Manuel COSMIC
ISO 19761

Partie 3 :
Exemples

Version 5.0

6 septembre 2020

Préface

Le manuel COSMIC ISO /IEC 19761 :2011 est composé de trois parties :

Partie 1 : Principes, définitions et règles.

Partie 2 : Recommandations du groupe COSMIC.

Partie 3 : Exemples des concepts COSMIC et de mesurages.

La troisième partie du manuel COSMIC présente des exemples de mesurages et de concepts de la méthode de mesure fonctionnelle (la « méthode COSMIC »). Les exemples sont organisés en fonction des phases de mesurage. Ils comprennent une introduction et une description de l'objectif en italiques.

Une version libre de droit du manuel COSMIC et des rapports techniques, y compris des traductions, sont disponibles dans la base de connaissance COSMIC www.cosmic-sizing.org.

Éditeurs		
Alain Abran École de technologie supérieure – U. Québec, Canada	Peter Fagg Pentad Royaume-Uni	Arlan Lesterhuis Pays-bas

Membres du comité des pratiques de mesure COSMIC		
Diana Baklizky TI Metricas Brésil	Jean-Marc Desharnais École de Technologie Supérieure – U. Québec, Canada	Cigdem Gencel Université libre Bozen-Bolzano, Italie
Dylan Ren Measures Technology LLC Chine	Bruce Reynolds Tecolote Research États-Unis	Hassan Soubra, Université allemande du Caire, Égypte
Sylvie Trudel UQAM, Canada	Francisco Valdés Souto Spingere Mexique	Frank Vogelezang Metri Pays-Bas

Traduction en français		
Alain Abran École de technologie supérieure – U. Québec, Canada	Jean-Marc Desharnais École de Technologie Supérieure – U. Québec, Canada	Anna Chapelle Safae Laqrichi Patrick Hamon Estimancy, France

Contrôle de version

La Table suivante récapitule les modifications apportées à cette 'ligne directrice'.

DATE	Réviseur(s)	Modifications / Additions
Avril	Estimancy	Création Fr – Part 3 – MM Examples

Table des matières

1 INTRODUCTION.

1.1 Exigences Fonctionnelles de l'Utilisateur (EFU).

De nombreux exemples d'Exigences Fonctionnelles de l'Utilisateur (EFU) sont documentés dans les Études de cas COSMIC (disponibles gratuitement sur cosmic-sizing.org).

1.2 Exigences Non Fonctionnelles (ENF).

La méthode COSMIC est uniquement utilisée pour mesurer la taille des EFU. Les exigences incluent souvent des Exigences Non Fonctionnelles qui ne peuvent pas être mesurées à l'aide de la méthode COSMIC. Cependant, des exigences systèmes qui sont, au départ, non fonctionnelles peuvent se transformer en EFU au fur et à mesure de l'évolution d'un projet. Il est donc primordial de distinguer les deux types d'exigences et de suivre leur évolution.

Exemple métier : Les exigences d'un nouveau système logiciel expliquent : « L'utilisateur doit pouvoir choisir de sécuriser les fichiers par cryptage ». Le projet de développement en est au stade de l'estimation de l'effort et des coûts. Il y a alors deux options possibles :

- Développer un logiciel de cryptage propriétaire. Il peut être nécessaire de mesurer la taille des EFU du logiciel de cryptage pour pouvoir estimer le projet.
- Acheter un pack COTS. Il peut être nécessaire de mesurer uniquement la taille des fonctionnalités indispensables à l'intégration du pack COTS pour pouvoir estimer le projet. Le coût, ainsi que les efforts d'intégration et de test du pack de cryptage de fichiers doivent également être pris en compte dans l'estimation du coût global du projet.

Exemple temps réel : Les exigences de fiabilité ou de tolérance aux erreurs des systèmes aérospatiaux sont principalement obtenues par une combinaison de redondance et de sauvegarde des systèmes physiques. Une fonctionnalité, telle que la surveillance d'un moteur, est exécutée sur au moins deux différents ordinateurs intégrés. Cette fonctionnalité présente une contrainte de temps stricte selon une ENF : « Chaque ordinateur doit répondre dans un temps imparti. » Si, à plusieurs reprises, un des ordinateurs ne répond pas dans le temps imparti, ou si ses résultats ne sont pas cohérents, il doit être écarté » (par un mécanisme décrit par une exigence fonctionnelle). Une exigence concernant la tolérance aux erreurs qui était non fonctionnelle évolue en EFU pouvant être mesurée. Le mécanisme de chronométrage peut également être partiellement intégrée à un logiciel. Cette fonctionnalité peut aussi être mesurée (pour voir un exemple, consulter la section 3.2 Recommandations pour le calcul de la taille de logiciels temps réel).

1.3 Le modèle contextuel COSMIC du logiciel.

Voir les Études de cas sur cosmic-sizing.org.

1.4 Le modèle générique COSMIC du logiciel.

Voir les Études de cas sur cosmic-sizing.org.

1.5 Types et occurrences.

La méthode COSMIC ne prend en compte que les types d'éléments, et non le nombre d'occurrences du même type. Comme cela peut fortement influencer la taille, il est primordial d'appliquer ce précepte à tous les éléments, tels que les utilisateurs fonctionnels, les objets d'intérêts, les groupes de données, etc., qui seront utilisés lors du mesurage.

Exemples de modèle contextuel COSMIC du logiciel

Exemple métier 1 : Prenons le système sur lequel repose un centre d'appel de 100 employés répondant aux questions des clients. Les employés font tous l'objet d'une même exigence (« répondre aux questions des clients »). Un modèle contextuel du système comprend un seul type d'utilisateur fonctionnel : « Employé du centre d'appel », qui a 100 occurrences.

Exemple temps réel 1 : Le logiciel intégré d'une radio envoie sa sortie vers une paire d'enceintes. Le logiciel envoie des signaux séparés, mais du même type, vers les deux enceintes. Elles convertissent toutes deux de la même manière le signal électrique reçu en son. Le modèle contextuel du logiciel présente un seul type d'utilisateur fonctionnel, l'« enceinte », qui a deux occurrences.

Exemples de modèles génériques COSMIC.

Exemple métier 2 : Prenons un processus fonctionnel de saisie et de validation des données concernant un nouveau client. Le mouvement de données Entrée sera exécuté, autrement dit, il y aura une occurrence à chaque fois qu'un utilisateur fonctionnel humain enregistre les données d'un nouveau client. Lors de son exécution, le processus fonctionnel doit valider les données saisies en cherchant à vérifier si le client n'existe pas déjà dans la base de données, d'où la Lecture pour la validation qui aura au moins une occurrence (en fonction de la conception de la base de données). Cependant, on compte une Entrée et une Lecture du client lors du mesurage de ce processus fonctionnel.

Exemple temps réel 2 : Prenons un processus fonctionnel devant contrôler la température d'un four toutes les dix secondes. Le processus fonctionnel sera exécuté, c'est-à-dire, qu'il aura une occurrence, toutes les dix secondes. Lors de cette exécution, la Sortie du processus permettant d'allumer ou d'éteindre le système de chauffe du four peut, ou non, être exécuté. Autrement dit, elle peut se produire ou non lors d'un cycle, en fonction de la nécessité d'allumer ou d'éteindre le système de chauffe, ou de le laisser dans son état actuel. La Sortie est comptée une seule fois au cours du processus fonctionnel, peu importe s'elle se produit, ou non, lors d'une exécution spécifique.

1.6 Le processus de mesurage COSMIC.

Voir les Études de cas sur cosmic-sizing.org.

2 LA PHASE DE STRATÉGIE DU MESURAGE.

2.1 Raison d'être du mesurage.

Un mesurage est effectué pour satisfaire les besoins d'informations d'une partie prenante. La personne en charge du mesurage doit en comprendre les raisons.

EXEMPLES : Les exemples suivants sont des raisons d'être classiques :

- Mesurer la taille des EFU au cours de leur évolution dans le cadre du processus d'estimation de l'effort de développement.
- Mesurer la taille des changements apportés aux EFU après validation, afin de gérer la « variation de périmètre » due à l'ajout ou à la modification d'exigences.
- Mesurer la taille du logiciel livré dans le cadre de l'évaluation de la productivité de l'organisation de développement.
- Mesurer la taille globale du logiciel livré, ainsi que la taille du logiciel développé, pour obtenir une mesure de la réutilisation fonctionnelle.

- Mesurer la taille du logiciel existant dans le cadre de l'évaluation de la performance du groupe responsable de la maintenance et de l'assistance.
- Mesurer la taille des modifications apportées à un système logiciel existant en tant que la taille du travail fourni par une équipe de projet d'évolution.
- Mesurer la taille d'une sous-partie de l'ensemble des fonctionnalités d'un logiciel devant être développé et qui sera livrée aux utilisateurs fonctionnels du logiciel en question.

2.2 Le périmètre du mesurage.

Le périmètre de mesurage peut inclure tout ou une partie du logiciel. Il découle de la raison d'être du mesurage.

Exemple métier : la figure 2.1 montre les différents morceaux d'un logiciel - le « périmètre global » - livrés par une équipe projet :

- les composants client et serveur d'un pack logiciel intégré
- un programme offrant une interface entre le composant serveur du nouveau pack et les applications existantes
- un programme utilisé une seule fois pour convertir des données existantes au format exigé par le pack logiciel. Ce programme a été élaboré à l'aide d'objets réutilisables développés par l'équipe projet
- le logiciel pilote du périphérique pour de nouveaux dispositifs sur lesquels le composant client du pack logiciel sera exécuté

Chaque morceau de logiciel pour lequel un périmètre de mesurage a été défini est représenté par un rectangle plein.

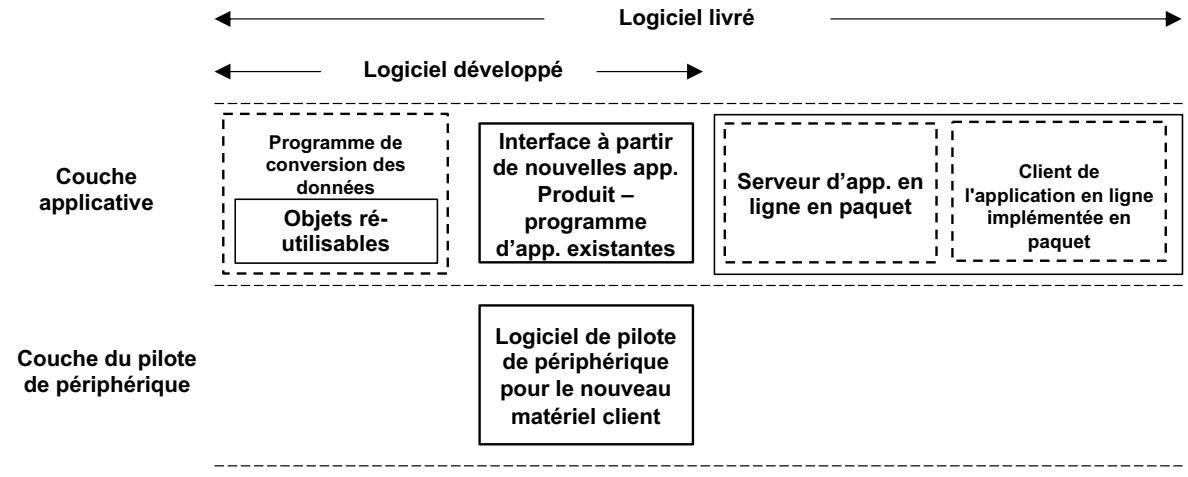


Figure 2.1 - Le périmètre global des livrables d'un projet logiciel et les périmètres de mesurage individuels.

Ce graphique montre que les morceaux de logiciel livrés sont à la fois des morceaux développés, ainsi que des morceaux implémentés par l'équipe projet. L'objectif étant de mesurer les EFU des morceaux individuels livrés pour les ajouter au portefeuille d'actifs logiciels de l'organisation, en prenant en compte le pack logiciel dans sa globalité, autrement dit, en ignorant la structure des composants client-serveur.

La taille du pack logiciel intégré a été ajoutée à celle du programme de l'interface afin de mettre à jour la taille totale du portefeuille d'actifs logiciels de l'organisation. Il n'est pas nécessaire de connaître la taille du programme de conversion de données car il n'est utilisé qu'une seule fois. Les tailles de chacun des objets réutilisables ont été répertoriées dans

l'inventaire de l'infrastructure logicielle de l'organisation, ainsi que celles des pilotes de périphériques. Une fois de plus, ces tailles ont été classées séparément.

En raison des différentes natures des livrables, il ne serait pas cohérent d'additionner les tailles de tous les morceaux de logiciel livrés pour évaluer la performance globale de l'équipe projet. Les performances des différentes équipes doivent être mesurées séparément.

Notez également qu'il est uniquement pertinent de mesurer le logiciel résultant de l'intégration d'un pack et non le pack en lui-même. La taille du pack n'intéresse peut-être que le fournisseur du pack.

2.3 Utilisateurs fonctionnels.

Plusieurs occurrences d'un seul utilisateur fonctionnel peuvent être responsables de la saisie de données dans un même processus fonctionnel. Cependant, la méthode COSMIC ne prend en compte que les types, et non les occurrences (voir la section 1.5).

Exemple métier 1 : Dans un système de commande, un certain nombre d'employés (utilisateurs fonctionnels humains) maintiennent les données. Le numéro d'identification d'un employé est ajouté à tous les groupes de données auxquels l'employé en question accède. Il faut identifier un type d'utilisateur fonctionnel « employé » car les EFU du système de commande sont les mêmes pour tous les employés.

Exemple temps réel 1 : Les quatre roues d'une voiture sont dotées d'un capteur pour mesurer la pression des pneus. À intervalles réguliers, un processus fonctionnel doit obtenir la pression des quatre pneus. Si la pression est trop faible ou trop élevée - la plage de pressions acceptables est renseignée dans le logiciel - le logiciel montre quel pneu a un problème de pression à l'aide d'un graphique représentant les quatre roues sur un écran intégré au tableau de bord. Les utilisateurs fonctionnels sont les quatre capteurs et les quatre indications sur le tableau de bord. Cependant, les quatre capteurs font l'objet de la même exigence (idem pour les indications sur l'écran), il faut alors identifier un type d'utilisateur fonctionnel « capteur » et un autre type pour les indications apparaissant sur l'écran.

Des utilisateurs fonctionnels différents nécessitent des fonctionnalités différentes.

Exemple métier 2 : Un système logiciel a pour fonctionnalité de maintenir des données personnelles de base qui sont à disposition de tout le personnel du département RH. Des données plus sensibles concernant les salaires ne sont accessibles que pour un sous ensemble des employés RH. Ce logiciel compte alors deux types d'utilisateurs fonctionnels. L'objectif du mesurage de ce logiciel doit déterminer si le périmètre de mesurage doit inclure les fonctionnalités accessibles pour tous les employés RH ou seulement les fonctionnalités disponibles pour l'un des deux types de personnel.

Les EFU sont rédigées du point de vue des utilisateurs fonctionnels. Chaque utilisateur peut avoir une vision différente du projet, alors les exigences seront exprimées en fonction des utilisateurs fonctionnels. Toutes les exigences à mesurer sont exprimées du point de vue du même utilisateur fonctionnel si leurs tailles sont comparables.

Exemple temps réel 2 : Prenons le logiciel intégré d'une photocopieuse. Les utilisateurs fonctionnels du logiciel peuvent être définis de deux manières différentes :
(a) l'utilisateur humain devant faire des photocopies,
(b) les différents éléments de la machine, comme les boutons de contrôle, l'écran sur lequel s'affiche les messages à destination de l'utilisateur humain, le mécanisme de transport du papier, les capteurs de bourrage papier, les contrôleurs d'encre, les indicateurs lumineux, etc., avec lesquels le logiciel interagit directement.

Ces deux types d'utilisateurs fonctionnels, humains ou matériels, ne voient pas les mêmes fonctionnalités. Par exemple, l'utilisateur humain ne se rendra compte que d'une partie de

toutes les fonctionnalités logicielles de la photocopieuse. Le développeur du logiciel intégré qui pilote la machine doit définir les éléments matériels comme étant les utilisateurs fonctionnels. Un employé marketing peut trouver utile de mesurer la taille des fonctionnalités de la photocopieuse produites par son entreprise, du point de vue de l'utilisateur fonctionnel humain. Il pourra ensuite comparer son prix et ses performances avec le produit d'un concurrent[1]. N'essayez surtout pas de mélanger les deux points de vue. Il serait très difficile d'interpréter la mesure d'une taille reposant sur des fonctionnalités « mixtes » (humain/matériel).

2.4 Modèles de stratégies de mesurage.

Voir des exemples dans Recommandations pour les modèles de stratégies de mesurage.

2.5 Couches.

Les logiciels ont souvent une architecture distribuant les fonctionnalités sur différents composants. Les composants communiquent entre eux à l'aide de messages suivant des règles définies dans les exigences. Les composants traitant de sujets similaires peuvent être compris dans une seule couche. Au sein d'une couche, les composants peuvent être de même niveau, clients ou serveurs. COSMIC explique que seules les tailles des composants se trouvant dans la même couche peuvent être comparées et additionnées.

Exemple 1 : Les morceaux de logiciel de la couche « application » présentée dans la figure 2.2 sont tous de même niveau.

Exemple 2 : Habituellement, la couche « supérieure » d'une architecture logicielle, autrement dit, la couche n'étant pas hiérarchiquement subordonnée aux autres, est la couche « application ». Dans cette couche, le logiciel repose sur les services de toutes les autres couches afin d'assurer son bon fonctionnement. Cette couche « supérieure » peut elle-même être formée de plusieurs couches. Par exemple, une architecture en trois couches : composants interface utilisateur, règles métier et services de données (voir l'exemple métier 2 ci-dessous).

Exemple métier 1 : La figure 2.2 représente la structure physique d'une architecture logicielle classique en couches sur laquelle repose un logiciel d'application métier.

[1] Par exemple, lors de l'atelier international sur la mesure logicielle intitulé « Defining measures for memory efficiency of the software in mobile terminals » qui a eu lieu à Magdebourg en Allemagne, en octobre 2020, M. Toivonen a comparé la taille de la fonctionnalité des téléphones portables destinée uniquement aux utilisateurs humains.

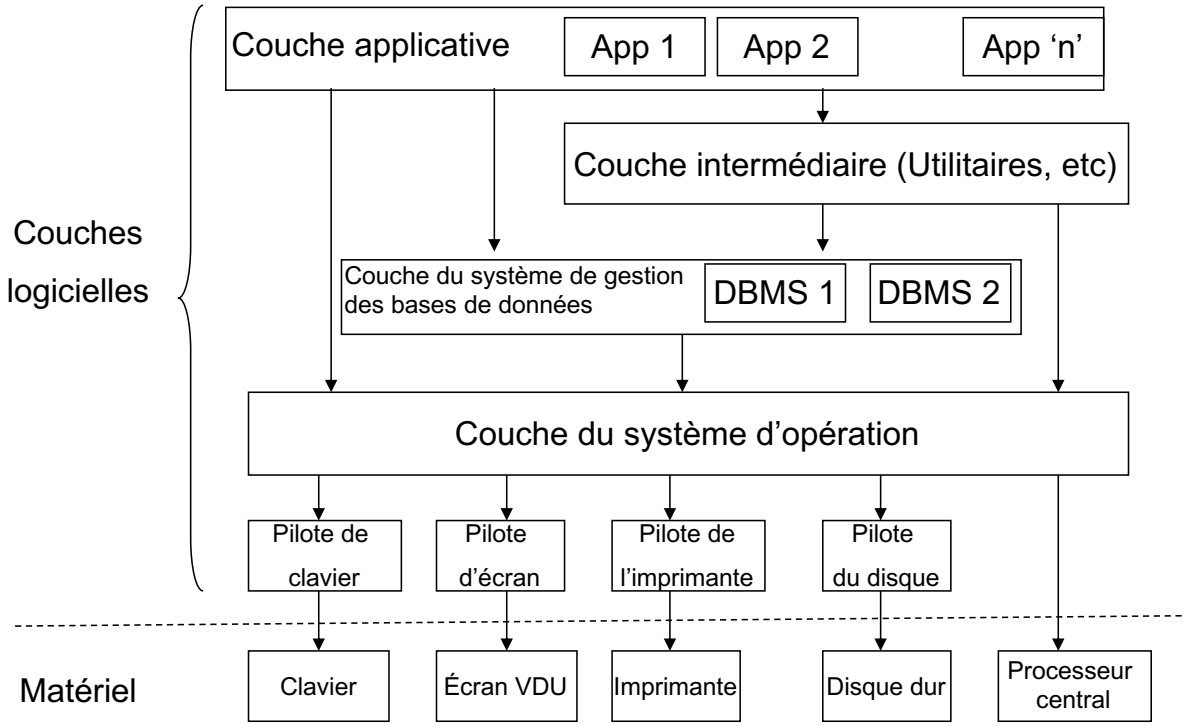


Figure 2.2- Architecture logicielle en couches classique d'un système métier/MIS.

Exemple temps réel 1 : La figure 2.3 représente la structure physique d'une architecture logicielle classique en couches d'un morceau de logiciel temps réel embarqué. (Remarque : un logiciel temps réel embarqué simple devant effectuer une seule tâche n'a pas nécessairement besoin d'un système d'exploitation temps réel.)

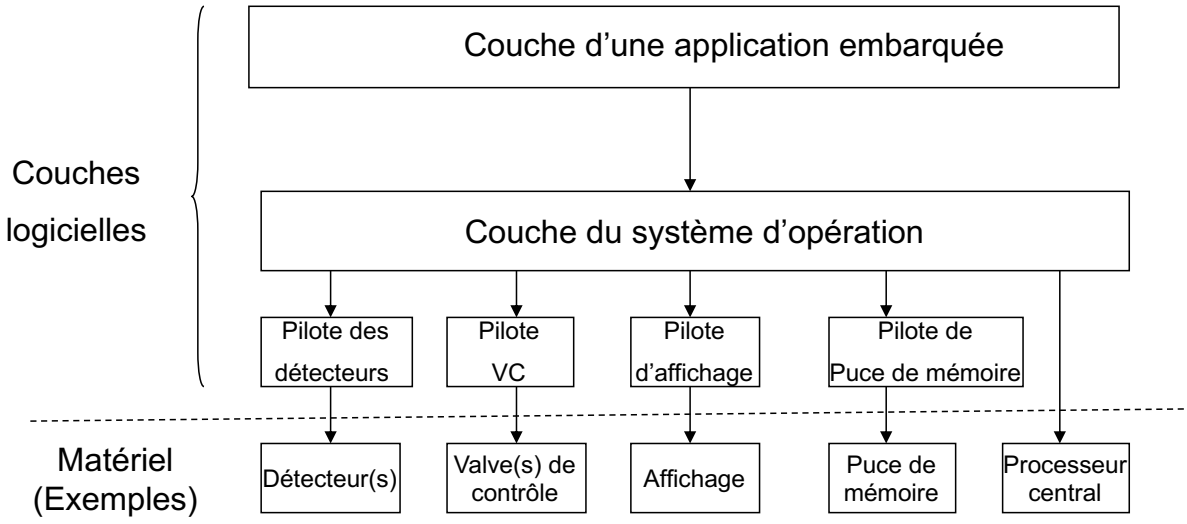


Figure 2.3 - Architecture en couche classique pour un système logiciel temps réel embarqué.

Exemple temps réel 2 : Le modèle ISO à 7 couches pour la télécommunication. Il définit une architecture en couche dans laquelle les règles d'organisation hiérarchique des couches recevant les messages sont l'inverse des règles des couches transmettant les messages.

Exemple temps réel 3 : L'architecture d'AUTOSAR, une application de l'industrie automobile, montre les différents types de règles de correspondance entre couches qui sont décrites dans les principes d'une couche.

Part 3 Figure 2.4 – The AUTOSAR architecture

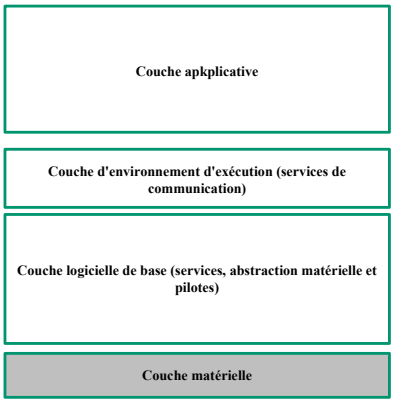


Figure 2.4 – Structure de l'architecture d'AUTOSAR

Une architecture logicielle qui montre différentes couches en fonction du point de vue.

Exemple métier 2 : Prenons une application A située dans une architecture logicielle en couche, comme sur la figure 2.5 ci-dessous, qui présente trois structures de couches possibles, a), b) et c) en fonction des différents points de vue architecturaux.

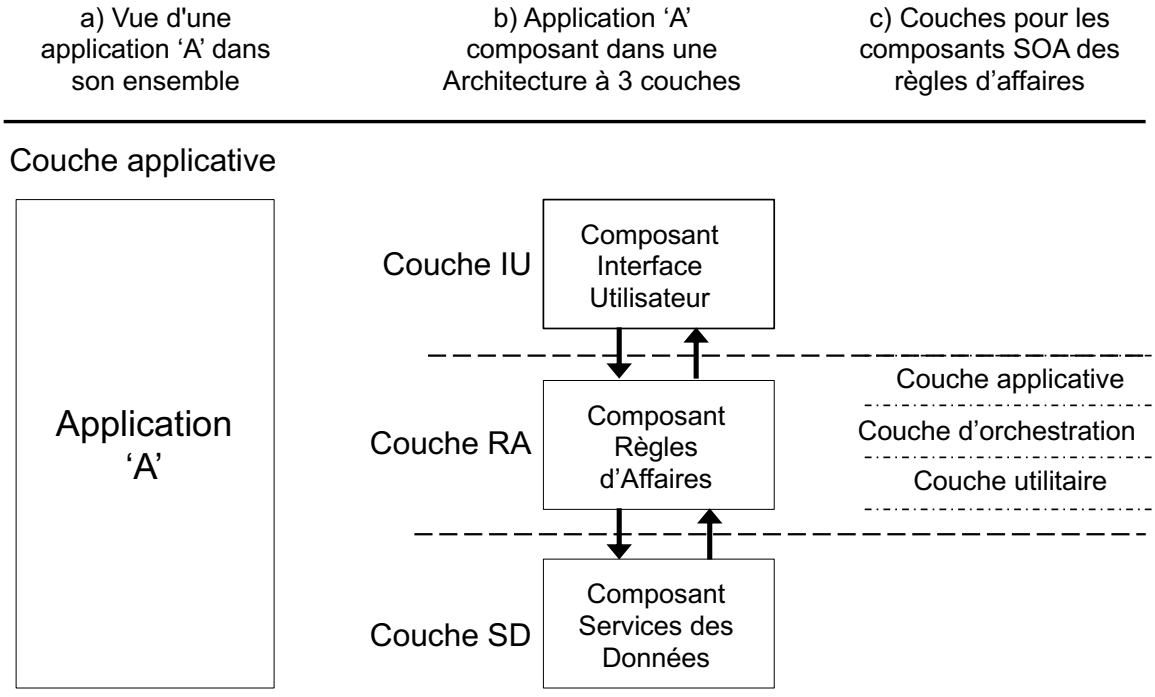


Figure 2.5 - Trois « vues » des couches d'une application

L'objectif 1 est de mesurer la taille fonctionnelle globale de l'application A, comme dans la vue a). Le périmètre de mesurage est la totalité de l'application A, qui se situe entièrement dans la couche application.

Objectif 2. L'application A a été créée d'après une architecture en trois couches comprenant les composants d'une interface utilisateur, des règles métier et des services de données. L'objectif 2 est de mesurer les trois composants séparément comme dans la vue b). Chaque composant est situé dans sa propre couche. Le périmètre de mesurage doit être défini séparément pour chaque composant.

Objectif 3. Le composant des règles métier de l'application a été construit à l'aide de composants réutilisables d'une architecture orientée service (AOS) qui a sa propre structure en couche. L'objectif 3 est de mesurer un composant AOS du composant règles métier comme dans la « vue » c). Chaque composant est situé dans une couche de l'architecture AOS et le périmètre de mesurage doit être défini séparément pour chaque composant AOS. (Notez que la terminologie AOS utilise également « couche application » dans son architecture.)

2.6 Les niveaux de décomposition.

Un logiciel peut être composé de différents niveaux de composants. Il s'agit des niveaux de décomposition de COSMIC. Il est important d'effectuer le mesurage au niveau de décomposition adapté à la raison d'être du mesurage, car les tailles des composants d'un morceau de logiciel ne peuvent être comparées qu'à celles de composants du même niveau de décomposition.

Exemple : Les recommandations de COSMIC concernant les Modèles de stratégies de mesurage reconnaissent trois niveaux standards de décomposition : « l'application entière », « les composants majeurs » et « les composants mineurs ». Voir l'exemple 2 ci-dessus, où les trois niveaux sont présentés sur la figure 2.5.

2.7 Graphique contextuel.

Un graphique contextuel permet de visualiser le logiciel à mesurer dans le contexte de ses utilisateurs fonctionnels et de son espace de stockage persistant.

Exemple métier : La figure 2.6 montre le graphique contextuel du logiciel client/serveur du pack d'application implémenté et devant être mesuré intégralement, comme dans l'exemple de la figure 2.1. Autrement dit, le fait que le pack application compte deux composants (client et serveur) est à ignorer pour mesurer la taille du pack.

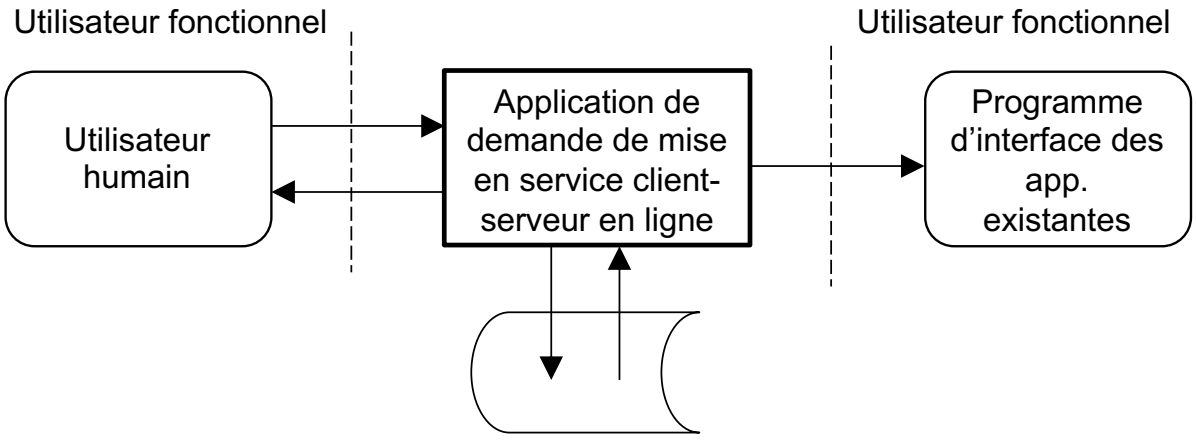


Figure 2.6 - Graphique contextuel pour l'application client-serveur.

Exemple temps réel : La figure 2.7 présente le graphique contextuel pour un système logiciel embarqué d'alarme anti-intrusion.

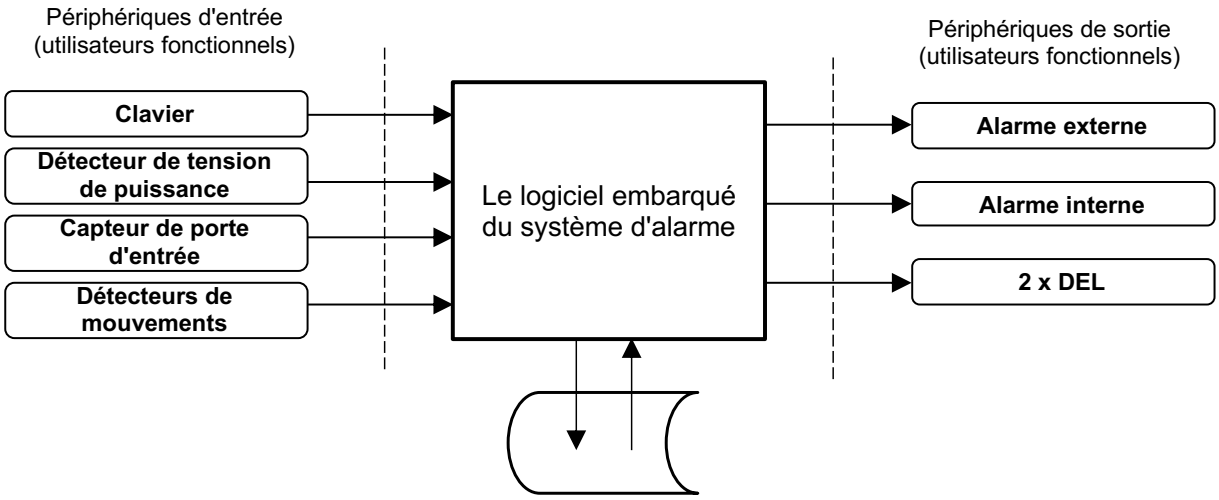


Figure 2.7 - Graphique contextuel pour le logiciel embarqué d'un système d'alarme anti-intrusion.

2.8 Les niveaux de granularité.

En début de développement, les exigences évoluent au fur et à mesure de la collecte d'informations et de la meilleure compréhension du projet. Aussi, les processus fonctionnels individuels et les informations nécessaires à leur identification et leur mesurage peuvent manquer. Dans ce cas, la taille peut être estimée ou évaluée en utilisant différentes méthodes (voir « La mesure logicielle avec COSMIC en début de projet : recommandations »). Les EFU d'un niveau de granularité supérieur décrivent, par exemple, des groupes d'utilisateurs fonctionnels, plutôt que des humains, des équipements ou des morceaux de logiciel individuels.

Lorsque les exigences pour le mesurage COSMIC sont connues, elles doivent se situer au niveau de granularité dit processus fonctionnel. Il est alors important de s'assurer que les exigences sont à ce niveau de granularité avant le début du mesurage.

Exemple : Un groupe d'utilisateurs fonctionnels peut être un « département » dont les employés gèrent plusieurs types de processus fonctionnels, ou un « tableau de bord » qui comporte plusieurs types d'instruments, ou un « système central ».

Dans des exigences fonctionnelles d'un haut niveau de granularité, un groupe d'événements peut être un flux d'entrée dans un système logiciel de comptabilité nommé « Opération commerciale » ou un flux d'entrée vers un système logiciel d'avionique nommé « Commandes pilote ».

Exemple temps réel : Pour voir des exemples de mesurages logiciels à différents niveaux de granularité et de décomposition, référez-vous à l'exemple du système de télécommunication du *Early Software Sizing with COSMIC: Experts Guidelines*.

Exemple métier : L'exemple de l'application métier fait partie du système d'e-commerce « Everest Ordering Application » (Application de Commande Everest). Cet exemple a pour but d'illustrer différents niveaux de granularité, ainsi que la découverte de processus fonctionnels à différents niveaux. La description ci-dessous a été extrêmement simplifiée pour illustrer les niveaux de granularité.

Pour mesurer cette application, il faut supposer que la raison d'être du mesurage est de déterminer la taille fonctionnelle du morceau d'application mis à disposition des utilisateurs clients humains (utilisateurs fonctionnels). Nous pouvons alors définir le périmètre de

mesurage : les morceaux de l'application Everest auxquels les clients peuvent accéder pour commander des articles. Notez cependant que l'objectif de cet exemple est d'illustrer différents niveaux de granularité. Nous n'aborderons alors que la partie des fonctionnalités du système nous permettant de comprendre le concept de niveau de granularité. Cet exemple porte sur les niveaux de granularité des EFU, il n'aborde pas l'éventuelle décomposition du logiciel.

Au niveau supérieur, le « niveau 1 (Fonctionnalités principales) », de ce morceau d'application, un énoncé des exigences de l'application de commande Everest serait un simple résumé, comme suit :

« L'Everest Odering Application doit permettre aux clients de rechercher, sélectionner, payer, se faire livrer un article provenant de la gamme de produits Everest, et des produits disponibles chez des fournisseurs tierces. »

Si on regarde cette exigence de haut niveau de plus près, on en déduit que le niveau inférieur, le niveau 2 de l'application, se compose de quatre sous-fonctionnalités, comme indiqué sur la figure 2.8 (a).

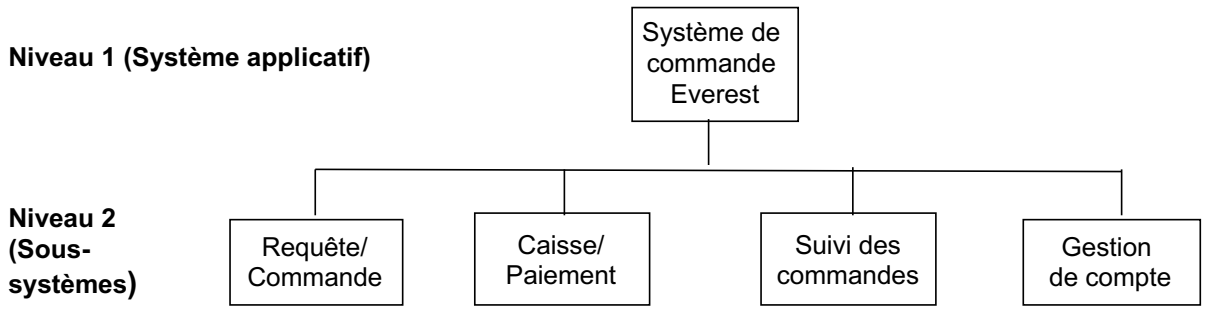


Figure 2.8 (a) - Analyse du système de commande Everest : les deux premiers niveaux de granularité

Les exigences des quatre sous-fonctionnalités sont les suivantes :

- La sous-fonctionnalité recherche/commande, qui permet à un client de trouver un produit, son prix et sa disponibilité dans la base de données d'Everest, et d'ajouter le produit sélectionné au panier pour l'acheter. Cette sous-fonctionnalité favorise également les ventes en suggérant des offres spéciales, en proposant des revues d'articles sélectionnés et en permettant des requêtes générales telles que les renseignements sur les conditions de livraison, etc. C'est une sous-fonctionnalité très complexe. Nous n'analysons donc pas cette sous-fonctionnalité plus en détail au-dessous du niveau 2 dans le cadre de cet exemple.
- La sous-fonctionnalité commande/paiement qui permet à un client de passer commande et de payer.
- La sous-fonctionnalité suivi de commande qui permet au client de se renseigner sur une commande existante, de mettre à jour les informations concernant la commande (ex : adresse de livraison) ou encore de retourner des articles.
- La sous-fonctionnalité gestion de compte qui permet à un client existant de gérer différentes informations de son compte comme son adresse, ses moyens de paiement, etc.

Les figures 2.8 (b) et (c) présentent certains détails qui apparaissent si l'on regarde les exigences des sous-fonctionnalités susmentionnées à un niveau de granularité inférieur. Lors de ce processus d'analyse, notez que :

- Le périmètre des fonctionnalités à mesurer n'a pas changé,

- Les différents niveaux de description de l'application Everest montrent toutes les fonctionnalités à disposition du client (en tant qu'utilisateur fonctionnel). Un client peut « voir » les fonctionnalités de l'application à tous ces niveaux de granularité.

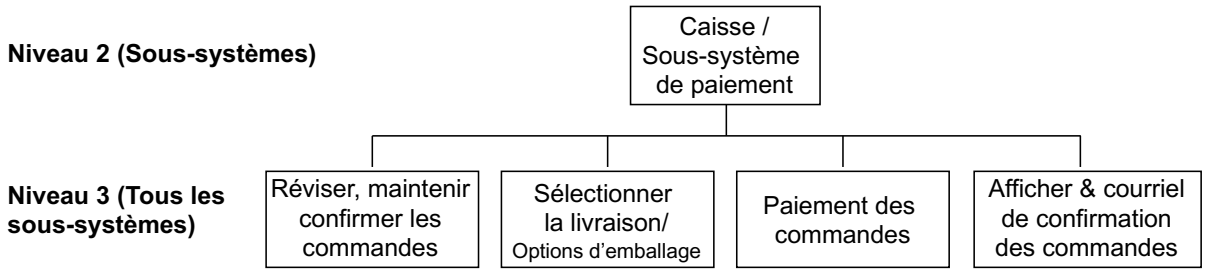


Figure 2.8 (b) - Analyse de la sous-fonctionnalité commande/paiement.

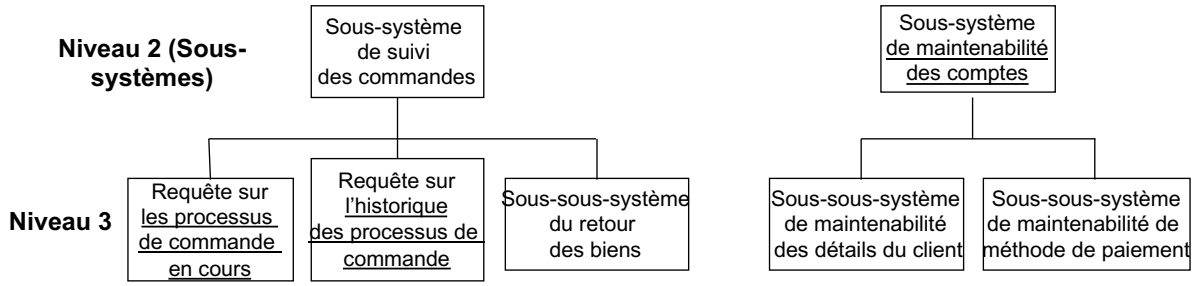


Figure 2.8 (c) - Analyse des sous-fonctionnalités suivi de commande et gestion de compte.

La figure 2.8 (c) montre que lorsque l'on regarde de plus près le niveau 3, le niveau le plus bas d'analyse de la sous-fonctionnalité « suivi de commande », il y a deux processus fonctionnels individuels[2] (pour deux demandes de la sous-fonctionnalité « suivi de commande »). D'autres processus fonctionnels émergeraient si l'on continuait d'affiner le niveau 3 de sous-sous-fonctionnalité vers les niveaux encore plus bas. Cet exemple démontre que lorsqu'une fonctionnalité est affinée du haut vers le bas, on ne peut pas être sûr qu'elle correspondra toujours au niveau de granularité auquel elle a été découverte, comme l'explique la méthode COSMIC. (Pour cette définition, Les fonctionnalités doivent avoir « un niveau de détails comparable » à chaque niveau de granularité.)

Par ailleurs, un autre analyste aurait pu faire un graphique totalement différent, mettant en avant d'autres groupes de fonctionnalités à chaque niveau. Il n'y a pas une seule bonne manière d'analyser les fonctionnalités d'un système aussi complexe.

Étant donné les variations, la personne effectuant la mesure doit examiner attentivement les différents niveaux d'un graphique d'analyse pour identifier les processus fonctionnels devant être mesurés. Si ce n'est pas possible, il faut appliquer une méthode de mesure approximative. C'est le cas par exemple, si l'analyse n'a pas encore atteint le niveau où tous les processus fonctionnels sont connus. Pour illustrer cela, examinons le cas de la sous-sous-fonctionnalité « Maintenir les informations client » (voir la figure 2.8 (c) ci-dessus), issue de la branche de la sous-fonctionnalité « Gestion de compte ».

Pour un mesureur expérimenté, le terme « Maintenir » suggère presque toujours un groupe d'événements et donc un groupe de processus fonctionnels. Nous pouvons donc supposer que la sous-sous-fonctionnalité « maintenir » comprend trois processus fonctionnels, à savoir, « rechercher détails clients », « mettre à jour détails client » et « supprimer détails client ». (Le processus « créer détails client » doit manifestement exister, mais il se trouve dans une autre branche du système, car il intervient lorsqu'un client passe une commande pour la première fois. Il se situe en dehors du périmètre de l'exemple simplifié.)

Un mesureur expérimenté devrait pouvoir estimer rapidement la taille de cette sous-sous-fonctionnalité en Points de Fonction COSMIC en multipliant le nombre de processus fonctionnels supposés (trois dans ce cas) par la taille moyenne d'un processus fonctionnel. Cette taille moyenne est obtenue par calibration d'autres morceaux du système ou d'autres systèmes comparables. On peut trouver des exemples de ce processus de calibration dans le document « Early software sizing with COSMIC : Experts Guide », qui contient également des exemples d'autres méthodes de mesure approximatives.

Bien évidemment, de telles méthodes d'approximation ont des limites. Si l'on applique une telle méthode au niveau des exigences données plus haut (« l'application Everest doit permettre aux clients de rechercher, sélectionner, payer, se faire livrer un article provenant de la gamme de produits Everest...), il est possible d'identifier quelques processus fonctionnels. Mais une analyse plus détaillée révèlerait que le vrai nombre de processus fonctionnels de cette application complexe doit être beaucoup plus important. Voilà pourquoi, habituellement, plus les exigences fonctionnelles sont détaillées, plus la taille fonctionnelle donne l'impression d'augmenter, même si le périmètre reste inchangé. À un niveau de granularité supérieur, ces méthodes d'approximation doivent être utilisées avec prudence car très peu de détails sont disponibles.

3 PHASE DE MISE EN CORRESPONDANCE.

3.1 Processus fonctionnels.

Selon la méthode COSMIC, il est essentiel d'identifier les processus fonctionnels pour obtenir une mesure correcte de la taille. Voici comment les processus fonctionnels peuvent être identifiés.

Un utilisateur fonctionnel humain saisit des données, ce qui déclenche un processus fonctionnel.

Exemple métier 1 : Dans une entreprise, on reçoit une commande (événement déclencheur), ce qui oblige un employé (utilisateur fonctionnel) à saisir les informations de la commande (Entrée de déclenchement transmettant des données concernant l'objet d'intérêt « commande »). C'est le premier mouvement de données du processus fonctionnel « Saisie de commande ».

Exemple métier 2 : Un processus fonctionnel d'un système logiciel RH peut être déclenché par une Entrée qui transfert un groupe de données décrivant un nouvel employé. Le groupe de données est généré par l'utilisateur fonctionnel humain qui saisit les données.

Un utilisateur fonctionnel matériel transfert des données, ce qui déclenche un processus fonctionnel.

Exemple temps réel 1 : Un processus fonctionnel d'un système logiciel temps réel peut être déclenché par une Entrée informant le processus fonctionnel du signal de l'horloge (processus fonctionnel). Le groupe de données transfert des données qui informent uniquement qu'un événement a eu lieu.

Exemple temps réel 2 : Un processus fonctionnel d'un système logiciel temps réel de détection d'incendie peut être déclenché par une Entrée initiée par un détecteur de fumée

spécifique (utilisateur fonctionnel). Le groupe de données générées par le détecteur de fumée transfert l'information « fumée détectée » (un événement a eu lieu). Il inclut le numéro du détecteur (autrement dit, ces données peuvent-être utilisées pour déterminer où l'événement s'est produit).

Exemple temps réel 3 : Un lecteur de codes-barres (utilisateur fonctionnel) d'une caisse de supermarché commence à scanner lorsqu'un code-barres passe devant (événement déclencheur). Le lecteur génère un groupe de données, comprenant une image du code-barres qui entre dans le logiciel de caisse. L'image du groupe de données est transférée vers son processus fonctionnel par une Entrée de déclenchement. Le processus fonctionnel ajoute le prix de l'article à la facture du client si le code est valide. Un bip sonore est émis pour informer le caissier que le produit est accepté. La vente est confirmée.

Identifier les événements déclencheurs distincts - et donc les processus fonctionnels distincts - lorsqu'un utilisateur fonctionnel humain prend des décisions indépendamment du logiciel concernant ce qu'il faut faire ensuite et qui demandent des réponses différentes de la part du logiciel.

Exemple métier 3 : Un utilisateur fonctionnel saisit la commande d'un client dont l'objet est un équipement industriel complexe, puis il confirme la commande auprès du client. Entre la saisie et la validation de la commande, l'utilisateur peut se renseigner pour savoir si la commande peut être livrée en temps voulu, si le client est solvable, etc. Bien que la saisie de la commande doive être suivie de la validation, dans ce cas, l'utilisateur peut décider séparément d'accepter ou non la commande. Cette exigence précise qu'il y a des processus fonctionnels séparés pour la saisie d'une commande et sa confirmation (et pour chaque requête).

Identifier les événements déclencheurs et les processus fonctionnels séparés lorsque les responsables des actions sont distincts.

Exemple métier 4 : Un système RH, où le responsable de la gestion des données personnelles de base n'est pas le même que celui de la gestion des données des effectifs, a des utilisateurs fonctionnels séparés ayant chacun son propre processus fonctionnel.

Exemple métier 5 : Prenons le reçu d'une commande de l'exemple métier 1. L'application de traitement des commandes doit transférer les données des nouveaux clients vers une application « d'inscription client » centrale devant être mesurée. L'application de traitement de commande est donc un utilisateur fonctionnel de l'application centrale. Après avoir reçu les données concernant le nouveau client, l'application de traitement de commande génère le groupe de données du client pour l'envoyer vers l'application centrale, qui va déclencher un processus fonctionnel pour stocker ces données.

Lors de la mesure de la taille des processus fonctionnels, il ne devrait pas y avoir de différence entre le traitement en ligne ou en lots (mode batch).

Exemple métier 6 : Imaginons que les commandes dans l'exemple métier 1 ci-dessus sont saisies par un processus « hors ligne », comme par exemple, le scannage de documents papiers qui sont temporairement stockés en vue d'un traitement automatique en lots. Comment le processus fonctionnel de la saisie de commande peut-il être analysé s'il doit être traité en lots ? L'utilisateur fonctionnel est l'humain qui saisit la commande prête à être traitée en lots. L'Entrée de déclenchement du processus fonctionnel qui traite les commandes en lots est le mouvement de données qui transfert le groupe de données de la commande vers le processus. S'il doit être mesuré, ne pas oublier que le processus hors ligne implique l'exécution d'un processus fonctionnel séparé qui charge les commandes dans un espace de stockage temporaire.

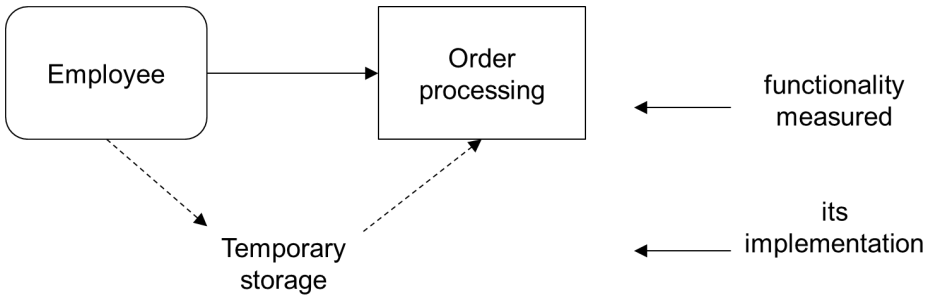


Figure 3.1 – Traitement par lots de l'exemple métier 4.

Exemple métier 7 : Imaginons qu'une EFU d'une application de traitement par lots demande, à chaque fin d'année, de générer un rapport déclarant le résultat de l'activité pour l'année et de mettre les compteurs à zéro pour l'année suivante. Physiquement, un « tic » d'horloge marquant la fin de l'année est généré par le système d'exploitation. C'est lui qui déclenche le processus qui est exécuté par l'application. Cependant, logiquement, chaque processus fonctionnel de l'application trouve ses données d'entrée dans le flux de données à traiter par lots. Cette exigence doit être analysée normalement (par exemple, les données d'un processus fonctionnel comprennent au moins une Entrée, la première étant l'Entrée de déclenchement du processus).

Cependant, supposons que l'un des processus fonctionnels de l'application de traitement par lots ne nécessite aucune donnée d'entrée pour générer ses rapports. Physiquement, l'utilisateur fonctionnel (humain) a délégué le déclenchement de ce processus fonctionnel au système d'exploitation. Étant donné que tout processus fonctionnel doit avoir une Entrée de déclenchement, on peut considérer que c'est le passage de l'horloge à la nouvelle année qui déclenche le processus. Ce processus fonctionnel peut avoir besoin de plusieurs Lectures et différentes Sorties afin de produire ses rapports. Logiquement, l'analyse de cet exemple reste la même si c'est l'utilisateur fonctionnel humain qui déclenche la génération d'au moins un rapport avec un clic de souris sur l'élément d'un menu en ligne, et non le système d'exploitation.

Pour les applications temps réel, c'est habituellement un capteur qui déclenche un processus fonctionnel.

Exemple temps réel 5 : Lorsqu'un capteur (utilisateur fonctionnel) détecte que la température atteint une certaine valeur (événement déclencheur), le capteur envoie un signal pour déclencher le mouvement de données de l'Entrée de déclenchement d'un processus fonctionnel permettant d'éteindre le chauffage (un autre utilisateur fonctionnel).

Exemple temps réel 6 : Un avion militaire est doté d'un capteur qui détecte l'événement « missile en approche ». Le capteur est un utilisateur fonctionnel du logiciel qui doit répondre à la menace. Dans le cas de ce logiciel, un événement a lieu seulement lorsque le capteur détecte quelque chose. C'est le capteur (utilisateur fonctionnel) qui génère le groupe de données d'une entrée de déclenchement (par exemple, « le capteur 2 a détecté un missile »), et un éventuel flux de données concernant la vitesse et les coordonnées du missile en question.

Identifier un processus fonctionnel sur les bases de l'organisation de la saisie de données, ou examiner les menus de certains logiciels installés.

Exemple métier 8 : Imaginons des EFU pour deux types de prestations sociales, le premier pour un enfant supplémentaire et le second pour un « crédit d'impôt » pour les faibles revenus. Ces exigences imposent au logiciel de répondre à deux événements distincts dans l'univers des utilisateurs fonctionnels humains. Même si un seul formulaire a été utilisé pour recueillir les données, il y a bien deux processus fonctionnels.

Identifier uniquement les mouvements de données d'un processus fonctionnel. Les différents chemins de traitement pouvant être empruntés ne mènent pas à des processus fonctionnels séparés.

Exemple métier 9 : Pour effectuer une recherche dans une base de données, un processus fonctionnel doit pouvoir accepter jusqu'à quatre paramètres de recherche (attributs de l'Entrée de déclenchement). Ce processus fonctionnel s'executera même si seules les valeurs d'un, deux ou trois paramètres de recherche sont renseignés.

Exemple métier 10 : Pour qu'un processus fonctionnel inscrive le nouveau client d'une société de location de véhicules, il doit saisir les données pour la plupart des attributs de données, mais certaines informations (par exemple, informations de contact) sont optionnelles. Peu importe si tout ou partie de ces attributs sont renseignés, il n'y a qu'un seul processus fonctionnel pour l'inscription d'un nouveau client.

Exemple métier 11 : Reprenons l'exemple 10. Lorsqu'un processus fonctionnel effectue une réservation de véhicule, plusieurs options sont proposées, comme par exemple, une assurance supplémentaire, un conducteur supplémentaire, un siège auto, etc. Toutes ces options mènent vers des chemins de traitement différents au sein d'un seul processus fonctionnel de réservation.

Exemple temps réel 12 : Dans un système d'avionique, l'Entrée de déclenchement (informations sur l'altitude de l'avion envoyées par GPS) d'un processus fonctionnel mène vers l'un des deux chemins de traitement du processus fonctionnel, en fonction de la valeur de l'Entrée (si l'altitude de l'appareil est supérieure ou inférieure à une altitude donnée). Les différents chemins affichent différents groupes de données sur la carte du pilote. Si l'altitude est trop basse, des alertes supplémentaires seront émises. Il n'y a qu'un seul processus fonctionnel.

3.2 Groupes de données et objets d'intérêt.

Les groupes de données et les objets d'intérêt peuvent prendre différentes formes dans les exigences. C'est le mesureur qui doit les identifier dans le contexte du processus fonctionnel mesuré.

Exemple 1 : En pratique, un groupe de données peut avoir différentes provenances, par exemple :

a) Une structure de données sur un périphérique de stockage (fichier, table de base de données, mémoire ROM, etc.).

b) Une structure de données au sein de la mémoire volatile de l'ordinateur (structure de données allouée dynamiquement ou au travers d'un bloc d'espace mémoire alloué au préalable).

c) Une présentation groupée d'attributs de données fonctionnellement liés concernant un périphérique en entrée/sortie (afficher l'écran, imprimer le rapport, contrôler l'écran d'affichage, etc.)

d) Un message en cours de transmission entre un dispositif et un ordinateur ou sur un réseau, etc.

Exemple 2 : La mémoire en lecture seule (ROM) est un stockage persistant si la récupération de ses données est requise par le FUR.

Exemple métier 1 : Dans une application métier, un objet d'intérêt peut être l'« employé » (physique) ou la « commande » (conceptuel). Dans le cas de la «commande», il découle généralement de la FUR des commandes multi lignes que deux objets d'intérêt sont identifiés: «commande» et «ligne de commande». Les groupes de données correspondants peuvent être nommés « données de commande » et « données de ligne de commande ».

Exemple métier 2 : Prenons la EFU d'un processus fonctionnel de requête ad hoc dans une base de données pour trouver le nombre d'employés dépassant un certain âge. Le paramètre d'entrée (l'âge limite) est un groupe de données qui définit un objet d'intérêt « l'ensemble des employés dépassant l'âge limite ». Le groupe de données de sortie, qui comprend le nombre d'employés dépassant l'âge limite, décrit le même objet d'intérêt que l'entrée.

Exemple métier 3 : Poursuivons avec l'exemple métier 2. En plus de sortir le nombre d'employés dépassant l'âge limite, la requête doit également donner la liste des noms. L'analyse sera la même que pour l'exemple métier 2, mais le processus fonctionnel doit maintenant sortir deux groupes de données : le nombre d'employés dépassant l'âge limite, ainsi qu'une liste de leurs noms (données dérivées des données persistantes). Ces deux groupes de données doivent être séparés car ils n'ont pas le même nombre d'occurrences (une pour les employés et éventuellement plusieurs pour les noms).

Exemple métier 4 : Une structure de données commune représente les objets d'intérêt mentionnés dans les EFU, pouvant être maintenus par des processus fonctionnels et auxquels la plupart de processus fonctionnels présents dans le logiciel mesuré peuvent accéder.

Identifier les groupes de données et les objets d'intérêt des logiciels temps réel.

Exemple temps réel 1 : Un groupe de données entrant dans un logiciel à partir d'un dispositif physique donne des informations sur l'état réel du dispositif. Dans ce cas, le dispositif physique est l'objet d'intérêt (et l'utilisateur fonctionnel). Le groupe de données transfère des informations sur son état, comme par exemple, si une vanne est ouverte ou fermée, ce qui déclenche un processus fonctionnel. L'objet d'intérêt est le dispositif physique. De la même manière, la sortie d'un groupe de données vers un appareil, comme par exemple, l'activation d'un signal lumineux, transfère des données sur l'état de l'objet d'intérêt « lampe ».

Exemple temps réel 2 : Un système logiciel d'échange de messages peut recevoir le groupe de données d'un message en tant qu'entrée et l'acheminer sans modification en tant que sortie, selon la FUR du morceau de logiciel en question. Les attributs du groupe de données du message peuvent être, par exemple : le numéro d'identification du message, le numéro d'identification de l'expéditeur, celui du destinataire, le code d'itinéraire et le contenu du message. L'objet d'intérêt du message est « message ».

Exemple temps réel 3 : Une structure de données de référence représente des objets d'intérêt dont les valeurs des attributs sont données dans des tableaux trouvés dans les EFU, qui sont maintenus dans des stockages persistants (mémoire ROM, par exemple) et accessibles pour la plupart des processus fonctionnels du logiciel mesuré.

Exemple temps réel 4 : Les fichiers communément appelés « fichiers plats » représentent les objets d'intérêt mentionnés dans les EFU qui se trouvent sur un appareil de stockage.

Un utilisateur fonctionnel peut être un objet d'intérêt.

Exemple métier 5 : Un utilisateur fonctionnel humain saisit un numéro d'identification et un mot de passe dans un processus d'identification pour se connecter à un système. L'objet d'intérêt du groupe de données saisi est l'utilisateur humain.

Exemple temps réel 5 : Prenons un capteur de température A, qui envoie une mesure de la température d'un matériau vers un processus fonctionnel pour qu'elle soit traitée. Le capteur fournit des informations concernant son propre état, il est donc l'objet d'intérêt du groupe de données.

3.3 Attributs de données.

La méthode COSMIC n'exige pas directement d'identifier les attributs de données associés à un groupe de données. Cependant, les recommandations concernant les règles 13 et 14 - Unicité des mouvements de données de la section 3.4 de la partie 2 – imposent un examen des attributs de données.

Exemple métier : L'objet d'intérêt « employé » peut être décrit par un groupe de données appelé « Données principales de l'employé », qui contient les attributs de données « ID employé », « adresse », « date de naissance », « sexe », « situation familiale », « numéro de sécurité sociale », « grade », « emploi », etc.

Exemples temps réel : Un capteur de température peut, sur demande, renseigner l'attribut « température ». Le capteur d'un système de sécurité peut détecter un intrus et envoyer l'attribut « mouvement détecté ». Un message transmis peut avoir les attributs « de (adresse), à (adresse), (contenu) ».

3.4 Mouvements de données.

Mouvements de données et « tics d'horloge ».

Exemple temps réel 1 : Pour un événement de type « tic d'horloge » ayant lieu toutes les trois secondes, identifier une entrée transférant un groupe de données d'un attribut de données. L'objet d'intérêt (et l'utilisateur fonctionnel) est l'horloge, le groupe de données informe sur l'état de l'horloge.

Mouvement(s) de données et obtention de la date et/ou de l'heure de l'horloge du système.

Exemple métier 1 : Lorsqu'un processus fonctionnel ajoute l'horodatage à un registre en vue de le stocker de manière persistante ou d'en faire une sortie, il n'y a pas d'Entrée. Par convention, obtenir la valeur de l'horloge du système est une fonctionnalité mise à disposition à tous les processus fonctionnels par le système d'exploitation.

Mouvement(s) de données et toutes données non liées à un objet d'intérêt d'un utilisateur fonctionnel.

Exemple métier 2 : Ne pas compter les mouvements de données permettant de déplacer les données générales d'une application, telles que les en-têtes et pieds de page (nom de l'entreprise, nom de l'application, date du système, etc.) qui apparaissent sur tous les écrans.

Exemple métier 3 : Ne pas compter les mouvements de données permettant de déplacer les commandes de contrôle (un concept défini uniquement pour les applications métier) qui permettent à un utilisateur fonctionnel de naviguer dans le logiciel, et non de déplacer des données. Par exemple, les commandes de défilement de la page, cliquer sur « OK » pour fermer un message d'erreur, etc. (voir la section 3.6).

Mouvement(s) de données pour toutes les données décrivant un objet d'intérêt.

Exemple métier 4 : Le cas le plus fréquent est qu'une Écriture, doit être identifiée, qui déplace un groupe de données contenant tous les attributs de données d'un objet d'intérêt devant être rendu persistant dans un processus fonctionnel donné.

Mouvement(s) de données lorsque différents utilisateurs fonctionnels déplacent différents groupes de données décrivant le même objet d'intérêt.

Exemple temps réel 2 : Un processus fonctionnel doit accepter différents groupes de données provenant de deux sismomètres différents (utilisateurs fonctionnels) répondant tous deux au même événement, comme par exemple, une explosion test. Il y a deux Entrées.

Exemple métier 5 : Prenons des EFU qui décrivent un unique processus fonctionnel produisant au moins deux sorties déplaçant différents groupes de données décrivant le même objet d'intérêt et destinés à différents utilisateurs fonctionnels. Par exemple, lorsqu'un nouvel employé est engagé, un rapport est généré et soumis à l'employé en question pour qu'il vérifie et valide ses données personnelles. Un message est ensuite envoyé à la sécurité pour autoriser l'employé à accéder aux locaux. Il y a deux Sorties.

Mouvement(s) de données pour déplacer différents groupes de données décrivant le même objet d'intérêt vers/depuis le stockage persistant.

Exemple métier 6 : Prenons les EFU d'un unique processus fonctionnel A pour stocker deux groupes de données dérivées des fiches de comptes d'une banque, en vue d'une utilisation ultérieure par d'autres programmes. Le premier groupe de données regroupe les informations des « comptes à découvert » (il comprend l'attribut « solde négatif »). Le second groupe de données regroupe les informations « comptes solde élevé » (il comprend uniquement le nom et l'adresse des propriétaires de comptes, en vue d'une campagne marketing). Le processus fonctionnel A a deux Écritures, une pour chaque groupe de données. Elles décrivent toutes deux le même objet d'intérêt « compte ».

Exemple métier 7 : Prenons les EFU d'un programme permettant de fusionner deux fichiers de données persistantes décrivant le même objet d'intérêt. Par exemple, un fichier contenant des données existantes concernant un objet d'intérêt « X » et un fichier contenant des attributs nouvellement définis décrivant le même objet d'intérêt « X ». Il y a deux Lectures, une pour chaque fichier, pour ce processus fonctionnel.

Mouvement(s) de données pour des occurrences répétées de tout mouvement de données décrivant le même objet d'intérêt.

Exemple métier 8 : Supposons qu'une lecture d'un groupe de données soit requise dans les EFU, mais que le développeur décide de l'implémenter à l'aide de deux commandes pour récupérer différents sous-groupes d'attributs de données du même objet d'intérêt provenant d'un stockage persistant à différents moments du processus fonctionnel. Identifier une Lecture.

Exemple métier 9 : Supposons qu'une lecture soit requise dans les EFU qui, en pratique, nécessite de nombreuses occurrences de récupération, comme dans une recherche dans un fichier. Identifier une Lecture.

Exemple temps réel 3 : Supposons que dans un processus fonctionnel temps réel, les EFU exigent que le même groupe de données soit saisi par un utilisateur fonctionnel donné (un dispositif matériel par exemple) deux fois à intervalles de temps prédéfinis afin de mesurer le taux de change au cours du processus. Dans COSMIC, les deux mouvements de données sont considérés comme étant plusieurs occurrences de la même Entrée. Une seule Entrée peut être comptée pour ce groupe de données au sein de ce processus fonctionnel. Notez qu'il n'y a aucune manipulation de données associée aux deux occurrences de l'Entrée. Le calcul du taux de change est associé à la Sortie qui rapporte le taux.

Exemple temps réel 4 : Imaginons un système de contrôle de processus pour une machine qui produit quelque chose de plat tel que du papier ou un film plastique. La machine est dotée de 100 capteurs identiques disposés transversalement à la direction du mouvement de la production afin de détecter les ruptures ou les trous dans le produit. Le processus fonctionnel en charge de la vérification reçoit les mêmes données de chaque capteur. Par exemple, la position d'un trou dans le produit peut être déterminée à partir de la position dans le flux de valeur de données provenant des capteurs. Le traitement des données de tous les capteurs est identique. Identifier un utilisateur fonctionnel pour tous les capteurs et une Entrée pour les données obtenues par l'ensemble des capteurs lors du processus fonctionnel.

Mouvement de données lorsque la Sortie d'une requête est un texte prédéfini.

Exemple métier 10 : Compter une Sortie pour le texte prédéfini, comme après un clic sur le bouton « Conditions générales » d'un site de vente en ligne par exemple.

Mouvement(s) de données lorsqu'un processus fonctionnel doit déplacer un groupe de données depuis un stockage persistant

Exemple 1 : Cet exemple concerne un logiciel A qui doit récupérer un groupe de données stocké. Les EFU du logiciel A ne sont pas concernés par la manière dont ces accès aux données sont gérés par tout autre logiciel dans les mêmes couches ou dans des couches différentes.

Les utilisateurs fonctionnels du logiciel A peuvent être, par exemple, des utilisateurs humains si le logiciel A se trouve dans la couche application lançant une requête sur le groupe de données stocké. La figure 3.2 représente les mouvements de données COSMIC de cette requête. Cette requête est déclenchée par une Entrée, suivie d'une Lecture du groupe de données provenant d'un espace de stockage persistant, puis d'une Sortie pour le résultat de la requête. Le processus fonctionnel A ne se préoccupe pas de l'endroit d'où proviennent les données, tant qu'il s'agit de données persistantes.

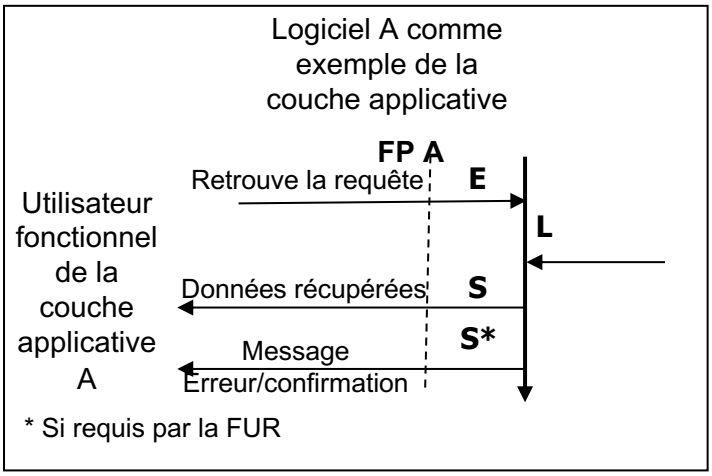


Figure 3.2 – Solution pour une Lecture provenant de la couche application du logiciel A.

Un modèle parfaitement analogue pourrait s'appliquer si le processus fonctionnel A devait rendre persistant un groupe de données à l'aide d'un mouvement de données de type éCriture. Selon la règle d) de la partie sur les conditions d'erreurs, les mouvements de données Lecture et éCriture prennent en compte tout code de retour ou rapport d'une condition d'erreur.

La figure 3.2 montre un éventuel message d'erreur spécifique à l'application pouvant provenir du processus fonctionnel A. Par exemple, l'enregistrement demandé est introuvable. Une condition d'erreur non spécifique à l'application ne sera pas comptée comme une Sortie du processus fonctionnel A. Voir également la section 4.9 de la partie 1 pour les messages d'erreur et de confirmation.

Mouvement(s) de données lorsqu'un processus fonctionnel doit obtenir certaines données provenant d'un autre morceau de logiciel.

Exemple 2 : On suppose que les morceaux de logiciel devant être mesurés ont une relation « client/serveur ». Autrement dit, le morceau de logiciel « client » obtient des services et/ou des données de l'autre morceau logiciel, le « serveur », dans la même couche logicielle ou dans une autre. La figure 3.3 présente l'exemple d'une relation du genre. Les deux morceaux sont les principaux composants de la même application. Dans une relation

client/serveur, les EFU du composant client C1 considèrent que le composant serveur C2 est l'un de ses utilisateurs fonctionnels, et réciproquement. La relation est la même et le graphique s'applique, peu importe si les deux morceaux de logiciels sont des applications distinctes, ou si l'un d'eux est le composant d'une application distincte.

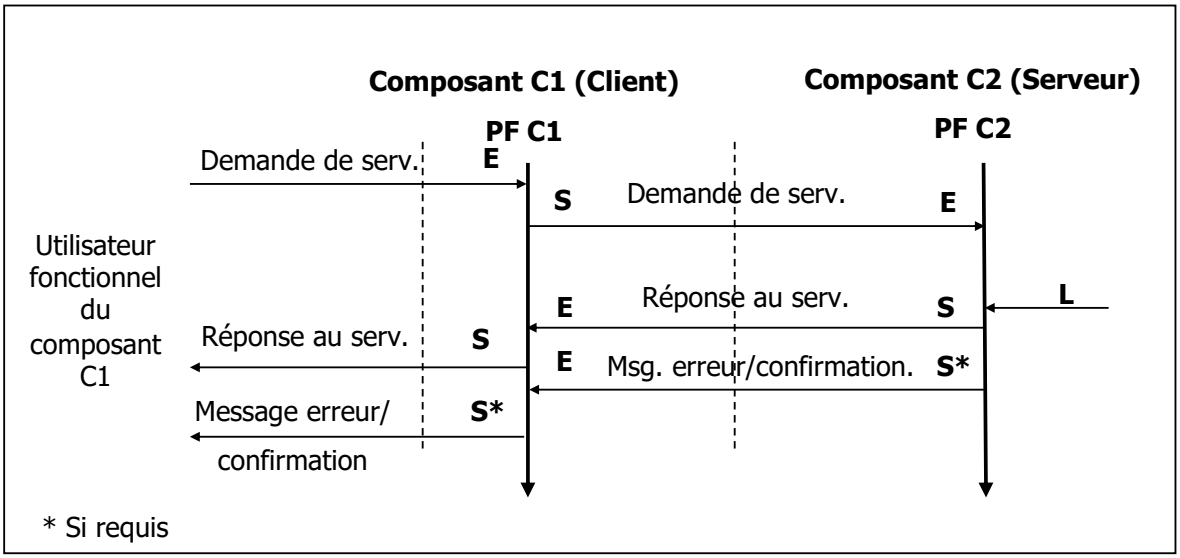


Figure 3.3 – Échanges de données entre composants client et serveur.

Physiquement, les deux composants pourraient être exécutés par des processeurs séparés. Dans ce cas, ils échangent des données via leurs systèmes d'exploitation respectifs et tout autre couche intermédiaire de leurs processeurs ayant une architecture logicielle comme celle présentée sur la figure 2.2. Mais logiquement, en appliquant les modèles COSMIC, les deux composants échangent des données à l'aide d'une Sortie, suivie d'une Entrée. Tous les logiciels ou dispositifs matériels intervenant ici sont ignorés.

La figure 3.3 montre qu'un processus fonctionnel FP C1 du composant client C1 est déclenché par une Entrée d'un utilisateur fonctionnel (tel qu'un humain) qui se compose, par exemple, des paramètres de la requête. Les EFU du composant C1 reconnaissent que ce dernier doit interroger le composant serveur C2 concernant les données requises, et préciser quel groupe de données est requis.

Afin d'obtenir le groupe de données requis, le processus fonctionnel PF C1 émet une Sortie contenant les paramètres de la requête à l'attention du composant C2. Cette Sortie traverse la frontière se trouvant entre C1 et C2, et devient donc l'Entrée de déclenchement du processus fonctionnel PF C2 dans le composant C2. Le processus fonctionnel PF C2 du composant C2 est supposé obtenir le groupe de données requis à l'aide d'une Lecture de son propre stockage persistant et renvoyer les données vers C1 à l'aide d'une Sortie. Le processus fonctionnel PF C1 du composant C1 reçoit ces données comme une Entrée. Le processus fonctionnel PF C1 transmet ensuite le groupe de données comme une Sortie pour satisfaire la demande de son utilisateur fonctionnel.

Compte tenu de l'éventuel message d'erreur/confirmation émis par le client, la requête de l'exemple 2 requiert alors 6 mouvements de données (soit 6 PFC) pour satisfaire la demande du composant C1 et 4 PFC pour le composant C2. Ce qui est comparable aux 4 PFC (1 Entrée, 1 Lecture et 2 Sorties) qui auraient été requis pour le composant C1 s'il avait pu récupérer le groupe de données à partir du stockage persistant de son côté de la frontière à l'aide d'une Lecture, comme présenté sur la Figure 3.3.

Le composant C2 utilisera probablement les services d'un pilote de périphérique de stockage dans une autre couche de l'architecture logicielle afin de récupérer les données sur le périphérique, comme dans l'exemple 4 illustré sur la figure 3.5 (b).

Mouvement(s) de données et différents droits d'accès aux données stockées.

Exemple 3 : Voir la figure 3.4. Imaginons que le morceau de logiciel A à mesurer est autorisé à récupérer certaines données stockées Z, mais il n'est pas autorisé à les maintenir directement (créer, mettre à jour ou supprimer). Le morceau de logiciel B doit garantir l'intégrité des données Z en assurant une validation cohérente. Il traite alors toutes les requêtes de maintenance des données Z. Lorsque A doit maintenir les données Z, il doit transmettre sa requête à B via une Sortie, suivie d'une Entrée.

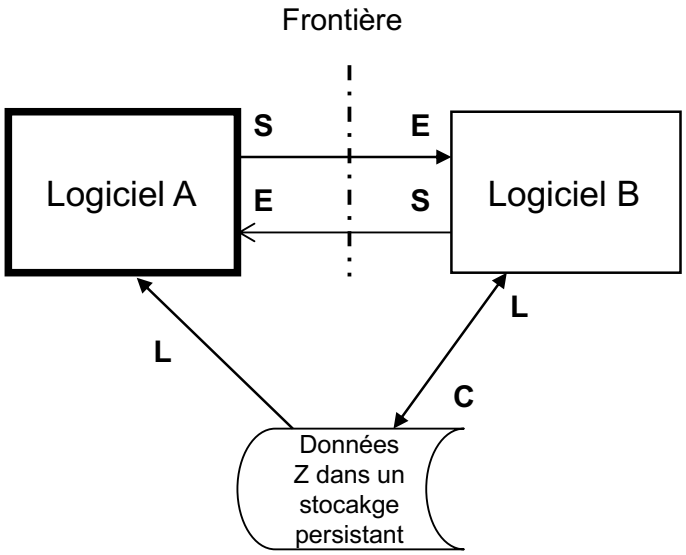


Figure 3.4 – Les données persistantes Z à travers les frontières des logiciels A et B pour une Lecture.

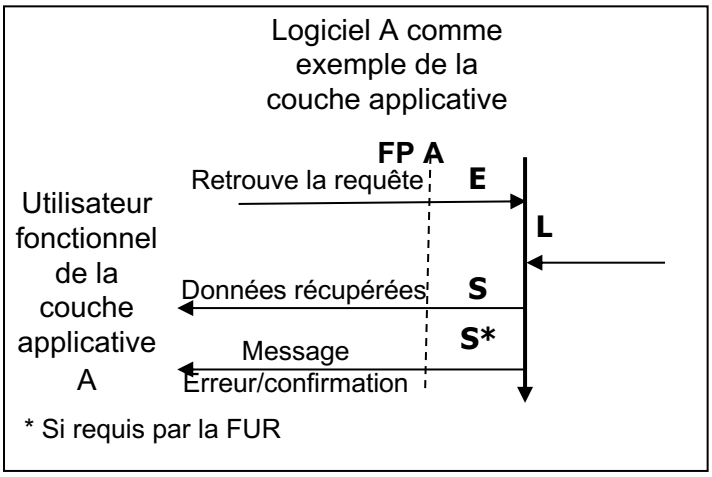
Mouvement(s) de données lorsque les données persistantes sont obtenues par le pilote qui interagit avec le périphérique de stockage.

Exemple d'infrastructure : Cet exemple concerne le morceau de logiciel A de l'exemple 1 qui doit récupérer un groupe de données stocké. Supposons qu'un morceau de logiciel B distinct est le pilote du périphérique de stockage intelligent contenant le groupe de données auquel le logiciel A doit accéder. (Pour simplifier, ignorer la présence probable d'un système d'exploitation, ce dernier transmet efficacement des requêtes d'application au pilote et renvoie les résultats des requêtes).

Les deux morceaux de logiciel se trouvent dans différentes couches d'une architecture comme celle présentée sur la figure 2.2. Le logiciel A se trouve, par exemple, dans la couche application, et le logiciel B dans la couche d'un pilote. Physiquement, il existe probablement une relation hiérarchique entre les deux morceaux et (toujours en ignorant le système d'exploitation) une interface physique entre les deux couches, comme présenté sur la Figure 2.2. Cependant, les modèles des processus fonctionnels des logiciels A et B ne dépendent pas de la nature de la relation entre les couches, qui peut être hiérarchique ou bidirectionnelle.

Les utilisateurs fonctionnels du logiciel B de la couche « pilote » sont le logiciel A (toujours en ignorant le système d'exploitation) et le périphérique de stockage intelligent contenant les données demandées. (Le terme « intelligent » signifie qu'il faut préciser au périphérique quelles sont les données requises.)

Supposons qu'un processus fonctionnel de requête PF A du logiciel A doive récupérer un groupe de données stocké. La Figure 3.5 (a) présente le modèle COSMIC de la requête. La Figure 3.5 (b) présente le processus fonctionnel PF B du logiciel B de la couche « pilote » qui gère l'extraction physique des données requises du périphérique de stockage (tel qu'un disque dur ou une clé USB).



Figures 3.5 (a) et (b) – Solution pour une lecture émise par le logiciel A dans la couche application vers le logiciel B dans la couche pilote de périphérique.

La Figure 3.5 (b) montre que la requête de Lecture du logiciel A est reçue en tant qu'Entrée de déclenchement du processus fonctionnel PF B, qui transmet la demande au périphérique de stockage en tant que Sortie. La réponse du périphérique dépend de la nature de celui-ci. Il se peut que le périphérique ne retourne que les données requises, représentées par l'Entrée E2 sur la Figure 3.5 b). Le périphérique peut également générer un message d'erreur séparé décrivant la réussite ou l'échec de la requête représentée par l'Entrée E1 sur la Figure 3.5 b). Par exemple, « Données introuvables » ou « Erreur disque ». Le processus fonctionnel B renvoie les données vers le logiciel A en tant que Sortie. Le processus fonctionnel B renvoie également un « code de retour » décrivant la réussite ou l'échec de la requête. (Bien que le code de retour puisse être physiquement attaché aux données renvoyées, il s'agit d'un groupe de données logiquement différent de ces dernières - ce groupe de données concerne les résultats du processus de requête). Pour un processus fonctionnel A, aucune Entrée pour ces messages n'est identifiée, car la Lecture prend en compte les données renvoyées et les messages d'erreurs. Les Lectures et les éCritures prennent en compte tous les rapports de conditions d'erreurs associés. Pour un processus fonctionnel PF A, une Sortie est identifiée pour un message d'erreur/confirmation, si demandé.

Remarque : En pratique, il peut y avoir plus de mouvements de données entre le pilote du périphérique et le périphérique de stockage intelligent qui sont présentés sur la Figure 3.5 b). Par exemple, cette figure ne montre pas l'impact du pilote de périphérique mesurant l'expiration du temps de réponse de l'appareil.

Mouvement(s) de données lorsqu'un processus fonctionnel n'a pas besoin de préciser à l'utilisateur fonctionnel quelles données envoyer.

Exemple temps réel 5 : Supposons qu'un processus fonctionnel d'un système logiciel de contrôle de traitement temps réel doive interroger un ensemble de capteurs classiques identiques. Au niveau de l'application, la requête de données par le processus fonctionnel et la réception des données sont représentées par une Entrée. (On identifie et compte une Entrée étant donné que les capteurs sont identiques).

Supposons que la requête de données doive, en pratique, être transmise à un morceau de pilote de périphérique dans une couche inférieure de l'architecture logicielle qui obtient physiquement les données requises provenant de l'ensemble des capteurs, comme illustré dans l'architecture en couches de la figure 2.3. Le processus fonctionnel du logiciel de contrôle de traitement et le pilote du périphérique des capteurs seront comme représenté sur la Figure 3.5 (a) et (b) ci-dessous.

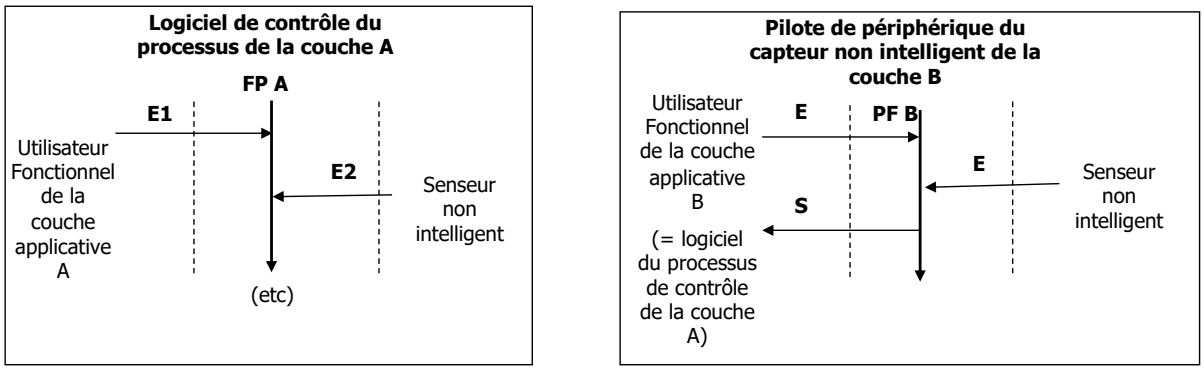


Figure 3.6 (a) et (b) – Solution pour le sondage des capteurs.

La figure 3.6 (a) montre que le processus fonctionnel PF A est déclenché par une Entrée E1, comme par exemple, le « tic d'une horloge ». Ce processus fonctionnel obtient alors des données via l'Entrée E2 provenant de l'ensemble des capteurs pour recevoir les nombreuses occurrences des lectures de capteurs. Ces capteurs sont également des utilisateurs fonctionnels du logiciel de contrôle de traitement. (Le pilote du périphérique est caché à ce niveau).

La Figure 6.3 (b) présente le modèle pour le logiciel qui pilote les capteurs. Il reçoit des données via une Entrée provenant du logiciel de contrôle de traitement (en pratique, probablement via un système d'exploitation) comme l'événement déclencheur du processus fonctionnel B. Ce processus fonctionnel obtient les données requises via une entrée E provenant de ses utilisateurs fonctionnels, c'est-à-dire, l'ensemble des capteurs.

Le groupe de données est renvoyé vers le logiciel de contrôle en tant que Sortie. Cette Sortie est reçue sous la forme de l'Entrée E2 par le processus fonctionnel PF A. Ce dernier continue alors de traiter les données des capteurs. Une fois de plus, le fait qu'il y ait plusieurs occurrences de ce cycle de collecte de données provenant de chacun de ces capteurs identiques importe peu.

L'incompatibilité apparente entre l'Entrée E2 provenant d'un capteur à destination du logiciel de contrôle et l'Entrée suivie d'une Sortie de données du pilote de périphérique est due à la convention selon laquelle une Entrée d'un capteur classique est supposée inclure toute fonctionnalité de « requête pour entrer » puisque l'utilisateur fonctionnel muet n'a pas la capacité de traiter un message d'un processus fonctionnel.

Mouvement(s) de données lorsqu'il faut préciser à l'utilisateur fonctionnel ce qu'il doit envoyer.

Exemple temps réel 6 : Supposons qu'un processus fonctionnel envoie à l'un de ses utilisateurs fonctionnels (un périphérique intelligent ou un morceau logiciel de même niveau) des paramètres pour une requête, les paramètres d'un calcul ou des données à compresser. La réponse de l'utilisateur fonctionnel est obtenue via le processus fonctionnel émettant une Sortie, suivie par la réception d'une Entrée, comme décrit dans la section 3.4, exemple 2.

3.5 Manipulations de données associées à des mouvements de données.

Une manipulation de données est considérée comme prise en compte par les mouvements de données associés. Pour la manipulation de données devant être mesurées dans le contexte de modifications apportées aux exigences, voir 3.10.

Exemple métier 1 : Une Entrée inclut toutes les manipulations nécessaires au formatage d'un écran permettant à un utilisateur humain de saisir et de valider des données, à l'exception de toute(s) Lecture(s) pouvant être requise(s) pour valider des données ou codes saisis, ou pour obtenir les descriptions de code associées.

Exemple métier 2 : Une Sortie inclut toutes les manipulations permettant le formatage du résultat et la préparation d'attributs de données pour l'impression (ou l'affichage sur écran), y compris les titres des champs lisibles par un humain[3], à l'exception de toute Lecture ou Entrée pouvant être requise pour fournir les valeurs ou les descriptions de certains des attributs de données imprimés.

Exemple : Imaginons deux occurrences d'un processus fonctionnel donné. Supposons que lors de la première occurrence, les valeurs de certains attributs à transférer à l'aide d'un mouvement de données prédéfini mènent au sous-processus de manipulation de données A et que, lors d'une autre occurrence du même processus fonctionnel, les valeurs des attributs engendrent un sous-processus de manipulation de données B. Dans de telles circonstances, les deux sous-processus de manipulations de données A et B sont associés au même mouvement de données. Voilà pourquoi seul le mouvement de données doit être identifié et compté dans le cas de ce processus fonctionnel.

3.6 Commandes de contrôle.

Selon la méthode COSMIC, les commandes de contrôle ne sont pas des mouvements de données. Il est important d'identifier les commandes de contrôle pour ne pas les confondre avec des mouvements de données.

Exemples de commandes de contrôle.

- Les commandes de défilement de pages (de haut en bas ou changement d'écrans physiques).
- Appuyer sur Tab, Entrée ou un bouton pour continuer.
- Cliquer sur un bouton OK pour confirmer ou annuler une action, ou pour accuser réception d'un message d'erreur ou confirmer la saisie de données, etc.
- Les fonctions qui autorisent un utilisateur à contrôler l'affichage (ou non) d'un titre ou de sous-totaux qui ont été calculés.
- Les commandes de menu qui autorisent les utilisateurs à naviguer vers au moins un processus fonctionnel spécifique mais qui ne déclenchent pas eux-mêmes de processus fonctionnels.
- Commandes pour afficher un écran blanc pour la saisie de données.

3.7 Messages d'erreur/confirmation et autres indicateurs de conditions d'erreur.

Les messages d'erreur et de confirmation, si demandés, sont pris en compte dans le mesurage.

[3] Cet exemple s'applique lors du mesurage d'applications logicielles à usage humain, indépendamment du domaine. Il ne s'applique évidemment pas lors du mesurage de la taille d'objets réutilisables sur lesquels reposent l'affichage de titres de champs individuels sur des écrans d'entrée ou de sortie.

Exemples : Les messages d'erreur ou de confirmation peuvent transmettre le succès ou l'échec de la validation de la saisie de données ou d'une requête de récupération de données ou d'une conversion en données persistantes, ou de la réponse d'un service rendu par un autre morceau de logiciel.

Exemple métier 1 : Dans un dialogue humain-ordinateur, des messages d'erreur apparaissant lors de la validation de données en cours de saisie peuvent être les suivants : « Mauvais format », « Client inconnu », « Erreur : merci de cocher la case indiquant que vous avez pris connaissance de nos conditions générales d'utilisation », « Vous n'avez plus de crédit. », etc. Tous ces messages d'erreur doivent être pris en compte comme étant plusieurs occurrences d'une même Sortie de chaque processus fonctionnel contenant des messages du genre.

Exemple métier 2 : Le processus fonctionnel A peut potentiellement émettre 2 messages de confirmation distincts et 5 messages d'erreur à l'attention de ses utilisateurs fonctionnels. Identifier une Sortie pour compter tous ces messages d'erreur (5+2=7). Le processus fonctionnel B peut potentiellement émettre 8 messages d'erreur à l'attention de ses utilisateurs fonctionnels. Identifier une Sortie pour compter ces 8 messages d'erreur.

Un message d'erreur/confirmation avec des données additionnelles.

Exemple métier 3 : Un processus fonctionnel d'un DAB (distributeur automatique de billets) peut émettre cinq types de messages en réponse à une requête pour retirer une certaine somme d'argent.

- Erreur : le distributeur n'a pas de billets.
- Erreur : le montant demandé doit être un multiple de 10.
- Retrait refusé. Compte bloqué. Contactez votre banque.
- Retrait refusé (la limite de retrait est dépassée de 139,14 €).
- Retrait accepté, le nouveau solde de votre compte est de 756,25 €.

Les quatre premiers messages décrivent une condition d'erreur et la première partie du cinquième message est une confirmation. D'après la règle sur les indications des conditions d'erreurs, il faut compter une Sortie pour tous ces messages. Les deux derniers messages comprennent également les attributs de données liés au compte client. Compter une Sortie pour ces données, pour tenir compte du mouvement des données du compte client. Au total, identifier deux Sorties pour ce processus fonctionnel, autrement dit, 2 PFC pour cette sortie.

Conditions d'erreur pour les utilisateurs humains qui ne sont pas spécifiés dans les EFU.

Exemple métier 4 : Un message provenant du système d'exploitation pourrait être « l'imprimante X ne répond pas », ignorer ce genre de message.

Exemple temps réel 1 : Dans un système temps réel, un processus fonctionnel qui vérifie périodiquement le bon fonctionnement de tous les équipements peut émettre le message suivant : « Le capteur S ne fonctionne pas », S étant une variable. Ce message devrait être identifié comme étant une Sortie de ce processus fonctionnel pour prendre en compte les données transférées concernant l'utilisateur fonctionnel (qui est aussi un objet d'intérêt) Capteur S.

Valeurs de données indiquant une condition d'erreur avec d'autres données.

Exemple temps réel 2 : Les EFU de l'exemple temps réel 6 de la section 3.4 peuvent également stipuler que les processus fonctionnels A et B doivent gérer une condition d'erreur lorsque le pilote de périphérique n'arrive pas à obtenir les données d'au moins un des capteurs. Un capteur classique ne peut pas, par définition, émettre un message d'erreur. Le pilote de périphérique PF B obtiendra certainement une série de valeurs provenant de

l’ensemble des capteurs. Par exemple, état 1, état 2, état 3, pas de réponse, état 5, pas de réponse, état 7, etc. Il va émettre ces valeurs sous la forme d'une Sortie vers le processus fonctionnel PF A de l’application où elle est reçue comme une Entrée. Aucun message d’erreur séparé ne doit être compté comme une Sortie provenant du processus fonctionnel PF B du pilote de périphérique, ni comme une Entrée du processus fonctionnel A de l’application de contrôle de traitement.

3.8 Mesurage des composants d'un système logiciel distribué.

Voir l’exemple 2 de la section 3.4.

3.9 Réutilisation de logiciel.

Processus fonctionnel qui inclut des fonctionnalités communes.

Exemple métier 1 : Plusieurs processus fonctionnels du même logiciel à mesurer peuvent nécessiter la même fonctionnalité de validation, par exemple, « date de commande », ou peut avoir besoin d’accéder aux mêmes données persistantes, ou d’effectuer le même calcul d'intérêts. Mesurer la fonctionnalité commune dans le cadre de chaque processus fonctionnel qui le nécessite.

Exemple temps réel 1 : Plusieurs processus fonctionnels du même logiciel mesuré peuvent nécessiter d’obtenir des données provenant du même capteur (mouvement commun du même groupe de données) ou d’effectuer le même calcul de conversion d’échelle, comme par exemple, de Fahrenheit à Celsius (manipulation de données commune). Mesurer cette fonctionnalité commune comme dans l’exemple précédent.

Fonctionnalité qui n’apparait pas dans les EFU.

Exemple métier 2 : Dans une application métier, l'utilisateur qui démarre l’application peut être un composant ordonnanceur du système d’exploitation, un opérateur de l’ordinateur ou tout autre utilisateur humain (par exemple, lorsqu’un utilisateur d’un PC ouvre un navigateur ou un logiciel de traitement de texte). Les données transférées par ces utilisateurs ne sont pas traitées par l’application comme défini dans les EFU, elles doivent donc être ignorées.

Exemple temps réel 2 : Pour les applications temps réel, l’utilisateur qui démarre l’application peut être le système d’exploitation ou le gestionnaire de réseau générant un signal d’horloge, ou un opérateur humain (par exemple, pour démarrer un système de contrôle de processus du poste de travail d'un opérateur). Cet utilisateur doit être ignoré dans le mesurage.

Exemple d’infrastructure : Pour le démarrage d’un système d’exploitation d'ordinateur, l'utilisateur, un programme d'amorçage qui est lancé lorsque l'ordinateur est mis sous tension, doit être ignoré dans le mesurage.

Exemple métier 3 : Une application « batch » permettant de traiter les données entrées pour une variété de processus fonctionnels en lots peut être démarré par un ordonnanceur de système d’exploitation. Si l’objectif est de mesurer les EFU de l’application batch, la fonctionnalité « démarrer le système » doit être ignorée. Les Entrées de déclenchement des processus fonctionnels de l’application batch et toutes les autres Entrées pouvant être requises constitueront les données d’entrée pour l’application batch.

Exemple métier 4 : Exceptionnellement, une application de traitement par lots pour produire des rapports sommaires à la fin d’une période de temps peut être démarrée sans nécessiter aucune donnée d'entrée fournie directement par l'utilisateur fonctionnel. Pour l’analyse, voir l’exemple métier 7 de la section 3.1.

Exemple temps réel 3 : Un véhicule moderne est doté d'un système distribué d’unités de commande électronique (UCE) pour contrôler différentes fonctionnalités, comme par exemple, le moteur, les freins, la climatisation, etc. Dans l’architecture d’AUTOSAR, dans un

système distribué, le module de « gestion réseau », qui tourne en permanence, est en charge de l'activation des UCE reliées ensemble par un réseau (« bus »). Le module de « gestion réseau » gère également la commutation coordonnée entre les états de fonctionnement des UCE : fonctionnement normale, alimentation faible et veille. C'est donc le module « gestion réseau » qui met en marche ou en veille les UCE. Lors du mesurage d'un logiciel d'UCE, la fonctionnalité de « gestion réseau » doit être ignorée.

3.10 Mesurage des modifications apportées à un logiciel.

Avec la méthode COSMIC, la taille des changements comprend les modifications apportées à tout élément contribuant à la taille, y compris les manipulation de données des mouvements de données.

Exemple 1 : Une manipulation de données est modifiée par exemple en modifiant le calcul, le formatage spécifique, la présentation et/ou la validation des données. « Présentation » peut faire référence, par exemple, à la police de caractère, la couleur d'arrière-plan, la longueur du champ, l'en-tête du champ, le nombre de décimales, etc.

Exemple 2 : Supposons qu'une demande de modification pour un processus fonctionnel requiert trois changements dans la manipulation des données associées à son Entrée de déclenchement et deux changements de la manipulation associée à une Sortie, ainsi que deux changements d'attributs du groupe de données déplacé par cette Sortie. Compter 2 PFC pour la taille des modifications, autrement dit, compter le nombre total de mouvements de données dont les attributs et les manipulations de données associées doivent être modifiés. Ne surtout pas compter le nombre de manipulations ou d'attributs de données devant être modifiés.

Exemple 3 : Supposons qu'une exigence impose d'ajouter ou de modifier les attributs de données d'un groupe de données D1 de sorte qu'après modification, il devienne D2. Dans le processus fonctionnel A, où cette modification est requise, tous les mouvements de données impactés par la modification devraient être identifiés et comptés comme étant modifiés. Ainsi, si le groupe de données modifié D2 est rendu persistant et/ou est sorti par le processus fonctionnel A, identifiez respectivement un mouvement de données d'Ecriture et / ou de Sortie comme modifié.

Cependant, il est possible que d'autres processus fonctionnels lisent ou entrent ce même groupe de données D2, mais leurs fonctionnalités ne sont pas impactées par les modifications car elles ne traitent pas les attributs de données modifiés ou ajoutés. Ces processus fonctionnels continuent de traiter les groupes de données déplacé comme s'il était encore D1. Ainsi, les mouvements de données dans les autres processus fonctionnels qui ne sont pas affectés par la modification du (des) mouvements de données du processus fonctionnel A ne doivent PAS être comptés comme modifiés.

Exemple 4 : Souvent, une partie obsolète d'une application est supprimée (ou plutôt « déconnectée ») en laissant le code du programme en place et en retirant simplement le lien vers la fonctionnalité obsolète. Supposons que la fonctionnalité du morceau obsolète soit de 100 PFC, et que le morceau peut être déconnecté en changeant, disons, deux mouvements de données. La taille des modifications fonctionnelles dépend de l'objectif du mesurage. Si le but est d'utiliser la taille pour estimer le projet, alors la taille est de 2 PFC. Si le but est de déterminer la taille globale de l'application, la taille de la modification est alors de 100 PFC.

À des fins d'estimation, il est recommandé d'utiliser une productivité différente pour une partie des modifications fonctionnelles, étant donné que la « déconnexion » de fonctionnalités est assez différente d'une « vraie » suppression. Alternativement*, toujours pour des fins d'estimation, il est préférable de mesurer la taille qui sera mise en œuvre, plutôt que la taille de l'exigence.*

Exemple 5 : Si dans l'exemple précédent la « taille du projet » de 2 PFC est mesurée, elle doit être clairement documentée et différenciée d'un mesurage des EFU qui requiert que la taille de l'application soit réduite de 100 PFC.

Exemple métier 1 : Si un message d'erreur/confirmation doit être modifié (ajout, modification ou suppression de texte), il doit être identifié pour le mesurage, peu importe si le texte modifié est la conséquence d'une exigence de modification d'un autre mouvement de données.

Modification des commandes de contrôle et données générales de l'application.

Exemple métier 2 : Lorsque la couleur de tous les écrans est modifiée, cette modification ne doit pas être mesurée.

3.11 Extension de la méthode COSMIC.

Bien que cela ne soit pas formellement expliqué dans la méthode COSMIC, tout aspects des EFU ne pouvant être mesurés à l'aide de la méthode COSMIC peuvent faire l'objet d'un processus de mesurage séparé. Cependant, la taille ne doit pas être exprimée en PFC.

Exemple 1 : La méthode COSMIC peut être étendue pour mesurer séparément et explicitement la taille des EFU des sous-processus de manipulation de données.

Exemple 2 : La méthode peut être étendue pour mesurer séparément l'influence du nombre d'attributs de données par mouvements de données sur la taille du logiciel.