taille fonctionnelle de ce morceau de logiciel sera la taille du logiciel de remplacement, mesuré d'après les règles usuelles du mesurage. Ce cas ne sera pas revisité dans cette section. Cependant, le mesureur devrait être conscient qu'il est nécessaire de distinguer les projets de développement d'un nouveau logiciel, des projets de 'redéveloppement' ou de 'remplacement' d'un logiciel existant, lors de la réalisation d'une mesure de performance ou d'estimation.

Souvent, les parties obsolètes d'une application sont supprimées (où 'déconnectée') en laissant le code du programme en place (en enlevant seulement le lien vers ces parties obsolètes). Lorsque la fonctionnalité de la partie obsolète atteint 100 CFP mais que le code qui a été modifié pour effectuer cette déconnection est de, disons, 2 mouvements de données, il faut compter 100 (et non pas 2) mouvements de données pour la taille de la modification fonctionnelle. Nous mesurons la taille de la FUR, et non pas la taille de la modification. [17]

Notons la différence entre la taille de la modification fonctionnelle et le changement dans la taille fonctionnelle du logiciel. Habituellement, elles sont différentes : La taille de cette dernière est adressée en section 4.4.2.

4.4.1 Modification des fonctionnalités

Tout mouvement de données de type E, S, L, et C implique deux types de fonctionnalités : il déplace un seul groupe de données et il possède une certaine quantité de manipulation de données (pour ce dernier, voir la section 3.5.6). Donc, pour les fins du mesurage, un mouvement de données est considéré comme étant fonctionnellement modifié comme suit:

- le groupe de données se déplace et/ou
- les manipulations de données associées sont modifiées de quelque façon que ce soit.

Un groupe de données est modifié si :

- des nouveaux attributs sont ajoutés à ce groupe de données et/ou ;
- les attributs existants sont supprimés du groupe de données et/ou ;
- un ou plusieurs attributs existants sont modifiés, c'est-à-dire selon leur signification, type ou format (en excluant les valeurs de ces attributs).

DÉFINITION – Modification (de la fonctionnalité des mouvements de données)
a) Un mouvement de données est considéré fonctionnellement modifié si au moins une des conditions suivantes s'applique: • le groupe de données déplacé est modifié,

- la manipulation de données associée est modifiée.

b) Un groupe de données est modifié, si au moins une des conditions suivantes s'applique:
   - un ou plusieurs nouveaux attributs sont ajoutés au groupe de données,
   - un ou plusieurs attributs existants sont enlevés au groupe de données,
   - un ou plusieurs attributs existants sont modifiés, par exemple la présentation ou le format (le changement des valeurs ne compte pas)

c) une manipulation de données est modifiée, si elle est, de quelque façon, fonctionnellement changée.

Exemple: Une manipulation de données est modifiée si elle est fonctionnellement changée de quelque façon que ce soit, par exemple en changeant les calculs, le formatage spécifique, la présentation, et/ou la validation des données. La 'présentation' peut signifier, par exemple la police de caractère, la couleur de l'arrière plan, la longueur des champs, le nombre de décimales, etc.

Les commandes de contrôle et les données générales d'application ne sont pas concernées par les mouvements de données, car il n'y a pas de données appartenant à un objet d'intérêt qui soit déplacée. En conséquence, les modifications affectant les commandes de contrôle et les données générales d'application ne doivent pas être comptés. Par exemple, si la couleur d'écran était changée pour tous les écrans, ce changement ne devrait pas être compté. (Voir la section 3.5.10 pour plus de détails sur les commandes de contrôle et les données générales d'application.)

RÈGLES – Modification d'un mouvement de données

a. Si un mouvement de données devait être modifié suite à un changement dans une manipulation de données associée et/ou suite à un changement du nombre ou du type d'attributs de données déplacé par ce mouvement de données, un CFP (modification) devra être mesuré, quelque soit le nombre réel de modifications dans ce mouvement de données ;
b. Si un groupe de données devait être modifié, les mouvements de données déplaçant ce groupe, dont la fonctionnalité n'est pas affectée par cette modification ne sera pas identifiés comme étant un mouvement de données modifié.

REMARQUE 1 : Le changement de la valeur actuelle d'un attribut, telle que la modification d'un code d'un groupe de données dont les valeurs des attributs est représenté par un système de codes, n'est pas une modification de 'signification, type ou format' d'attribut.

REMARQUE 2 : Une modification de toute donnée qui apparait sur des écrans d'entrée ou de sortie qui ne représentent pas d'objet d'intérêt pour un utilisateur fonctionnel ne devrait pas être identifiée comme un CFP de modification (Voir section 3.3.4 pour des exemples d'une telle donnée).

EXEMPLE pour les règles (a) et (b) : Supposons une requête visant à ajouter ou modifier des attributs d'un groupe de données 'D1', telle sorte qu'après la modification, le groupe de données devienne 'D2'. Dans le processus fonctionnel 'A' où cette modification a été demandée, tous les mouvements de données affectés par cette modification devraient être identifiés et comptés comme étant modifiés. Ainsi, selon la règle (a), si le groupe de données modifié 'D2' a été envoyé vers un stockage persistant et/ou est sortie du processus fonctionnel 'A', il faudra identifier et compter respectivement des mouvements de données en éCriture et/ou en Sortie comme étant modifiés. Cependant, il est possible que d'autres processus fonctionnels fassent entrer et/ou lisent ce même groupe de données 'D2', mais leur fonctionnalité reste inchangée par la modification, car ils n'utilisent pas les attributs de données modifiés. Ces processus fonctionnels continuent à traiter le groupe de données déplacé comme si c'était toujours 'D1'. Ainsi, selon la règle (b), ces mouvements de données des autres processus fonctionnels qui ne sont pas affectés par la modification du(des) mouvement(s) de données du processus fonctionnel 'A', ne doivent pas être identifiés et comptés comme étant modifiés.

Exemple d'affaires: Si un message d'erreur/confirmation doit être modifié (c'est-à-dire des textes ajoutés, modifiés ou supprimés) il sera identifié pour la mesure, indépendamment de si oui ou non le texte modifié est une conséquence d'une exigence de changer un autre mouvement de données

4.4.2 La taille d'un logiciel fonctionnellement modifié

Après avoir fonctionnellement modifié un morceau de logiciel, sa nouvelle taille totale est:

- la même que la taille originale,
- plus (+) la taille fonctionnelle de tous les mouvements de données supplémentaires,
- moins (-) la taille fonctionnelle de tous les mouvements de données supprimés.

Les mouvements de données modifiés n'ont aucune influence sur la taille de ce morceau de logiciel car ils existaient déjà avant et après que les modifications aient été faites.

4.5 Extension de la méthode de mesurage COSMIC.

4.5.1 Introduction.

La méthode de mesurage de taille fonctionnelle COSMIC ne prétend pas mesurer tous les aspects de la taille du logiciel. Aujourd'hui, la méthode de mesurage COSMIC n'est pas conçue pour fournir une manière standard de mesurer la taille de certains types de FUR, tel que la mesure d'algorithmes mathématiques complexes ou des FUR comportant des règles complexes telles que l'on en trouve dans les systèmes experts, par exemple. Aussi, l'influence du nombre d'attributs de données par mouvement de données sur la taille fonctionnelle d'un logiciel n'est pas prise en compte par la méthode de mesurage. D'un autre côté, l'influence de la manipulation de données des sous-processus sur la taille fonctionnelle n'est prise en compte par l'intermédiaire d'une simplification des règles qui est valide que pour certains domaines de logiciel, comme défini dans la section 2.1 sur l'applicabilité de la méthode.

D'autres paramètres tels que la 'complexité' (quel que soit sa définition) pourraient être considérés influençant la taille fonctionnelle. Une discussion constructive sur cette question exigerait un accord commun sur les définitions des autres éléments mal connus influençant la taille du logiciel. De telles définitions font toujours l'objet de nouvelles recherches et de nombreuses discussions.

Néanmoins, la mesure de la taille COSMIC est considérée comme étant bon moyen pour les fins visés et le domaine d'applicabilité de la méthode. Il est possible que dans le contexte d'une organisation où la méthode de mesurage COSMIC est utilisée, que l'on puisse tenir compte de différents types de fonctionnalité de la méthode, de manière à ce qu'elle soit significative dans le cadre de la règle locale de l'organisation. Pour cette raison, la méthode de mesurage COSMIC prévoit des adaptations locales. Dans le cas où de telles adaptations ou extensions locales sont utilisées, les résultats du mesurage doivent être rapportés selon la convention utilisée, telle que présentée dans la section 5.1. Les sections suivantes montrent comment 'étendre' la méthode à une règle locale.

4.5.2 Mesure d'un logiciel 'riche en données de manipulation'

La méthode de mesurage COSMIC a été conçue pour mesurer du logiciel 'riche' en mouvement de données. Comme toutes les autres méthodes reconnues du mesurage de la taille fonctionnelle (FSM), le modèle du mesurage ne vise pas explicitement la mesure des fonctionnalités de manipulation des données. Au lieu de cela, la méthode suppose que le nombre de fonctionnalités de manipulation de données s'explique par le nombre de types de mouvements de données (voir plus loin). Cette hypothèse s'est avérée être raisonnable dans la pratique pour la mesure de performance des projets et l'estimation pour laquelle la méthode a été conçue et pour les domaines dans lesquels il est couramment utilisé.

Toutefois, l'expérience a montré que la méthode peut aussi souvent être appliquée avec succès pour mesurer la taille des logiciels 'riche en données de manipulation', par exemple des logiciels scientifique et d'ingénierie. C'est vrai, par exemple, lorsque le logiciel doit gérer de gros volumes de données, conduisant à un grand nombre de mouvements de données (type). Ce dernier peut effectivement expliquer toute manipulation de données mathématiquement complexes qui peuvent être également être présente. 'Appliquée avec succès' signifie que la méthode a produit des tailles significatives et utiles en lien avec la raison d'être du mesurage. Les exemples incluent la taille des systèmes experts, des logiciels

pour traiter numériquement des variables continues, qui recueille et analyse les données d'expériences scientifiques ou des mesures d'ingénierie, etc.

Cependant, compte tenu du modèle fondamental de la méthode COSMIC, les utilisateurs de la méthode, lorsqu'ils sont confrontés avec une taille fonctionnelle d'un logiciel qui est riche en manipulation de données, doivent décider eux-mêmes si la méthode produit vraiment des tailles fonctionnelles significatives et utiles en relation avec la raison d'être du mesurage. Lorsque la méthode ne peut expliquer adéquatement la manipulation de données, il est possible de développer une extension locale à la méthode pour surmonter cette limitation – Voir la section 4.5.

4.5.3 Restrictions sur les facteurs qui contribuent à la taille fonctionnelle

Dans ses domaines d'application, la méthode COSMIC ne tente pas de mesurer tous les aspects possibles des fonctionnalités pouvant contribuer à la taille du logiciel. Par exemple, la méthode de mesurage ne capture pas explicitement l'influence de la 'complexité' du logiciel. Il existe plusieurs types de complexité par exemple d'architecturale, de sémantique, de calendrier, de processus, de données, etc. et dans la mesure de la taille fonctionnelle, la méthode COSMIC tient compte d'une façon simplifiée de la contribution de la complexité à la taille du processus (et donc indirectement de la complexité des données).

La méthode ne considère pas l'influence du nombre d'attributs de données pour chaque mouvement de données pour le calcul de la taille fonctionnelle du logiciel. Tel que décrit dans la section 4.5.6, si vous le souhaitez, ces aspects de la taille fonctionnelle peuvent être soutenus par une extension localede la méthode de mesurage COSMIC.

4.5.4 Restrictions sur la mesure des très petits morceaux de logiciel

Toutes les méthodes du mesurage de la taille fonctionnelle sont basées sur les hypothèses d'un modèle simplifié de la fonctionnalité du logiciel qui est raisonnable 'en moyenne' pour son domaine prévu d'applicabilité. Il est donc nécessaire lors du mesurage de prendre certaines précautions, en comparant ou en utilisant des tailles de très petits morceaux de logiciel et surtout de très petits changements à un morceau de logiciel, où l'hypothèse de la 'moyenne' peut s'avérer fausse. Dans le cas de la méthode COSMIC, "très petits morceaux signifie 'quelques mouvements de données'.

4.5.5 Extension locale avec des algorithmes complexes

Si l'organisation juge nécessaire de tenir compte des algorithmes complexes, une norme locale peut être établie pour cette fonctionnalité exceptionnelle. Dans n'importe quel processus fonctionnel où il y a un sous-processus fonctionnel comportant des manipulations de données anormalement complexes, le mesureur est libre d'assigner ses propres points de fonction localement déterminés.

> *EXEMPLE : Une extension d'une norme locale pourrait être dans notre organisation, un point de fonction local assigné pour des algorithmes mathématiques comme (liste d'exemples ou de règles locales significatifs et bien compris). Deux points de fonction sont assignés à (une autre liste d'exemples de règles locales), etc.*

4.5.6 Extension locale avec des fractions d'unité du mesurage.

Lorsqu'il faut davantage de précision pour la mesure des mouvements de données, alors une fraction de l'unité du mesurage peut être définie. Par exemple, un mètre peut être subdivisé en 100 centimètres ou 1000 millimètres. Par analogie, le mouvement d'un attribut de données simple peut être utilisé comme une fraction d'unité du mesurage. Des mesures sur un petit échantillon de logiciel lors des essais COSMIC indiquent que, dans un échantillon du mesurage, le nombre moyen d'attributs par mouvement de données ne varie pas beaucoup d'un mouvement de données à un autre pour les quatre types de mouvements de données. Pour cette raison et pour conserver la logique du mesurage, l'unité du mesurage COSMIC 1 CFP, a été fixée au niveau d'un mouvement de données. Cependant, il faut être prudent lorsqu'on compare des tailles fonctionnelles CFP, de deux morceaux de logiciel différents où le nombre moyen d'attributs de données par mouvement de données diffère drastiquement entre les deux morceaux de logiciel.

Si l’on souhaite raffiner la méthode COSMIC en présentant une fraction d’unité du mesurage il est possible de le faire, mais il faut indiquer clairement que les résultats qui résultent d’une telle méthode de mesurage de taille fonctionnelle ne sont pas exprimées en points de fonction CFP selon les normes COSMIC.

5

RAPPORTER LE MESURAGE

Sommaire du chapitre

Lorsqu'une mesure est terminée et acceptée, le résultat doit être présenté et les données sur la mesure archivées afin de s'assurer que le résultat est toujours clairement interprétable. Ce chapitre répertorie les paramètres qui devraient être considérés pour l'enregistrement.

5.1 Étiquetage

Le modèle générique de logiciel peut être présenté sous forme de matrice où

- les rangées peuvent représenter les processus fonctionnels (qui peuvent être groupés par couches),
- les colonnes ; des groupes de données et
- les cellules ; d'identifier les mouvements de données (Entrée, Sortie, Lecture et éCriture).

Cette représentation du modèle générique de logiciel est présentée en annexe A.

Les résultats du mesurage COSMIC doivent être signalés et archivés selon les conventions suivantes. Lorsque vous signalez une taille fonctionnelle COSMIC elle devrait être étiquetée selon la convention suivante, conformément à la norme ISO/IEC 14143-1: 2007.

RÈGLE – Étiquetage du mesurage COSMIC
Les résultats du mesurage COSMIC seront notés comme suit, « **x** PFC (v.**y**) », où : • "**x**": représente la valeur numérique de la taille fonctionnelle ; • "v.**y**": représente l'identification de la version de la norme de la méthode COSMIC utilisée pour obtenir la valeur numérique "**x**". Remarque: Si une méthode d'approximation locale a été utilisée pour obtenir la mesure, elle devra être notée ailleurs. Si la mesure a été effectuée à l'aide des conventions d'une version standard COSMIC, la convention étiquetage ci-dessus doit être utilisée, voir section 5.2.

*Exemple: Un résultat obtenu en utilisant les règles de ce Manuel de mesurage est noté de la façon suivante: '**x** PFC (v4.0)'*

Lorsque les extensions locales sont utilisées, tels que définis à l'article 4.5, le résultat du mesurage doit être rapporté tel que défini ci-après.

Règle – Étiquetage des extensions locales
Un résultat du mesurage COSMIC utilisant des extensions locales doit être noté de la

façon suivante, « x PFC (v. y) + z Local PF », où:

- "**x**": représente la valeur numérique obtenue en additionnant tous les résultats des mesures individuelles selon la norme de la méthode COSMIC, version v.y ;
- "v.**y**": représente l'identification de la version de la norme de la méthode COSMIC utilisée pour obtenir la valeur numérique fonctionnelle "**x**" ;
- "**z**": représente la valeur numérique obtenue en additionnant tous les résultats des mesures individuelles à partir des extensions locales de la méthode COSMIC.

5.2 Archivage des résultats du mesurage COSMIC

Lorsqu'on archive les résultats du mesurage COSMIC, les informations suivantes doivent être conservées pour s'assurer que les résultats sont toujours interprétables.

Règle – Rapporter la mesure COSMIC

En plus des résultats du mesurage tels qu'enregistrés dans la section 5.1, les caractéristiques de chaque mesurage doivent être enregistrées, comme la raison d'être du mesurage et le niveau désiré de comparabilité aux autres mesures, par exemple pour des fins d'analyse comparative.

a. Une identification du composant logiciel mesuré (nom, numéro de version ou numéro de configuration).
b. Le nom des sources d'information utilisées pour identifier la FUR pour le mesurage ;
c. Le domaine d'applicabilité du logiciel mesuré.
d. Une description de l'architecture du logiciel, les couches pour lesquelles les mesures sont réalisées, s'il y a lieu.
e. Un énoncé sur la raison d'être du mesurage.
f. Une description du périmètre du mesurage, et sa relation avec les autres périmètres du mesurage, s'il y a lieu. (Utiliser les catégories génériques du périmètre dans la section 2.2)
g. Le modèle du mesurage utilisé (COSMIC ou local), avec le mode de traitement (en ligne ou par lots).
h. Les utilisateurs fonctionnels du logiciel.
i. Le niveau de granularité des artéfacts logiciels disponibles et le niveau de décomposition du logiciel.
j. La phase du cycle de vie du projet lorsque la mesure a été faite (est-ce que la mesure est une estimation approximative basée sur des exigences incomplètes, ou elle a été effectuée sur la base des fonctionnalités réellement livrées).
k. La cible ou la marge d'erreur connue du mesurage.
l. Des indications sur si la méthode de mesurage COSMIC standard a été employée, et/ou si une approximation locale de la méthode standard a été appliquée, et/ou si des extensions locales ont été employées (voir la section 4.5). Employer les conventions d'étiquetage des sections 5.1 ou 5.2.
m. Une indication sur si l'on mesure une fonctionnalité développée ou déjà livrée (la fonctionnalité 'développée' est obtenue en créant un nouveau logiciel ; la fonctionnalité 'déjà livrée' inclut la fonctionnalité 'développée' en plus d'autres fonctionnalités comme par exemple toutes formes de réutilisation de logiciels existants ou d'utilisation de paramètres existants pour ajouter ou modifier la fonctionnalité, etc.).
n. Une indication sur si la mesure provient d'une fonctionnalité nouvellement fournie

ou si elle est le résultat d'une activité de ‘maintenance’ ou 'd'amélioration' (i.e. la somme des ajouts, modification et suppressions - voir 4.4).

o. Le nombre de composants majeurs dont les tailles ont été additionnées ensemble pour obtenir la taille totale telle qu’enregistrée, s’il y a lieu.

p. Le pourcentage de fonctionnalités implémentées par le logiciel réutilisé.

q. Saisir, pour chaque périmètre de logiciel du le périmètre global du mesurage, une matrice du mesurage, telle qu'indiqué dans l'annexe A.

r. Le nom du mesureur et de toutes les qualifications et certifications COSMIC; la date du mesurage.

RÉFÉRENCES

On trouvera sur le Web tous les documents COSMIC énumérés ci-dessous, y compris les traductions dans d'autres langues. Voir www.cosmicon.com.

Les titres des documents COSMIC ne donnent pas le numéro de version de la méthode à laquelle ils se rapportent. Tous les documents sont mis à jour pour les faire coïncider avec la version 4.0 de la méthode.

[1] ISO/IEC 14143/1:2011 Software Engineering – COSMIC: a functional size measurement method, www.iso.org

[2] Introduction to the COSMIC Method of measuring software

[3] (Example of several papers by the same authors) Al-Sarayreh, K.T. and A. Abran, Specification and Measurement of System Configuration Non Functional Requirements, 20th International Workshop on Software Measurement (IWSM 2010), Stuttgart, Germany, 2010

[4] Guideline for Sizing Real-time Software

[5] Guideline for 'Measurement Strategy Patterns'

[6] Guideline for approximate COSMIC functional size measurement (under construction)

[7] Guideline for Sizing Business Application Software

[8] Guideline for sizing Data Warehouse Application Software

[9] Guideline for Sizing Service-Oriented Architecture Software

[10] Quick Reference Guide to the COSMIC method for sizing Business Application Software

[11] Quick Reference Guide to the COSMIC method for sizing Real-Time Application Software

[12] Advanced and Related Topics

[13] Guideline for Convertibility (under construction)

[14] International Vocabulary of Basic and General Terms in Metrology, International Organization for Standardization, Switzerland, 2nd edition, 1993, ISBN 92-67-01075-1

[15] ISO/IEC 15939:2002, Systems and Software Engineering – Measurement Process, definition 3.24

[16] Adapted from Merriam Webster's Collegiate Dictionary, 10th Edition

[17] Adapted from Merriam Webster's Collegiate Dictionary, 10th Edition, and La Petit Larousse Illustré, 1996 Edition

ANNEXE A – DOCUMENTATION DU MESURAGE DE LA TAILLE COSMIC

La matrice ci-dessous peut être employée pour tenir à jour les résultats du mesurage pour chaque composant identifié, d'un périmètre global qui a été retracé à partir du modèle générique de logiciel. Chaque périmètre du périmètre global du mesurage doit posséder sa propre matrice.

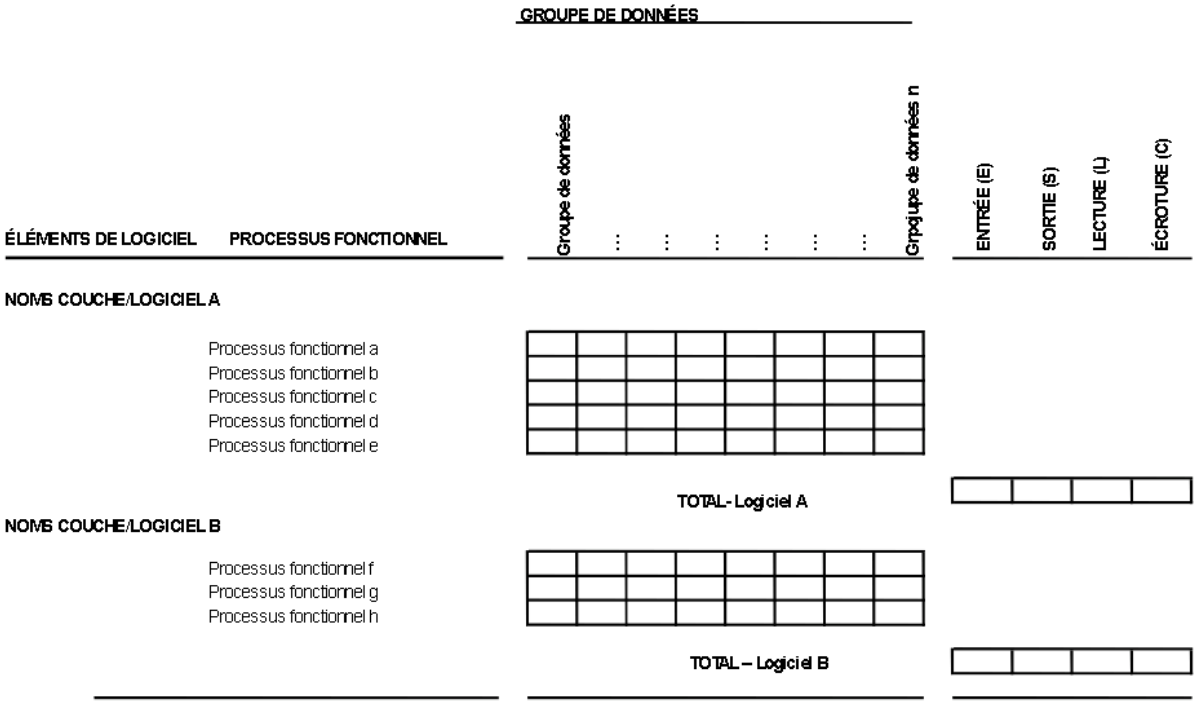


Figure A – Matrice du modèle générique de logiciel

Phase d'arrimage.

- Chaque groupe de données identifié est enregistré dans la colonne ;
- Chaque processus fonctionnel est enregistré sur une ligne spécifique, selon la couche auquel il appartient.
- Pour chaque processus fonctionnel identifié, les mouvements de données identifiés, qu'ils soient nouveaux ou modifiés, sont notés dans la cellule correspondante en utilisant la convention suivante: "E" pour une Entrée, "S" pour une Sortie, "L" pour une Lecture and "C" pour une éCriture ;
-

Phase du mesurage.

- Pour chaque processus fonctionnel identifié, chacun des mouvements de données sont comptabilisés par type, et chacune des sommes doit être enregistrée dans la colonne appropriée, à droite de la matrice ;
- La somme des mesures peut alors être calculée et enregistrée dans les cellules de chaque couche, sur la ligne « TOTAL ».

ANNEXE B – ÉVOLUTION DES EXIGENCES NON FONCTIONNELLES – EXEMPLES

Le tableau suivant répertorie quelques exemples d'énoncés d'exigences qui peuvent figurer au départ au niveau système (même avant que les exigences aient été attribuées aux logiciels ou matériels) ou au niveau logiciel comme non fonctionnel, mais qui ont évolués, en tout ou en partie au fur et à mesure de la progression du projet, dans un mélange de FUR pour les logiciels et d'énoncés d'exigences vraiment 'non fonctionnelles' (ENF).

- Colonne 1: exemples d'énoncés de système ou logiciel ENF
- Colonne 2: exemples de ce qui pourrait résulter d'un FUR de logiciel, au fur et à mesure de la progression du projet, à partir de l'ENF dans la colonne 1. La FUR peut être pour le logiciel à développer ou à être acquis par exemple un logiciel 'COTS' (logiciel commercial prêt à être utilisé) logiciel.
- Colonne 3: exemples des exigences et des contraintes sur le système ou le projet qui pourrait rester après avoir éliminé ce qui reste de la FUR du logiciel dans la colonne 2. Ce sont alors des exigences vraiment 'non fonctionnels'.

Exigences du système ou du logiciel qui peuvent apparaitre au départ comme 'non fonctionnels'	**Exemples de FUR pour les logiciels, devant être développés ou acquis, et qui peuvent évoluer à partir du système initial ENF**	**Exemples de vrais ENFs qui peuvent rester après que quelques exigences du système initial aient évoluées vers la FUR du logiciel**
Le temps de réponse du système pendant les heures de pointe ne doit pas dépasser une moyenne de X secondes.	Logiciel pour: • Alimentation externe des données requises par le système en temps réel • Surveiller et faire rapport sur les temps de réponse moyen	• Spécifique au matériel (rapidité) • Du logiciel pouvant être écrits dans un langage de bas niveau • Énoncé de temps réponse pour une cible spécifique
L'accès au système doit dépasser Y% en moyenne au cours de chaque année civile	Logiciel pour activer le changement rapide de traitement vers un processeur de sauvegarde sans interruption de service	• Un processeur de matériel de sauvegarde fonctionnant en mode 'éveil automatique' • un énoncé spécifique de disponibilité de la cible
Les paramètres de l'application doivent être facilement maintenues par le personnel de l'utilisateur	Logiciel permettant aux utilisateurs de maintenir les tables de paramètres	(aucun)
Le système doit être utilisable par le grand public sans formation avec un taux d'achèvement de Z%	Logiciel pour: • fournir une aide globale d'installation • fournir des menus bien structurés pour permettre la facilité d'utilisation • support des utilisateurs malvoyants	• exigences pour des claviers Braille • essais élaborés par les membres du public en général • un taux en achèvement avec une cible spécifique de Z %
L'utilisateur doit avoir la possibilité de sécuriser les fichiers de cryptage	Logiciel pour crypter et décrypter des fichiers à la demande de l'utilisateur	Utilisation d'une 'clef USB' ou un périphérique de chiffrement clé
Le système doit être	Une couche logicielle pour isoler la	Utilisation d'un langage très portable

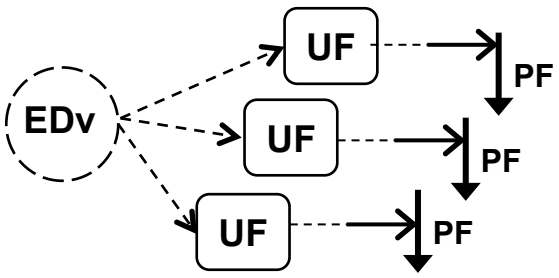
portable à travers les environnements matériels/logiciels X, Y et Z	fonctionnalité principale des exigences spécifiques des interfaces des environnements X, Y et Z	tels que Java

ANNEXE C –CARDINALITÉ DE DÉCLENCHEMENT DES ÉVÉNEMENTS, LES UTILISATEURS FONCTIONNELS ET LES PROCESSUS FONCTIONNELS

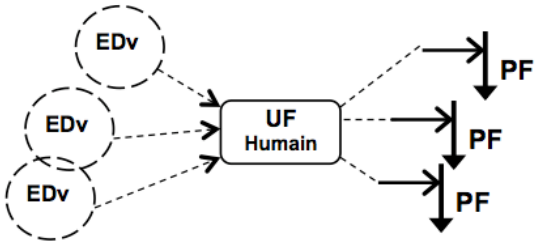
Toutes les relations, liées à l'évènement déclencheur / l'utilisateur fonctionnel / l'entrée de déclenchement / la chaine de processus fonctionnels (comme illustré à la Figure 3.3), peuvent être de plusieurs-à-plusieurs, en principe, à une exception près. (L'exception est que toute entrée de déclenchement ne peut engager qu'un seul processus fonctionnel – Voir la règle b) pour un processus fonctionnel à la section 3.2.2.).

Le tableau suivant montre des exemples de relations possibles. Notez que les cas ne sont peut-être pas exhaustifs. Le tableau utilise les abréviations et symboles suivants.

EDv	Évènement déclencheur	UF	Utilisateur Fonctionnel
----→	Le groupe de données (partie en pointillé) est déplacé par une entrée de déclenchement (flèche pleine)	↓PF	Processus Fonctionnel

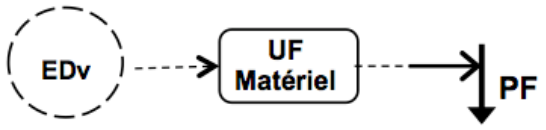
1. Un seul évènement déclencheur peut être causé par plusieurs UFs chacun initiant une entrée de déclenchement dans un ou plusieurs systèmes logiciels différents. Chaque entrée de déclenchement initie ses propres PFs	
	*Exemple temps réel: L'évènement déclencheur d'un tremblement de terre peut être détecté par plusieurs senseurs UFs indépendants. Chaque UF initie une entrée de déclenchement qui commence son PF dans un seul système ou dans différents systèmes.* *BUSINESS EXAMPLE: L'évènement déclencheur de l'embauche d'un nouvel employé provoqué par l'UF d'une intervention humaine qui entre les données de base de l'employé dans le 'système du personnel' et une autre UF humaine pour entrer les données sur les salaires au 'système de paie'.*

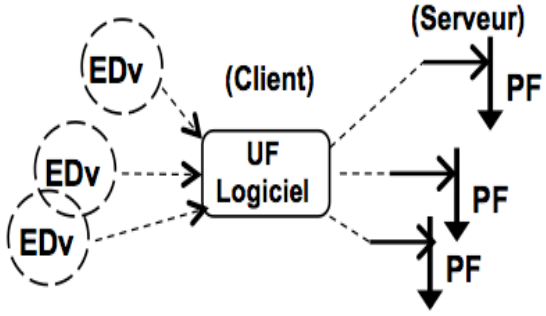
2. Chaque évènement déclencheur provoque une UF humaine qui initie une entrée de déclenchement différente. Chaque entrée de déclenchement commence son PF dans un ou plusieurs systèmes logiciels différents



Dans un système de gestion d'appel téléphonique d'urgence de la police, de nombreux types d'évènements déclencheurs peuvent être rapportés incitant une UF humaine à amorcer différentes entrées de déclenchement. Chacun de ces départs commence son PF pour enregistrer l'évènement. En outre, l'utilisateur humain peut engager des requêtes différentes d'entrée de déclenchement. Chacun de ces départs commence son PF dans le système de gestion des appels ou dans d'autres systèmes.

3. Un matériel ou un logiciel UF peut être conçu pour déceler (ou 'générer') un ou plusieurs types d'évènements spécifiques. Chacun de ces évènements engage l'UF à démarrer une Entrée de déclenchement. Chacun démarre le PF dans le système de logiciel existant.



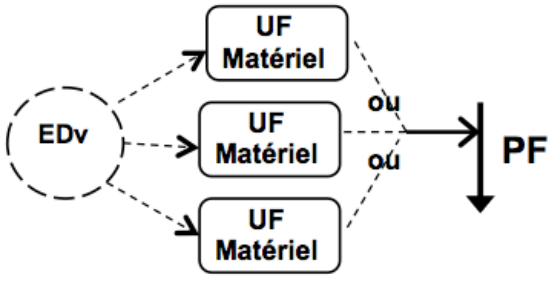


Exemple temps réel: Lorsque la température du liquide atteint un niveau prédéterminé (l'évènement déclencheur), un thermocouple

UF initie une Entrée de déclenchement pour démarrer le PF dans un système logiciel spécifique.

Exemple d'affaires: Dans une application de logiciel distribué, le composant client est une UF du composant serveur. Différents besoins d'information (les évènements déclencheurs) du composant client pourrait initier les différentes Entrées de déclenchement, chacun démarrant le PF du composant serveur, pour chaque type de service dont il a besoin.

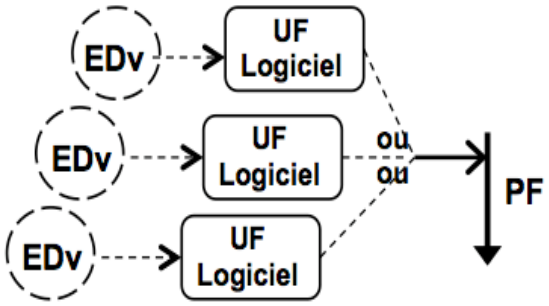
4. Deux ou plusieurs matériels d'UF du même logiciel peuvent détecter un seul évènement déclencheur. Chaque UF peut démarrer l'Entrée de déclenchement qui démarre ce PF.



Exemple temps réel: L'évènement déclencheur d'une situation anormale dans un système de contrôle de processus en temps réel, peut être capté par un ou plusieurs matériels UF. Chaque UF peut engager le PF d'arrêt d'urgence.

Remarque : Toute occurrence de ce PF sera initiée par le premier UF qui détectera l'évènement déclencheur.

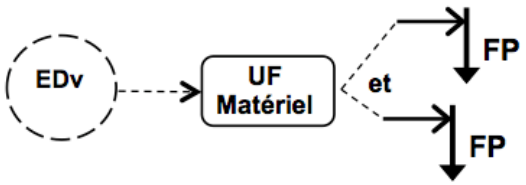
5. Deux ou plusieurs logiciels UF peuvent initier chacun une Entrée de déclenchement qui démarre ce PF.



Exemple d'infrastructure: Plusieurs logiciels UF peuvent s'appeler *l'un l'autre, c'est-à-dire initier le même PF dans le même composant logiciel réutilisable. (Dans ce cas le logiciel UF 'génère' l'évènement lorsqu'il appel le composant.)*

Remarque : Toute une occurrence de ce PF peut être initiée par un seul des logiciels possibles UF à la fois.

6. En détectant un évènement déclencheur, une UF peut engager deux ou plusieurs Entrées de déclenchement. Chaque Entrée de déclenchement démarre son PF dans un seul logiciel.



Exemple temps réel: Dans un système de double contrôle essentiel à la sécurité, un évènement déclencheur peut provoquer une UF (généralement le matériel) pour initier deux Entrées de déclenchement, chacune démarrant les PFs. Les deux PFs pourraient, par exemple, avoir la même FUR mais être mis au point par des groupes séparés grâce à une stratégie de diversité.

ANNEXE D – SOMMAIRE DES PRINCIPES ET DES RÈGLES DE LA MÉTHODE COSMIC

Le tableau ci-dessous identifie chaque principe et règle qui se trouve dans la méthode de mesurage COSMIC dans le but d'un référencement précis, avec le numéro de section dans la colonne de gauche.

Sec.	DESCRIPTION DES PRINCIPES ET RÈGLES
1.3.1	Le modèle contextuel de logiciel COSMIC **Principes** a) Le logiciel est limité par le matériel ; b) Le logiciel est typiquement structuré en **couches**; c) Une couche peut contenir un ou plusieurs morceaux de logiciel, de **même niveau**; d) Tout morceau de logiciel à mesurer doit être défini par un **périmètre de mesurage** , lequel doit aussi être entièrement compris à l'intérieur d'une seule couche de logiciel ; e) Le périmètre du morceau de logiciel à mesurer doit dépendre de la **raison d'être** du mesurage ; f) Les **utilisateurs fonctionnels** doivent être identifiés à partir des Fonctionnalités Utilisateurs Requises (FUR) du morceau de logiciel à mesurer comme les émetteurs et/ou les destinataires visés pour les données ; g) La FUR d'un logiciel peut être exprimée à différents **niveaux de granularité** ; h) Une mesure précise de la taille d'un morceau du logiciel exige que le niveau de granularité de la FUR soit celui ou le processus ou sous-processus fonctionnel peut être identifié; i) Une mesure approximative COSMIC de la taille d'un morceau de logiciel est possible si la FUR est mesurée à un haut niveau de granularité par une approche approximative et mise à l'échelle au niveau de granularité des processus et sous-processus fonctionnels.
1.3.2	**Le modèle générique de logiciel COSMIC** **Principes** a) Un morceau de logiciel interagit avec ses utilisateurs fonctionnels par-delà la **frontière**, et un **stockage persistent** à l'intérieur de sa frontière. b) Les exigences fonctionnelles de l'utilisateur d'un morceau de logiciel à mesurers peuvent être arrimées dans des processus fonctionnels uniques. c) Chaque processus fonctionnel comprend des sous-processus. d) Un sous-processus peut être soit un **mouvement de données** ou une **manipulation de données**. e) Un mouvement de données déplace un seul groupe de données. f) Il y a quatre types de mouvement de données, **entrée**, **sortie**, **écriture** et **lecture**. Une entrée déplace un groupe de données d'un processus fonctionnel à partir d'utilisateur fonctionnel. Une sortie déplace un groupe de données d'un processus fonctionnel vers un utilisateur fonctionnel. Une écriture déplace un groupe de données d'un processus fonctionnel vers un stockage persistant. Une lecture déplace un groupe de données d'un stockage persistant vers un processus fonctionnel. g) Un groupe de données consiste en un ensemble unique **d'attributs de données** décrivant un seul **objet d'intérêt.**

Sec.	DESCRIPTION DES PRINCIPES ET RÈGLES
	h) Chaque processus fonctionnel est démarré par le **déclenchement en entrée** d'un mouvement de données. Le groupe de données déplacé par l'entrée de déclenchement est généré par un utilisateur fonctionnel en réponse à un **évènement déclencheur**. i) Un processus fonctionnel doit comprendre au moins un déplacement de données d'entrée et une écriture ou un mouvement de données de sortie, c'est-à-dire qu'il doit comprendre un minimum de deux mouvements de données. Théoriquement, il n'y a pas de limite supérieure pour le nombre de mouvements de données dans un processus fonctionnel. j) En tant qu'approximation pour les objectifs , les manipulations de données des sous-processus ne sont pas mesurées séparément ; la fonctionnalité de toute manipulation de données est censée tenir compte du déplacement de données auquel il est associé.
1.4	**Principes du mesurage COSMIC** **Principes** a) La taille d'un processus fonctionnel est égale au nombre de ses mouvements de données. b) La taille fonctionnelle d'un morceau de logiciel, pour une portée définie, est égale à la somme des tailles de ses processus fonctionnels.
2.2	**Périmètre du mesurage** **Règles** a) Le périmètre de tout morceau de logiciel à mesurer doit être dérivé de la raison d'être du mesurage. b) Le périmètre de toute mesure ne doit pas couvrir plus d'une couche du logiciel à mesurer.
2.2.2	**Couche** **Principes** a) Le logiciel d'une couche fournit un ensemble de services qui est cohérent selon certains critères définis, et le logiciel dans d'autres couches peut être utilisé sans savoir comment ces services sont implémentés. b) La relation entre le logiciel dans n'importe laquelle des deux couches est définie par une « règle de correspondance » qui peut-être être soit: c) 'hiérarchique', c'est-à-dire que le logiciel de la couche A est autorisé à utiliser les services fournis par le logiciel de la couche B, mais pas vice versa (où la relation hiérarchique peut être vers le haut, vers le bas ou latérale), ou d) 'bidirectionnelle', c'est-à-dire que le logiciel de la couche A est autorisé à utiliser le logiciel de la couche B, et vice versa. e) Le logiciel d'une couche échange des groupes de données avec le logiciel dans un autre couche via leurs processus fonctionnels respectifs. f) Le logiciel d'une couche n'utilise pas nécessairement tous les services fonctionnels fournis par le logiciel d'une autre couche. g) Le logiciel d'une couche d'une architecture logicielle définie peut être partitionnée dans d'autres couches selon une architecture logicielle définie différemment.
2.3.1	**Utilisateurs fonctionnels** **Règles** a) Les utilisateurs fonctionnel d'un morceau de logiciel à mesurer doivent dérivés de la raison d'être du mesurage. b) Lorsque la raison d'être d'une mesure d'un morceau de logiciel est liée à l'effort de développement ou à l'effort de modification d'un morceau de logiciel, alors

Sec.	DESCRIPTION DES PRINCIPES ET RÈGLES
	les utilisateurs fonctionnels devraient être tous les expéditeurs et/ou destinataires visés des données vers et à partir des fonctionnalités nouvelles ou modifiées, tel que requis par les FURS.
2.4.3	**Niveau de granularité du processus fonctionnel** **Règles** a) Le mesurage exact de la taille fonctionnelle d'un morceau de logiciel exige que sa FUR soit connue à un niveau de granularité pour lesquels son processus fonctionnel et ses sous-processus de mouvements de données peuvent être identifiés. b) Si certaines exigences doivent être mesurées avant d'avoir été définis avec suffisamment de détails pour un mesurage précise, les exigences peuvent être mesurés à l'aide d'une approche approximative. Ces approches définissent comment les exigences peuvent être mesurées à un ou des niveau(x) de granularité plus élevés. Les mises à d'échelle sont ensuite appliquées au(x) niveau(x) plus élevés de granularité pour produire une taille approximative des processus fonctionnels et des sous-processus de mouvements de données à leur niveau de granularité. Voir 'Guideline for approximate COSMIC functional size measurement' [6].
3.2.2	**Processus fonctionnel** **Règles** a) Un processus fonctionnel doit appartenir entièrement à un morceau logiciel *dont le périmètre de mesurage a été défini*, et ce, pour une seule et unique couche . b) Toute Entrée de déclenchement d'un morceau de logiciel devant être mesurer peut initier seulement un processus fonctionnel de ce logiciel. c) Un processus fonctionnel contient au moins deux mouvements de données, une en Entrée plus une en Sortie et/ou une en éCriture Il n'y a pas de limite supérieure au nombre de mouvements de données dans un processus fonctionnel. d) l'exécution d'un processus fonctionnel doit être considéré comme terminé lorsqu'il a satisfait aux exigences de la FUR en réponse à une Entrée de déclenchement. Une pause pendant l'exécution pour des raisons techniques ne doit pas être considéré comme étant la terminaison du processus fonctionnel.
3.3.1	**Groupe de données** **Principes** a) Chaque groupe de données identifié doit être unique quant à l'ensemble des attributs qu'il contient ; b) Chaque groupe de données doit être directement lié à un objet d'intérêt décrit dans les FUR du logiciel à mesurer ; c) Le groupe de données doit être matérialisé dans le système informatique supportant le logiciel.
3.5.2	**Entrée (E)** **Principes** a) Une Entrée déplacera un seul groupe de données décrivant un seul objet d'intérêt d'un utilisateur fonctionnel, à travers la frontière du logiciel vers le processus fonctionnel auquel cette Entrée fait partie. Si l'Entrée d'un processus fonctionnel comporte plus d'un groupe de données, il faudra identifier une seule Entrée pour chaque groupe de données. (voir également la section 3.5.7 sur 'l'unicité des mouvements de données' ;

Sec.	DESCRIPTION DES PRINCIPES ET RÈGLES
	b) Une Entrée ne doit pas sortir des données au-delà de la frontière, ou lire ou écrire des données depuis/vers un stockage persistant. **Règles** a) Un groupe de données d'une Entrée de déclenchement peut comporter un seul attribut de données informant tout simplement le logiciel qu'un événement 'Y' est survenu. Très souvent, plus particulièrement pour les logiciels d'application d'affaires, le groupe de données de l'Entrée de déclenchement a plusieurs attributs de données qui informent le logiciel qu'un événement 'Y' s'est produit et présente ainsi les données au sujet de cet événement particulier ; b) Les signaux périodiques d'une l'horloge comme déclencheur d'événements sera toujours 'extérieur' au logiciel mesuré. Par conséquent lorsque qu'une horloge émet un signal périodique pour un événement se produisant toutes les 3 secondes, ce signal sera associé à une Entrée déplaçant un groupe de données d'un seul attribut de données. Noter qu'il n'y a aucune différence si l'événement déclencheur est produit périodiquement par du matériel ou par un autre morceau de logiciel en dehors de la frontière du logiciel mesuré ; c) À moins qu'un processus fonctionnel spécifique soit nécessaire, le fait d'obtenir le temps d'une horloge d'un système ne sera pas considéré comme la cause suffisante de caractérisation d'un mouvement de données en Entrée ; d) Si une occurrence d'un évènement spécifique déclenche l'Entrée d'un groupe de données comportant 'N' attributs (d'un objet d'intérêt particulier), et que la FUR admet qu'une autre occurrence du même évènements puisse déclencher une Entrée d'un groupe de données ayant les même valeurs pour un sous-ensemble de ces 'N' attributs, alors une seule Entrée sera identifiée : celle comportant tous les 'N' attributs. e) Lors de l'identification des Entrées dans un écran qui permet aux utilisateurs fonctionnels (humain) d'entrer des données dans les processus fonctionnels, par exemple dans une application d'affaires en ligne, d'analyser seulement des écrans qui contiennent des données. Ignorer 1) n'importe quel écran qui est formaté normalement 'vide', sauf pour les valeurs possibles par défaut et 2) ignorer tout les champs et autres rubriques permettant aux utilisateurs humains de comprendre les données d'entrée requises. REMARQUE. Il peut être nécessaire d'envisager les champs et autres rubriques lors du mesurage de la FUR pour les modifications apportées aux entrées – Voir la section 4.4.1.
3.5.3	**Sortie (S)** **Principes** a) Une Sortie déplacera un seul groupe de données décrivant un seul objet d'intérêt d'un processus fonctionnel, à travers la frontière du logiciel vers un utilisateur fonctionnel. Si la Sortie d'un processus fonctionnel comprend plus d'un groupe de données, il faudra identifier une seule Sortie pour chaque groupe de données. (voir également la section 4.1.7 sur 'l'unicité des mouvements de données'.) ; b) Une Sortie ne doit pas entrer des données à travers la frontière du logiciel, ni lire, ni écrire des données. à partir/cers un stockage persistent. **Règles** a) Une requête qui affiche le texte fixe (où « fixe » signifie le message ne contient aucune valeur de données variables, par exemple le résultat de la pression sur un bouton pour « Termes et Conditions » d'un site commerçant), doit être modélisée comme ayant une Sortie pour la visualisation du texte fixe. b) Remarque: Pour la sortie de la fonctionnalité 'Aide', voir 'Guideline for sizing Business Application Software'. Pour la sortie des messages liés aux conditions

Sec.	DESCRIPTION DES PRINCIPES ET RÈGLES
	d'erreurs ou de réussites confirmées, voir la section 3.5.11 de ce Manuel de mesurage. c) Si la Sortie d'un processus fonctionnel déplace un groupe de données comportant 'N' attributs (d'un objet d'intérêt particulier), et que la FUR admet qu'une autre occurrence du même événement puisse déclencher une Sortie d'un groupe de données ayant les même valeurs pour un sous-ensemble de ces 'N' attributs, alors une seule Sortie sera identifiée : celle comportant tous les 'N' attributs. d) Lors de l'identification des Sorties, ignorer tout les champs et autres 'en-têtes' qui sont là seulement pour permettent à des utilisateurs humains de comprendre les données de sortie. Remarque: Il peut être nécessaire de considérer le champ et les autres rubriques lors de la
3.5.4	**Lecture (L)** **Principes** a) La Lecture déplacera un seul groupe de données décrivant un seul objet d'intérêt d'un stockage persistant, à travers la frontière du logiciel, vers le processus fonctionnel auquel cette Lecture fait partie. Si le processus fonctionnel doit lire plus d'un groupe de données à partir du stockage persistant, il faudra identifier une seule Lecture pour chaque groupe de données à lire. (voir également la section 3.5.7 sur 'l'unicité des mouvements de données') ; b) La Lecture ne doit pas recevoir de données à travers la frontière du logiciel, ni sortir, ni écrire des données dans un stockage persistant; c) Le mouvement (ou la manipulation) des donnés qui ne proviennent pas d'une FUR mais qui sont internes à un processus fonctionnel et qui peuvent être changé seulement par : un programmeur, par un calcul de résultats intermédiaires, ou à partir de données conservées par un autre processus fonctionnel résultant de l'implantation, ne sera pas considéré comme étant un mouvement de données en Lecture ; d) Un mouvement de données en Lecture inclura toujours toutes fonctionnalités de 'demande de Lecture'. Ainsi un mouvement de données additionnel ne sera jamais nécessaire pour représenter une éventuelle fonctionnalité distincte de 'demande de Lecture'. Voir également la section 3.5.9 **Règles** a) Identifier une Lecture lorsque, selon la FUR, le logiciel à mesurer doit récupérer un groupe de données d'un stockage persistant. b) Ne pas identifier une Lecture lorsque la FUR du logiciel à mesurer spécifie n'importe quel logiciel ou matériel utilisateur fonctionnel comme source, ou comme moyen d'extraction, d'un groupe de données. (Dans ce cas voir les principes et règles pour les Entrées et les Sorties.)
3.5.5	**ÉCriture (C)** **Principes** a) Une éCriture déplacera un seul groupe de données décrivant un seul objet d'intérêt d'un processus fonctionnel duquel l'éCriture fait partie, à travers la frontière du logiciel, vers le 'stockage persistant'. Si le processus fonctionnel doit écrire plus d'un groupe de données dans le stockage persistant, il faudra identifier une seule éCriture pour chaque groupe de données à écrire. (voir également la section 3.5.7 sur 'l'unicité des mouvements de données'.) ; b) Une éCriture ne doit pas recevoir de données à travers la frontière du logiciel, ni sortir, ni lire des données dans un stockage persistant; c) Une manipulation de données de suppression d'un groupe de données du

Sec.	DESCRIPTION DES PRINCIPES ET RÈGLES
	'stockage persistant' ne sera mesurée que par un seul mouvement de données en éCriture ; d) Ce qui suit n'est pas est considéré comme des mouvements de données éCriture: e) Le déplacement ou la manipulation des données qui n'existaient pas au début d'un processus fonctionnel et qui ne sont pas devenues persistantes à la fin du processus fonctionnel; f) Création ou mise à jour des variables ou des résultats intermédiaires qui sont internes au processus fonctionnel; g) Stockage des données par un processus fonctionnel résultant uniquement de son implémentation, plutôt que de la FUR. (Un exemple serait le stockage temporaire de données lors d'un tri d'un grand nombre d'informations dans un traitement en lot). **Règles** a) Identifier une éCriture quand, selon la FUR, le logiciel à mesurer doit déplacer un groupe de données vers un stockage persistant. b) Ne pas identifier une éCriture lorsque la FUR du logiciel à mesurer spécifie n'importe quel logiciel ou matériel utilisateur fonctionnel comme destination, ou comme moyen de déplacement, du groupe de données. (Dans ce cas voir les principes et règles pour les Entrées et les Sorties.)
3.5.6	**Manipulation des données associées aux mouvements de données** **Principe** Toute manipulation de données d'un processus fonctionnel sera associée aux quatre types de mouvements de données (E, S, L, et C). Par convention, on suppose que les mouvements de données d'un processus fonctionnel représentent des manipulations de données pour le même processus fonctionnel. **Règles** a) Un mouvement de données d'Entrée comprend toutes les manipulations de données pour permettre à un groupe de données d'être entré par un utilisateur fonctionnel (par exemple les manipulations de mise en forme et de présentation) et permettre leur validation, b) Un mouvement de données de Sortie comprend toutes les manipulations de données créant les attributs d'un groupe de données en sortie et/ou permettant au groupe de données en sortie (par exemple les manipulations de mise en forme et les présentations) d'être acheminé à l'utilisateur fonctionnel prévu, c) Un mouvement de données de Lecture inclut tous les calculs et/ou traitements logiques nécessaires visant à récupérer un groupe de données à partir du stockage persistant. d) Un mouvement de données éCriture inclut tous les calculs et/ou traitements logiques pour créer ou mettre à jour un groupe de données à écrire, ou de supprimer un groupe de données. e) La manipulation de données associée à n'importe lesquels de ces mouvements de données n'inclut aucune des manipulations de données nécessaires après que le déplacement des données a été mené à bien, pas plus qu'il n'inclut les manipulations de données associées à tout autre mouvement de données.
3.5.7	**Unicité des mouvements de données et exceptions possibles** **Règles** a) Sauf si la FUR l'indique autrement, tous les groupes de données, décrivant un objet d'intérêt exigé dans un processus fonctionnel, de même que toutes les manipulations de données associées, seront identifiées et comptées comme étant des mouvements de données en Entrée (-type) ;

Sec.	DESCRIPTION DES PRINCIPES ET RÈGLES
	(REMARQUE: Un processus fonctionnel pourrait, en principe, manipuler de multiples Entrées (-type), chacune déplaçant un groupe de données décrivant un objet d'intérêt (-type) distinct). La même règle correspondante s'applique à n'importe quel mouvement de données de Lecture, d'éCriture ou de Sortie dans tous les processus fonctionnels. b) Plus d'un mouvement de données d'Entrée (type), chacun déplaçant, un groupe de données décrivant le même objet d'intérêt (type), dans un processus fonctionnel (type) donné, peuvent être identifiés et comptés s'il existe une FUR pour ces entrées multiples et qu'au moins une des conditions suivantes s'applique: • les Entrées diffèrent en déplaçant leurs groupes de données vers un autre utilisateur fonctionnel • les Entrées déplacent différents groupes de données, • les Entrées manipulent différentes données associées. La même règle correspondante s'applique pour les Sorties dans n'importe quel processus fonctionnel donné. Remarque : Tout processus fonctionnel ne peut avoir qu'une seule entrée de déclenchement. c) Plus d'un mouvement de données (type), chacun déplaçant un groupe de données décrivant un objet d'intérêt (type) dans un processus fonctionnel (type) donné, peuvent être identifiés et comptés s'il existe une FUR pour ces Lectures multiples et qu'au moins une des conditions suivantes s'applique: • les Lectures déplacent différents groupes de données, , • les Lectures manipulent différentes données associées. La même règle correspondante s'applique pour les éCritures dans n'importe quel processus fonctionnel donné. d) Plusieurs *occurrences* d'un même type de mouvement de données (i.e. déplaçant le même groupe de données avec la même manipulation de données) ne seront pas identifiées ou comptées plus d'une fois dans tout le processus fonctionnel. e) Dans le cas où des *occurrences* multiples d'un mouvement de données (associés à un processus fonctionnel donné), diffèreraient dans leur manipulation de données en raison des différentes valeurs des attributs du groupe de données, cela résulterait dans le suivi de différents chemins de traitement. Ce type de mouvement de données ne sera pas identifié et compté plus d'une fois dans ce processus.

3.5.9	**Lorsqu'un processus fonctionnel demande des données à un utilisateur fonctionnel.** **Règles** a) Un processus fonctionnel doit obtenir un groupe de données via un mouvement de données en Entrée provenant d'un utilisateur fonctionnel, et ce, dans le seul cas où le processus fonctionnel n'a pas besoin de préciser à cet utilisateur, de quelle donnée il s'agit. Voici certains de ces cas, soit : • Quand un utilisateur fonctionnel envoie un déclenchement en Entrée qui initie le processus fonctionnel ; • Quand un processus fonctionnel, ayant déjà commencé, attend une prochaine Entrée de l'utilisateur fonctionnel (typique dans les logiciels d'application d'affaires, où utilisateur fonctionnel est un humain) ; • Quand un processus fonctionnel, ayant déjà commencé, demande à l'utilisateur fonctionnel d'envoyer des données maintenant s'il en a : L'utilisateur fonctionnel envoie alors ses données (typiquement connues sous les termes 'd'invitation à émettre' ou de 'scrutation' dans les logiciels de type temps réel) ; • Quand un processus fonctionnel, ayant déjà commencé, inspecte l'état de l'utilisateur fonctionnel et récupère les données dont il a besoin. REMARQUE : Dans les deux derniers cas (en général, se produisant des le cas de logiciel de type temps réel), par convention, aucune Sortie du processus fonctionnel ne sera compté pour obtenir la donnée requise. Le processus fonctionnel enverra plutôt un message à l'utilisateur fonctionnel pour l'obtention de cette donnée et ce message sera considéré comme faisant partie de l'Entrée. Par contre, en aucun cas ce message sera comptabilisée comme étant une Entrée en soit : une seule Entrée sera comptabilisée dans ce cas, soit la donnée qui était attendue. b) Dans le cas où un processus fonctionnel requiert les services d'un utilisateur fonctionnel (par exemple, pour obtenir des données) et que cet utilisateur fonctionnel a besoin qu'on lui précise la donnée à envoyer (par exemple, quand cet utilisateur fonctionnel provient d'un autre morceau de logiciel ne faisant pas parti du périmètre du mesurage), une paire de mouvements de données en 'Sortie et Entrée' devra être compté. La Sortie contient la donnée de « demande de données » spécifiques et l'Entrée contient les données renvoyées.
3.5.10	**Commande de contrôle dans les applications avec une interface humaine** **Règle** Dans les applications avec une interface humaine, la 'commande de contrôle' doit être ignorée puisqu'elle n'implique pas de mouvement de données des objets d'intérêt.
3.5.11	**Messages d'erreur/confirmation** **Règles** a) Une Sortie doit être identifiée pour tenir compte de tous les types de messages d'erreur/confirmation émis par un processus fonctionnel du logiciel à mesurer, à partir de toutes les causes possibles selon sa FUR, par exemple les succès ou les échecs de validation des données saisies pour un appel pour récupérer des données ou pour créer des données persistantes ou pour répondre à un service demandé d'un autre morceau de logiciel. Remarque : Si la FUR du processus fonctionnel ne nécessite pas la présence d'un message d'erreur/confirmation, ne pas identifier une Sortie correspondante. b) Toutes les autres données, émises ou reçues par le logiciel à mesurer, vers ou à partir de son matériel ou du logiciel des utilisateurs fonctionnels, doivent être analysées selon la FUR comme Sorties ou Entrées respectivement, selon les

	règles normales de COSMIC, indépendamment du fait que les valeurs des données indiquent ou pas une condition d'erreur. c) On doit prendre en considération seulement les Lectures et éCritures des présentations associées aux conditions d'erreur. Donc aucune Entrée vers le processus fonctionnel à mesurer ne doit être identifiée pour une indication d'erreur reçue suite à une Lecture ou éCriture de données persistantes. d) *Aucune Entrée ou Sortie ne doit être identifiée pour les messages qui indiquent une condition d'erreur, qui pourrait apparaitre lors de l'utilisation du logiciel à mesurer, et qui n'a pas besoin d'être traité par la FUR de ce logiciel, par exemple un message d'erreur émis par le système d'exploitation.*
4.3.1	**Additivité des résultats du mesurage** **Règles** a) Pour chaque processus fonctionnel, la taille fonctionnelle des mouvements de données est regroupée en une seule valeur de taille fonctionnelle par l'intermédiaire d'une addition arithmétique du nombre de mouvements de données ; Taille $_{CFP}$ (processus fonctionnel) = Σ taille (Entrée i) + Σ taille (Sortie i) + Σ taille (Lecture i) + Σ taille (éCriture i) b) Pour chaque processus fonctionnel, la taille fonctionnelle de chaque modification de FUR est additionnée à la somme des tailles fonctionnelles des mouvements de données, pour lesquels les données ont réalisé une manipulation de donnée par l'entremise d'un processus fonctionnel. Voici la formule : Taille $_{CFP}$ (modification (processus fonctionnel $_i$)) = Σ taille (mouv. de donnée $_i$ --> manip. 'ajouté') + Σ taille (mouv. de donnée $_i$ --> manip. 'modifié') + Σ taille (mouv. de donnée $_i$ --> manip. 'supprimé') Pour en savoir plus sur l'additivité de la taille fonctionnelle, voir la section 4.3.2. Pour mesurer la taille d'un morceau de logiciel modifié, voir la section 4.4.2 c) La taille d'un morceau de logiciel pour un périmètre défini doit être obtenue en additionnant les tailles fonctionnelles des processus fonctionnels de ce morceau, selon les règles (e) et (f) ci-dessous ; d) La taille d'une modification d'un morceau logiciel d'un périmètre donné doit être obtenue en additionnant les tailles fonctionnelles de toutes les modifications de tous les processus fonctionnels de ce morceau, selon les règles (e) et (f) ci-dessous ; e) La taille de chaque morceau de logiciel à mesurer dans une couche peut être obtenue en les additionnant, mais seulement au même niveau de granularité des processus fonctionnels de leur FUR ; f) Les tailles des morceaux de logiciel et/ou des modifications apportées à un morceau de logiciel dans leur couche respective ou dans des couches différentes, ne devraient pas être additionnées entre elles, sauf si il est logique de le faire, pour les fins du mesurage ; g) La taille d'un morceau de logiciel ne peut pas être obtenue en additionnant les tailles de ses composants (quelques soit le mode de décomposition) à moins que la taille des mouvements de données entre ces composants soient éliminée du calcul; h) Si la méthode COSMIC est étendue localement (par exemple pour mesurer certains aspects de la taille non couverte par la méthode normalisée), alors la taille mesurée en utilisant une extension locale doit être rapportée séparément tel que décrit dans la section 5.1, et ne peut pas être additionné pour obtenir une taille selon l'étalon du mesurage COSMIC (voir la section 4.5).
4.4.1	**Modification d'un mouvement de données** **Règles**

	a) Si un mouvement de données devait être modifié suite à un changement dans une manipulation de données associée et/ou suite à un changement du nombre ou du type d'attributs de données déplacé par ce mouvement de données, un CFP (modification) devra être mesuré, quelque soit le nombre réel de modifications dans ce mouvement de données ; b) Si un groupe de données devait être modifié, les mouvements de données déplaçant ce groupe, dont la fonctionnalité n'est pas affectée par cette modification ne sera pas identifiés comme étant un mouvement de données modifié. REMARQUE 1 : Le changement de la valeur actuelle d'un attribut, telle que la modification d'un code d'un groupe de données dont les valeurs des attributs est représenté par un système de codes, n'est pas une modification de 'signification, type ou format' d'attribut. REMARQUE 2 : Une modification de toute donnée qui apparait sur des écrans d'entrée ou de sortie qui ne représentent pas d'objet d'intérêt pour un utilisateur fonctionnel ne devrait pas être identifiée comme un CFP de modification (Voir section 3.3.4 pour des exemples d'une telle donnée).
5.1	**Étiquetage du mesurage COSMIC** **Règles** Les résultats du mesurage COSMIC seront notés comme suit, « **x** PFC (v.**y**) », où : • "**x**": représente la valeur numérique de la taille fonctionnelle ; • "v.**y**": représente l'identification de la version de la norme de la méthode COSMIC utilisée pour obtenir la valeur numérique "**x**". Remarque: Si une méthode d'approximation locale a été utilisée pour obtenir la mesure, elle devra être notée ailleurs. Si la mesure a été effectuée à l'aide des conventions d'une version standard COSMIC, la convention étiquetage ci-dessus doit être utilisée, voir section 5.2. **Étiquetage des extensions locales COSMIC**
5.1	**Règle** Un résultat du mesurage COSMIC utilisant des extensions locales doit être noté de la façon suivante, « x PFC (v. y) + z Local PF », où: • "**x**": représente la valeur numérique obtenue en additionnant tous les résultats des mesures individuelles selon la norme de la méthode COSMIC, version v.y ; • "v.**y**": représente l'identification de la version de la norme de la méthode COSMIC utilisée pour obtenir la valeur numérique fonctionnelle "**x**" ; • "**z**": représente la valeur numérique obtenue en additionnant tous les résultats des mesures individuelles à partir des extensions locales de la méthode COSMIC.
5.2	**Rapportage du mesurage COSMIC** **Règle** En plus des résultats du mesurage tels qu'enregistrés dans la section 5.1, les caractéristiques de chaque mesure doivent être enregistrés, selon la raison d'être du mesurage et le niveau désiré de comparabilité aux autres mesures, par exemple pour des fins d'analyse comparative. a) Une identification du composant logiciel mesuré (nom, numéro de version ou numéro de configuration). b) Le nom des sources d'information utilisées pour identifier la FUR pour la mesure ; c) Le domaine d'applicabilité du logiciel mesuré. d) Une description de l'architecture du logiciel, les couches pour lesquelles les mesures sont réalisées, s'il y a lieu.

	e) Un énoncé sur l'objet du mesurage. f) Une description du périmètre du mesurage, et sa relation avec les autres périmètres du mesurages, s'il y a lieu. (Utiliser les catégories génériques du périmètre dans la section 2.2) g) Le modèle du mesurage utilisé (COSMIC ou local), avec le mode de traitement (en ligne ou par lots). h) Les utilisateurs fonctionnels du logiciel. i) Le niveau de granularité des artéfacts logiciels disponibles et le niveau de décomposition du logiciel. j) La phase du cycle de vie du projet lorsque la mesure a été faite (est-ce que la mesure est une estimation basée sur des exigences incomplètes, ou elle a été effectuée sur la base des fonctionnalités réellement livrées). k) La cible ou la marge d'erreur connue du mesurage. l) Des indications sur si la méthode de mesurage COSMIC standard a été employée, et/ou si une approximation locale de la méthode standard a été appliquée, et/ou si des extensions locales ont été employés (voir la section 4.5). Employer les conventions d'étiquetage des sections 5.1 ou 5.2. m) Une indication sur si l'on mesure une fonctionnalité développé ou déjà livrée (la fonctionnalité 'développée' est obtenue en créant un nouveau logiciel ; la fonctionnalité 'déjà livrée' inclut la fonctionnalité 'développée' en plus d'autres fonctionnalités comme par exemple toutes formes de réutilisation de logiciels existants ou d'utilisation de paramètres existants pour ajouter ou modifier la fonctionnalité, etc.). n) Une indication sur si la mesure provient d'une fonctionnalité nouvellement fournie ou si elle est le résultat d'une activité de 'maintenance' ou 'd'amélioration' (i.e. la somme des ajouts, modification et suppressions - voir 4.4). o) Le nombre de composants majeurs dont les tailles ont été additionnées ensemble pour obtenir la taille totale telle qu'enregistrée, s'il y a lieu. p) Le pourcentage de fonctionnalités implémentées par le logiciel réutilisé. q) Saisir, pour chaque périmètre de logiciel du le périmètre global du mesurage, une matrice du mesurage, telle qu'indiqué dans l'annexe A. r) Le nom du mesureur et de toutes les qualifications et certifications COSMIC; la date du mesurage.

ANNEXE E – PRINCIPAUX CHANGEMENTS ENTRE LA VERSION 3.0.1 ET LA VERSION 4.0

Cette annexe contient un résumé des principales modifications apportées à la méthode de mesure de taille fonctionnelle COSMIC de la version 3.0.1 à la présente version 4.0

Pour retracer l'évolution de la méthode, veuillez consulter le manuel de mesurage de chaque version de la méthode (2.2, 3.0 et 3.0.1).

Un 'MUB' est une un bulletin de mise à jour de la méthode publié entre les versions majeures du Manuel mesurage pour annoncer et expliquer les modifications.

V4.0 Réf.	Changement
-	Numérotation simplifiée pour les figures, séquencées par chapitre plutôt que par (sous) section (par exemple la 'Figure 3.5.8.3' est maintenant la 'Figure 3.8').
Chs.1 to 5	Un résumé a été ajouté pour chacun des chapitres 1 à 5.
1.1	Les sections 1.1.2, 1.1.3 et 1.1.4 sur 'les limitations de la méthode' ont été déplacées à la section 4.5. Le titre de la sous-section 1.1.1 a été supprimé, car son texte est maintenant l'ensemble de la section 1.1.
1.2	La définition de la FUR a été déplacée de la section 2.2 et est définie ultérieurement lorsqu'elle est utilisée avec la méthode COSMIC (MUB 11).
1.2.3	Une nouvelle section 1.2.3 sur les Exigences Non fonctionnelles (ENF) a été ajoutée (MUB 10).
1.3.1	Le principe g) du Modèle de contexte de logiciel COSMIC (MCL) a été simplifié (MUB 7) et par la suite a été supprimé pour être fusionné avec le principe a) du Modèle générique de logiciel (MGL), où il appartient plus logiquement.
1.5	Cette nouvelle section a été transmise vers la section 4.5, qui contient maintenant les trois 'limitations' qui ont été discutées dans les sous-sections 1.1.2, 1.1.3 et 1.1.4.
2.2	Les règles sur le périmètre sont maintenant des règles sur le 'périmètre du mesurage'
2.2.2	La définition de 'couche' a été révisée pour la rendre conforme aux concepts du SEI (MUB 9). La figure 2.4 a été ajoutée pour montrer comment les couches dépendent des vues architecturales du logiciel.
2.2.2	La définition et les principes de 'composant de même niveau' ont été supprimés car ils ne sont plus nécessaires dans la méthode COSMIC (MUB 9).
2.3.2	La remarque 1 sur la définition du stockage persistant a été reformulée pour plus de clarté (MUB 7) (cette définition était à la section 3.3.1 dans la version 3.0.1 du MM.
2.3.3	Une section sur le 'Diagrammes de contexte' a été ajoutée.
2.4.3	L'exemple très élaboré des niveaux de granularité du système 'Everest' a été réécrit pour plus de clarté.
3.1	Le principe "Appliquer le modèle générique de logiciel " a été supprimé car il est déjà mentionné dans la description de la MCL et de la MGL ainsi que dans la description du processus de mesurage.
3.2	Les définitions des concepts d'évènement déclencheur, d'utilisateur fonctionnel, d'Entrée de déclenchement et de processus fonctionnel ont été améliorées (MUB 11).

3.2	Les degrés possibles de relations (c'est-à-dire les cardinalités) entre les évènements, les utilisateurs fonctionnels, l'entrée de déclenchement et les processus fonctionnels sont plus précis, comme illustré à la figure 3.3, et beaucoup d'exemples sont fournis à l'annexe C (MUB 11).
3.2.2	Le processus d'identification des processus fonctionnels est maintenant décrit plus clairement.
3.2.2	Les règles pour déterminer un processus fonctionnel ont été améliorées et simplifiées (MUB 11).
3.2.3 b)	La discussion sur la manière d'analyser et de mesurer les applications d'affaires et en lot a été élargie et clarifiée, avec l'ajout d'une nouvelle figure (3.4) (MUB 11)
3.2.7	Une nouvelle section a été ajoutée en expliquant comment mesurer les processus fonctionnels qui partagent des fonctionnalités communes ou qui sont très similaires (MUB 11)
3.2.8	Une nouvelle section a été ajoutée en expliquant la nécessité de distinguer l'évènement déclencheur qui démarre un système logiciel de l'évènement déclencheur qui démarre un processus fonctionnel (MUB 11).
3.5	L'ancienne section 4.1, 'Identifier les mouvements de données' a été déplacée au chapitre 3, section 3.5.
3.5.1	La figure 3.5 (anciennement la figure 4.1.1) montre plus clairement les relations entre un processus fonctionnel et les quatre types de mouvements de données (E, S, L et C).
3.5.5	Le principe d) pour une éCriture a été modifié pour éviter la contradiction avec le principe c) (MUB 7).
3.5.6	Les instructions sur la manipulation des données associées à des mouvements de données sont en fait des règles et ont donc été dotées d'une boite de règles. De plus, le libellé est simplifié.
3.5.7	La règle b) qui concerne deux ou plusieurs mouvements de données de même type et déplaçant des données relatives à l'objet d'intérêt figurant dans le même processus fonctionnel, a été ventilé en deux règles b) et c) plus compréhensibles. Plusieurs exemples ont été ajoutés dans cette section.
3.5.8	Plusieurs modifications ont été apportées dans cette section. L'exemple 1 a été scindé en deux exemples (1 et 4). L'exemple 3 a été ajouté pour illustrer le cas du logiciel mesuré qui est autorisé (ou non) à accéder aux données requises sur un stockage persistant (MUB 7 v2). L'analyse des messages d'erreur est maintenant discutée pour chacun des exemples, se référant à la nouvelle sous section 3.5.11 qui régule les messages d'erreur/confirmation.
3.5.10	Le concept de 'commande de contrôle' a été généralisé à toutes les applications avec une interface humaine.
3.5.11	Une nouvelle section a été ajoutée avec une définition et des règles précises pour mesurer les messages d'erreur/confirmation et autres conditions d'erreur.
4.4.1	La remarque1 des règles de modification d'un mouvement de données a été supprimée, car elle est incompatible : un changement d'une valeur de code n'est pas une modification d'un type d'attribut (c.-à-d. il ne doit pas être compté). Mais un changement nécessaire à un format ou un format d'entête n'est également pas nécessairement un changement d'un type d'attribut, mais il doit être compté.
4.4.1	La définition de changement d'un mouvement de données, et donc au groupe de données ou associées aux manipulations de données, a été transformée en une définition de 'Modification de la fonctionnalité d'un mouvement de données'.

4.5	Les sous sections 1.1.2, 1.1.3 et 1.1.4 sur les limites de la méthode COSMIC ont été déplacées pour devenir les sous sections 4.5.2, 4.5.3 et 4.5.4 respectivement. La précédente section 1.1.2 (maintenant 4.5.2) 'Domaine non applicable' a été retitrée comme 'Logiciel riche en manipulation de données' et le premier paragraphe de cette sous section a été remplacé, puisque la pratique a montré que la méthode COSMIC peut être utilisée pour mesurer la taille de plusieurs types de logiciels répertoriés dans cette déclaration de 'limitations' (MUB 8).
Références	Toutes les références sont maintenant dans cette section.
Annexe A	Le diagramme a été mis à jour pour afficher les totaux.
Annexe B	Cette annexe a été ajoutée avec des exemples d'Exigences non fonctionnelles qui évoluent en partie ou en totalité vers des FURs (MUB 10)
Annexe C	L'annexe a été ajoutée avec des exemples et des diagrammes sur la cardinalité des relations entre l'évènement déclencheur, les utilisateurs fonctionnels, l'Entrée de déclenchement et les processus fonctionnels.
Annexe D	Le sommaire des principes et des règles a été mis à jour.
Annexe F	Le Glossaire des termes a été déplacé du document 'Documentation Overview and Glossary of Terms' V3.0 vers cette annexe de la version 4.0 du MM et aussi mis à jour.

ANNEXE F - GLOSSAIRE

Les termes suivants sont utilisés tout au long de la méthode de mesurage de la taille fonctionnelle COSMIC (la 'méthode COSMIC'), selon les définitions figurant dans ce chapitre. Les termes déjà définis par l'ISO, comme "Mesure de la taille fonctionnelle" ou "unité de mesurage", ainsi que leur définition de l'ISO, ont également été adoptées pour la méthode COSMIC.

Pour plusieurs termes figurant dans le Glossaire, lorsque cela est approprié, le suffixe 'type' s'affiche. Étant donné que toute méthode de mesurage de la taille fonctionnelle vise à identifier les 'types' et non les 'occurrences, des données ou des fonctions, presque invariablement, tout au long de la méthode COSMIC nous serons intéressés aux 'types' et non aux 'occurrences'. Par conséquent, dans les textes nous allons laissé tomber le suffixe 'type' des termes dans un souci de lisibilité, sauf lorsque nous avons spécifiquement besoin de distinguer le type et l'occurrence. C'est aussi la convention adoptée par la norme internationale (ISO/IEC 19761: 2003) de la méthode COSMIC. Parfois, cette convention conduit à des difficultés lors de l'élaboration de ces définitions – voir par exemple la remarque 3 de la définition de 'type de mouvement de données' ci-dessous, qui n'apparait pas dans la norme internationale.

Remarque: Les termes qui sont utilisés uniquement dans des domaines spécifiques aux guides COSMIC sont définis dans ces guides; ils ne figurent pas ci-dessous.

Dans les définitions présentées ci-dessus:

- les termes qui sont définis ailleurs dans ce Glossaire sont soulignés pour faciliter les renvois.
- les termes qui origines de la norme ISO de la méthode COSMIC (ISO/IEC 19761) ou qui sont d'une certaine façon spécifiques à la méthode COSMIC sont indiqués en ***gras italique***
- les autres termes qui ont été adoptés par ISO mais ne sont pas spécifiques àa la méthode COSMIC sont montrés en **gras**.

Application. Un système de logiciel pour la collecte, l'enregistrement, traitement et présentation de données au moyen d'un ordinateur.

Remarque : C'est une adaptation de la définition fournie par 'ISO/IEC 24570:2005 Software engineering -- NESMA functional size measurement method version 2.1'.

(Définition alternative pour 'application logicille'). Logiciel conçu pour aider les utilisateurs à effectuer des tâches particulières ou pour traiter certains types de problèmes, par opposition à un logiciel qui contrôle l'ordinateur lui-même.

Remarque: Il s'agit d'une légère adaptation de la définition fournie par 'ISO/IEC 24765:2010 Systems and software engineering-vocabulary, 4.5').

Application- données générales. Toutes les données relatives à une application en général et qui ne sont pas liés à un objet d'intérêt d'un processus fonctionnel spécifique.

Composant fonctionnel de base (CFB). Unité élémentaire des Fonctionnalités Utilisateurs Requises (FUR), définie par la méthode de mesure de la taille fonctionnelle (MTF) pour les fins du mesurage [1].

Remarque: La méthode COSMIC définit un type de mouvement de données comme un CFB.

Composant fonctionnel de base (ou type de CFB)). Une catégorie définie de CFB [1]. La méthode COSMIC a 4 types de CFB, Entrée, Sortie, Lecture et éCriture (types).

***Frontière*.** Une interface conceptuelle entre le logiciel à mesurer et ses utilisateurs.

Remarque: Il découle de la définition qu'il y a une limite entre n'importe lequel des deux morceaux de logiciel d'une même couche ou de différentes couches qui échangent des données, lorsqu'un morceau de logiciel est un utilisateur fonctionnel de l'autre et/ou vice versa.

Composant. Toute partie d'un système logiciel qui est séparé pour des raisons d'architecture du logiciel, et/ou qui a été spécifié, conçu ou mis au point séparément.

Commande de contrôle. Une commande qui permet à un utilisateur fonctionnel de contrôler l'utilisation du logiciel mais qui n'implique pas de mouvement de données à propos d'un objet d'intérêt.

Remarque: Une commande de contrôle n'est pas un mouvement de données parce que la commande ne déplace pas des données d'un objet d'intérêt. Exemples: commande de 'page haut/bas', peser sur Tab ou la clef Enter (Entrée), cliquer sur 'OK' pour confirmer une action précédente, etc.

Unité de mesurage COSMIC. 1 PFC (Point de Fonction COSMIC), définit comme la taille d'un mouvement de données

Attribut de données (synonyme d'élément de données type) La plus petite parcelle d'information codée, dans un groupe de données, possédant une signification dans la perspective des Fonctionnalités Utilisateurs Requises (FUR) du logiciel

Groupe de données (type). Tout ensemble distinct, non vide, non ordonné et non redondant de types d'attributs de données où chaque type d'attribut de donnée décrit un aspect complémentaire du même objet d'intérêt.

Manipulation des données. Tout ce qui survient aux données autres qu'un mouvement de données vers l'intérieur ou l'extérieur d'un processus fonctionnel, ou entre un processus fonctionnel et un stockage persistant.

Mouvement de données (type). Un composant fonctionnel de base qui déplace un seul type de groupe de données.

Remarque 1: Il y a quatre sous-types de mouvements de données: entrée (E), sortie (S), lecture (L), écriture (C).

Remarque 2 : Pour le besoin de la mesure, chaque sous-type de mouvement de données est considéré comme incluant un certain nombre de manipulation de données qui y sont associées – voir le Manuel de Mesure pour plus de détails.

Remarque 3 : Pour préciser, c'est l'occurrence d'un mouvement de données, non son type, qui en pratique déplace les *occurrences* d'un groupe de données. Cette remarque s'applique aussi aux définitions de: Entrée (E), Sortie (S), Lecture (L), éCriture (C).

E. Abréviation pour 'Entrée' (type).

Entrée (type). Un type de mouvement de données qui déplace, à travers la frontière, un groupe de données depuis un utilisateur fonctionnel vers le processus fonctionnel où il est requis.

Remarque 1 : Un type d'Entrée inclut aussi certaines manipulations de données associées (voir le Chapitre 3 pour plus de détails).

Message erreur/confirmation. C'est une Sortie émise par un processus fonctionnel à l'attention d'un utilisateur humain qui soit, confirme que les données saisies ont été acceptées, ou confirme qu'il y a une erreur dans les données saisies.

Remarque : Si un message à un utilisateur humain fournit des données autres que la confirmation que des entrées données ont été acceptées, ou que les données saisies sont erronées, alors ces autres données doivent être déterminées comme un groupe de données proposé par une Sortie et non comme un message d'erreur/confirmation.

Type d'évènement. Quelque chose qui arrive.

Sortie (type). Un type de mouvement de données qui déplace, à travers la frontière, un groupe de données d'un processus fonctionnel vers l'utilisateur fonctionnel qui le demande.

Remarque : Une sortie type inclut aussi les manipulations de données associées (voir le Chapitre 3 pour plus de détails).

Processus fonctionnel (type)

a) Un ensemble de mouvements de données représentant une partie élémentaire des FURs pour le logiciel à mesurer, qui est unique à cette FUR et qui peut être défini indépendamment de tout autre processus fonctionnel dans cette FUR.

d) Un processus fonctionnel a seulement un évènement déclencheur. Chaque processus fonctionnel démarre le traitement dès réception d'un groupe de données déplacé par l'Entrée de déclenchement du processus fonctionnel.

e) L'ensemble de tous les mouvements de données d'un processus fonctionnel est l'ensemble qui est nécessaire pour satisfaire sa FUR pour toutes les réponses possibles par l'entrée de déclenchement.

f) Remarque 1: Lors de l'implémentation, c'est une occurrence d'un processus fonctionnel qui déclenche l'exécution, et ce à la réception d'une occurrence d'un groupe de données déplacé par une occurrence de l'Entrée de déclenchement.

g) Remarque 2: La FUR pour un processus fonctionnel peut exiger une ou plusieurs autres entrées en plus de l'Entrée de déclenchement.

h) Remarque 3: Si un utilisateur fonctionnel envoie un groupe de données avec des erreurs, par exemple parce qu'un capteur-usager a un mauvais fonctionnement ou une commande d'entrée par un être humain comporte des erreurs, c'est habituellement la tâche du processus fonctionnel de déterminer si l'évènement a vraiment eu lieu et/ou les données entrées sont vraiment valables. Le processus fonctionnel indique aussi comment réagir.

***Niveau de granularité du processus fonctionnel*.** Un niveau de granularité de la description d'un morceau de logiciel pour lequel :

- Les utilisateurs fonctionnels sont des individus humains ou des objets d'ingénierie ou d'autres morceaux de logiciel (et non pas des groupes de ceux-ci) ET ;
- Détectent des occurrences uniques d'évènements auxquels un morceau du logiciel doit répondre (et pas au niveau où les groupes d'évènements sont définis).

REMARQUE 1: En pratique, la documentation du logiciel contenant les FURs décrit souvent les fonctionnalités à divers niveaux de granularité, plus particulièrement quand la documentation est toujours en cours d'élaboration.

REMARQUE 2: Les 'Groupes ciblés (utilisateurs fonctionnels) peuvent être, par exemple, est un 'département' dont les membres peuvent effectuer plusieurs types de processus fonctionnels, ou un 'panneau de contrôle' qui a plusieurs types d'instruments, ou des 'systèmes centraux'.

REMARQUE 3: Les 'Groupes d'évènements' peuvent, par exemple, être exprimés dans l'énoncé d'une FUR à un haut niveau de granularité par un flux d'intrants pour un système comptable nommé 'transactions de vente' ou par un flux d'intrants à un logiciel avionique appelée 'commande de pilotage'.

Taille fonctionnelle. Une taille du logiciel dérivé en quantifiant les besoins fonctionnels des utilisateurs.[1]

Mesurage de la taille fonctionnelle (MTF). Le processus de mesurage de la taille fonctionnelle. [1]

Méthode de mesurage de la taille fonctionnelle (MMTF). Une implémentation spécifique de la MTF définie par un ensemble de règles, qui sont conformes aux caractéristiques obligatoires de la norme ISO/IEC 14143-1:1998. [1]

***Utilisateur fonctionnel*.** Un sous-ensemble des besoins de l'utilisateur. Les besoins qui décrivent ce que le logiciel doit accomplir, en termes de tâches et service.

Remarque : La FUR inclut (mais n'est pas limitée au) :

- Transfert de données (exemple: données d'entrée du client, envoyer un signal de contrôle);
- Transformation de données (exemple: calculer l'intérêt bancaire, déterminer la température moyenne);
- Stockage de données (exemple: emmagasiner la commande de l'usager, enregistrer la température ambiante sur une période);
- Extraction des données (exemple : liste (actuelle) des employés actuels, récupérer la position de l'avion).

Exemples de Requis Utilisateur qui ne sont pas des Fonctionnalités Utilisateur Requises qui incluent, mais ne sont pas limités à des:

- contraintes de qualité (exemple: utilisabilité, fiabilité, efficience et portabilité);
- contraintes organisationnelles (exemple: locations pour opération, matériel cible et conformité aux normes);
- contraintes environnementales (exemple: inter-opérabilité, sécurité, protection de la vie privée et sécurité);

contraintes d'implémentation (exemple: langage de développement, cédule de livraison).

Input (entrées). Les données pour lesquelles la valeur est indépendante du logiciel et qui sont entrées par un utilisateur et qui sont utilisées par le logiciel lors de son opération. La définition du terme 'entrée' utilisée dans ce manuel est différente de la définition utilisée par l'International Function Point Users Group (IFPUG) pour le terme " input ". Pour COSMIC, le terme 'input' comprend toutes les entrées qui participent dans un processus fonctionnel particulier.

***Couche*.** Un partitionnement résultant d'une division fonctionnelle d'une architecture de logiciel.

***Niveau de décomposition*.** N'importe quel niveau résultant de la Division un morceau de logiciel en composants (nommé 'niveau 1', par exemple), puis diviser les composants en sous-composantes ('niveau 2'), et ensuite diviser les sous-composants en sous-sous composants ('niveau 3'), etc..

Remarque 1: Ne pas confondre avec 'niveau de granularité'.

Remarque 2: Le mesurage de la taille d'un morceau du logiciel peut être comparé directement seulement aux composants d'un même niveau de décomposition.

***Niveau de granularité*.** Tout niveau d'expansion de la description d'un seul morceau de logiciel (c.-à-d. un énoncé d'une exigence, ou la description d'une structure d'un morceau de logiciel) tel que chaque niveau supérieur d'expansion, la description de la fonctionnalité du morceau du logiciel est à un niveau supérieur et uniforme de détails.

REMARQUE: Les mesureurs doivent tenir compte de l'évolution précoce des exigences dans le cycle de vie d'un projet. À tout moment, différentes parties des fonctionnalités requises du logiciel seront typiquement documentées à différents niveaux de granularité..

Méthode de mesurage [14]. Une suite logique d'opérations, décrite de manière générique, utilisée pour l'exécution du mesurage.

***Modèle de stratégie du mesurage*.** Un modèle normalisé qui peut-être être appliqué lors du mesurage d'un morceau de logiciel provenant d'un domaine fonctionnel du logiciel donné, qui définit les types d'utilisateurs fonctionnels et qui peut interagir avec le logiciel, le niveau de décomposition du logiciel et les types de mouvements de données que le logiciel peut manipuler.

Modèle [16]. Une description ou une analogie utilisée pour aider à visualiser un concept qui ne peut être observé directement.

Modification (de la fonctionnalité d'un mouvement de données)

a) Un mouvement de données est considéré fonctionnellement modifié si au moins une des conditions suivantes s'applique:
   - le groupe de données déplacé est modifié,
   - la manipulation de données associée est modifiée.

b) Un groupe de données est modifié, si au moins une des conditions suivantes s'applique:
   - un ou plusieurs nouveaux attributs sont ajoutés au groupe de données,
   - un ou plusieurs attributs existants sont enlevés au groupe de données,
   - un ou plusieurs attributs existants sont modifiés, par exemple la présentation ou le format (le changements des valeurs ne compte pas)

c) une manipulation de données est modifiée, si elle est, de quelque façon, fonctionnellement changée.

Exigences non fonctionnelles. Toute exigence ou contrainte sur un système (équipement/logiciel) ou sur un produit logiciel, ou sur un projet pour développer ou maintenir ce système ou produit, sauf une exigence fonctionnelle (FUR) pour un logiciel.

Remarque : Les exigences système ou logiciel, initialement exprimés comme non fonctionnelles, peuvent évoluer, en tout ou en partie, au fur et à mesure que le projet progresse, en FUR pour le logiciel.

Object d'intérêt (type). Toute 'chose' qui est identifiée du point de vue des Fonctionnalités Utilisateurs Requises (FUR). Ce peut être une chose physique, aussi bien qu'un objet conceptuel ou partie d'un objet conceptuel dans le monde de l'utilisateur fonctionnel pour lequel un logiciel est requis pour un traitement et/ou un stockage de données.

Remarque : Dans la méthode COSMIC, le terme 'objet d'intérêt' est utilisé pour éviter d'utiliser des termes reliés à des méthodologies spécifiques du génie logiciel. Le terme ne correspond pas nécessairement au terme 'objet' selon le sens utilisé dans les méthodes orientées objets.

Environnement opérationnel (logiciel). L'ensemble des logiciels fonctionnant simultanément sur un ordinateur spécifié.

Output. Les données dont la valeur dépend de l'opération du logiciel et qui sont ainsi créées ou encore modifiées par le logiciel pendant son opération avant d'être déplacées vers l'utilisateur. La définition générique de l'expression « données de sortie » utilisée dans ce manuel est différente de la définition utilisée par l'International Function Point Users Group (IFPUG) pour le terme « OUTPUT ». Pour COSMIC, le terme « output » comprend toutes les Sorties qui participent dans un type particulier de processus fonctionnel.

Morceaux de logiciel de même niveau. Deux morceaux d'un logiciel sont de même niveau si chacun réside dans la même couche.

Stockage persistent. Un type de stockage qui permet à un processus fonctionnel de conserver des données au-delà de la vie du processus fonctionnel et/ou qui permet à un processus fonctionnel de retrouver une donnée stockée par un autre processus fonctionnel, ou stockée plus tôt au cours d'une occurrence du même processus fonctionnel ou stockée par un autre processus.

Remarque 1 : Dans le modèle COSMIC, puisque la partie de stockage persistant est du côté logiciel de la frontière, elle n'est pas considérée comme un utilisateur du logiciel à mesurer.

Remarque 2 : Un exemple d'un autre type de stockage persistant est la fabrication de mémoires à lecture seulement.

Morceau de logiciel. N'importe quel élément discret du logiciel à n'importe quel niveau de décomposition à partir d'un système logiciel entier jusqu'à un élément se trouvant au niveau du plus petit composant d'un système logiciel.

Raison d'être du mesurage. Un énoncé qui indique pourquoi une mesure est exigée, et comment le résultat sera employé..

L. Abréviation pour Lecture (type).

Lecture (type). Un type de mouvement de données qui, dans le contexte de son processus fonctionnel, déplace un groupe de données depuis sa partie de stockage persistant pour le mettre à la portée du processus fonctionnel (type) qui le requiert.

Remarque : Une Lecture inclut certaines manipulations associées de données nécessaires - voir le Manuel de mesurage pour plus de détails.

Mise à l'échelle (du mesurage). Le processus de conversion d'un mesurage de la taille d'une unité de mesure à une mesure d'une autre unité de mesure.

Périmètre (du mesurage). C'est l'ensemble des Fonctionnalités Utilisateur Requises (FUR) qui doivent être incluses dans une occurrence spécifique du mesurage de taille fonctionnelle [1].

REMARQUE : (spécifique à la méthode COSMIC) une distinction doit être faite entre le 'périmètre global', i.e. tout le logiciel qui doit être mesuré selon la raison d'être du mesurage, et le 'périmètre, de chaque morceau de logiciel à l'intérieur de cette raison d'être globale, et pour lequel la taille doit être mesurée séparément. Dans le Manuel de mesurage, le terme 'périmètre' (ou l'expression 'périmètre du mesurage') sera relié à tout morceau de logiciel dont la taille doit être mesurée séparément.

Logiciel [17]. Un ensemble d'instructions informatiques, de données, de procédures et éventuellement de documentations opérées comme un tout, pour répondre à un ensemble de buts spécifiques, lesquels peuvent être décrits dans une perspective fonctionnelle via un ensemble fini de Fonctionnalités Utilisateurs Requises (FUR) ainsi que d'exigences techniques et qualitatives.

Système logiciel*. Un système composé uniquement de logiciels*.

Sous-processus (type). Une partie d'un processus fonctionnel qui soit, déplace des données (dans le logiciel d'un utilisateur fonctionnel ou hors le logiciel pour un utilisateur fonctionnel, ou partir du stockage persistant) ou manipule des données.

Système*. Une combinaison de matériel, de logiciels et de procédures manuelles organisé pour répondre à des fins établies à l'avance.*

Remarque: La définition ci-dessus est une adaptation de la définition de la norme ISO/IEC 15288:2008. Dans la définition COSMIC, 'matériel, logiciel et manuel de procédures' remplacent 'éléments qui interagissent' dans la définition de la norme ISO/IEC.

Entrée de déclenchement (type). Le mouvement de donnée 'Entrée' d'un processus fonctionnel qui déplace un groupe de données, généré par un utilisateur fonctionnel, nécessaire au processus fonctionnel pour son démarrage.

Remarque: La FUR d'un processus fonctionnel peut exiger une ou plusieurs autres entrées en plus de l'Entrée de déclenchement.

Remarque: L'horloge et les évènements de calendrier peuvent être de évènements déclencheurs.

Unité de mesurage [14]. Grandeur particulière, définie et adoptée par convention, à laquelle on compare les autres grandeurs de même nature pour les exprimer quantitativement par rapport à cette grandeur. Les unités de mesurage ont par convention des noms et des symboles correspondants.

Voir aussi 'Unité de mesurage COSMIC'

Utilisateur [1]. N'importe quelle personne ou chose qui communique or interagit avec le logiciel à n'importe quel moment.

Remarque : les exemples de 'chose' incluent, mais ne sont pas limités à, des applications logicielles, des animaux, des senseurs or d'autre matériel.

Valeur (d'une grandeur) [14]. Expression quantitative d'une grandeur particulière, généralement sous la forme d'un étalon de mesure multiplié par un nombre.

C. Abréviation pour 'éCriture - type'.

ÉCriture (type). Un type de mouvement de données qui déplace un groupe de données depuis un processus fonctionnel vers un stockage persistant.

Remarque : Un type d'écriture inclut certaines manipulations associées (voir le Chapitre 3 pour plus de détails).

S. Abréviation pour Sortie (type).

ANNEXE G - PROCÉDURE DE DEMANDE DE MODIFICATION ET COMMENTAIRE

Le Comité pratiques des mesures (CPM) COSMIC souhaite recevoir vos commentaires, observations et, le cas échéant, les demandes de modifications de ce Manuel de mesurage. Cette annexe indique comment communiquer avec le CPM de COSMIC. Toutes les communications à le CPM COSMIC doivent être envoyées par courriel à l'adresse suivante : mpc-chair@cosmicon.com

Commentaires informels généraux et rétroaction

les commentaires et/ou rétroactions concernant de Manuel de mesurage, tels que des difficultés de compréhension ou d'application de la méthode COSMIC, suggestions d'améliorations générales, etc. sont à adresser par courriel à l'adresse ci-dessus. Les messages seront enregistrées et un avis de réception/ou réponse sera transmis dans les deux semaines suivant la réception.

Demandes formelles de changements

Lorsqu'un lecteur de la documentation COSMIC détecte une erreur dans le texte ou que le texte semble avoir besoin de clarifications ou encore d'amélioration, une demande formelle de changement peut être soumise.

Les demandes formelles de changement sont conservées et un accusé de réception est envoyé au demandeur dans un délai de deux semaines suivant la réception du message. Un numéro séquentiel est alloué à chaque demande de changement et la demande est communiquée à un groupe d'experts répartis à travers le monde, soit aux membres du comité des pratiques du mesurage COSMIC. Le cycle de révision normal peut prendre un mois au minimum, mais ce délai peut être plus long lorsque le comité fait face à des demandes de changement qui présentent des défis important quant à la solution à apporter.

Le résultat ou la solution trouvée suite au processus de révision peut être de nature diverse. Le comité peut décider d'accepter, de refuser ou mettre en suspend la décision finale pour en discuter davantage (par exemple, dans le cas où il y aurait une dépendance par rapport à une autre demande de changement). Le résultat est communiqué dès que possible au demandeur.

Une demande formelle de changement n'est acceptée que si elle est accompagnée des informations suivantes :

- Nom, titre et organisation de la personne soumettant la demande de changement ;
- Les coordonnées nécessaires pour communiquer avec la personne soumettant la demande de changement ;
- Date de la soumission de la demande de changement ;
- Déclaration générale de la raison d'être de la demande de changement (i.e. « nécessité d'améliorer le texte … »);
- Le texte actuel qui a besoin d'être changé, remplacé ou enlevé (ou une référence à ce texte) :
- Le texte de remplacement ou le texte additionnel proposé :
- Une description détaillée expliquant pourquoi le changement est nécessaire ;

Il est nécessaire de saisir le formulaire de « demande de changement » pour soumettre une demande. Celui-ci est disponible à cette adresse : www.cosmicon.com

La décision du comité des pratiques du mesurage COSMIC suivant le processus de révision de la demande, de même que la version du texte COSMIC à laquelle ce changement s'applique, sont définitives.

Questions sur l'application de la méthode COSMIC

Le comité des pratiques du mesurage COSMIC regrette de ne pouvoir répondre directement aux questions portant l'utilisation ou l'application de la méthode de mesurage COSMIC. Plusieurs organisations de nature commerciales œuvre dans la formation et le support (certaines proposent parfois des outils) de la méthode COSMIC. Il est possible d'obtenir plus de détails sur ces organisations à cette adresse : www.cosmicon.com.

Commentaires pour la version française : contacter Jean-Marc Desharnais à l'adresse courriel suivante : desharnaisjm@gmail.com.