ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

THÈSE PRÉSENTÉE À
L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
L'UNIVERSITÉ DE VERSAILLES SAINT-QUENTIN-EN-YVELINES
(COTUTELLE)

COMME EXIGENCE PARTIELLE
À L'OBTENTION DU
DOCTORAT EN GÉNIE
Ph.D.

COTUTELLE FRANCE-QUÉBEC

PAR
Hassan SOUBRA

AUTOMATISATION DE LA MESURE FONCTIONNELLE COSMIC -ISO 19761 DES LOGICIELS TEMPS-RÉEL EMBARQUÉS, EN SE BASANT SUR LEURS SPÉCIFICATIONS FONCTIONNELLES

MONTRÉAL, LE 23 NOVEMBRE 2011

PRÉSENTATION DU JURY

CETTE THÈSE A ÉTÉ ÉVALUÉE

PAR UN JURY COMPOSÉ DE :

M. Alain Abran, directeur de thèse
Professeur au Département de génie Logiciel à l'École de technologie supérieure

M. Amar Ramdane-Cherif, codirecteur de thèse
Professeur des Universités à l'Université De Versailles Saint-Quentin-En-Yvelines

Mme Sophie Stern, membre du jury
Responsable scientifique thèse, Renault SAS

M. Chakib Tadj, président du jury
Professeur au Département de génie Électrique à l'École de technologie supérieure

M. Alain April, rapporteur
Professeur au Département de génie Logiciel à l'École de technologie supérieure

M. Kamel Barkaoui, rapporteur
Professeur des Universités au Conservatoire National des Arts et Métiers de Paris

Mme Nicole Levy, examinatrice externe
Professeure des Universités au Conservatoire National des Arts et Métiers de Paris

ELLE A FAIT L'OBJET D'UNE SOUTENANCE DEVANT JURY

LE 25 OCTOBRE 2011

À L'UNIVERSITÉ DE VERSAILLES SAINT-QUENTIN-EN-YVELINES

AUTOMATISATION DE LA MESURE FONCTIONNELLE COSMIC -ISO 19761 DES LOGICIELS TEMPS-RÉEL EMBARQUÉS, EN SE BASANT SUR LEURS SPÉCIFICATIONS FONCTIONNELLES

Hassan SOUBRA

RÉSUMÉ

De plus en plus de prestations véhicule sont réalisées par du logiciel embarqué, notamment celles liées au véhicule électrique, au multimédia et aux aides à la conduite. Piloter les coûts de développement de ces logiciels embarqués en augmentation est un enjeu majeur pour garantir la compétitivité de Renault. L'objectif du projet industriel dans lequel ce projet de recherche s'inscrit est de piloter ces coûts efficacement.

Le premier problème industriel à résoudre est l'estimation des coûts de développement des futurs logiciels en ayant seulement leurs spécifications fonctionnelles modélisées avec un langage/outil de modélisation donné. Renault SAS, avec son propre processus d'estimation a priori, entre autre, aura des benchmarks pour pouvoir négocier et choisir les futurs fournisseurs auxquels il va confier le développement de ces produits logiciels. Suite à un appel d'offre et aux réponses des différents fournisseurs, le constructeur pourra chercher et demander des explications on se basant sur des mesures et des benchmarks concrets.

Un processus d'estimation basé sur l'utilisation de modèles de simulation de productivité de développement logiciel permet ce pilotage des coûts. La création d'un modèle de productivité repose d'une part sur la mesure d'un attribut du logiciel, sa taille fonctionnelle, et d'autre part sur les données d'effort transmises par le fournisseur.

Le but principal de ce projet de recherche est l'automatisation de la mesure de la taille fonctionnelle, selon la méthode COSMIC ISO 19761, des logiciels des systèmes réactifs, temps-réel et embarqués en utilisant leurs spécifications fonctionnelles. Le but est donc d'identifier les parties automatisables dans le processus de mesurage de la taille fonctionnelle de ces logiciels, et de concevoir et prototyper un outil qui les automatise pour accélérer la mesure (processus qui consomme beaucoup de temps et de ressources s'il est fait manuellement, sans oublier le facteur 'humain' d'erreur) afin que l'utilisation de cette méthode en industrie soit exploitable et efficace.

Ces tailles fonctionnelles seront utilisées avec l'historique de l'effort des fournisseurs dans les développements antérieurs et les modèles de productivité correspondant pour estimer les efforts de développement des nouveaux logiciels et des évolutions des logiciels.

Il est impossible d'automatiser une méthode de mesure de la taille fonctionnelle directement parce qu'une méthode de mesure définit des concepts et des descriptions génériques. Il faut créer une procédure de mesurage pour chaque modèle-cible en entrée à un processus de mesurage. Une procédure de mesurage est définie comme un ensemble d'opérations décrites explicitement pour permettre de mesurer suivant les principes d'une méthode de mesure

donnée. En conséquence, pour pouvoir atteindre le but principal de ce projet de recherche, deux objectifs ont été définis. Le premier objectif est le design des procédures de mesurages basées sur le standard COSMIC ISO 19761, et le deuxième est le développement d'un prototype qui implémente ces procédures.

L'atteinte du premier objectif s'est faite en deux étapes. La première étape était l'analyse et la compréhension des modèles des outils de modélisations utilisés chez Renault, qui sont Statemate et Simulink. La deuxième étape était la création et la documentation des deux procédures de mesurages, une pour les logiciels dont les spécifications fonctionnelles sont modélisées avec Statemate, et une procédure pour les logiciels dont les spécifications fonctionnelles sont modélisées avec Simulink.

La première étape de l'ensemble des étapes nécessaires pour atteindre le deuxième objectif, était la création d'une représentation semi-formelle qui a comme but principal l'extraction des informations nécessaires et suffisantes pour pouvoir mesurer le logiciel en suivant les principes de la méthode COSMIC ISO 19761. Cette représentation semi-formelle peut être produite manuellement lors de l'application manuelle des procédures de mesurages, mais sa production manuelle consomme d'avantage d'effort et de ressources. Pour l'automatisation, toutefois, la production de cette représentation est très importante et nécessaire pour la phase de mesurage qui produit la taille finale en unités internationales de mesure CFP. La deuxième étape était le design et le développement du prototype de mesurage qui implémente la procédure de mesurage proposée pour les spécifications fonctionnelles des logiciels modélisées avec Simulink. Ce prototype a été utilisé chez Renault. Une version industrielle a aussi été développée. Un dépôt de demande de brevet a été fait.

Et bien sur, le prototype a été testé avant son utilisation pour s'assurer que les résultats de la mesure faite par le prototype sont conformes au standard COSMIC ISO 19761. L'évaluation du prototype a été faite en appliquant un protocole d'évaluation proposé dans le cadre de ce projet de recherche pour les outils de mesure automatisés qui sont basés sur le standard COSMIC et en utilisant un ensemble de spécifications fonctionnelles de Renault.

Mots clés : Taille fonctionnelle, systèmes temps-réel, automatisation, procédures de mesurage, prototype d'automatisation, Statemate, Simulink, COSMIC ISO 19761.

AUTOMATISATION DE LA MESURE FONCTIONNELLE COSMIC -ISO 19761 DES LOGICIELS TEMPS-RÉEL EMBARQUÉS, EN SE BASANT SUR LEURS SPÉCIFICATIONS FONCTIONNELLES

Hassan SOUBRA

ABSTRACT

More and more automotive services are designed using embedded software, including those related to electrical vehicles as well as multimedia and driving aids. Managing the development costs of embedded software is a major challenge to ensure the competitiveness of Renault. The thesis is part of an industry research project to improve the management of these software development costs.

An estimation process based on the use of software development productivity models allows the management of development costs. The creation of a productivity model is based on the measurement of an attribute of the software, its functional size, and on the effort data provided by the supplier(s).

The industrial problem to be addressed first is the cost estimation of future software development using their functional requirements modeled with a specific language or modeling tool. By measuring the functional size of its functional requirements using the COSMIC ISO 19761 ISO standard, Renault SAS is able to benchmark, select suppliers to whom it will entrust the development of its software products and negotiate specific development projects.

The main purpose of this research project is the automation of the measurement of functional size of reactive, real-time and embedded systems software using their functional requirements, and according to the ISO 19761 COSMIC method. Hence, this requires to identify the parts in the measurement process that can be automated, and then to design and prototype a tool to automate those parts to speed up the measurement process (which consumes a lot of time and resources if done manually, not to mention the 'human' error factor) so that the use of this method in the industry is feasible and effective.

These functional sizes are combined with project efforts to produce the suppliers' productivity models which are used next for estimation purposes.

It is challenging to automate a functional size method because measurement methods define generic concepts and measurement rules. The set up of a measurement procedure for each input to the measurement process is hence needed. A measurement procedure is defined as a set of operations described explicitly in order to measure according to a specific measurement method. Therefore, in order to achieve the goal of this research project, two research objectives were selected. The first research objective is the design of measurement procedures based on the COSMIC ISO 19761 standard, and the second research objective is the development of a prototype that implements these procedures.

The first research objective was achieved in two steps:

1. The analysis of the models of the two modeling tools used at Renault (i.e. Simulink and Statemate)
2. The design of the two measurement procedures, one for the functional requirements modeled with Statemate, and another for Simulink.

The second research objective required a number of steps, including first the design of a semi-formal representation for the extraction of necessary and sufficient information to be able to measure software by following the principles of the COSMIC ISO 19761 method. This semi-formal representation can be generated manually throughout the manual application of the measurement procedures, but its manual production is time consuming.

Another step of the second research objective was the design and the development of a prototype that implements the measurement procedure proposed for the measurement of software functional requirements modeled with Simulink. This research prototype was tested to ensure that the results of the measurement made by the prototype were in accordance with the COSMIC ISO 19761 standard. The prototype was evaluated with an evaluation protocol proposed in the context of this research project for the automation tools which are based on the COSMIC standard and by using a set of functional requirements of Renault. An industrial version of this prototype-tool was also developed and is now in use at Renault; a patent request was submitted.

Keywords: Functional size measurement, real-time systems, estimation, measurement procedure, automation tool, Statemate, Simulink, COSMIC ISO 19761.

TABLE DES MATIÈRES

Page

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

Page

INTRODUCTION

Beaucoup de constructeurs industriels dans les domaines de l'aéronautique, des télécommunications et de l'automobile, - comme Renault SAS, l'entreprise chez laquelle ce projet de recherche a eu lieu - font développer leurs produits finaux chez des tiers externes, les fournisseurs. Dans le monde du temps-réel, le logiciel prend une place de plus en plus importante dans ces systèmes et désormais le logiciel est développé séparément du matériel. Ceci permet aux constructeurs industriels de faire développer leurs modules logiciels chez des fournisseurs différents, et donc ces constructeurs ne sont plus dans l'obligation de confier tous leurs logiciels aux mêmes fournisseurs.

La problématique de la gestion de l'impartition des produits logiciels, notamment les logiciels temps-réel embarqués, a des impacts économiques importants pour les constructeurs et spécifiquement pour le constructeur automobile partenaire de ce projet de recherche. Un important problème à résoudre pour le partenaire industriel est l'estimation des coûts de développement des futurs composants logiciels à partir de leurs exigences fonctionnelles logicielles spécifiées et documentées avec un outil-logiciel de modélisation.

Renault SAS, avec son propre processus d'estimation a priori, entre autre, aura des benchmarks pour pouvoir négocier et choisir les futurs fournisseurs auxquels il va confier le développement de ces produits logiciels. Suite à un appel d'offre et aux réponses des différents fournisseurs, le constructeur pourra chercher et demander des explications en se basant sur des mesures et des benchmarks concrets.

Bien entendu, le processus d'estimation doit couvrir non seulement les nouveaux logiciels à développer, mais aussi les «carry-overs» (logiciels recyclés : réutilisable après modification(s)) et les nouvelles versions (évolutions) des logiciels déjà existants.

Une des retombées finales prévues de ce projet de recherche est donc de pouvoir permettre aux constructeurs de gérer l'impartition de leurs produits logiciels (composants ou logiciels)

des systèmes temps-réel embarqués en leur permettant de suivre et d'évaluer la productivité de leurs fournisseurs, et donc d'améliorer leurs performances économiques. Pour arriver à cela, il faut pouvoir faire une estimation a priori des coûts des développements des nouveaux logiciels et des évolutions des logiciels des systèmes temps-réel réactifs embarqués via un processus d'estimation.

Le cœur de ce processus d'estimation est un modèle de productivité. Le modèle de productivité est un modèle de simulation qui a comme but d'utiliser l'historique des projets antécédents et de donner des statistiques en se basant sur les données de l'effort du fournisseur et sur la taille fonctionnelle des logiciels développés précédemment (i.e. l'historique).

Ce projet s'inscrit dans un contexte idéal pour pouvoir répondre aux attentes du constructeur partenaire de ce projet de recherche quant à la gestion de leurs performances économiques.

La problématique à laquelle doit répondre ce projet de recherche ce divise en deux sous-problématiques :

1. La première est la mesure de la taille fonctionnelle, suivant la méthode COSMIC ISO 19761, des logiciels temps-réel embarqués d'une manière répétable et reproductible. Cette taille doit être mesurée à partir des spécifications fonctionnelles des logiciels spécifiées et documentées avec les outils de modélisations utilisés chez Renault.

2. La deuxième sous-problématique est l'étude de la faisabilité de l'automatisation de la mesure de la taille fonctionnelle COSMIC de ses logiciels et le développent d'un prototype qui constituera la base d'un outil industriel. L'automatisation de cette mesure a comme but de remplacer la mesure manuelle qui a plusieurs désavantages.

CHAPITRE 1

REVUE DE LA LITTÉRATURE

1.1 Introduction

L'étude de la littérature porte sur la mesure de la taille fonctionnelle des logiciels et surtout des logiciels temps-réels embarqués. Comme la problématique à laquelle ce projet de recherche doit répondre est l'automatisation de procédures de mesure, l'étude bibliographique couvrira principalement ce thème. Les procédures de mesure de la taille fonctionnelle proposées dans la littérature seront détaillées tandis que seulement des généralités sur la mesure logicielle et sur les modèles d'estimations seront présentées.

1.2 La mesure du logiciel en général

La mesure du logiciel est un sujet de recherche dont les premiers travaux datent des années 1960s [54]. Depuis cette époque, plusieurs modèles d'estimation ont été proposés en se basant sur la mesure de différentes caractéristiques du logiciel : lignes de codes, complexité, taille fonctionnelle, etc.

La mesure du logiciel peut être faite pour divers buts comme le soulignent McGarry et al. dans [41]: l'estimation, l'analyse de la faisabilité, l'analyse de la performance, etc.

Le processus de mesure du logiciel est décrit dans le standard international ISO 15939 [25] en termes d'activités et de tâches : l'engagement et le support de la mesure, la planification du processus de mesure, l'exécution du processus et l'évaluation de la mesure. Le standard ISO 15939 présente aussi un modèle d'information de la mesure ('measurement information model'), qui montre que la mesure doit répondre à des demandes d'information bien précises (buts de la mesure).

Dans le contexte du projet industriel dans lequel cette thèse s'inscrit, Renault SA cherche à utiliser les résultats des mesures principalement pour construire les modèles de simulation de productivité qui seront utilisés dans son processus d'estimation.

1.3 Modèles d'estimations

Plusieurs modèles d'estimation ont été proposés dans la littérature. Dans son livre [53], Marvin V. Zelkowitz classe les modèles de la façon suivante : modèles empiriques paramétriques, modèles empiriques non-paramétriques, modèles analogiques, modèles

théoriques et des modèles heuristiques. Dans la section 5 de son livre [53], Zelkowitz présente plusieurs modèles et montre les détails de certains modèles.

Un exemple de modèle d'estimation est COCOMO (Constructive Cost Model), proposé par Barry Boehm dans [8] en 1981 : ce modèle a subi une évolution et sa nouvelle version, COCOMO II, a été publiée en 2000 dans [9].

On se limitera à ce niveau de détails quant aux modèles d'estimation, et on exposera plus en détail les moyens, les méthodes ainsi que les éléments des modèles de productivité, bases des modèles d'estimation.

Pour pouvoir faire un modèle de productivité pour un type de développement donné, on doit avoir « mesuré » les produits déjà développés (on choisissant une des caractéristiques mesurables).

1.4 Mesure fonctionnelle

Une des caractéristiques mesurables/mesurées est la fonctionnalité du logiciel: cette caractéristique a été le sujet d'une étude par Allan Albrecht en 1979 [1]. Cette étude est la racine de la mesure de la taille fonctionnelle des logiciels, le cœur de ce projet de recherche.

Il existe actuellement cinq méthodes de mesure de la taille fonctionnelle du logiciel approuvées par l'Organisation internationale de normalisation (ISO):

- COSMIC - ISO/IEC 19761: 2011 Software Engineering - COSMIC - A Functional Size Measurement Method [27].

- IFPUG - ISO/IEC 20926: 2009 Software Engineering - Function Point Counting Practices Manual [26].

- MKII - ISO/IEC 20968: 2002 Information Technology - Software Engineering - Mk Ii Function Pont Analysis - Counting Practices Manual:2002 [21].

- NESMA - ISO/IEC 24570: 2005 Information Technology - Definitions And Counting Guidelines For The Application Of Function Point Analysis [22].

- FISMA - ISO/IEC 29881:2008 Information technology - Software and systems engineering - FiSMA 1.1 functional size measurement method [23].

La première méthode de mesure de la taille fonctionnelle publiée est la méthode « IFPUG », maintenant sous le standard ISO/IEC 20926:2009. D'après son abstract (présenté à l'ISO), ce standard fournit la description claire et détaillée de calcul de point de fonction. Il fournit aussi une base pour assurer que les calculs sont cohérents, ainsi que des conseils pour permettre de faire le comptage des points de fonction des exigences des utilisateurs fonctionnels à partir des livrables des méthodologies et des techniques de développement logicielles populaires. Et finalement, toujours d'après son abstract, ce standard présente un framework pour permettre l'automatisation du comptage des points de fonction. Toutefois, d'après nos recherches à ce jour, aucun outil ou prototype d'automatisation de cette méthode n'a été publié. D'autre coté, une étude [13] publiée par Diab et al. présente uniquement une formalisation des règles d'application de la méthode IFPUG pour la mesure des logiciels spécifiés avec la méthode B. Elle prétende aussi que cette formalisation peut être utilisée pour l'automatisation d'IFPUG pour les spécifications conçues avec la méthode B, mais cette étude ne le démontre pas. Finalement les auteurs de cet article présentent des imperfections dans la définition d'IFPUG FP et proposent des modifications pour la compléter.

Des recherches supplémentaires sur d'autres travaux concernant l'automatisation de l'application de la méthode IFPUG sont en cours de réalisation. D'après les auteurs du standard IFPUG, ce standard international peut être appliqué par n'importe quelle personne qui souhaite utiliser les résultats de la mesure des points de fonction pour des analyses et des statistiques. Il a été aussi conçu pour être utilisé par des personnes débutantes (sans expérience dans le domaine de mesure) aussi bien que par celles avec une expérience intermédiaire et avancée.

La deuxième méthode standardisée, considérée comme la méthode de la deuxième génération de mesure [52] [27] de la taille fonctionnelle : la méthode COSMIC - ISO 19761:2011. Ce standard spécifie l'ensemble de définitions, les conventions et les phases de la méthode

COSMIC. D'après le standard, cette méthode est applicable aux logiciels des domaines d'application suivants :

1. Applications d'administration : système d'information de gestion ou système de pilotage;
2. Applications temps-réel, dont la mission est de suivre ou de contrôler des événements dans le monde réel;
3. Applications hybrides de ce qui précède.

Le standard ISO 19761 précise que la méthode COSMIC n'a pas été conçue pour mesurer la taille fonctionnelle des logiciels qui sont caractérisés par des algorithmes mathématiques complexes et des règles complexes (exemple : logiciels de simulation, d'auto-apprentissage et des systèmes de prévisions météorologiques..) et les logiciels qui «manipulent» des variables continues (signaux audio, images vidéo…) qu'on retrouve dans ces types de logiciels : jeux, instruments de musique et ainsi de suite. Toutefois, ISO 19761:2003 contient des dispositions pour la personnalisation de la méthode : dans l'environnement d'une organisation utilisant la méthode COSMIC pour la mesure de la taille fonctionnelle, il est possible de « créer » des extensions locales de mesure (i.e. des normes locales).

Plusieurs études sur les deux méthodes-standards ont été menées sur des thèmes différents. Une étude a été faite par Serge Oligny et Alain Abran sur la compatibilité entre les deux méthodes [46]. Cet article étudie la question de la compatibilité dans le design de la mesure des méthodes COSMIC et IFPUG. La compatibilité entre les deux méthodes est absolument obligatoire pour pouvoir additionner les résultats des deux méthodes en les mélangeant dans une seule mesure fonctionnelle. La compatibilité, d'une part, entre les objets de mesure et d'autre part entre les processus de mesure pour les deux méthodes est analysée et l'exactitude de leur agrégation est identifiée comme étant fonction du niveau de granularité d'application des mesures.

L'étude [52] par Xunmei et al. avait comme but de comparer les deux méthodes. Cet article souligne que la méthode IFPUG FPA a été conçue avant la publication du standard ISO

14143, et que la méthode COSMIC a été après ISO 14143 [24]. Ainsi, les auteurs expliquent que COSMIC a été conçu pour assurer la pleine conformité avec la norme ISO 14143, tandis que certains éléments de la méthode IFPUG n'ont pas été adoptés par l'ISO. Le standard ISO 14143-6:2006 fournit un résumé des standards internationaux en relation avec les méthodes de mesure fonctionnelle en donnant des définitions et des concepts pour les procédures de mesurage. [52] montre aussi que les domaines de logiciels qui peuvent être mesurés avec COSMIC est nettement plus large que la gamme couverte par IFPUG FPA.

Puisque la méthode COSMIC a été conçue pour mesurer les logiciels de plusieurs domaines d'applications, et surtout le domaine des systèmes temps-réel embarqués, cible de ce projet de recherche, les prochaines sections de cette revue de littérature se focalisent sur les travaux faits en utilisant la méthode COSMIC.

L'application de la méthode COSMIC aux logiciels temps-réel a été l'objet d'une recherche menée par Diab et al. qui proposent dans [12] une formalisation des règles d'application de la méthode COSMIC pour mesurer les spécifications conçues avec le langage de modélisation temps-réel ROOM. Un premier but de cette formalisation est d'éliminer les écarts éventuels dans les mesures parce que sans une procédure bien définie, d'après les auteurs [12], certaines définitions dans COSMIC pourront être sujettes à différentes interprétations par différents mesureurs, ce qui peut conduire à des résultats différents d'une même spécification, en fonction de l'interprétation faite par chacun des évaluateurs. Un deuxième but de ces définitions formelles est d'avoir une éventuelle piste pour pouvoir automatiser la mesure COSMIC pour les spécifications sous ROOM, ce qui réduit les coûts de mesure.

Dans [14] Diab et al. présentent un prototype appelé µcROSE qui mesure automatiquement la taille fonctionnelle des logiciels, telle que définie par la méthode COSMIC, pour les modèles Rational Rose RealTime. µcROSE sera présenté plus en détail dans le paragraphe « Outils d'automatisation basée sur la méthode COSMIC » de cette section.

La plus récente version de la méthode COSMIC est la version 3.0.1 et il est pertinent d'identifier les travaux faits avec cette version. Beatriz Marín, et ses collègues, ont travaillé sur l'application de la méthode COSMIC aux modèles conceptuels de logiciels des applications orientées-objet (non temps-réel). Ainsi, dans [36] Marín et al. présentent une procédure de mesure définie selon la dernière version de la norme ISO 19761 (Méthode de mesure COSMIC). La procédure de mesure a été conçue pour mesurer la taille fonctionnelle d'applications orientées objet générées à partir de leurs modèles conceptuels par moyen de transformations de modèles. La procédure de mesure est structurée en trois phases :

- phase de la stratégie, où l'objectif de la mesure est défini,
- phase de « mapping » où les éléments du modèle conceptuel qui contribuent à la taille fonctionnelle sont sélectionnés, et
- phase de mesure, qui génère la taille fonctionnelle de l'application générée.

Leur méthode OOmCFP (OO-Method COSMIC Function Points) est donc une procédure de mesure qui a été conçue pour mesurer la taille fonctionnelle des applications orientées objet générées à partir de leurs modèles conceptuels par le biais de transformations de modèles.

Marín et al. ont aussi travaillé sur le thème de précision de mesure dans [34]: ils précisent le sens terminologique du mot « précision » (qui assure deux propriétés : la répétabilité et la reproductibilité) et proposent une méthode d'évaluation de la précision de mesure du logiciel conformément à la norme ISO 5725. Cette méthode a été utilisée pour évaluer une procédure de mesure de la taille fonctionnelle. Une étude pilote a été aussi conçue et appliquer dans le but d'évaluer la précision de la procédure de mesure OOmCFP (OO-Method COSMIC Function Points) proposée dans [36]. L'autre but est de révéler des lacunes éventuelles dans la conception de leur méthode d'évaluation de la précision.

1.5 Automatisation de la mesure fonctionnelle

1.5.1 Généralités

Coman et al. présentent dans [10] une étude de cas sur l'adoption et l'utilisation à long terme, plus précisément de neuf mois de planification, suivi de deux ans d'utilisation, d'un système AISEMA (Automated Inprocess Software Engineering Measurement and Analysis) dans le département logiciel d'une grande entreprise italienne (son nom n'est pas cité) : les systèmes AISEMA visent à collecter automatiquement les données et également à fournir des analyses sur mesure pour l'aide à la décision. AISEMA réduirait le coût de la collecte des données en n'ajoutant aucune charge de travail supplémentaire. Ils peuvent collecter une large variété de données.

Les résultats de Coman et al. [10] tiennent en compte les caractéristiques de l'entreprise dans laquelle la recherche s'est déroulée et du système étudié. En offrant un aperçu des avantages et défis de l'adoption et l'utilisation d'un système AISEMA dans l'industrie, leur article a comme but d'aider d'autres entreprises à envisager l'adoption d'un système de mesure. Les enseignements tirés peuvent aussi guider l'amélioration des systèmes AISEMA car ils identifient à l'aide d'un cas des défis actuels dans l'utilisation et de proposer des solutions. Leur cas d'étude montre que l'entreprise qui envisage d'adopter un système AISEMA devrait être prête à accepter un long délai initial. À la fin de ce délai, le système commencera à offrir des bénéfices à l'entreprise. Les auteurs soulignent aussi que la présentation des données et des analyses est aussi importante que leur exactitude et leur intégrité.

Bien que [10] traite de l'automatisation de la collecte de donnés, ce qui est intéressant dans cet article est qu'il illustre par un cas que l'automatisation joue un rôle très important face à la collecte manuelle qui consomme beaucoup de temps, qui est fastidieuse, sujette à l'erreur et biaisée.

Komi-Sirvio et al. [29] proposent un processus d'automatisation de mesures qui est fondé sur les principes de mesurage dirigé par objectif. [29] présente deux exemples industriels

montrant ce qu'est en pratique l'automatisation de mesurage. Komi-Sirvio et al. discutent également les exigences des outils et comme exemple, ils présentent un outil de gestion du mesurage qui permet l'automatisation du mesurage.

Les auteurs en [29] précisent que par « automatisation du mesurage » ils entendent l'automatisation de la collecte des données, du calcul et de la formation des graphiques d'analyses en suivant un chemin prédéfini et en utilisant les données stockées dans une ou plusieurs sources de données. Donc l'horizon de [29] se situe aux frontières de notre projet de recherche. Le travail en [29] peut être utile en ce qui concerne la production des données utilisées pour la formation des graphes par exemple.

1.5.2 Études sur les outils et/ou prototypes qui automatisent la mesure de la taille fonctionnelle

Cette section présente les études du marché et les publications de recherche sur les outils ou les prototypes, incluant les pistes qui permettent l'automatisation de la mesure de la taille fonctionnelle des logiciels. Dans cette section, on trouvera aussi quelles parties des méthodes de mesures sont visées par chacun de ces outils/prototypes/pistes.

Dans [50], Stambollian *et al.* présentent un cadre de référence constitué de l'ensemble des fonctions qui intéressent les praticiens pour mettre en œuvre les normes ISO sur les mesures de la taille fonctionnelle. Cet article contient également une étude de 2006 qui présente des outils liés à COSMIC qui étaient alors disponibles sur le marché et dans la communauté scientifique. [50] est un complément à Mendes *et al.* [44] et reprend la présentation du framework général des fonctions de la mesure de la taille fonctionnelle ainsi que les quatre dimensions et les catégories-clés de la mesure de la taille fonctionnelle:

- support (documentation et formation);
- mesurage (manuel ou automatisé);
- stockage des résultats de mesures (référentiel des résultats de mesure) ;

- utilisation des résultats (planification de projet, de baselining, d'estimation et de prévision, gestion, modélisation de productivité, benchmarking et analyse comparative).

En termes de support du processus de mesurage, les outils de support COSMIC varient considérablement. L'étude a montré qu'en 2006 ils couvraient les différentes phases de la méthode elle-même, les phases de la méthode et les résultats de la méthode utilisés à des fins différentes. Les sept outils étudiés sont :

1. COSMICXpert
2. ISBSG
3. MeterIT-Cosmic, MeterIT-Project, PredictIT;
4. ExperiencePro;
5. KnowledgePlan;
6. SIESTA.

En plus des outils commerciaux, Stambollian *et al.* ont aussi identifié dans leur étude les prototypes/pistes de recherche suivants:

- XForms-Format
- μcROSE, ROOM
- COSMIC-RUP
- Ontological formalization

D'après [17], la classification concise des outils de support COSMIC - ISO 19761 dans cette étude a permis :

- d'effectuer une analyse d'écart entre eux;
- d'identifier les fonctions qui n'avaient pas encore été abordées en 2006 [50].

Les outils « commerciaux » présentés sont assez variés car ils couvrent différents aspects de la méthode COSMIC ou phases de cette méthode. Toutefois aucun de ces outils ne prend en charge l'automatisation de la fonction-clé du mesurage, bien qu'ils couvrent les autres fonctions ou dimensions présentées en [44] et repris dans [50].

Marin *et al.* [39] présente une revue de la littérature de 2008 sur les procédures de mesure de la taille fonctionnelle basées sur la méthode COSMIC et qui peuvent être appliquées aux modèles conceptuels. Cette étude comprend les propositions de Bevo *et al.* [5], Jenner [28], Diab *et al.* [12], Poels [48], Nagano *et al.* [45], Azzouz *et al.* [4], Condori-Fernández *et al.* [11], Habela *et al.* [18], Grau *et al.* [17], Lévesque *et al.* [32], et Marín *et al.* [38].

Dans [39], les auteurs résument les propositions basées sur la mesure COSMIC en utilisant les critères suivants :

- la version de la méthode de mesure ;
- le contexte de la proposition (de la procédure) ;
- le domaine fonctionnel (par exemple : les systèmes temps réel, les systèmes de gestion d'information) ;
- l'artefact en entrées (i.e. modèle d'exigences, modèle d'analyse, ou modèle de conception);
- les règles d'application de la procédure ;
- l'instrument d'application de la procédure ;
- et la vérification de la procédure.

Cette étude fournit aux chercheurs un aperçu de l'état actuel des procédures de mesure de la taille fonctionnelle basée sur COSMIC, ainsi que des informations sur ces procédures.

Parmi les procédures étudiées [39], notre projet de recherche s'intéresse aux procédures qui ont un instrument d'application ou un outil qui aide les praticiens à les appliquer, soit :

Bevo *et al.* 1999 :

Cette étude se base sur un mapping entre les concepts de diagrammes UML version 1.0 (cas d'utilisation, scénarios, de classes) et les concepts de COSMIC version 2.0. Elle concerne les systèmes d'informatique de gestion.

Jenner 2001 :

Cette étude traite l'aspect de la granularité des cas d'utilisation dans la proposition de Bevo *et al.* introduite ci-dessus. Pour cette raison, les caractéristiques générales de la proposition de Jenner constituent un affinement des caractéristiques de Bevo *et al.*

Diab *et al.* 2001 :

La proposition de Diab *et al.* présente un ensemble de règles formelles (équations mathématiques) qui permettent la mesure de la taille fonctionnelle, en utilisant un mapping avec COSMIC version 2.0, pour les applications temps-réel qui sont spécifiées sous Real-Time Object Oriented Modelling (ROOM). Les spécifications ROOM sont utilisées par l'outil Rational Rose Real Time (RRRT) pour la conception et la spécification des applications des systèmes temps-réel. Un outil-prototype nommé « µcRose » met en œuvre cette procédure de mesure (mise à jour vers la version 2.2 de COSMIC).

Azzouz *et al*. 2004 :

Azzouz *et al.* présentent un prototype qui automatise la mesure de la taille fonctionnelle d'applications développées avec le Rational Unified Process (RUP) pour les systèmes d'information de gestion. Cette méthode utilise UML pour la spécification des systèmes. Ils basent leur proposition sur les règles décrites par Bevo *et al*. et Jenner. La version de la méthode COSMIC est la 2.2. Ce prototype est toujours resté de type exploratoire et n'a pas été soumis à des tests.

Grau *et al.* 2007 :

La proposition de Grau *et al.* présente un ensemble de règles de mapping pour mesurer la taille fonctionnelle des modèles i* générés par le biais de la réingénierie de systèmes en utilisant PRiM. Ici encore, COSMIC version 2.2. est utilisé, et le domaine d'application visé est celui des système d'information de gestion. L'outil d'application de la mesure s'appel J-PRIM.

Marín *et al.* 2008 :

La proposition de Marín *et al.* présente une procédure de mesure pour mesurer la taille fonctionnelle des applications orientés-objets générés dans des environnements MDA à partir de leurs modèles conceptuels. Cette proposition utilise la Méthode de développement OO-Method version 3.8 comme référence pour l'approche MDA. La version de la méthode COSMIC utilisée est la version 3.0.

Dans [39], on retrouve aussi une analyse globale des critères utilisés dans l'étude présentée ci-dessus, incluant la version de la méthode de mesure COSMIC : quatre propositions (Bevo, Jenner, Diab, et de Nagano), utilisent la version 2.0, deux propositions (Poels et Lévesque), utilisent la version 2.1, quatre propositions (Azzouz, Condori -Fernández, Habela, et Grau) utilisent la version 2.2, et seule la proposition (Marin) utilise la version 3.0. Il est important de noter que la proposition par Nagano en 2003 utilise la version 2.0 en dépit du fait des nouvelles versions de COSMIC déjà publiées en 2003. Il est également important de noter que la proposition de Lévesque de 2008 utilise la version 2.1 en dépit du fait que la version 3.0 de COSMIC existait déjà en 2008. Selon [39], est que les nouvelles versions de COSMIC apportent des améliorations et des clarifications qui aident à mieux comprendre la méthode de mesure et d'obtenir des mesures exactes. Par conséquent, ils considèrent que l'utilisation de la dernière version de la méthode est très importante pour le développement correct des procédures de mesure.

Une deuxième remarque importante dans [39] est que les modèles UML, MERODE et i* n'ont pas suffisamment d'expressivité pour spécifier toutes les exigences fonctionnelles des applications (par exemple, ces modèles ne permettent pas la spécification des valeurs données aux attributs des classes, les unités d''interaction, etc.). De même pour les modèles d'exigences de la méthode OO-method. Par conséquent, les propositions fondées sur ces modèles ne font qu'une estimation de la taille fonctionnelle des applications, et non pas une mesure. Toutefois, les propositions fondées sur les modèles RRRT, le cahier des charges xUML, et le modèle conceptuel de la méthode OO-method ont assez de sémantique et de formalisation pour préciser toutes les exigences fonctionnelles, ce qui permet la mesure de la taille fonctionnelle des applications.

Seules les propositions de Diab *et al.* et de Nagano *et al.* ont été développées pour le domaine des systèmes temps-réel. Les autres propositions ont été développées pour le domaine des systèmes de l'information de gestion. D'après [39], aucune proposition de procédure de mesure n'existe pour d'autres domaines (les systèmes algorithmiques, de systèmes géographiques, systèmes ubiquitaires, etc.), en dépit du fait que la méthode de mesure COSMIC peut être appliquée à plusieurs domaines.

En ce qui concerne l'artefact d'entrée, toutes les propositions utilisent plus qu'un artefact d'entrée.

Sept propositions (Bevo, Jenner, Azzouz, Condori-Fernández, Habela, Grau, et Lévesque) utilisent les entrées obtenues dans la phase des exigences. D'un autre coté, trois propositions (Diab, Poels, Nagano) utilisent les objets d'entrée obtenus dans la phase de spécification d'exigences, et une seule proposition (Marín) utilise les artefacts d'entrée obtenus dans les phases de spécification d'exigences et de conception de modèles.

Le design d'une procédure de mesurage est une étape clé dans l'élaboration de l'abstraction correcte des éléments à mesurer, car, autrement, la procédure risque de ne pas mesurer ce qui doit être mesuré selon les principes de la méthode de mesure de base sélectionnée. La

documentation et le détails des phases de l'application d'une procédure est donc nécessaire. Il est également important de garder à l'esprit l'influence directe que la conception d'une procédure de mesure a sur les éventuelles automatisations de la procédure. D'après [39] seules deux propositions (Condori-Fernández et Marín) proposent et montrent la phase de conception (ou design) de la procédure de mesure, la définition des objectifs de la procédure, la caractérisation du concept à mesurer, le mapping avec les concepts de la méthode COSMIC, et les règles de mesure. Les autres neuf propositions se limitent à définir un mapping partiel entre les concepts de COSMIC et les concepts des modèles conceptuels à mesurer.

En conclusion, [39] donne un résumé détaillé des propositions des procédures de mesure actuelles de la taille fonctionnelle qui sont basées sur la méthode de mesure COSMIC et qui utilisent des modèles conceptuels comme artefacts d'entrée pour effectuer la mesure. L'étude présentée dans [39] offre aux chercheurs un aperçu général actualisé de l'état en 2008 des procédures de mesure de la taille fonctionnelle qui sont fondées sur COSMIC. Cette étude [39] fournit également aux praticiens des informations sur les procédures de mesure de taille fonctionnelle qui sont disponibles. Il est important de noter que les procédures de mesure présentées dans [39] ont été développées pour appliquer la méthode de mesure COSMIC aux modèles conceptuels afin d'obtenir la taille fonctionnelle des applications finales dans les premiers stades du processus du développement logiciel. Par conséquence, les conclusions importantes de [39] sont :

- la procédure de mesure doit être basée sur la dernière version de la méthode de mesure. L'artefact d'entrée utilisé doit avoir suffisamment de sémantique et de formalisme pour permettre la spécification de toutes les exigences fonctionnelles.

- la conception de la procédure de mesure doit être effectuée en définissant clairement des règles pour préciser la stratégie de la mesure, les règles pour effectuer le mapping entre les concepts de COSMIC et ceux des modèles conceptuels, ainsi que des règles pour identifier les mouvements de données et effectuer le mesurage.

- l'automatisation de la procédure aide à réduire le coût de l'exécution du mesurage et à accroître l'efficacité du processus de mesure.

- la vérification de la procédure doit être effectuée pour assurer la qualité des résultats obtenus.

1.5.3 Évaluation des outils d'automatisation de la mesure fonctionnelle

IFPUG a établi un programme de certification pour les outils de comptage de points de fonction (FP) qui reconnaît trois types d'outils [43] :

1. Les outils de type 1 : c'est à l'utilisateur d'effectuer l'identification des fonctions et l'outil est utilisé seulement pour le calcul.

2. Dans les outils du type 2 : le comptage est déterminé d'une façon interactive : le système pose des questions à l'utilisateur et fait le calcul sur la base des réponses fournies.

3. Dans les outils du type 3 : le système effectue le comptage de façon autonome sur la base d'une description stockée dans l'application.

En 2006, il existait cinq outils reconnus comme type 1 et un seul outil de type 2. Aucun outil de type 3 n'avait réussi à obtenir une certification officielle [16].

Dans [43], Mendes présente un protocole d'évaluation pour les outils informatisés de comptage automatique de points de fonction. La méthode de calcul de taille fonctionnelle utilisée dans le cadre ce projet est la méthode IFPUG. L'objectif principal du projet de Mendes est de bâtir ce protocole en identifiant les enjeux de l'automatisation de calcul des PF, les moyens d'évaluation de l'exactitude des résultats du calcul par les outils, la démarche d'évaluation des outils, les composants internes aux outils responsables – s'ils ont lieu - des

écarts entre résultats (manuels/automatisés). Un des buts du projet [43] est de déterminer les cas où les résultats de(s) l'outil(s) de mesurage peuvent être considérés comme corrects. La stratégie de recherche que Mendes a adoptée se divise en 2 phases :

1. La première concerne une étude de marché des outils de mesurage de Point de Fonctions ;
2. La deuxième concerne la proposition du protocole énoncé ci-dessus.

Mendes a identifié quatre dimensions concernant les Points de Fonctions :

1. Comptage
2. Stockage
3. Utilisation
4. Support au comptage.

Les dimensions identifiées ci-dessus ont permis la création d'un cadre référentiel pour supporter la structuration des informations relatives au cycle de vie des Points de Fonctions. Ce cadre référentiel a permis le classement dans des catégories taxonomiques des outils trouvés dans le cadre de cette recherche. Il est à noter que les catégories proposées ne sont pas exclusives, i.e. elles peuvent contenir différentes caractéristiques : essentielles, souhaitables et optionnelles.

L'auteur [43] a inclus dans ce document une liste des outils d'automatisation du comptage PF présents dans le marché en 1996, avec les différentes caractéristiques identifiées.

Dans le cadre de sa recherche, l'auteur [43] a constaté que la description logique des systèmes est beaucoup plus utilisée par rapport à la description physique, dans le contexte de l'automatisation du calcul des points de fonctions. L'auteur [43] a décidé d'utiliser le modèle d'un processus générique – cible de comptage de PF - proposé par Paton et Abran [47]. Ce modèle est basé sur les notions de : processus cible (frontières), dépôts de données et flux de

données. Toutefois, Mendes annonça que le modèle présente quelques limitations et il propose des extensions pour éliminer ces limitations.

Les extensions proposées viennent compléter le modèle et surtout compléter la couverture des situations ou des scénarios possibles qui peuvent se produire dans les cas pratiques lors de l'application de la méthode de comptage. Donc l'auteur a ajouté plusieurs éléments et notions pour atteindre la description la plus proche des cas pratiques. Le but de l'auteur est de présenter tous les cas possibles et valides, des situations les plus simples jusqu'au plus complexes, d'une manière, d'après l'auteur, systématique et méthodique. Les analyses ont été faites pour un seul processus cible isolé en premier lieu, puis, en présence de plusieurs processus cibles.

Toutefois, l'auteur n'utilise que des «études de cas» pour ses analyses pour la définition du protocole d'évaluation, en vision de simplification.

L'auteur propose que l'évaluation des outils ait deux volets : les caractéristiques statiques des outils d'une part, et les caractéristiques dynamiques d'autre part. Le protocole proposé couvre ces caractéristiques.

La démarche d'évaluation proposée comporte les étapes suivantes :

1. Evaluer les caractéristiques statiques (initiation);
2. Planifier l'étendue des essais dynamiques et choisir les modules d'évaluation dynamique à utiliser (structuration);
3. Evaluer les caractéristiques dynamiques (réalisation);
4. Analyser et à documenter les résultats obtenus ainsi que la procédure exécutée.

Pour les caractéristiques dynamiques, la partie majeure de ce protocole, trois étapes d'essais ont été proposées :

1. Unitaires : Le but des essais unitaires est d'évaluer de manière isolée chacun des composants transactionnels avec des scénarios de complexité croissante (simple, moyenne et élevée).
2. Intégration à petite échelle : Le but des essais d'intégration à petite échelle est d'évaluer le comptage automatique et d'identifier les différents composants transactionnels dans le cas où ils sont mélangés. Ceci et aussi par des scénarios de complexité croissante.
3. Intégration à grande échelle : Et toujours par les trois niveaux croissant de complexité, les essais d'intégration à grande échelle ont comme but d'évaluer l'outil pour les applications « complètes » comportant une gamme de situations d'identification de composant très diversifiés.

Ce protocole est intéressant pour le projet de recherche de cette thèse. Même s'il s'agit d'un protocole qui vise les outils de mesurage basés sur la méthode IFPUG, il trace un chemin pour savoir comment et quoi tester dans les prototypes à venir. Toutefois, ce protocole reste théorique et selon l'auteur il est nécessaire de tester ce protocole dans des cas réels, pour s'assurer de la faisabilité de son application.

1.5.4 Outils d'automatisation de la méthode IFPUG

Quelques chercheurs ont travaillé sur le sujet de l'automatisation de la mesure de taille fonctionnelle avec la méthode IFPUG.

Dans [30], Lamma *et al.* présentent un outil de mesure des points de fonction (FP) du logiciel à partir de sa spécification exprimée sous la forme d'une combinaison d'un diagramme d'entité-relation (ER) et d'un diagramme de flux de données (DFD).

La première étape de [30] consiste à traduire les règles informelles de comptage et générale FP – selon les auteurs - en règles rigoureuses exprimant des propriétés de l'ER-DFD. Cette étape a un double but : le premier est de réduire la divergence des résultats de comptages

d’une même application faits par des personnes différentes. Le deuxième but est de fournir un système pour l’automatisation de la mesure des points de fonction.

Ensuite, les règles sont converties en Prolog. Et finalement les auteurs déclarent qu’ils ont mesuré six applications avec leur outil. Bien que les chiffres de la comparaison ne figurent pas dans l’article, les résultats obtenus par l’outil se rapprochent, selon les auteurs, à ceux des experts humains. Toutefois, vu que les résultats des mesures manuelles effectuées par les experts humain est absente dans l’article, l’affirmation des auteurs ne peut pas être vérifiée. Les auteurs [30] annoncent de futurs travaux pour tester l’outil avec plusieurs cas d’application et comparer les résultats à ceux d’un expert humain.

Dans [16], Fraternali *et al.* ont illustré les résultats de la mise en œuvre de la mesure des points de fonction dans le contexte de développement « Model Driven » pour les applications Web. Leur approche explicite et concrétise l'application des règles d’IFPUG au modèle conceptuel d'une application Web, qui donc consiste à produire un schéma de données et un schéma d’hypertexte. Le modèle conceptuel a suffisamment d'information pour permettre le comptage, jugé précis par les auteurs, de l'application, sans qu'il soit nécessaire d'intervenir sur le cahier des charges avec des extensions de modèle dites « non-naturelle ». En conséquence, la mesure des points de fonction, selon les auteurs, ne demandera pas plus d’effort au niveau des designs puisqu’elle utilise directement les modèles conceptuels utilisés pour la génération du code. Toutefois, les auteurs ne donnent pas de chiffres comparatifs pour confirmer cette affirmation.

L'approche proposée a été mise en œuvre dans le cadre d'une notation spécifique de modélisation (WebML) et une suite d'outils de développement (WebRatio). Toutefois, les auteurs [16] prétendent que l’algorithme proposé est bien adapté pour chaque profil UML ou même pour tout langage de description de domaine précis qui peut être utilisé pour décrire les données back-end ou des objets, la structure du front-end, et le lien entre ces deux couches.

Fraternali *et al.* suggèrent pour des futurs travaux l'application du système présenté pour une grande collection des projets développés en utilisant WebML et WebRatio, avec un double objectif : poussez l'évaluation de l'outil et d'optimiser sa précision d'une part. Et d'autre part, assurer l'évaluation de la productivité du développement Model-Driven.

Quant à Uemura *et al.* ils proposent dans [51] des règles de mesure FPA détaillées pour les spécifications de conception basée sur UML (Unified Modeling Language) et annoncent qu'ils ont développé un outil de mesure de points de fonction qui utilise en entrée les spécifications de conception sous Rational Rose. Cet outil est conçu et mis en œuvre à l'aide de Visual C++ sous Windows 98. Les entrées de l'outil sont des diagrammes de séquences et des diagrammes de classes de Rational Rose et la sortie comprend les résultats de la mesure, ainsi que d'autres éléments identifiés. L'outil est composé principalement de trois unités et deux bases de données : l'unité d'analyse, l'unité de d'analyse des bases de données, l'unité de comptage, la base de données de comptage et l'unité d'interface.

Dans [51] l'outil a été testé avec une spécification de conception réelle et les différences ont été examinées entre les valeurs données par l'outil et celles d'un spécialiste FPA. Et selon [51], les résultats obtenus montrent l'applicabilité de leur outil, mais cette affirmation ne s'appuie sur aucun chiffre fourni.

1.5.5 Outils d'automatisation basés sur la méthode COSMIC

L'application de la méthode COSMIC aux logiciels temps-réel a fait l'objet d'une recherche menée par Diab *et al*. qui proposent dans [12] une formalisation des règles d'application de la méthode COSMIC pour mesurer les spécifications conçues avec le langage de modélisation temps-réel ROOM. Les buts de cette formalisation sont tout d'abord d'éliminer les écarts éventuels dans les mesures parce que, d'après [12], certaines définitions dans COSMIC pourront être sujettes à différentes interprétations par différents mesureurs : ce qui peut conduire à des résultats différents d'une même spécification, en fonction de l'interprétation des spécifications faite par chacun des évaluateurs.

Un deuxième but de ces définitions formelles est d'avoir une éventuelle piste pour pouvoir automatiser la mesure COSMIC pour les spécifications sous ROOM, ce qui réduirait les coûts de mesure.

Finalement, ces définitions formelles de COSMIC pour ROOM fournissent un chemin qui est peut être utile pour l'application de la mesure COSMIC pour d'autres langages de modélisation orienté-objet comme UML.

Les principales conclusions de [12] montrent que les règles présentées sont spécifiques au vocabulaire du langage ROOM, d'où l'éventuelle automatisation sera exclusivement applicable aux spécifications ROOM. Deuxièmement, en créant les règles formelles, dans [12] les auteurs affirment ont trouvé plusieurs possibilité pour appliquer les principes de la méthode COSMIC au vocabulaire du langage ROOM (i.e. en créant les procédures de mesurage, les élément du langage ROOM présentent plusieurs choix valables pour le mapping nécessaire avec les éléments COSMIC , donc un choix devait être fait pour mapper chaque élément COSMIC à un et seulement un seul élément ROOM) et donc ils ont choisi l'interprétation qui leur paraît la plus « raisonnable » : donc les règles doivent être validées avec un groupe d'experts COSMIC pour s'assurer de leur validité. Et finalement, les règles formelles doivent encore être appliquées à plusieurs systèmes pour les « tester » et pouvoir comparer les résultats avec les mesures effectuées par plusieurs experts pour valider leur homologie.

Dans [14], Diab *et al.* présentent un outil appelé µcROSE qui mesure automatiquement la taille fonctionnelle des logiciels, tel que défini par la méthode COSMIC, pour les modèles Rational Rose RealTime. µcROSE simplifie et organise le processus de mesure, afin d'assurer la cohérence et la répétabilité de la mesure tout en réduisant les coûts de mesure. Il est le premier outil à adresser l'automatisation de la mesure fonctionnelle avec COSMIC et peut être intégré dans l'outil Rational Rose RealTime.

Toutefois, cet outil présente plusieurs contraintes et défauts. Pour commencer, les résultats fournis par l'outil, pour un logiciel donné, ont été comparés avec ceux d'un expert COSMIC indépendant : les tailles fonctionnelles obtenus sont différentes. En second, la principale limitation de mcROSE concerne le module qui effectue l'analyse du code C++ qui ne peut examiner que certains opérateurs de base, et vu que l'outil utilise des segment de code C++ pour pouvoir effectuer la mesure, donc la mesure peut être non précise.

Marín *et al.* présentent une définition des mécanismes nécessaires pour automatiser la procédure OOmCFP [35]. Ce document présente également l'outil qui met en œuvre la procédure OOmCFP, c.-à-d. qui automatise la procédure.

Comme cet outil mesure la taille fonctionnelle des applications industrielles générées dans des environnements MDA à partir de leurs modèles conceptuels, il n'est pas nécessaire d'effectuer la tâche de mesure sur le code final. Toutefois, plusieurs hypothèses en relation avec la performance du processus de la mesure fonctionnelle ainsi qu'en relation avec l'implémentation de l'outil ont été prises en compte.

L'aspect performance est axé sur la réduction du temps de mesure, tandis que l'aspect implémentation concerne l'exécution correcte du processus de mesure en évitant des problèmes reliés aux débordements éventuels produits lors de la mesure de « grands » modèles conceptuels .

L'étude montre aussi, par le biais d'un exemple, que l'outil OOmCFP est plus efficace que les mesures effectuées manuellement, en termes de précision des résultats de mesure et du fait qu'il n'est pas sujet à l'erreur 'humaine'. Cet exemple montre également comment l'outil OOmCFP obtient des résultats de mesures précises des modèles conceptuels de la Méthode OO-Method.

Finalement, l'outil OOmCFP intègre les avantages que la méthode de mesure COSMIC fournit : ces avantages sont montrés par le biais d'une analyse comparative entre la procédure

OOmCFP qui implémente la méthode COSMIC et une autre procédure (OOmFP) qui implémente la méthode FPA.

Dans [4], Azzouz *et al.* présentent une approche et un prototype pour l'automatisation de la mesure de la taille fonctionnelle suivant le processus RUP. La conception de cet outil est basée sur un mapping direct entre COSMIC et les concepts et les notations UML, une base à partir de laquelle les composants de Rational Rose peuvent être extraits pour procéder à la mesure des projets logiciels. Ce prototype a comme but de permettre non seulement l'obtention de la taille fonctionnelle, une fois que toutes les spécifications ont été accomplies, mais aussi d'avoir des indicateurs de la taille quand seulement des informations de haut niveau sont disponibles.

Toutefois, ce prototype présente plusieurs limitations surtout liées au fait que plusieurs étapes doivent toujours être faites manuellement. D'autre part, une phase de test du prototype n'a pas été encore faite, de même que des tests indépendants et à grande échelle.

Li *et al.* [33] présentent un système de support de la mesure fonctionnelle pour les logiciels interactifs. Ce système a été élaboré en se basant sur le format XForms afin de faciliter le processus de mesure. XForms est présenté comme la nouvelle génération de formulaires Web. Et il est aussi capable de décrire l'objectif et la présentation du formulaire Web séparément, les sémantiques des interactions utilisateur-ordinateur, qui sont essentiel à la mesure de la taille fonctionnelle (FSM), peuvent alors être extraites de la spécification. Dans cette étude, les auteurs proposent plusieurs extensions à XForms de W3C pour l'appliquer à la mesure fonctionnelle. La procédure de mesure de la taille fonctionnelle, proposée dans une autre étude, est automatisée en se basant sur les spécifications UI format XForms.

Finalement, la partie interactive d'un produit logiciel de voiture-navigation a été mesurée avec le système de support et les résultats suggèrent que ce système est efficace dans l'automatisation de la génération de résultats de mesure et, donc, dans la réduction des coûts

de mesure. Des futurs travaux sont annoncés pour améliorer la précision des résultats de la mesure.

Dans [6], Bévo *et al.* explorent une vision ontologique d'une procédure d'une méthode de mesure (processus d'application d'une méthode de mesurage). Les bases d'une formalisation ontologique de la procédure (basée sur COSMIC) sont présentées: l'accent est mis sur les ontologies du domaine et des tâches associées à la procédure. Outre le fait qu'elles peuvent être utilisées pour la conception des outils de mesure, les ontologies sont utiles car elles fournissent une meilleure compréhension des procédures des méthodes de mesure, et servent comme points de ralliement consensuel pour structurer, représenter, échanger et interpréter des informations, des concepts liés à la procédure de mesure et de la mesure en général. Le formalisme utilisé pour la présentation de ces ontologies doit être approprié. Le formalisme orienté objet est utilisé dans cette étude. Les tâches et les concepts sont décrits selon la méthodologie CommonKADS [49].

Dans [6], Bévo *et al.* présentent aussi un exemple d'une ontologie de domaine associée à la procédure de mesure COSMIC, ainsi qu'un exemple d'ontologie de tâches associées à cette procédure. Donc un des objectifs principaux de ce document est d'utiliser l'ontologie proposée et présentée dans un formalisme simple et expressif, avec une description claire et précise des tâches de mesure. Cela constitue un pas vers l'automatisation complète ou partielle de ces tâches, ainsi qu'il contribue à une meilleure compréhension de la procédure de la méthode de mesure.

Les travaux de Bévo *et al.* [6][7] se concentrent sur deux volets principaux : le premier volet touche à la formalisation de l'ontologie du processus de l'application d'une procédure de mesure de la taille fonctionnelle du logiciel, et la présentation des avantages de cette ontologie qui aide à l'automatisation de la procédure de mesure. Le deuxième volet touche à une application du premier volet, i.e. l'automatisation du processus d'application de la méthode de mesure COSMIC à partir de spécifications UML.

Dans [6], les auteurs soulignent qu'il faut prendre en compte, dans la conception des outils de mesurage, les aspects suivants :

- l'identification de tous les concepts traités en procédure d'une méthode de mesure, ainsi que les relations entre ces concepts (ontologie de domaine),
- l'identification de toutes les tâches associées à la procédure d'une méthode de mesure, ainsi que les liens entre ces tâches (ontologie des tâches).

D'après Bévo *et al.* l'existence de telles ontologies dans un formalisme simple, expressif, claire et précis, permettrait non seulement une meilleure compréhension de la procédure de mesure mais aussi de servir pour la représentation, l'échange et l'interprétation des informations, des concepts liés à la procédure de mesure et de la mesure en général. Ensuite, [38] présente le contenu exact de ces ontologies en proposant un exemple d'une ontologie de domaine associé à la COSMIC procédure de mesure est présenté, ainsi qu'un exemple d'ontologie de tâches associées à cette procédure.

Dans [38], le prototype proposé vise à automatiser, d'après l'auteur, « une bonne partie du processus » de la méthode COSMIC avec UML, en utilisant les études présentées en [38] : Le prototype mentionné dans [6], est développé avec le langage de programmation Java et sous la plate-forme de développement Sun One Studio, Community Edition. Les données relatives aux spécifications UML des applications, données relatives aux modèles des applications suivant la méthode COSMIC, aussi dites « données persistantes», sont gérées avec le SGBD Oracle 8i. L'auteur explique qu'il s'est contenté d'un prototype partiel (qui utilise les diagrammes de cas d'utilisations, les diagrammes de séquences correspondants et les modèles objet du domaine). D'autre coté, l'auteur ne présente pas des chiffres comme résultats de validation et d'analyses des résultats, il annonce juste que les tests du prototype sont orientés pour couvrir l'aspect fonctionnel et l'aspect de précision des résultats. Aucune affirmation concluante n'est fournie.

1.5.6 Études basées sur l'ingénierie inversée

Dans cette section des procédures basées sur les techniques d'ingénierie inversée seront présentées. Les techniques d'ingénierie inversée consistent à obtenir la taille (mesurée ou estimée) d'un logiciel à partir de son code source.

Dans [19], Ho et *al.* proposent un cadre général visant à construire un modèle pour l'analyse automatique des points de fonction (FPA) à partir du code source COBOL en utilisant la technique de découpage des programmes (program slicing).

Le cadre proposé permet de construire des modèles permettant d'automatiser le comptage des points de fonctions à partir du code source en se basant sur la méthode IFPUG. En suivant la procédure, le code source du système COBOL est scanné pour produire des calculs de points de fonction. Les fichiers source de l'application sont utilisés pour définir les frontières de l'application, étape nécessaire pour faire le comptage.

Le modèle prend en compte la structure du langage COBOL pour identifier les fichiers physiques et les transactions. Donc la technique repose sur l'identification des mots clefs (i.e. les mots réservés, les déclarations des fichiers d'entrée / de sortie (READ & WRITE), les instructions de manipulation de données (ACCEPT, DISPLAY et MOVE)) qui sont utilisés comme des informations de base pour le technique de découpage du programme pour identifier les candidats aux fichiers physiques et aux transactions.

Une fois que les candidats physiques sont identifiés, les règles heuristiques, proposées par les auteurs, sont appliquées afin de mapper les fichiers physiques et les transactions (candidats) dans des fichiers et des transactions logiques pour pouvoir ensuite extraire les informations nécessaires au comptage : les éléments de la méthode IFPUG (les données et les transactions).

Une remarque importante sur cette étude est que cette proposition se base sur une démarche de « programme slicing » : donc la qualité de l'outil de découpage jouera un rôle très important dans la qualité de résultats fournis (nuance théorique/ pratique). Donc il sera nécessaire de tester et de valider le modèle par des exemples pratiques et de comparer ses résultats aux mesurages manuels.

Une autre étude [3], April *et al.* proposent une approche pour le comptage des PF à partir du code source en utilisant des techniques d'ingénierie inversée (reverse engineering) d'un point de vue conceptuel et théorique. Cette étude a comme but principal de permettre une évaluation plus précise des règles IFPUG. Pour cela, April *et al.* utilisent la représentation semi-formelle décrite par Paton *et al.* (et approfondie ultérieurement par Mendes dans l'étude décrite ci-dessus), pour développer une notation semi-formelle qui peut identifier les combinaisons valides (cf. l'étude de Mendes) de types de fonctions transactionnelles de FP en fonction des règles IFPUG. Les combinaisons sont un concept utile, d'après les auteurs, pour l'automatisation de l'identification du type d'une fonction transactionnel.

Il faut noter que la seule fonction transactionnelle donnée en tant qu'exemple dans cette étude est la « requête externe » (Externel enquiry - EQ) de la méthode IFPUG. April *et al.* proposent leur propre extension du modèle de Paton *et al.* : ce raffinement de la représentation précédente est fait pour pouvoir représenter les unités de calcul et les fichiers physiques qui seront utilisés comme intrants pour la technique d'ingénierie inversée et pour les algorithmes, utilisés pour mécaniser l'interprétation des règles de comptage IFPUG. Et donc une traduction physique-logique est proposée pour finalement avoir les règles IFPUG exprimées avec les annotations qui représentent les entités physiques. Une analyse de propagation est aussi faite dans cette étude pour couvrir les situations où une variable peut atteindre le point de sortie par l'intermédiaire d'une propagation d'une série de copies, tandis que la variable d'origine elle-même peut être modifiée en toute sécurité après la copie.

Donc dans cette recherche des définitions formelles de quatre des sept règles de requête externe (EQ) de comptage IFPUG ont été introduits. Donc il serait

intéressant de poursuivre la recherche sur l'ensemble des règles d'IFPUG et de développer les techniques d'analyse liées. D'un autre coté, une validation empirique de l'approche proposée serait utile.

1.6 Estimation des points de fonction

Face aux procédures de calculs ou de comptage de PF basées sur les méthodes de PF standards, il existe des méthodes d'estimation (non pas un calcul précis et complet) de PF du logiciel.

Dans [42] Meli *et al.* présentent les caractéristiques de certaines méthodes proposées dans la littérature et ils présentent aussi un modèle d'analyse comparative générale utile pour l'évaluation de toute méthode supplémentaire.

Les auteurs argumentent qu'il y a au moins deux situations dans lesquelles avoir une méthode d'estimation compatible, mais alternative aux règles standard pour le comptage des PF (suivant la méthode IFPUG, dans leur étude), pourrait être décisive :

1. Le premier cas se produit lorsqu'un projet de nouveau développement ou d'évolution est dans une phase très précoce qu'il n'est tout simplement pas possible d'effectuer un comptage de FP selon les normes IFPUG.

2. Le deuxième cas se produit quand une évaluation des logiciels existants est nécessaire, mais la documentation nécessaire ou le temps et les ressources nécessaires pour effectuer un calcul détaillé FP ne sont pas disponibles.

Le modèle d'analyse comparative générale, ou le modèle de « benchmarking » défini par les auteurs tient compte d'un ensemble d'attributs ou paramètres, liés à l'objet et au but de comparaison. Chaque paramètre, a, selon le cas, une valeur appropriée. Et vu que selon les circonstances et les besoins réels des organisations, une même méthode d'estimation peut être

évaluée comme la solution préférable dans certains cas, ou non dans d'autres cas : le modèle d'analyse comparative exprime à la fois les variables objectives et subjectives pour répondre à ce besoin. Donc les auteurs ont proposé les variables suivantes: poids, score et score pondéré. Une liste des paramètres se trouve aussi dans le document.

Les classes de méthodes étudiées sont: les méthodes d'estimation directe et les méthodes d'estimation dérivée.

La principale conclusion de cette étude est qu'aucune des méthodes n'est meilleure que les autres dans l'absolu: en effet, le modèle général d'analyse comparative a été mis en place afin de trouver la méthode d'estimation optimale pour chaque situation donnée. Le niveau de détail des informations nécessaires pour estimer les PF suivant la plupart des méthodes d'estimation dérivée est très semblable à celui requis par un «calcul standard» et cela signifie que les méthodes d'estimation correspondantes ne sont pas particulièrement puissantes. Seule la méthode «Early Function Points» peut être considérée comme puissante, car elle permet d'avoir un gain en temps et en efforts entre 50% et 90% pour estimer des résultats de mesure qui varient de plus ou moins 10% des résultats de mesurage réels par la méthode de mesure utilisée. Et finalement, les forces et les faiblesses des différentes méthodes sont complémentaires.

Lemaitre *et al.* présentent dans [31] une nouvelle approche pour l'automatisation du calcul des points de fonctions basée sur les accès du logiciel à la base de données. La précision de cette méthode dépend du lien entre l'application et ses accès aux données qui se trouvent dans la base (i.e. Système d'exploitation : absence de SGBD / système de gestion d'une bibliothèque : possibilité de l'application de la méthode proposée). L'avantage principal de cette méthode, d'après les auteurs, est la facilité de son automatisation. Cette méthode est basée sur la méthode COSMIC.

L'hypothèse que le schéma logique permet de retrouver le schéma conceptuel d'une manière complètement automatisée, est considérée dans le cadre de cette recherche pour pouvoir

appliquer la méthode. Les auteurs proposent des règles théoriques qui seront utilisées pour l'automatisation FFP suivant deux principes :

1. Le principe de décomposition des prédicats. Il permet de décomposer n'importe prédicat SQL en un ensemble d'associations.

2. Le principe de l'association des variables : associer une variable provenant du langage ayant appelé SQL et se trouvant dans un des deux éléments d'une association avec l'ensemble des tables de l'autre élément de l'association.

Les requêtes et le schéma de la base de données sont utilisés en entrée du processus d'estimation du nombre de points de fonction du programme. Pour cela, le mapping entre les accès à la base de données et COSMIC via des définitions des fonctions et des règles de mesurage. Pour illustrer, les auteurs fournissent un exemple d'application à un système réel.

1.7 Méthodes de mesure avec extension

Pour pouvoir l'appliquer à un type de logiciel spécifique, certains chercheurs proposent leur propre extension d'un standard pour pouvoir s'approcher des notions d'un type spécifique de logiciel.

Dans [15], Felferning *et al.* illustrent comment la méthode IFPUG peut être appliquée à l'estimation des efforts de développement des systèmes de configuration expert (knowledge-based configuration systems). Le but des auteurs est d'étendre le champ d'application des estimations des logiciels (générales) à ce type particulier de logiciel parce que les auteurs considèrent que les caractéristiques particulières de ce type de système ne peuvent pas être transparentes dans les processus d'estimation de l'effort industriel de développement logiciel et donc avec les projets de développement logiciel traditionnels, vu que ce type de système a des caractéristiques spécifiques.

Ils présentent le processus d'estimation de l'effort s'appliquant au développement des systèmes de configuration en ajoutant une dimension de complexité additionnelle aux principes de la méthode, vu que l'approche conventionnelle FPA ne considère pas explicitement la complexité de la logique métier qui a une forte influence sur le temps et les coûts de développements. Afin d'examiner cet aspect important dans leur approche d'estimation, ils ont introduit de nouvelles règles (Business Rules).

Un cadre d'application pour une société spécifique a été présenté qui réduit les taux d'erreur de prédiction, selon les auteurs, par rapport à l'application des approches conventionnelles des Points de Fonctions. Aucune démonstration n'a été présentée pour montrer la validité de cette affirmation qu'en utilisant cette approche, les techniques d'estimation des efforts de développement conventionnel peuvent être intégrés dans les processus de développement des systèmes de configuration (systèmes experts).

1.8 Synthèse

Les tableaux 1.1, 1.2 et 1.3 récapitulent les principaux éléments retenus de la revue de la littérature sur l'automatisation de la mesure de la taille fonctionnelle:

Tableau 1.1 Tableau de synthèse des conclusions des études (1/3)

	Stambollian	**Mendes**	**Marín**	**Lamma**	**Fraternali**
Thème de l'étude	étude de marché / cadre référentiel	Evaluation des procédures de mesure	Procédure de mesure	Procédure de mesure	Procédure de mesure
Méthode de base	COSMIC	IFPUG	COSMIC	IFPUG	IFPUG
Domaine d'application	Divers	Divers	Informatique de gestion	Informatique de gestion	Informatique de gestion
Validation	N/A	N/A	N/A	N/A	N/A
Niveau de tests	N/A	Peu développés	Peu développés	Plusieurs et comparaisons	Peu développés
Type de l'approche	Utilization des dimensions et categories clés de la mesure	Identification des caractéristiques statiques et dynamiques	Up-bottom avec OO-Method	Up-bottom avec Utilisation des ER-DFD	Up-bottom avec UML
Outil de support	Non	Non	Oui- (prototype d'étude)	Non	Non
Référence #	[50]	[44]	[39]	[30]	[16]

Tableau 1.2 Tableau de synthèse des conclusions des études (2/3)

	Uemura	**Diab**	**Azzouz**	**Li**	**Bévo**
Thème de l'étude	Procédure de mesure	Procédure de mesure	Procédure de mesure suivant le processus RUP	Procédure de mesure	Ontology + Procédure d'automatisation
Méthode de base	IFPUG	COSMIC	COSMIC	COSMIC	COSMIC
Domaine d'application	Informatique de gestion	Temps réel	Informatique de gestion	logiciels interactifs	Divers
Validation	N/A	formelle	N/A	N/A	N/A
Niveau de tests	Peu développés	Peu développés	N/A	1 test	Peu développé
Type de l'approche	Up-bottom	Up-bottom avec ROOM	Up-bottom avec UML	Up-bottom avec XForms	Up-bottom avec UML
Outil de support	Oui	Oui-(prototype d'étude)	Prototype très limité	Prototype d'étude	Prototype
Référence #	[51]	[12] [14]	[4]	[33]	[6][7]

Tableau 1.3 Tableau de synthèse des conclusions des études (2/3)

	Ho	**April**	**Meli**	**Lemaitre**	**Felfernig**
Thème de l'étude	Procédure de mesure	Procédure d'évaluation précise des règles	étude / cadre référentiel	Procédure d'estimation	Procédure de mesure avec extension
Méthode de base	IFPUG	IFPUG	IFPUG	COSMIC	IFPUG
Domaine d'application	Informatique de gestion	Informatique de gestion	Informatique de gestion	Informatique de gestion	systèmes de configuration (systèmes experts
Validation	N/A	formelle	N/A	N/A	N/A
Niveau de tests	Pas de tests	N/A	N/A	N/A	N/A
Type de l'approche	Bottom-up avec COBOL	Bottom-up avec traduction physique-logique	ensemble d'attributs / paramètres, liés à l'objet / but de comparaison	Bottom-up avec SQL	Up-bottom
Outil de support	Non	Non	Non	Non	Non
Référence #	[19]	[2]	[42]	[31]	[15]

CHAPITRE 2

BUT, OBJECTIFS ET MÉTHODOLOGIE DE RECHERCHE

2.1 Introduction

Ce chapitre présente le but principal de la recherche et les objectifs choisis pour atteindre ce but. Une section intitulée 'Originalité des travaux envisagés' montre que les travaux de ce projet s'inscrivent dans l'innovation pour la mesure de la taille fonctionnelle des systèmes temps-réel et expose l'intérêt des résultats de ces travaux.

Ce chapitre présente aussi les différentes phases de la méthodologie de recherche nécessaires pour atteindre le but final, qui est l'automatisation de la mesure fonctionnelle des logiciels temps-réel en utilisant la méthode de mesure COSMIC – ISO 19761 et les objectifs spécifiques de recherche.

2.2 But et objectifs de la recherche

Le but principal de ce projet de recherche est l'automatisation de la mesure de la taille fonctionnelle des logiciels des systèmes réactifs, temps-réel et embarqués en utilisant leurs spécifications fonctionnelles exprimées par les outils de modélisations temps-réel utilisés en industrie.

Le but principal se divise en deux objectifs:

1. Le design des procédures de mesurages basées sur la méthode COSMIC ISO 19761 pour chacun des outils de modélisations utilisés chez Renault: Statemate et Simulink.
2. L'automatisation de ces procédures de mesurage et la réalisation d'un prototype de mesurage qui implémente ces procédures.

Ces tailles fonctionnelles seront utilisés avec l'historique de l'effort du (des) fournisseur(s) dans les développements antérieurs et les modèles de productivité correspondant pour estimer les efforts de développement des nouveaux logiciels et des évolutions de logiciels.

Le premier volet de la thèse (i.e. objectif 1) couvrira la création et l'application de procédures de mesurage basées sur la méthode de mesure fonctionnelle COSMIC – ISO 19761 aux spécifications fonctionnelles des logiciels des systèmes temps-réel, i.e. obtenir à partir des spécifications d'un logiciel temps-réel embarqué, sa taille fonctionnelle en mappant les principes de la méthode COSMIC avec les éléments des outils de modélisation utilisés chez Renault (i.e. StateMate et Simulink).

Les procédures de mesure fonctionnelle créées dans la première partie de cette thèse seront la base pour la deuxième partie de cette thèse qui est l'automatisation du mesurage de cette taille fonctionnelle (FSM) COSMIC (i.e. objectif 2).

2.3 Méthodologie de recherche

Pour atteindre les objectifs de ce projet de recherche, plusieurs étapes de recherche doivent être franchies. La méthodologie de recherche proposée ci-après présente les étapes, les phases et définit les objectifs pour chaque étape.

La vue générale du périmètre de la thèse est présentée à la figure 2.1.

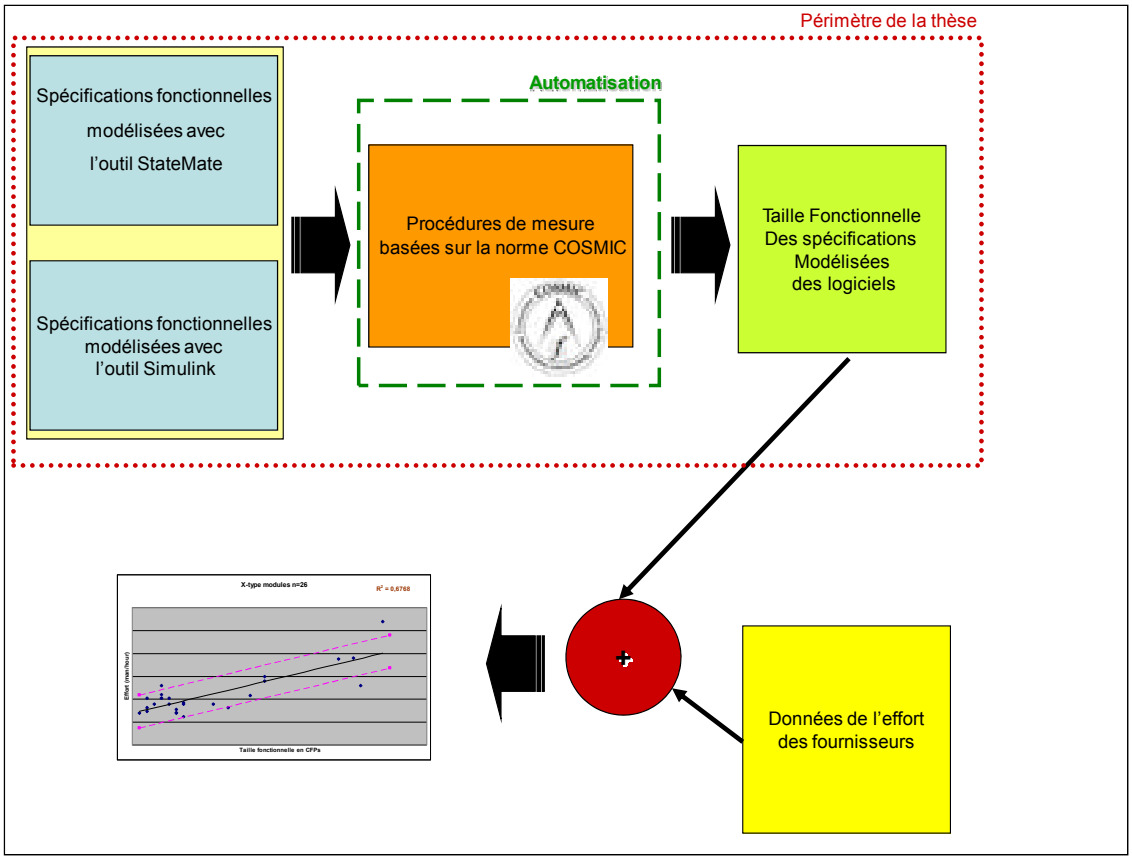


Figure 2.1 Périmètre du projet de recherche

Le périmètre présenté dans la figure 2.1 montre les deux axes de travail de ce projet de recherche:

1. Création des procédures de mesurage : les procédures de mesurages spécifiquement conçues pour ces outils sont appliquées sur les spécifications fonctionnelles complètes modélisées via un des outils de modélisations utilisés chez Renault,. Ces procédures sont construites pour répondre aux principes et à la terminologie de la méthode COSMIC version 3.0.1 et pour pouvoir appliquer le processus de mesurage clairement et sans ambigüité. Cette partie constitue la première partie de la thèse.

2. Automatisation des procédures de mesurages : l'automatisation est faite en passant par toute étape intermédiaire nécessaire, jusqu'à la phase de mesurage proprement dite -

application de la fonction mathématique - suivant la méthode COSMIC ISO 19761 - est appliquée pour obtenir la taille fonctionnelle du logiciel. Cette partie constitue la deuxième partie de la thèse.

Et comme retombée de ce projet de recherche : La taille fonctionnelle obtenue automatiquement est couplée avec les données de l'effort des fournisseurs après, pour obtenir des modèles de Simulation de productivité. Les modèles de productivité obtenus sont après utilisés dans le processus d'estimation de l'entreprise. Cette partie se situe hors du périmètre de la thèse.

2.3.1 Méthodologie de recherche

Ce projet de recherche est divisé en plusieurs phases. Chaque phase comprend au moins un processus, des entrées et des produits. La figure 2.2 montre les phases de la méthodologie de recherche avec les chapitres correspondants.

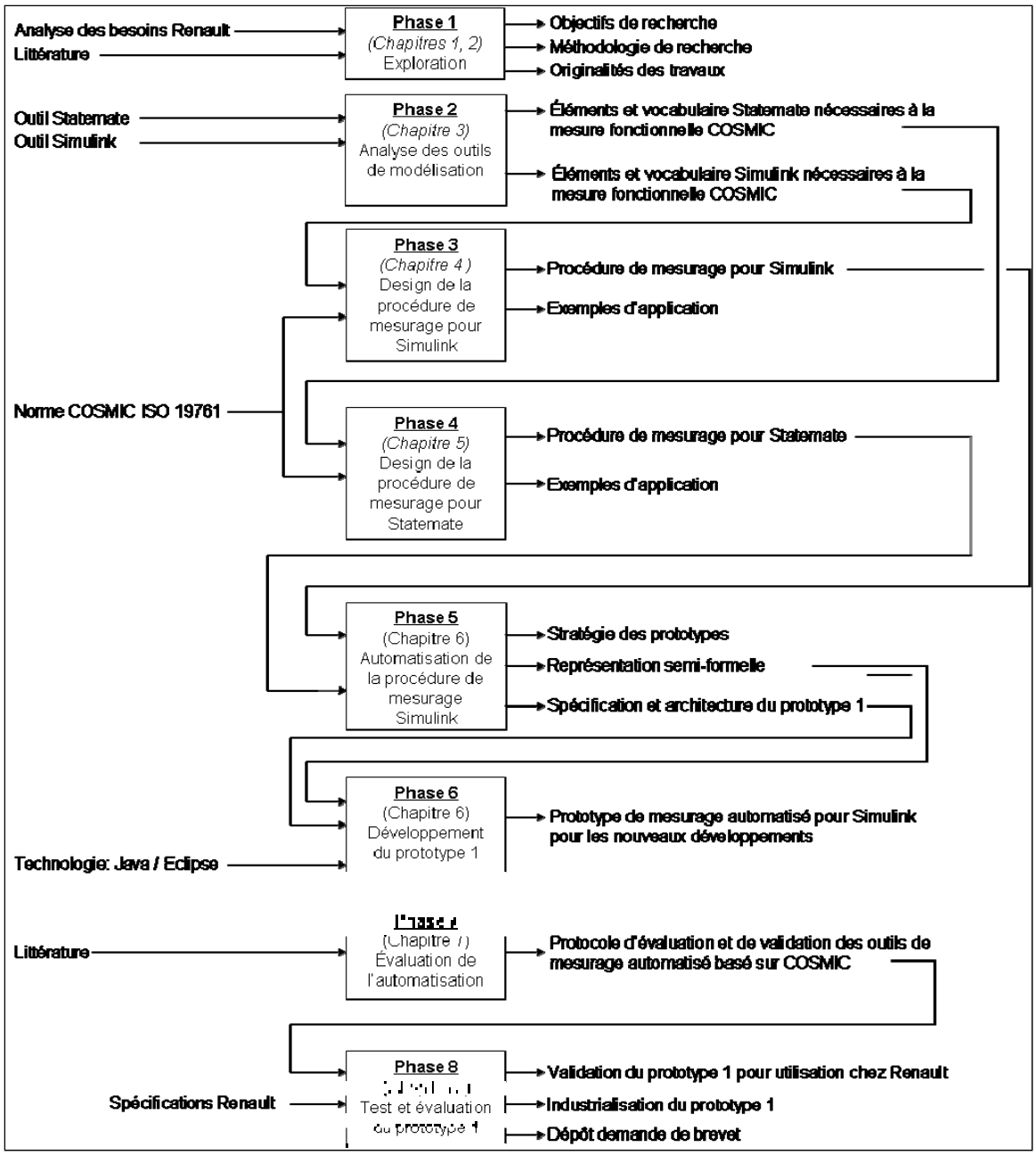


Figure 2.2 Les phases de la méthodologie de recherche

1. Phase 1 : Exploration

Le processus de cette phase consistait à analyser les documents de la littérature, et puis de synthétiser les contenus des publications pour pouvoir trier et utiliser les travaux utiles pour ce projet de recherche. D'autre part, comme ce projet de recherche s'inscrit dans le cadre d'un projet industriel chez Renault SA, les besoins spécifiques de Renault ont été analysés.

Le principal livrable intermédiaire produit à l'issue de ce processus sont les objectifs de la recherche. Dans ce livrable, on trouve la synthèse de toutes les étapes à suivre pour atteindre le but principal de la thèse, la synthèse de tous les travaux faits dans le domaine de ce projet de recherche ainsi que des publications et des propositions de démarche qui pouvaient éventuellement être utiles pour ce projet de recherche.

2. Phase 2 : Analyse des outils de modélisation

Dans cette phase l'analyse des outils de modélisation utilisés chez Renault a été faite dans le but de trouver tous les éléments importants et nécessaires pour l'application de la mesure fonctionnelle COSMIC. Les livrables de cette phase constituent les entrées pour les phases de design des procédures de mesurage développées dans ce projet de recherche.

3. Phase 3 : Design de la procédure de mesurage pour Simulink

Dans cette phase le design de la procédure de mesurage spécifique aux spécifications fonctionnelles modélisées avec l'outil de modélisation Simulink). Cette procédure est basée sur la méthode de mesure fonctionnelle COSMIC ISO 19761.

Les règles de mesure listées dans cette procédure constituent les étapes à suivre pour mesurer la taille fonctionnelle des nouveaux développements des logiciels ainsi que les

règles pour la mesure des évolutions des logiciels existants en utilisant leurs spécifications fonctionnelles en Simulink.

Finalement, des projets de nouveaux logiciels et des projets d'évolutions de versions de logiciels déjà existants sont mesurés avec les règles proposées (les règles pour les nouveaux développements et les règles pour les évolutions). Le résultat de ces mesures a été utilisé dans le processus d'estimation interne chez Renault.

4. Phase 4 : Design de la procédure de mesurage pour Statemate

Dans cette phase le design de la procédure de mesurage spécifique aux spécifications fonctionnelles modélisées avec l'outil de modélisation Statemate). Cette procédure est basée sur la méthode de mesure fonctionnelle COSMIC ISO 19761.

Les règles de mesure listées dans cette procédure constituent les étapes à suivre pour mesurer la taille fonctionnelle des nouveaux développements des logiciels ainsi que les règles pour la mesure des évolutions des logiciels existants en utilisant leurs spécifications fonctionnelles en Statemate.

Finalement, des projets de nouveaux logiciels et des projets d'évolutions de versions de logiciels déjà existants sont mesurés avec les règles proposées (les règles pour les nouveaux développements et les règles pour les évolutions). Le résultat de ces mesures a été utilisé dans le processus d'estimation interne chez Renault.

5. Phase 5 : Automatisation de la procédure de mesurage Simulink

Cette phase commence par l'étude de la faisabilité de l'automatisation des procédures proposées. La priorité pour Renault était d'automatiser le mesurage pour les spécifications Simulink. Donc, pour satisfaire à cette demande la stratégie des prototypes suivante a été adoptée :

Proto (n) = Proto (n-1) + X + Y
Où *Proto (n)* : la version *n* du prototype
Proto (n-1) : la version *n-1* du prototype
X : les éléments et les fonctionnalités ajoutés entre deux prototypes.
Y : les limitations éventuelles ou fonctionnalités qui ne peuvent par être développées pour des raisons diverses.

Le prototype 1 est celui pour le mesurage automatisé pour Simulink pour les nouveaux développements (phase 6). Et une fois le prototype 1testé, validé et déployé (phases 7 et 8), le prototype 2 devrait contenir les mêmes fonctionnalités du prototype 1 et en ajoutant, si possible, les fonctionnalités de mesurage pour les évolutions des spécifications Simulink (phase 9).

La représentation semi-formelle « commune » est un livrable important de la phase 5 : elle constitue un modèle qui montre tous les flux de données du logiciel modélisé et à mesurer. Donc à l'aide de ce modèle de mesure, le mesureur identifie tous les mouvements de données. Cette représentation semi-formelle a été utilisée lors de l'automatisation de la procédure de mesurage de la taille fonctionnelle du logiciel dont la spécification fonctionnelle est modélisée avec Simulink .

Cette représentation semi-formelle est dite «commune » parce qu'elle peut être construite à partir de différents outils ou langages de modélisation utilisés pour la production des spécifications fonctionnelles des logiciels des systèmes temps-réel embarqués. En d'autres termes, pour un module logiciel donné, sa représentation semi-formelle doit théoriquement être toujours la même, même s'il a été conçu avec un outil de modélisation X ou un outil Y: ceci comprend la condition que les spécifications soient au même niveau de détail: claires, cohérentes et non-ambigües.

La spécification fonctionnelle et l'architecture du prototype ont été le deuxième livrable de cette phase, ainsi que le processus du choix de la technologie, des outils, environnement et autres détails qui sont utilisés dans la phase 6 qui ont joué un rôle important dans la réalisation du prototype 1.

6. Phase 6 : Développement du prototype 1

Cette phase vise à créer l'outil-prototype d'automatisation de la procédure de mesure de la taille fonctionnelle pour les logiciels en nouveaux développements dont les spécifications sont modélisées avec Simulink. Pour cela, une première version - prototype - est construite au début de cette phase en utilisant le design conçu en phase 5. Un livrable de documentation sera aussi produit à l'issu du développement de ce prototype.

Ce prototype va être soumis à plusieurs tests et les résultats de ces tests seront documentés dans un autre livrable intermédiaire : donc une sous-étape de création et de sélection de tests aura lieu aussi dans cette phase.

7. Phase 7 : Évaluation de l'automatisation

La revue de littérature sur l'évaluation de l'automatisation des outils qui mesurent la taille fonctionnelle a montré que de telles études sont rares: en fait, un seul protocole d'évaluation proposé pour IFPUG [26] a été répertorié. Et il n'y a pas de protocoles spécifiques d'évaluation proposé pour les outils qui implémentent la méthode de mesure COSMIC.

Un protocole d'évaluation des outils de mesurage automatique de la taille fonctionnelle basés sur la méthode de mesure COSMIC a été proposé et documenté pour évaluer la performance de ce type d'outils. Ce protocole utilise des échantillons au niveau d'entrée de l'outil d'automatisation pour couvrir la plupart des types de cas d'entrée. Pour fins de

vérification, les échantillons doivent aussi être mesurés manuellement, en fonction des procédures FSM proposées.

8. Phase 8 : Test et évaluation du prototype 1

Cette phase constitue la mise en œuvre et l'utilisation du protocole d'évaluation proposé pour l'évaluation des outils d'automatisation basés sur COSMIC - ISO 19761. Le protocole a été appliqué sur un ensemble de 77 modèles de spécifications distinctes. Aux fins de vérification, le protocole comprend en parallèle des mesures manuelles de cet ensemble de modèle de spécifications. Différentes tailles de spécifications ont été choisies parmi un certain nombre de fonctions logicielles qui représentent différents ECM (Engine Control Module) du département dans lequel l'outil d'automatisation sera initialement utilisé. Après la validation du prototype 1, une demande de dépôt de brevet a été faite.

2.3.2 Originalité des travaux proposés

Pour démontrer cette originalité, il faut commencer par montrer les éléments non couverts par les travaux antérieurs de recherche.

D'après les besoins spécifiques du constructeur partenaire de ce projet de recherche, la revue de la littérature sur le sujet de la mesure fonctionnelle en générale et les travaux sur l'automatisation des procédures de mesures, on en tire les conclusions principales suivantes :

1. La mesure fonctionnelle est relativement récente, mais elle devient de plus en plus importante pour obtenir des modèles de productivité, des processus d'estimation, gestion de l'impartition, des budgets, etc.

2. la plupart des procédures de mesurage proposées dans la littérature concernent le domaine d'application de l'informatique de gestion.

3. la méthode COSMIC est la méthode de mesure la plus adaptée au domaine des logiciels temps-réel réactifs embarqués. Plusieurs procédures développées récemment sont basées sur COSMIC.

4. Les quelques rares travaux dans le domaine du temps-réel, ont été faits avec la version 2.0 de COSMIC.

5. Aucun projet de recherche avec la toute nouvelle version de COSMIC (version 3.0.1) n'a visé les applications temps-réel.

6. Le sujet de l'automatisation de la mesure fonctionnelle avec la méthode COSMIC est toujours dans une phase exploratoire : aucun outil n'a encore été proposé comme outil assez générique et qui automatise complètement le processus de mesure c.-à-d. le processus d'obtention de la taille fonctionnelle des logiciels. Des prototypes et des pistes ont été développés et présentés, mais ceux-ci ont plusieurs limitations. Certains nécessitent plusieurs interventions manuelles pour pouvoir obtenir les résultats et/ou ils n'ont pas encore été testés largement et indépendamment pour s'assurer de la précision des résultats qu'ils fournissent.

Ce projet de recherche adresse principalement les éléments non couverts ou sous-développés dans les projets antécédents :

- d'une part l'étude, la conception et la proposition de procédures de mesure de la taille fonctionnelle basées sur la version la plus nouvelle de COSMIC (version 3.0.1) pour les logiciels temps-réel en utilisant leurs spécifications fonctionnelles créées par des outils de modélisations utilisés en industrie.

- d'autre part, le principal défi de ce projet est l'automatisation de ces procédures, et plus précisément, l'automatisation du processus d'application de ces procédures.

Le processus de mesure consiste à appliquer les définitions et les règles de la procédure de mesure fonctionnelle proposée qui suit les étapes de la méthode COSMIC. L'application manuelle de ce processus consomme beaucoup d'effort et de temps, et les résultats obtenus peuvent être différents selon les mesureurs. L'outil attendu doit normalement éliminer un certain nombre de ces problèmes.

En effet pour les nouveaux développements, idéalement et théoriquement, le prototype doit automatiser toutes les étapes du processus de mesurage, c.-à-d. l'obtention automatique de la taille fonctionnelle d'un logiciel en ayant comme entrée sa spécification fonctionnelle. Toutefois pour arriver à cela, il faut couvrir toutes les étapes clés suivantes :

1. Identification des Utilisateurs Fonctionnels.
2. Détermination des Frontières.
3. Identification des Processus Fonctionnels.
4. Identification des Objets D'intérêts.
5. Identification des Mouvements des Groupes de Données.

Si les spécifications en entrée sont parfaites et complètes les étapes pourraient être automatisées à 100%. Mais ce n'est pas le cas car plusieurs informations importantes et nécessaires se trouvent dans d'autres documents (par exemple toute l'information sur les Utilisateur Fonctionnels ne se trouve pas toujours dans les spécifications fonctionnelles).

Un autre problème se pose au niveau des identifications des mouvements de données: dans cette étape un certain nombre (pourcentage) des mouvements peut être identifié sans problème. Les autres nécessitent une intervention humaine dans le cas où l'information nécessaire pour les déterminer est absente ou mal spécifiée; il y a également le cas où des limitations causées par l'implémentation du prototype lui-même existent..

Les résultats de ce projet seront utilisés pour répondre aux attentes de Renault : le processus de mesure est le cœur du processus de création des modèles de productivité qui, lui-même, est le cœur du processus de production des modèles d'estimation des futurs développements. Donc, ce projet de recherche cherche à répondre à des problèmes existants actuellement en industrie et en recherche, par rapport à la gestion du coût des développements des nouveaux logiciels des systèmes temps-réel et embarqués utilisés dans le secteur industriel de l'automobile et des évolutions de ces logiciels.

CHAPITRE 3

OUTILS DE MODÉLISATION ÉTUDIÉS ET FAISABILITÉ DE L'APPLICATION DE LA MÉTHODE COSMIC AUX MODÈLES DE CES OUTILS

3.1 Introduction

Dans ce chapitre, les outils de modélisations utilisés pour la spécification des logiciels temps réels sont présentés. Deux exemples explicatifs (un par outil) de modélisation sont aussi présentés ainsi que la faisabilité de l'application de la méthode COSMIC aux modèles de ces outils.

3.2 Aperçu de l'outil Simulink et faisabilité de l'application de la méthode COSMIC au modèle Simulink

Simulink est un outil Matlab [40] utilisé pour modéliser, simuler et analyser le comportement de systèmes dynamiques, notamment électrique, mécanique, thermodynamique et de systèmes embarqués temps-réel. Avec Simulink, on peut créer un modèle visuel du système à développer. Ensuite, il est possible de simuler le comportement de ce système.

En tant que concept de base principal, Simulink utilise l'approche des schémas de blocs pour modéliser un système dynamique de toute nature. Les «Blocs» sont donc les principaux éléments de Simulink. Qu'ils soient prédéfinis ou définis par l'utilisateur, dans un modèle ils sont des symboles qui peuvent se déplacer et changer. Les blocs Simulink ont des entrées et des sorties, ainsi que des paramètres configurables. Le bloc agit comme un système de base qui opère un certain traitement. Les paramètres peuvent influencer le comportement exact du bloc.

Les blocs peuvent (dans certains cas) avoir des états, i.e. les blocs Stateflow : l'état peut modifier la sortie de blocs selon une valeur qui est donnée en fonction de la valeur de l'état antérieur et/ou des entrées précédentes.

L’outil Simulink permet la création de sous-systèmes (i.e. blocs Subsystem), permettant ainsi une modélisation de systèmes dans des systèmes. Le type de bloc Subsystem encapsule un groupe de blocs et de signaux dans un seul bloc. Les entrées et sorties sont donc précisées lors de la construction du bloc. Simulink permet donc la modélisation hiérarchique, la gestion des données à l'aide de personnalisation sous-système et référencement de modèles.

Les blocs Simulink sont reliés entre eux par des «lignes» : ces lignes sont l'autre type d'éléments dans Simulink. Les lignes sont utilisées pour relier les différents blocs ensemble. Elles permettent de relier les sorties d'un bloc aux entrées d’autres blocs. Le comportement du système est donc décrit par la combinaison des blocs utilisés selon leurs types, en plus des flux de données qui sont prévus par les lignes. Ainsi, l'architecture du modèle tient également compte de son comportement.

Par conséquent, Simulink contient les détails nécessaires pour identifier et classer les fonctions du logiciel d'un module et le déplacement des données a lieu pendant le traitement d'une fonction.

3.2.1 Exemple d’un système modélisé avec Simulink

La figure 3.1 montre la modélisation des exigences fonctionnelles d'un simple système d'équations en utilisant l'outil Simulink: ce système simple possède deux sous-systèmes: Equation_1 et Equation_2.

Equation_1 exécute deux séries d'opérations: une fois déclenchée, la première série d'opérations consiste à ajouter 5 à la valeur de B, puis en multipliant le résultat avec la valeur de A. Le résultat final est obtenu sous forme de la valeur de C. L'équation mathématique de cette série est: C = A x (B +5).

La deuxième série d'opérations consiste à ajouter 5 à la valeur de B1 puis en multipliant le résultat avec la valeur d'A1. Mais au lieu d'avoir le résultat final immédiatement, comme dans la première série, la spécification fonctionnelle précise que le résultat doit être écrit dans la mémoire par l'intermédiaire de la «variable» M. L'équation mathématique de cette série est: M = A1 x (B1 +5).

Les blocs Simulink utilisé pour modéliser Equation_1, tel que présenté dans la figure 3.2, sont les suivants: 4 blocs Inport (A et B, A1 et B1), 1 bloc Outport (C), 2 blocs constants (Constant1, Constant2), 2 blocs de Somme (Add, Add1), 2 blocs de produit (Product, Product1), 1 bloc Data Store Write (Data Store Write) et 1 bloc Triggerport (Sched). Et tous les blocs sont mis à l'intérieur d'un bloc sous-système appelé Equation_1, comme le montre la figure 3.1.

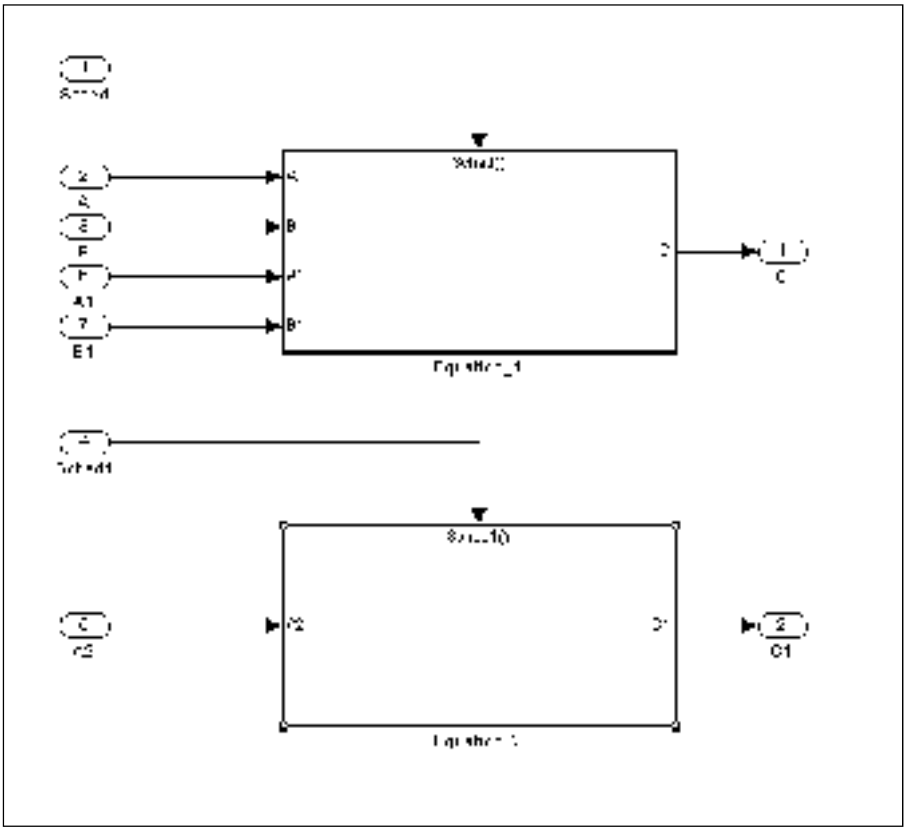


Figure 3.1 Les blocs Subsystem: Equation_1 and Equation_2

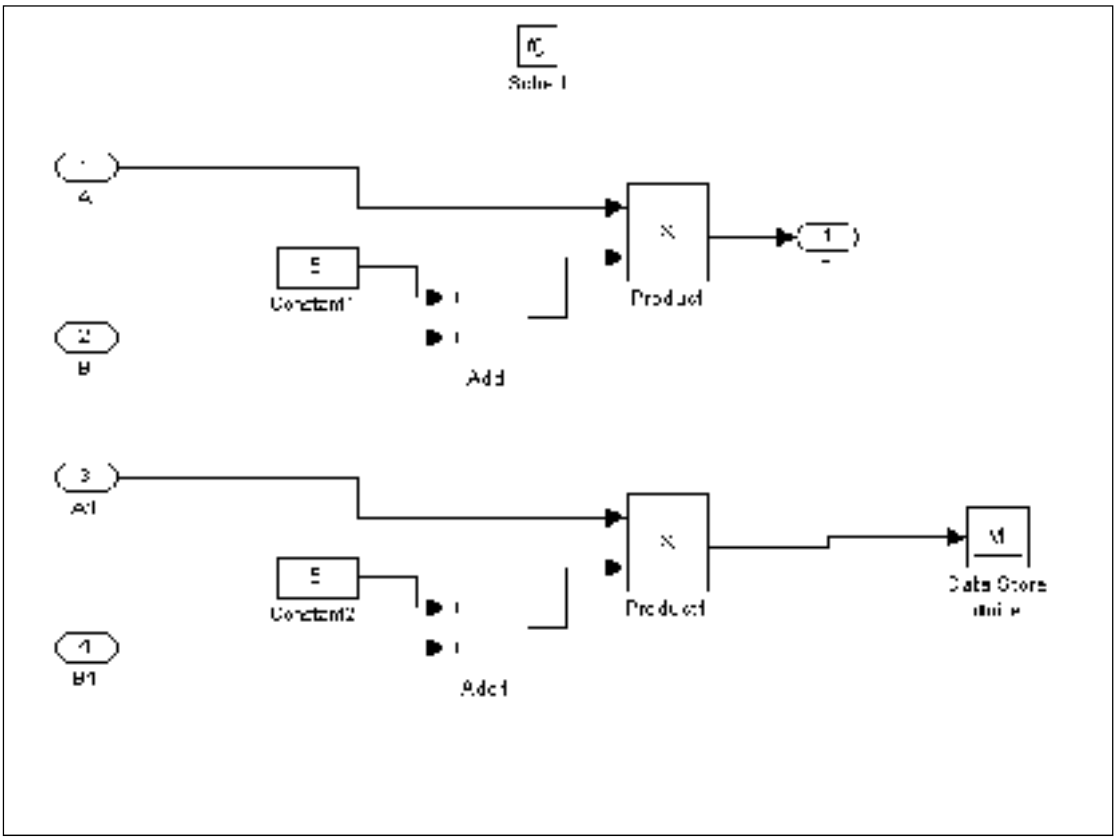


Figure 3.2 A l'intérieur du bloc subsystem 'Equation_1'

Equation_2, une fois déclenchée, consiste à ajouter la valeur de M (en mémoire) à la valeur de A2. Le résultat final de cette somme est obtenu en tant que la valeur de C1. L'équation mathématique de cette série modélisée est: C1 = A2 + M.

Les blocs Simulink utilisé pour modéliser cette équation (equation_2), tel que présenté dans la figure 3.3, sont les suivants: 1 bloc Inport (A2), 1 bloc Outport (C1), un bloc Sum (ADD2), 1 bloc Data Store Read (Data Store Read) et 1 bloc Triggerport (Sched1). Et tous les blocs sont mis à l'intérieur d'un bloc sous-système appelé Equation_2, comme le montre la figure 3.1.

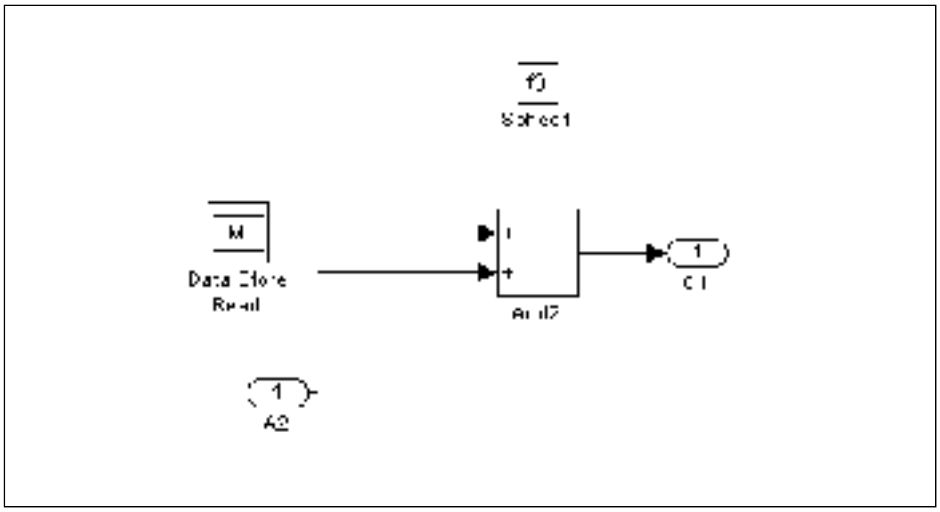


Figure 3.3 A l'intérieur du bloc subsystem 'Equation_2'

3.3 Aperçu de l'outil Simulink et faisabilité de l'application de la méthode COSMIC au modèle Statemate

Statemate est un outil graphique de conception, de simulation et de prototypage pour le développement rapide des systèmes embarqués complexes. Il offre la possibilité de spécifier les systèmes sans ambiguïté et il permet alors de créer des modèles graphiques des systèmes, basés sur des diagrammes d'ingénierie standards. Statemate montre les états actifs, les fonctions, et les scénarios pendant les simulations, et il permet d'analyser les spécifications afin de s'assurer que leur comportement est correct et il permet la capture des données des tests

L'élément principal de base du modèle Statemate s'appelle «activité». Une activité est un schéma hiérarchique des flux de données dans lequel se trouvent les capacités fonctionnelles ou la description du système. Des données et des signaux circulent entre les différentes activités d'un modèle. Les capacités fonctionnelles décrites dans les «activités» sont dynamiquement non exclusives. En d'autres termes, les boîtes "activité" montrent que des activités existent, qu'elles peuvent être déclenchées pour exécuter des fonctionnalités, que des données peuvent circuler, ainsi de suite, mais elles ne précisent pas ce qui va arriver, ni quand ni pourquoi.

Une "activité de contrôle" reflète les aspects comportementaux d'une "activité". Un "activité de contrôle" contrôle la dynamique des activités et leurs flux de données, à savoir une "Activité de contrôle" est capable d'activer et de désactiver les activités, d'envoyer des signaux, de causer l'écriture, la modification, et la lecture des données. "Activité de contrôle" est responsable aussi de détecter quand de telles opérations arrivent. Les "activités de contrôle" peuvent être spécifiées en utilisant des diagrammes d'états ou en utilisant des "Minispecs", ou en utilisant des diagrammes d'états et des "Minispecs" simultanément. Une Minispec est un pseudo-code qui fournit des informations non-graphiques sur des éléments. Si une "Activité de contrôle" est décrite en utilisant un diagramme d'états, ou un diagramme d'états et une "Minispec" simultanément, une boîte "Activité de contrôle" avec un symbole «@» qui précède son nom se trouvera dans la boîte de l''activité'. Si une "Activité de contrôle" est décrite en utilisant uniquement une Minispec, il n'y aura pas de boîte de "Activité de contrôle" dans la boîte activité. (Il y a un signe ">" qui suivra le nom de l'activité).

Comme le modèle Statemate est basé sur les "Activités" et les "activités de contrôle", le design peut être vu via deux points de vue différents: un point de vue structurel et un point de vue comportemental.

- le point de vue structurel est le diagramme des "Activités": c'est le squelette du modèle et il s'agit d'un schéma hiérarchique montrant les flux de données entre les «Activités externes" et le module (appelé Top Activité).
- le point de vue comportemental est au niveau des "activités de contrôle".

Pour rappel, une "Activité de contrôle" contrôle la dynamique de son activité: envoyer et recevoir des données, activer et désactiver les autres activités, etc. C'est à ce niveau là où les mouvements de données ont lieu. Par conséquent, Statemate contient les détails nécessaires pour identifier et classer les fonctionnalités logicielles d'un module et les mouvements de données qui se déroulent pendant le traitement d'une fonctionnalité.

La manière précise dont les Statecharts et les «Mini-Specs » décrivent le comportement, et donc contrôlent le comportement de l'intégralité des activités et des données au fil du temps, est au cœur du modèle du système. Cette description est faite avec un ensemble d'expressions. Les expressions dans Statemate prennent la forme de trigger/action:

- un 'trigger' est un événement (event) et/ou une condition qui définit les critères pour un changement dans l'état du système.

- une 'action' précise ce qu'il faut faire à la suite de l'occurrence d'un déclencheur.

Cependant, d'après nos connaissances, il n'y a pas de schéma Statemate qui montre l'interaction complète entre les différentes activités de contrôle (ni pour les activités non plus) à l'intérieur d'un module. Par conséquent, il est nécessaire de "décortiquer" le modèle du module logiciel afin d'identifier tous les mouvements internes de données (à l'intérieur de l'Activité Top).

3.3.1 Exemple d'un système modélisé avec Statemate

La figure 3.4 montre la conception et la structure d'un système simple temps-réel en utilisant l'outil Statemate: une simple horloge telle que présentée dans I-Logix MAGNUM Tutorial 2001 [20]. L'horloge affiche l'heure de la journée en heures et minutes. Elle dispose également d'un mode "SET TIME" permettant à l'utilisateur de régler l'heure de la journée grâce à deux boutons dans le panneau de saisie. Un autre bouton dans le panneau est le bouton MODE qui donne à l'utilisateur la possibilité de sélectionner le mode de l'horloge. L'horloge a aussi un écran qui reste vide lorsque le courant n'est pas disponible. Lorsque le courant revient, l'horloge se met automatiquement en " SET TIME " mode.

La conception de l'exemple d'horloge montre trois activités en orange (l'activité Top (CLOCK) et deux autres "sous"-activités (KEEP_TIME et SET_TIME)).

Le CLOCK_CNTL et le KEEP_TIME_BHVR sont les activités de contrôle (en vert) respectivement de l'activité CLOCK et de l'activité KEEP_TIME. Elles sont décrites à l'aide des diagrammes d'états. L'activité de contrôle de l'activité SET_TIME est décrite via une Minispec.

Les deux «Activités externes » avec lesquelles CLOCK interagit sont les INPUT_PANEL et DISPLAY (boîtes en pointillé).

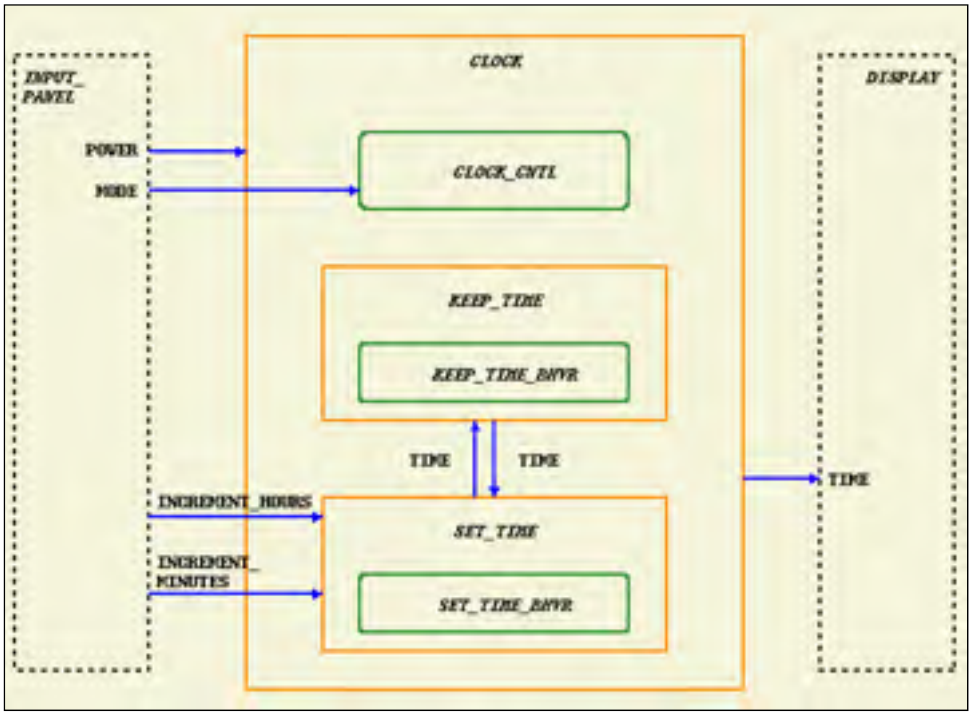


Figure 3.4 Le système d'horloge comme présenté
dans I-Logix MAGNUM Tutorial 2001

La figure 3.5 montre l'activité de contrôle 'CLOCK_CNTL' qui contrôle le comportement général de l'horloge : quand l'horloge est mise sous tension, le système entre par défaut dans le mode de réglage (i.e. l'activité de contrôle SET_TIME_BHVR). Et la valeur de 'Mode' permet à l'utilisateur de basculer entre les deux modes de l'horloge (mode de réglage et mode normal).

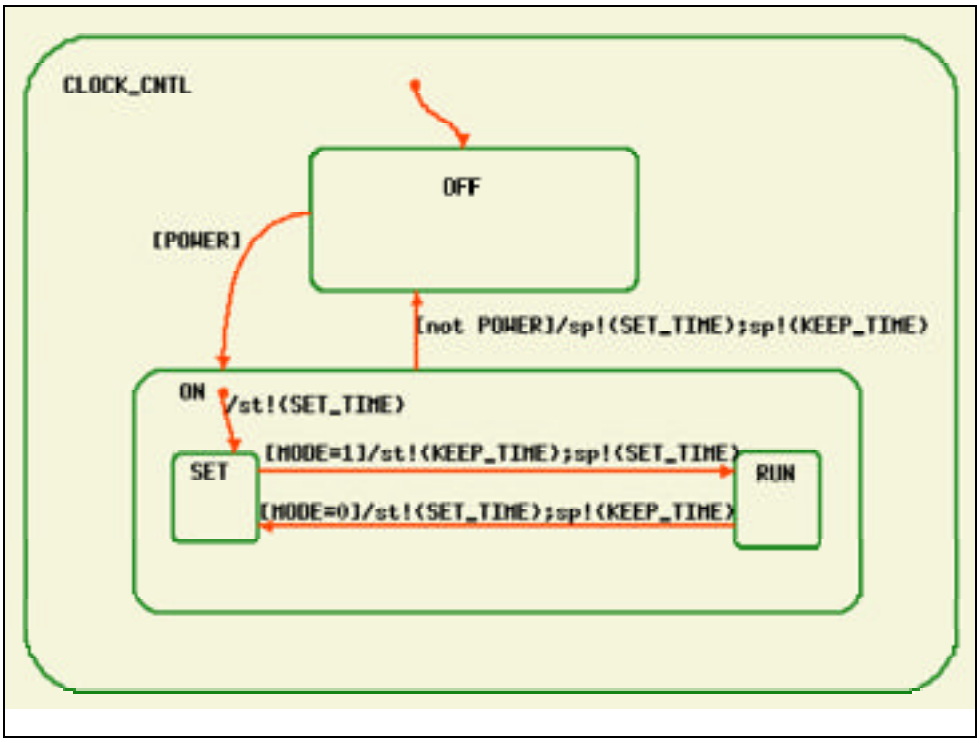


Figure 3.5 L'activité de contrôle CLOCK_CNTL

La figure 3.6 présente l'activité de contrôle qui permet à l'horloge de fonctionner dans son mode normal de mise à jour du temps pour qu'il soit correctement affiché à l'écran en heures et en minutes. Elle incrémente le compteur des minutes jusqu'à 59, et à la prochaine incrémentation, elle met à jour le compteur des heures et fait passer celui des minutes à 0. D'autre part, le compteur des heures est incrémenté jusqu'à 12 et puis il est mis à zéro.

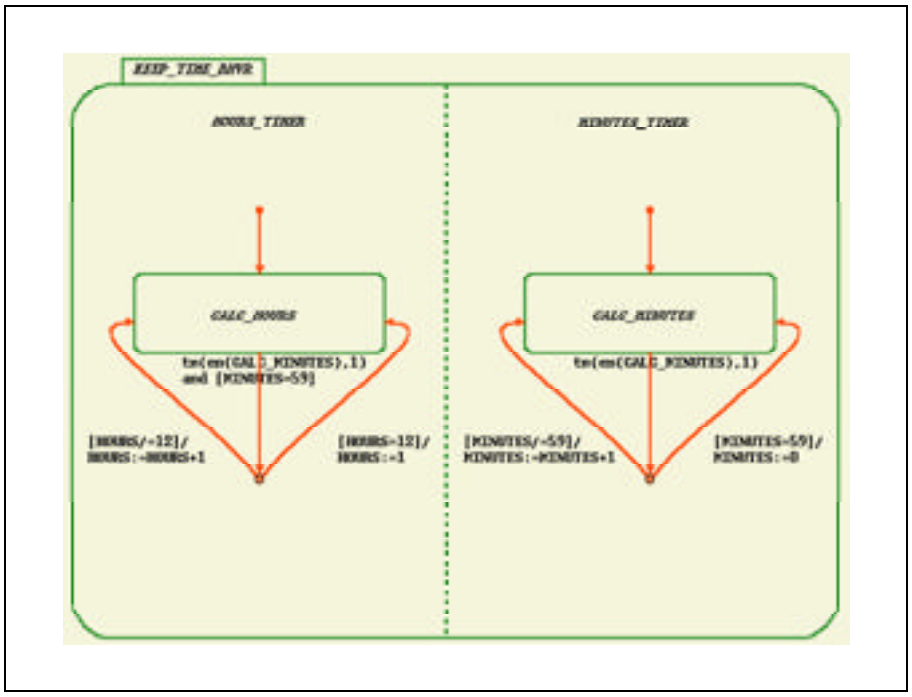


Figure 3.6 L'activité de contrôle KEEP_TIME_BHV

La figure 3.7 présente l'activité de contrôle qui permet à l'horloge de fonctionner dans son mode de réglage. Elle permet l'incrémentation du compteur des heures et celui des minutes par les boutons du panel. L'incrémentions des minutes est possible jusqu'à 59, et à la prochaine incrémentation il passe à 0. D'autre part, le compteur des heures est incrémenté jusqu'à 12 et puis il est mis à zéro.

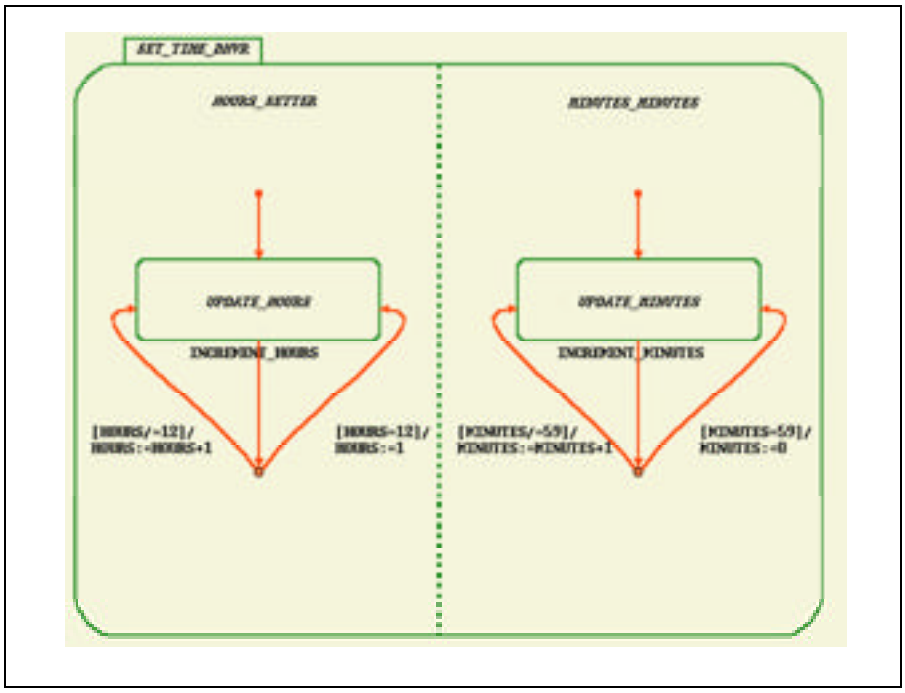


Figure 3.7 L'activité de contrôle SET_TIME_BHVR

.

CHAPITRE 4

PROCÉDURE DE MESURAGE DE LA TAILLE FONCTIONNELLE POUR LES LOGICIELS CONÇUS AVEC L'OUTIL DE MODÉLISATION SIMULINK

4.1 Introduction

Ce chapitre propose une procédure de mesure de la taille fonctionnelle: cette procédure est basée sur la méthode COSMIC V 3.0.1 et elle est conçue pour les logiciels embarqués temps réel dont les spécifications sont documentées en utilisant l'outil de modélisation Simulink. Cette procédure de mesurage couvre à la fois les nouveaux développements des logiciels et les évolutions des logiciels. La taille fonctionnelle est obtenue par le mapping des concepts élémentaires définis dans Simulink et ceux de la méthode COSMIC. Cette procédure de mesurage a comme but d'éliminer toute variance éventuelle de mesurage qui peut être causée par une mauvaise application des principes de la méthode COSMIC par les mesureurs et qui peut donc conduire à des résultats différents pour une même spécification. Cette procédure fournit également une base pour l'automatisation du processus de mesure pour les spécifications Simulink.

4.2 Une procédure de mesurage pour les systèmes temps-réels conçus avec Simulink

Afin de bien mesurer la taille fonctionnelle des logiciels dont les spécifications fonctionnelles sont modélisées par l'outil Simulink, l'objectif de la mesure, son champ d'application et d'autres éléments doivent être bien identifiés et définis. La procédure de mesurage devrait également avoir un ensemble de règles de mesure, appliquée aux modules afin d'obtenir leurs dimensions fonctionnelles. Ces règles doivent être claires et cohérentes afin d'avoir des résultats précis, rendre le processus de mesure simple et permettre l'automatisation future de la procédure. Cette procédure de mesurage est basée sur la version COSMIC 3.0.1 (dernière version) et ses trois phases de mesurage. Ainsi, dans ce chapitre, les éléments tels que définis dans la méthode COSMIC sont mappés aux éléments du modèle Simulink. Nous

avons choisi d'introduire une catégorie appelée «blocs élémentaires» qui est nécessaire pour la conception de cette procédure de mesurage.

Une liste des dits «blocs élémentaires» est fournie ci-après. Un bloc élémentaire est utilisé afin d'identifier un processus fonctionnel. Les blocs élémentaires exécutent des fonctionnalités élémentaires et/ou des opérations qui transforment les données d'entrée des blocs en données de sortie. Les blocs élémentaires sont toutes les catégories que l'on trouve dans la bibliothèque Matlab Simulink bloc R2006b :

1. Liste des blocs Sink dans la librairie Matlab Simulink R2006b:

{Display; Floating Scope; Out; Scope; Stop Simulation; To File; Terminator; To Workspace; XY Graph}

2. Liste des blocs Source dans la librairie Matlab Simulink R2006b:

{ Signal Generator ; Ramp ; Sine Wave; Step; Chirp Signal; Pulse Generator; Repeating Sequence; Clock; Constant; Digital Clock; From File; From Workspace; Random Number; Band-Limited White Noise }

3. Les blocs élémentaires sont toutes les catégories suivantes de blocs, que l'on trouve dans la librairie Matlab Simulink R2006b:

{Continuous; Discontinuities; Discrete; Logic And Bit Operations; Lookup Tables; Math Operations; Model Verification ; Model_Wide Utilities; Signal Attributes; Signal Routing; User Defined Functions; Additional Math And Discrete}.

4.2.1 La phase de stratégie de mesure

Raison d'être de la procédure :

L'objectif de cette procédure de mesurage est d'appliquer la méthode COSMIC au modèle Simulink, pour mesurer la taille des FUR d'un logiciel basé sur les spécifications fonctionnelles décrites en utilisant le modèle Simulink.

La taille des FUR sera utilisée pour estimer l'effort de développer des logiciels (conçus en utilisant Simulink).

Périmètre de la mesure :

Le périmètre de cette procédure est au niveau du détail des sous-systèmes (Blocs Subsystem) du modèle Simulink.

Niveau de granularité :

Le niveau de granularité est au niveau des blocs du modèle Simulink.

Les utilisateurs fonctionnels :

Les utilisateurs fonctionnels sont tous les logiciels externes (Blocs Subsystem - Root Systems) ou utilisateurs humains, qui interagissent avec le logiciel à mesurer.

4.2.2 La phase de mapping

Le tableau 4.1 résume l'ensemble des correspondances entre les éléments COSMIC et les éléments Simulink.

Processus fonctionnels :

Un sous-système contenant un bloc élémentaire est un processus fonctionnel: une fois déclenché, il reçoit, manipule et déplace des données externes et les données internes également. Les blocs Stateflow sont aussi des processus fonctionnels.

Frontières :

Il ya une frontière entre un logiciel externe (utilisateur fonctionnel) et le logiciel à mesurer (système). Il y a aussi une frontière entre deux sous-systèmes (composants homologues au niveau de la même couche).

Groupes de données :

Identifier les groupes de données est un élément clé pour l'identification correcte des mouvements des groupes de données dans chaque processus fonctionnel. L’hypothèse que chaque ligne transporte un groupe de données unique est considérée.

Tableau 4.1 Règles de mapping Simulink/COSMIC

COSMIC element	Rule n°	Rule description
Functional user	1	Identify 1 functional user for each external subsystem that interacts (sends and/or receives data to/from) any FP identified according to the rules #4 & #5
Boundary	2	Identify 1 boundary between any external subsystem interacting with the to-be-measured subsystem
Boundary	3	Identify 1 boundary between two functional processes interacting with each other
Functional process	4	Identify 1 functional process for each subsystem containing at least one elementary block
Functional process	5	Identify 1 functional process for each StateFlow Block

4.2.3 La phase de mesurage : Mesurage de la taille fonctionnelle d'un "nouveau" logiciel

Dans cette section, les règles d'identification des mouvements de données d'un logiciel conçu en utilisant le modèle Simulink sont présentées: les règles qui sont utilisées pour mesurer la taille fonctionnelle d'un nouveau logiciel sont expliquées.

Les mouvements de données de chaque processus fonctionnels sont identifiés dans cette phase. Une fois que tous les mouvements de données dans un processus fonctionnels sont identifiés, l'unité de mesure de 1 CFP est attribuée à chaque mouvement de données. La dernière étape, comme présentée dans le tableau 4.4 consiste à additionner les résultats pour obtenir la taille fonctionnelle de chaque processus fonctionnels (règle 18).

La taille fonctionnelle des processus fonctionnels sont ensuite agrégés pour obtenir la taille fonctionnelle du module mesuré (règle n° 19).

Identification des mouvements de groupes de données :

Les règles pour l'identification correcte des 4 types de mouvement de groupes de données COSMIC sont présentées dans les tableaux 4.2, 4.3 et 4.4. Les déclencheurs COSMIC sont identifiés en utilisant la règle n ° 6. Les mouvements de groupes de données COSMIC d'entrée (E) et de sortie (X) sont identifiés respectivement, en utilisant le type de bloc Source Port (règle 7) et le type de bloc Sink (règle 9). Les mouvements de groupe de données de Lecture (R) et d'écriture (W) sont liés aux blocs DataStoreRead et DataStoreWrite respectivement (règles 11 et 12). Ils sont également identifiés si le processus fonctionnel mesuré est un bloc Stateflow (règles 16 et 17).

Tableau 4.2 Règles pour identifier les mouvements de groupes de données d'un logiciel modélisé avec Simulink (cas des blocs non-StateFlow)

COSMIC element	**Rule n°**	**Rule description**
Data group movements	6	Identify 1E data movement -and COSMIC triggering event- for the **TriggerPort, EnablePort or Function-Call Generator**
Data group movements	7	Identify 1E data movement for each **InPort** (or any other **Source Port**) connected via a **line** to any **elementary block** inside this subsystem.
Data group movements	8	Identify 1E data movement for each **line** coming from a **Subsystem** (identified as a functional process) (line's source) and connected to an elementary block (line's destination)
Data group movements	9	Identify 1X data movement for each **OutPort** (or any other **Sink Port**) connected via a **line** to any elementary block in this functional process
Data group movements	10	Identify 1X data movement for each **line** going to a **Subsystem** (identified as a functional process) (destination) and coming from an elementary block (source) inside this FP
Data group movements	11	Identify 1R data movement for each **DataStoreRead** identified in this functional process and coming from an elementary block (source) inside this FP
Data group movements	12	Identify 1W data movement for each **DataStoreWrite** identified in this functional process and connected to an elementary block (line's destination)

.

Tableau 4.3 Règles pour identifier les mouvements de groupes de données d'un logiciel modélisé avec Simulink (cas des blocs StateFlow)

COSMIC element	**Rule n°=**	**Rule description**
Data group movements	13	Identify 1E and COSMIC triggering event data movement for an **event** starting this Functional Process
Data group movements	14	Identify 1E data movement for each **Data object** that is a*n input to the chart from a Simulink block* and used by a **condition** in this Stateflow chart
Data group movements	15	Identify 1X data movement for each **Data object** that is a*n output from the chart to a Simulink block* and used by an **action** in this Stateflow chart
Data group movements	16	Identify 1R data movement for each **Data object** that is local to the Stateflow chart and used by a **condition** in this Stateflow chart
Data group movements	17	Identify 1W data movement for each **Data object** that is local to the Stateflow chart and used by an **action** in this Stateflow chart

Tableau 4.4 Règles pour obtenir la taille fonctionnelle
des processus fonctionnels et du logiciel

COSMIC element	Rule n°=	Rule description
Functional Process	18	Aggregate the CFP related to the data movements identified in a specific FP to obtain the functional size of that process.
Whole Software	19	Aggregate the CFP related to the data movements of (identified in) the functional processes of (identified in) the whole system to obtain the functional size of that system.

4.3 Application de la procédure à l'exemple du système d'Équations

Dans cette section, la procédure de mesure proposée est appliquée à l'exemple du système d'équations présenté au chapitre 3.

- **Les processus fonctionnels du système :** Selon la règle n°1, il existe 2 sous-systèmes qui contiennent au moins 1 bloc élémentaire dans l'exemple: Equation_1 et Equation_2 donc il ya seulement 2 processus fonctionnels dans cet exemple. La prochaine étape consiste à obtenir la taille fonctionnelle de chacun des processus fonctionnels identifiés.

- **La taille fonctionnelle du processus fonctionnel 'Equation_1':** Ce processus fonctionnel est décrit, comme présenté dans la figure 3.2, en utilisant 4 blocs Inport, 1 bloc Outport, 2 blocs Constant, 4 blocs élémentaires (2 blocs Sum et 2 blocs Product) et 1 bloc Data Store Write. Il est déclenché par un bloc Triggerport. Le tableau 4.5 montre tous les mouvements de groupes de données identifiés dans ce processus fonctionnels. La taille totale du processus fonctionnel Equation_1 est de 9 CFP.

Tableau 4.5 Détail de la taille fonctionnelle totale du processus fonctionnel Equation_1

Number of the applied rule	**Type(s) of the identified block (s)**	**Name(s) of the identified block(s)**	**Data movement type**	**CFP value**
6	Triggerport	Sched	E	1
7	Inport ; Product	A; Product	E	1
7	Inport ; Sum	B; Add	E	1
7	Constant ; Sum	Constant1; Add	E	1
9	Outport; Product	C; Product	X	1
7	Inport ; Product	A1; Product1	E	1
7	Inport ; Sum	B1; Add1	E	1
7	Constant ; Sum	Constant2; Add1	E	1
12	Data Store Write; Product	M	W	1
				Total: 9 CFP

- **La taille fonctionnelle du processus fonctionnel 'Equation_2':** Ce processus fonctionnel est décrit, comme présenté dans la figure 3.3, en utilisant 1 bloc Inport, 1 bloc Outport, un bloc Constant, 1 bloc élémentaire (un bloc Sum) et 1 bloc Data Store Read. Il est déclenché par un bloc Triggerport. Le tableau 4.6 montre tous les mouvements de données identifiées dans ce processus fonctionnels. La taille totale du processus fonctionnel Equation_2 est de 4 CFP.

Tableau 4.6 Détail de la taille fonctionnelle totale du processus fonctionnel Equation_2

Number of the applied règle	**Type(s) of the identified block (s)**	**Name(s) of the identified block(s)**	**Data movement type**	**CFP value**
6	Triggerport	Sched1	E	1
7	Inport ; Sum	A2; Add2	E	1
9	Outport; Sum	C1; Add2	X	1
11	Data Store Read; Product	M	R	1
				Total: 4 CFP

- **La taille fonctionnelle du système:** Afin d'obtenir la taille fonctionnelle de l'ensemble du système, les tailles de tous les processus fonctionnels du module sont agrégées (règle n ° 27) :

  Taille (système d'équations) = Taille (Equation_1) + Taille (Equation_2) = 13 CFP.

.

CHAPITRE 5

PROCÉDURE DE MESURAGE DE LA TAILLE FONCTIONNELLE POUR LES LOGICIELS CONÇUS AVEC L'OUTIL DE MODÉLISATION STATEMATE

5.1 Introduction

Ce chapitre propose une procédure de mesure de la taille fonctionnelle : cette procédure est basée sur la méthode COSMIC V 3.0.1, et elle est conçue pour les logiciels embarqués temps réel dont les spécifications sont documentées en utilisant l'outil de modélisation Statemate. Cette procédure de mesurage couvre à la fois les nouveaux développements des logiciels et les évolutions des logiciels. La taille fonctionnelle est obtenue par le mapping des concepts élémentaires définis dans Statemate et ceux de la méthode COSMIC. Cette procédure de mesurage a comme but d'éliminer toute variance éventuelle de mesurage qui peut être causée par une mauvaise application des principes de la méthode COSMIC par les mesureurs et qui peut donc conduire à des résultats différents pour une même spécification. Cette procédure fournit également une base pour l'automatisation du processus de mesure pour les spécifications Statemate.

5.2 Une procédure de mesurage pour les systèmes temps-réels conçus avec Statemate

Afin de bien mesurer la taille fonctionnelle des logiciels dont les spécifications fonctionnelles sont modélisées par l'outil Statemate, l'objectif de la mesure, son champ d'application et d'autres éléments doivent être bien identifiés et définis. La procédure de mesurage devrait également avoir un ensemble de règles de mesure, appliquée aux modules afin d'obtenir leurs dimensions fonctionnelles. Ces règles doivent être claires et cohérentes afin d'avoir des résultats précis, rendre le processus de mesure simple et permettre l'automatisation future de la procédure. Cette procédure de mesurage est basée sur la version COSMIC 3.0.1 (dernière version) et ses trois phases de mesurage. Ainsi, dans ce chapitre, les éléments tels que définis dans la méthode COSMIC sont mappés aux éléments du modèle Statemate.

5.2.1 La phase de stratégie de mesure

Raison d'être de la procédure :

L'objectif de cette procédure de mesurage est d'appliquer la méthode COSMIC au modèle Statemate, pour mesurer la taille fonctionnelle d'un logiciel en utilisant sa spécification fonctionnelle décrite par le modèle Statemate. La taille obtenue sera utilisée pour estimer l'effort de développer des logiciels conçus en utilisant Statemate.

Périmètre de la mesure :

Le périmètre de cette procédure de mesure est au niveau du détail des «activités de contrôle» (Control Activity) du modèle Statemate.

Niveau de granularité :

Le niveau de granularité est au niveau des « activités de contrôle » (Control Activity) du modèle Statemate.

Les utilisateurs fonctionnels :

Les utilisateurs fonctionnels sont toutes les activités externes interagissant (envoi et réception de données) avec le logiciel en question.

5.2.2 La phase de mapping

Le tableau 5.1 résume l'ensemble des correspondances entre les éléments COSMIC et les éléments Statemate.

Processus fonctionnels :

Une activité de contrôle est un processus fonctionnel, puisque quand elle est déclenchée, elle reçoit, manipule et déplace des données externes (de/vers d'autres activités de contrôle et/ou activités extérieures), et également parfois, des données internes (données créées et utilisées exclusivement dans le processus).

Frontières :

Il y a une frontière située entre les activités externes (qui sont les utilisateurs fonctionnels) et le logiciel à mesurer (Top Activity). Il ya aussi une frontière entre deux activités de contrôle.

Groupes de données :

Identifier les groupes de données est un élément clé pour l'identification correcte des mouvements des groupes de données dans chaque processus fonctionnel. L'hypothèse que chaque variable représente un groupe de donnée a été fixée.

Tableau 5.1 Les règles de mapping COSMIC et Statemate

COSMIC element	**Rule n°**	**Rule description**
Functional user	1	Identify 1 functional user for each External Activity that sends and/or receives data from any Control Activity
Boundary	2	Identify 1 boundary between the module and any external activity interacting with it
Boundary	3	Identify 1 boundary between two control activities interacting with each other
Functional process	4	Identify 1 functional process for each Control Activity

5.2.3 La phase de mesurage : Mesurage de la taille fonctionnelle d'un "nouveau" logiciel

Dans cette section, les règles d'identification des mouvements de données d'un logiciel conçu en utilisant le modèle Statemate sont présentées: les règles qui sont utilisées pour mesurer la taille fonctionnelle d'un nouveau logiciel sont expliquées.

Les mouvements de données de chaque processus fonctionnels sont identifiés dans cette phase. Une fois que tous les mouvements de données dans un processus fonctionnels sont identifiés, l'unité de mesure de 1 CFP est attribuée à chaque mouvement de données. La dernière étape, comme présentée dans le tableau 5.3 consiste à additionner les résultats pour obtenir la taille fonctionnelle de chaque processus fonctionnels (règle 19). La taille fonctionnelle des processus fonctionnels sont ensuite agrégés pour obtenir la taille fonctionnelle du logiciel mesuré (règle n° 20).

Identification des mouvements de groupes de données :

Les règles permettant d'identifier correctement les 4 sous-types de mouvement de données COSMIC sont dans le tableau 5.2. Les déclencheurs COSMIC sont identifiés à l'aide des règles 5 et 6. Les mouvements de données COSMIC d'Entrée (E) et de Lecture (R) peuvent être des expressions Statemate dites 'events' ou expressions du type 'condition' (règles 7, 8, 9, 10 et 15); les lectures (R) peuvent aussi être liées à l'utilisation de diagrammes d'états (règle 16).

Les mouvements de groupes de données de Sorties (X) sont des expressions Statemate du type 'action' (règles 11, 12, 13 et 14). Enfin, les mouvements de groupes de données d'Écriture (W) sont obtenus par une expression d'action (règle 17), ou liées à l'utilisation de diagrammes d'états (règle 18).

Tableau 5.2 Règles pour identifier les mouvements de groupes de données d'un logiciel modélisé avec Statemate

COSMIC element	**Rule n°**	**Rule description**
Data group movements	5	Identify 1E (and COSMIC trigger) data movement if this Control Activity is triggered by the system's clock (there is no explicit start (st!) action expression triggering this Control Activity)
Data group movements	6	Identify 1E (and COSMIC trigger) data movement if this Control Activity is triggered by the start(st!) action expression occurring in another Control Activity
Data group movements	7	Identify 1E data movement if this Control Activity is stopped by a stop (sp!) action expression occurring in another Control Activity
Data group movements	8	Identify 1E data movement for each variable defined in an external activity and consumed by an event expression inside this Control Activity
Data group movements	9	Identify 1E data movement for each variable defined in an external activity and consumed by a condition expression inside this Control Activity
Data group movements	10	Identify 1E data movement for each variable consumed by a condition expression inside this Control Activity, but was created in a different Control Activity
Data group movements	11	Identify 1X data movement for each start (st!) action expression occurring in this Control Activity and triggering another Control Activity
Data group movements	12	Identify 1X data movement for each stop(sp!) action expression occurring in this Control Activity and stopping another Control Activity
Data group movements	13	Identify 1X data movement for each variable defined or assigned by an action expression of this Control Activity and used or consumed in an external activity
Data group movements	14	Identify 1X data movement for each variable defined or assigned by an action expression in this Control Activity and used or consumed in a different Control Activity
Data group movements	15	Identify 1R data movement for each local variable consumed by a condition expression inside this Control Activity
Data group movements	16	Identify 1R data movement if this Control Activity is described, fully or partially, by Statecharts

Tableau 5.2 (Suite) Règles pour identifier les mouvements de groupes de données d'un logiciel modélisé avec Statemate

COSMIC element	**Rule n°**	**Rule description**
Data group movements	17	Identify 1W data movement for each local variable assigned by a action expression inside this Control Activity
Data group movements	18	Identify 1W data movement if this Control Activity is described, fully or partially, with Statecharts

Tableau 5.3 Règles pour obtenir la taille fonctionnelle des processus fonctionnels et du logiciel

COSMIC element	Rule n°	Rule description
Functional Process	19	Aggregate the CFP related to the data movements identified in a given Control Activity to obtain the functional size of that process.
Whole Software	20	Aggregate the CFP related to the data movements of (identified in) the functional processes of (identified in) the module to obtain the functional size of that module.

5.3 Application de la procédure à l'exemple de l'horloge

Dans cette section, la procédure de mesure proposée est appliquée à l'exemple du système de l'horloge présenté au chapitre 3.

- **Les utilisateurs fonctionnels du logiciel d'horloge :** en considérant la règle 1, l'horloge a deux utilisateurs fonctionnels: INPUT_PANEL et DISPLAY.

- **Les processus fonctionnels du système :** en considérant la règle 3, il existe 3 processus fonctionnels dans le module d'horloge:
    1. CLOCK_CNTL,
    2. KEEP_TIME_BHVR, et
    3. SET_TIME_BHVR.

  La prochaine étape consiste à obtenir la taille fonctionnelle de chacun des processus fonctionnels identifiés.

- **La taille fonctionnelle du processus fonctionnel 'CLOCK_CNTL':** Ce processus fonctionnel est décrit en utilisant un diagramme d'états : donc d'après les règles 16 et 18, il y a 1 W et 1 R. Il est déclenché par l'horloge interne du système, donc il y a 1 E. Les

variables POWER et MODE sont vérifiées à partir de INPUT_PANEL, ce qui correspond à 2 E. CLOCK_CNTL contrôle le déclenchement et l'arrêt des deux autres activités du système d'horloge (i.e. KEEP_TIME_BHVR et SET_TIME_BHVR), ce qui correspond à 1X pour le déclenchement de KEEP_TIME_BHVR, 1X pour le déclenchement de SET_TIME_BHVR, à 1X pour l'arrêt de KEEP_TIME_BHVR et 1X pour l'arrêt de SET_TIME_BHVR. Le tableau 5.4 montre la taille totale du processus fonctionnel CLOCK_CNTL qui est de 9 CFP.

Tableau 5.4 Détail de la taille fonctionnelle totale du processus fonctionnel 'CLOCK_CNTL'

Applied Rule n°=	**Expression**	**Expression source/destination**	**Data movement type**	**Data group CFP value**
5	*Internal clock*	*system*	E	1
9	[POWER]	INPUT_PANEL	E	1
9	[MODE=1]	INPUT_PANEL	E	1
11	st!(KEEP_TIME)	KEEP_TIME_BHVR	X	1
12	sp!(KEEP_TIME)	KEEP_TIME_BHVR	X	1
11	st!(SET_TIME)	SET_TIME_BHVR	X	1
12	sp! (SET_TIME)	SET_TIME_BHVR	X	1
16 & 18	*Statechart*		W+R	2

- **La taille fonctionnelle du processus fonctionnel 'KEEP_TIME_BHVR':** Ce processus fonctionnel est décrit en utilisant un diagramme d'états : donc d'après les règles 16 et 18, il y a 1 W et 1 R. Il est déclenché par CLOCK_CNTL, donc il y a 1 E. Il est arrêté par CLOCK_CNTL, donc il y a 1 E. Les variables, MINUTES et HOURS sont des variables locales, ce qui correspond à 2 W et 2R. CALC_MINUTES est une expression qui incrémente de 1 le compteur des minutes à chaque occurrence d'horloge (i.e. 1 E). la variable TIME (variable externe) est mise à jour par ce processus

fonctionnel, ce qui correspond à 1X. Le tableau 5.5 montre la taille totale du processus fonctionnel CLOCK_CNTL qui est de 10 CFP.

Tableau 5.5 Détail de la taille fonctionnelle totale
du processus fonctionnel 'KEEP_TIME_BHVR'

Applied Rule n°=	**Expression**	**Expression source/destination**	**Data movement sub-type**	**CFP**
6	st!(KEEP_TIME)	CLOCK_CNTL	E	1
7	sp!(KEEP_TIME)	CLOCK_CNTL	E	1
9	En(CALC_MINUTES)	KEEP_TIME_BHVR	E	1
15	[MINUTES/=59]	KEEP_TIME_BHVR	R	1
15	[HOURS/=12]	KEEP_TIME_BHVR	R	1
17	MINUTES:= MINUTES+1	KEEP_TIME_BHVR	W	1
17	HOURS:= HOURSS+1	KEEP_TIME_BHVR	W	1
14	*TIME*	SET_TIME & DISPLAY	X	1
16 & 18	*statechart*		W+R	2

- **La taille fonctionnelle du processus fonctionnel 'SET_TIME_BHV':** Ce processus fonctionnel est décrit en utilisant un diagramme d'états : donc d'après les règles 16 et 18, il y a 1 W et 1 R. Il est déclenché par CLOCK_CNTL, donc il y a 1 E. Il est arrêté par CLOCK_CNTL, donc il y a 1 E. Les variables, MINUTES et HOURS sont des variables locales, ce qui correspond à 2 W et 2R. Les variables INCREMENT_HOURS et INCREMENT_ MINUTES sont des variables externes (i.e. 1 E + 1 E). la variable TIME (variable externe) est mise à jour par ce processus fonctionnel, ce qui correspond à 1X. Le tableau 5.6 montre la taille totale du processus fonctionnel CLOCK_CNTL qui est de 10 CFP.

Tableau 5.6 Détail de la taille fonctionnelle totale du processus fonctionnel 'SET_TIME_BHVR'

Applied Rule n°=	**Expression**	**Expression source/destination**	**Data movement type**	**CFP**
6	st!(SET_TIME)	CLOCK_CNTL	E	1
7	sp!(SET_TIME)	CLOCK_CNTL	E	1
8	INCREMENT_HOURS	INPUT_PANEL	E	1
8	INCREMENT_ MINUTES	INPUT_PANEL	E	1
15	[HOURS =12]	SET_TIME_BHVR	R	1
15	[MINUTES=59]	SET_TIME_BHVR	R	1
17	HOURS:=1	SET_TIME_BHVR	W	1
17	MINUTES:= 0	SET_TIME_BHVR	W	1
14	*TIME*	KEEP_TIME & DISPLAY	X	1
16 & 18	*statechart*		W+R	2

- **La taille fonctionnelle du système:** Afin d'obtenir la taille fonctionnelle de l'horloge, on agrège la taille de tous les processus fonctionnels du module (règle n ° 20):

  Size (clock) = Size (CLOCK_CNTL) + Size (KEEP_TIME_BHVR) + Size (SET_TIME_BHVR) = 9 + 10 + 10 = 29 CFP.

CHAPITRE 6

AUTOMATISATION DES PROCÉDURES DE MESURE POUR DES LOGICIELS EN 'NOUVEAUX DÉVELOPPEMENTS'

6.1 Introduction

Il est nécessaire d'automatiser le processus de mesure pour obtenir une solution qui peut être appliquée dans un contexte industriel où on manque souvent de temps et de ressources pour la mesure. En industrie, il est essentiel de faire les mesures rapidement, et d'une façon précise. Par conséquent, un outil qui permet la mesure automatique des logiciels en se basant sur leurs modèles conceptuels est nécessaire pour éviter les délais et les erreurs humaines impliquées dans un processus de mesure manuelle. Ce chapitre présente les deux étapes suivies pour automatiser le mesurage de la taille fonctionnelle des logiciels sur la base de leurs spécifications fonctionnelles :

1- La première étape est de préparer une représentation semi-formelle où tous les éléments nécessaires et suffisants des outils de modélisation sont présentés par les éléments COSMIC correspondants.

2- La deuxième étape est la conception et le prototypage de l'outil-prototype d'automatisation pour les logiciels dont les spécifications fonctionnelles sont modélisées avec l'outil Simulink.

6.2 La représentation semi-formelle

L'objectif principal derrière la représentation semi-formelle est de faire la synthèse de toutes les informations nécessaires et suffisantes pour obtenir la taille fonctionnelle des logiciels mesurés selon les principes de la méthode COSMIC. Ainsi, dans ce modèle, les éléments définis dans la méthode COSMIC sont identifiés pour assurer un mapping complet entre les règles de la méthode COSMIC et les outils de modélisation temps-réel Statemate et Simulink.

6.2.1 Éléments nécessaires pour le mesurage

Les éléments nécessaires définis dans la méthode COSMIC [2] sont :

1. **Utilisateur fonctionnel** : Un (type d') utilisateur qui envoie et/ou est un récepteur de données des Fonctionnalités Utilisateurs Requises d'une partie du logiciel.

2. **Processus fonctionnel**: Un composant élémentaire d'un ensemble des Fonctionnalités Utilisateur Requises (FUR), comprenant un ensemble unique de mouvements de groupe de données, cohésif et indépendamment exécutable. Il est déclenché par un mouvement d'un groupe de données d'un utilisateur fonctionnel qui informe le morceau de logiciel à mesurer que l'utilisateur fonctionnel a identifié un évènement déclencheur. Il est complet lorsqu'il a exécuté tout ce qui est requis en réponse au type d'évènement déclencheur.

3. **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.

4. **Mouvement de données** : Un composant fonctionnel de base qui déplace un seul type de groupe de données. Il y a quatre sous-types de mouvements de données: entrée (E), sortie (S), lecture (L), écriture (C). Pour le besoin de la mesure, chaque sous-type de mouvement de données est considéré comme incluant un certain nombre de manipulations de données qui y sont associées.

6.2.2 Éléments correspondants dans la représentation

Le tableau 6.1 montre les éléments nécessaires, les figures correspondantes et une description rapide de ces figures.

Tableau 6.1 Tableau des éléments de la représentation semi-formelle

Nom de l'élément	***functional users***	***functional processes***	***data groups movements***	***persistent storage***
Abbreviation	FU	FP	E/X/W/R	
Figure				
Eléments Simulink correspondants	Subsystem	Subsystem et bloc StateFlow	Blocs + lignes; Action/condition + data objets	Blocs Data Store; Data objects
Règles Simulink correspondantes	1	4,5	6->10 ; 13->15 ;	11-12 ; 16, 17
Eléments Statemate correspondants	External Activity	Control Activity	Variables+ (Events ou Conditions ou Actions)	Variables+ (Events ou Conditions ou Actions)
Règles Statemate correspondantes	1	3	5->14	15->18

6.3 Le prototype d'automatisation de la mesure de la taille fonctionnelle des logiciels conçus avec Simulink

Le design du prototype 1 se divise en trois modules :

1. Filtre de données (ou Data Filter & Synthesizer / Input Generator): le but de ce module est d'extraire, d'organiser et de présenter les informations nécessaires pour l'application de la procédure de mesure définie.

2. Measurement unit : le module dans lequel le processus de mesurage se déroule.

3. GUI- Graphical User Interface : interface d'interaction avec l'utilisateur de l'outil.

Les modules 1 et 2 constituent le cœur du prototype. Le module 3 (GUI) concerne l'interface avec l'utilisateur du prototype (pour pouvoir lancer le processus de mesurage).

La relation entre les modules est présentée à la figure 6.1. Les figures 6.2 et 6.3 montrent les modules comme modélisés avec l'outil Simulink.

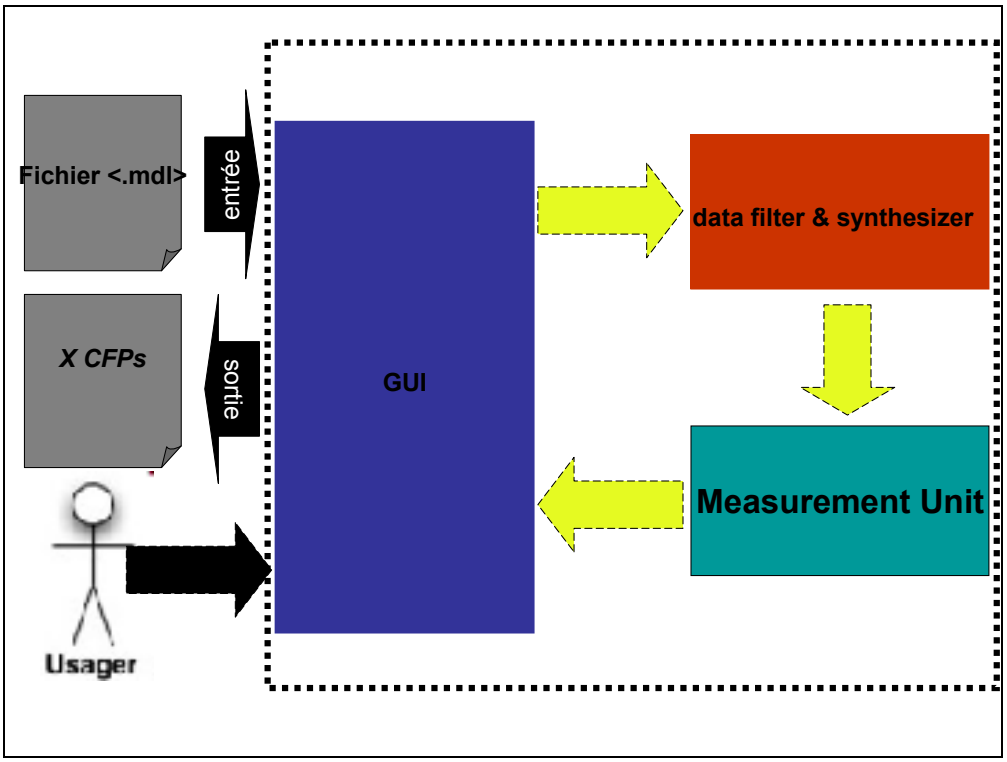


Figure 6.1 Modules du prototype 1

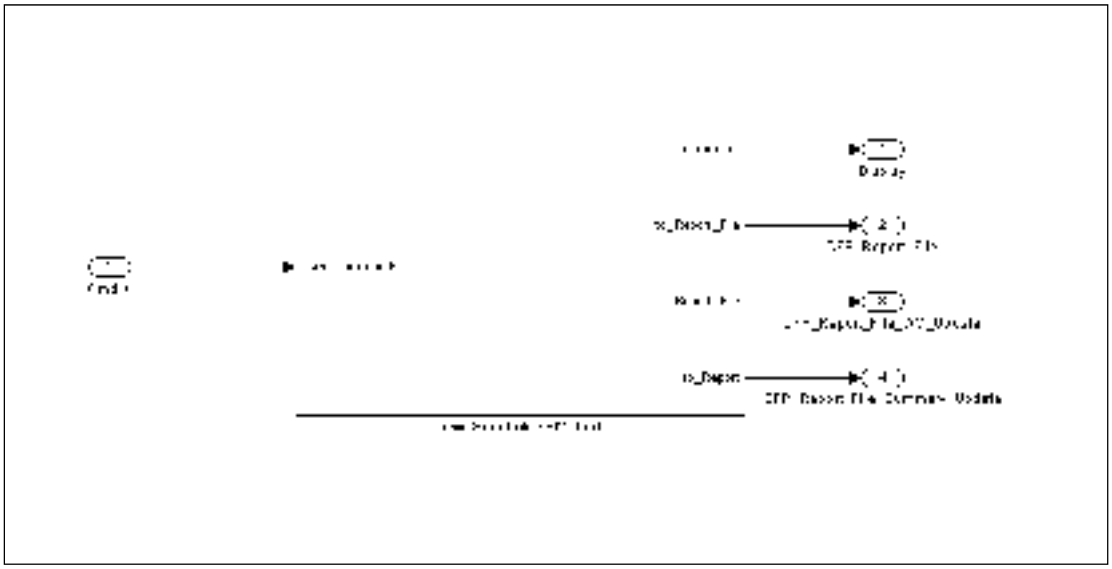


Figure 6.2 SimuLink Subsystem qui enveloppe les 3 modules du prototype 1

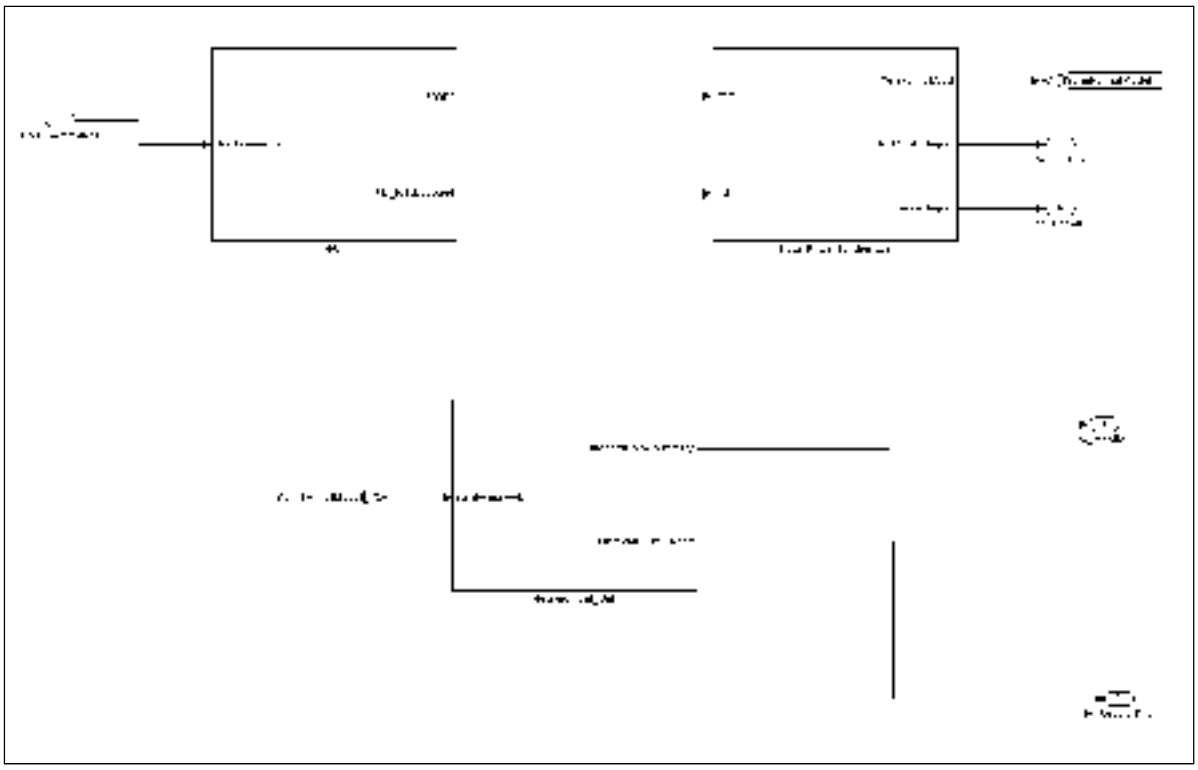


Figure 6.3 Les 3 modules SimuLink du prototype 1

6.3.1 Module 1 du prototype 1 : le filtre de données

Le module ‘filtre de données, se divise en 2 sous-modules :

1. Le sous-module ‘parseur’ qui décortique le fichier Simulink (extension .mdl) pour pouvoir extraire l’information nécessaire pour identifier les éléments pour le mesurage.

2. Le sous-module ‘générateur d’entrée’ qui présente ces informations dans un format donné.

Un stockage persistant est nécessaire pour stocker certains éléments identifiés au niveau de ce module.

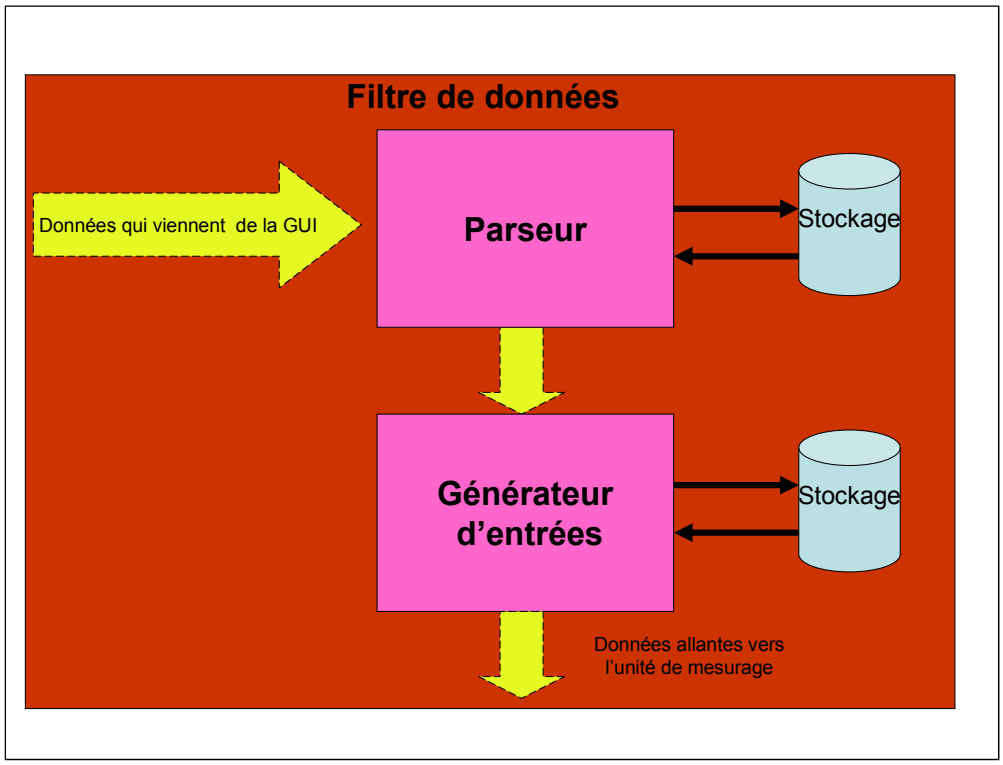


Figure 6.4 Les sous-modules du module 1 : Filtre de données

La figure 6.5 montre l'intérieur du module 1 suivant le modèle Simulink : le sous module « Parseur » est présenté par la fonction (OpenSpec). Le sous module « Générateur d'entrées » est aussi présenté par les fonctions (Rip_FP et GenerateSemiFormalModel) dans cette figure.

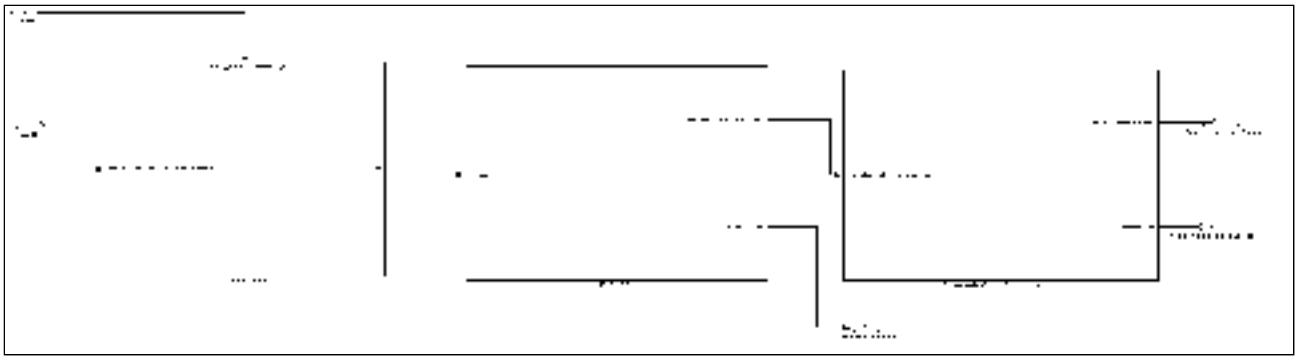


Figure 6.5 L'intérieur du module 1 du prototype 1

6.3.2 Le sous-module 'parseur'

Le but du sous-module 'parseur' est d'assurer le filtrage et extraction des éléments qui aident à l'identification des processus fonctionnels (PF) et des mouvements de données:

1. La première fonction (OpenSpec) de ce sous-module est d'ouvrir le fichier Simulink (.mdl) à mesurer (figure 6.6).

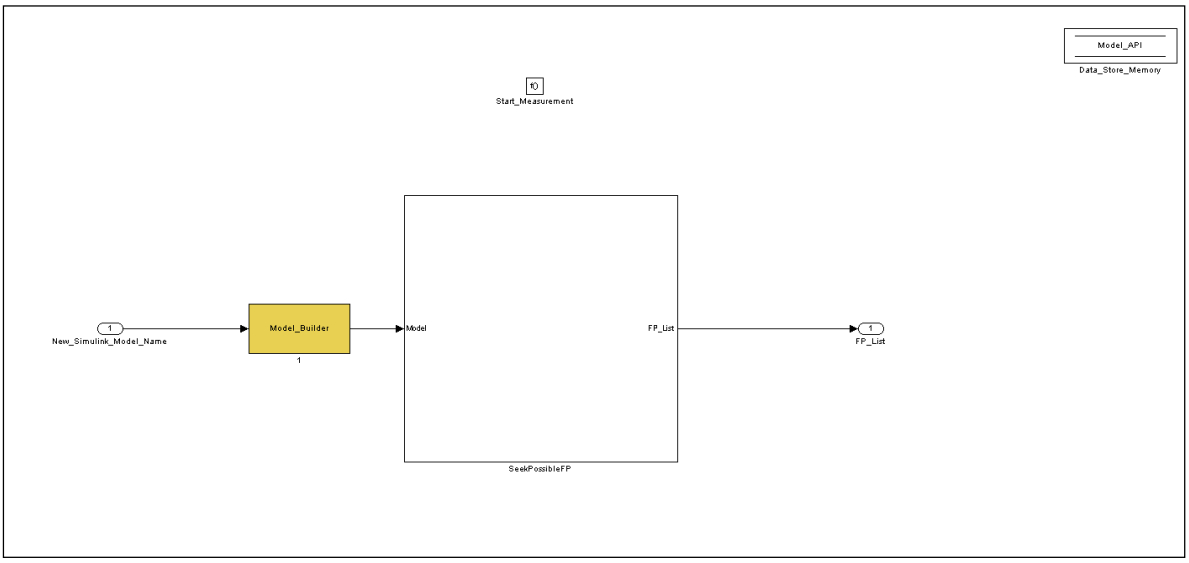


Figure 6.6 La fonction OpenSpec

2. La deuxième fonction (seekpossibleFP) de ce sous-module est de « décortiquer » les fichiers entrés, de réorganiser l'information présente dans ces fichiers pour finalement donner une liste des Processus Fonctionnels candidats (figure 6.7).

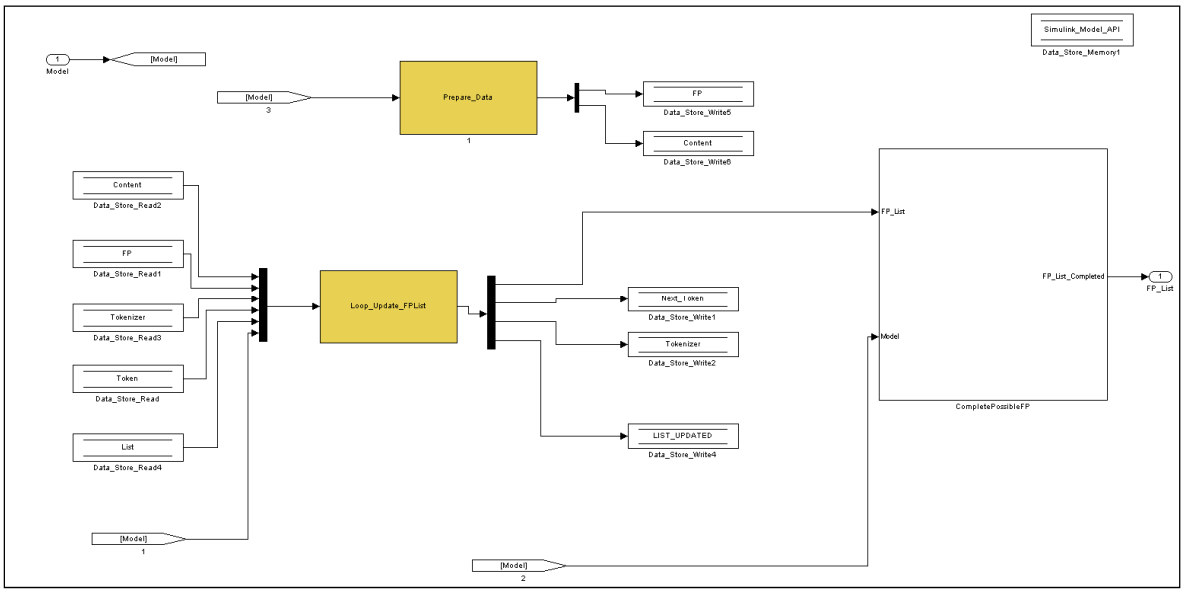


Figure 6.7 La fonction seekpossibleFP

3. Une fois que la liste des Processus Fonctionnels candidats est construite, la troisième fonction (completePossibleFP) du sous-module complète cette liste en affectant à chaque Processus Fonctionnel potentiel la liste complète des sous-blocs présents (figure 6.8). Toutefois, cette fonction fait appel à une autre fonction (ReorganizeBlocks) qui ne récupère que les informations nécessaires pour le mesurage suivant un format défini (figure 6.9). Cette fonction utilise aussi une fonction (AlreadyCheckedBlocks) qui vérifie que chaque sous-block n'est identifié et utilisé qu'une seule fois (suppression de toute redondance éventuelle) (figure 6.10).

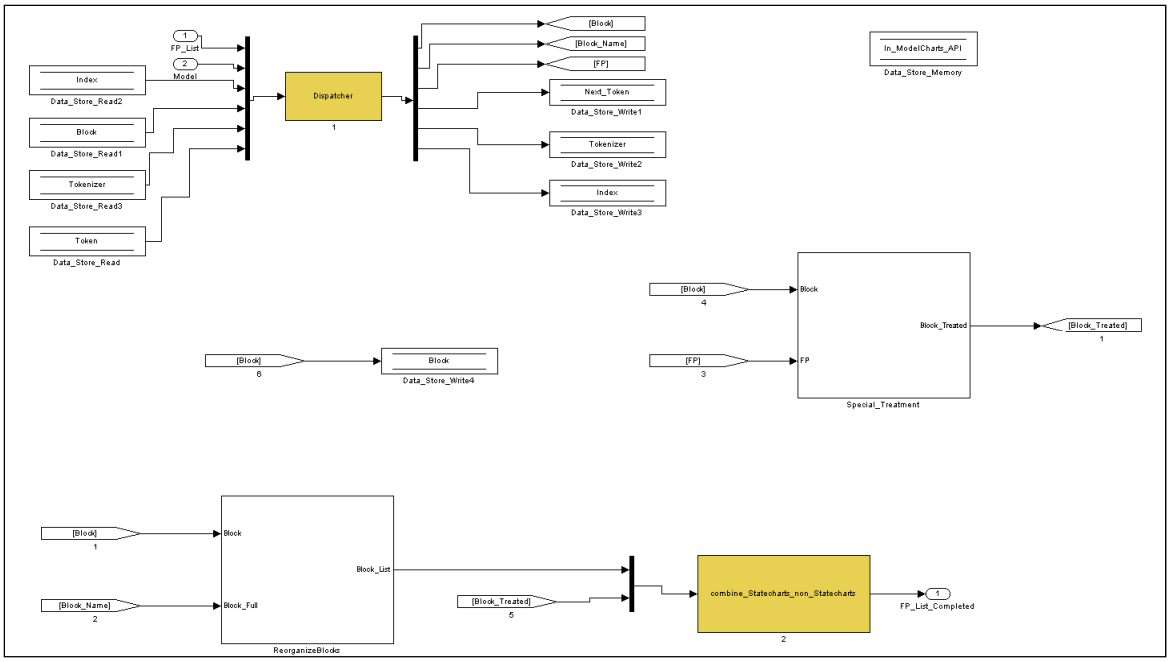


Figure 6.8 La fonction completePossibleFP

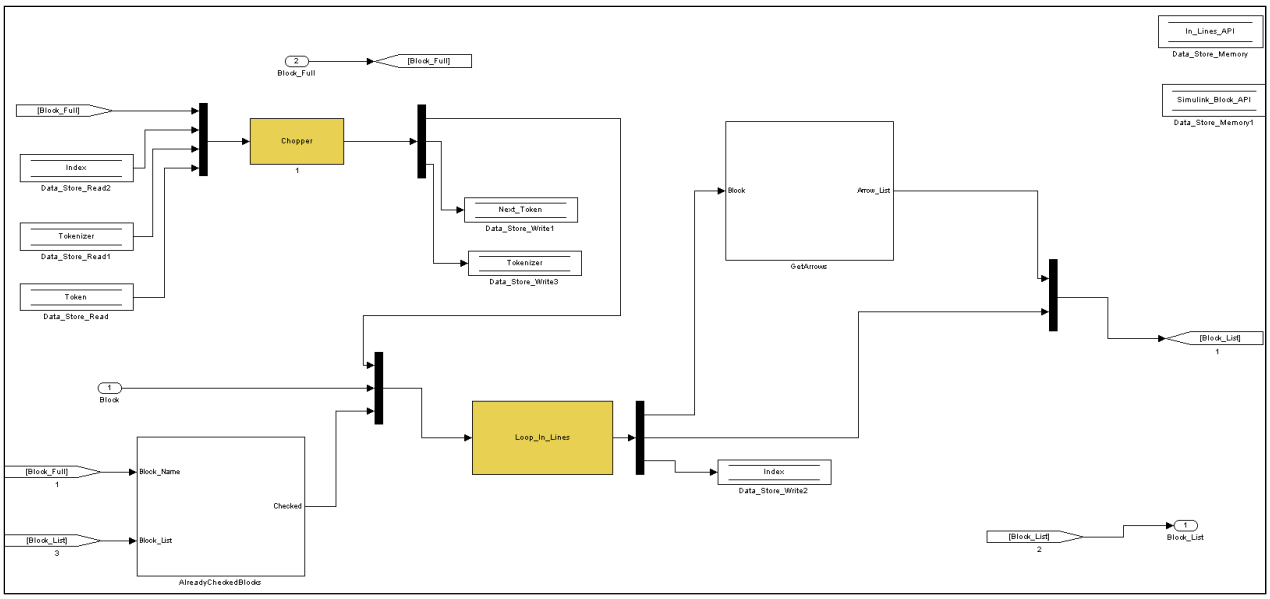


Figure 6.9 La fonction ReorganizeBlocks

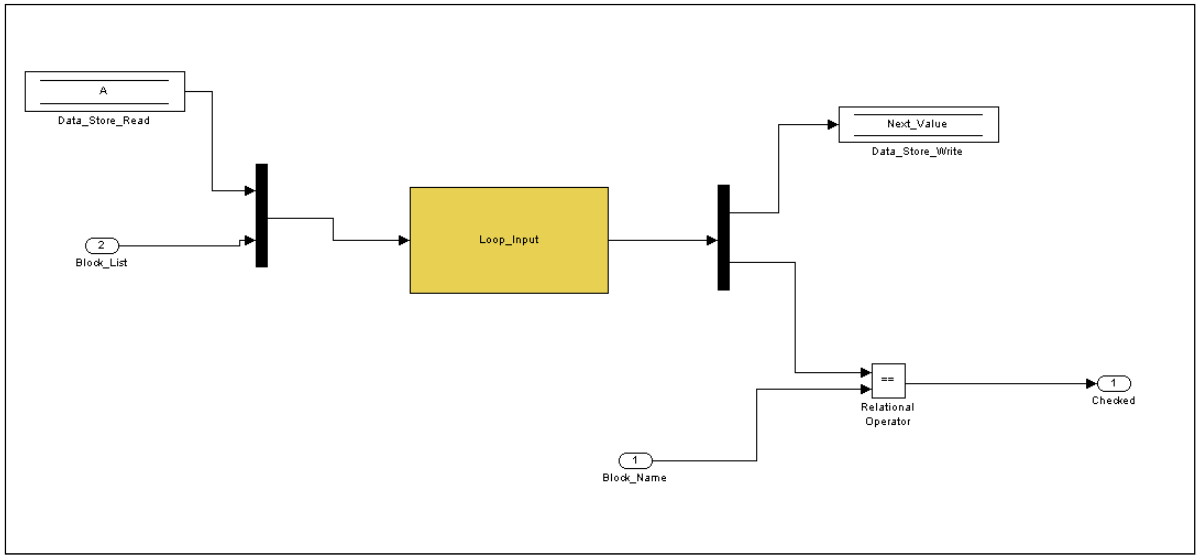


Figure 6.10 La fonction AlreadyCheckedBlocks

4. La quatrième fonction (GetArrows) de ce sous-module consiste à ajouter pour chaque sous-block retenu la liste de ses lignes de connexions (figure 6.11).

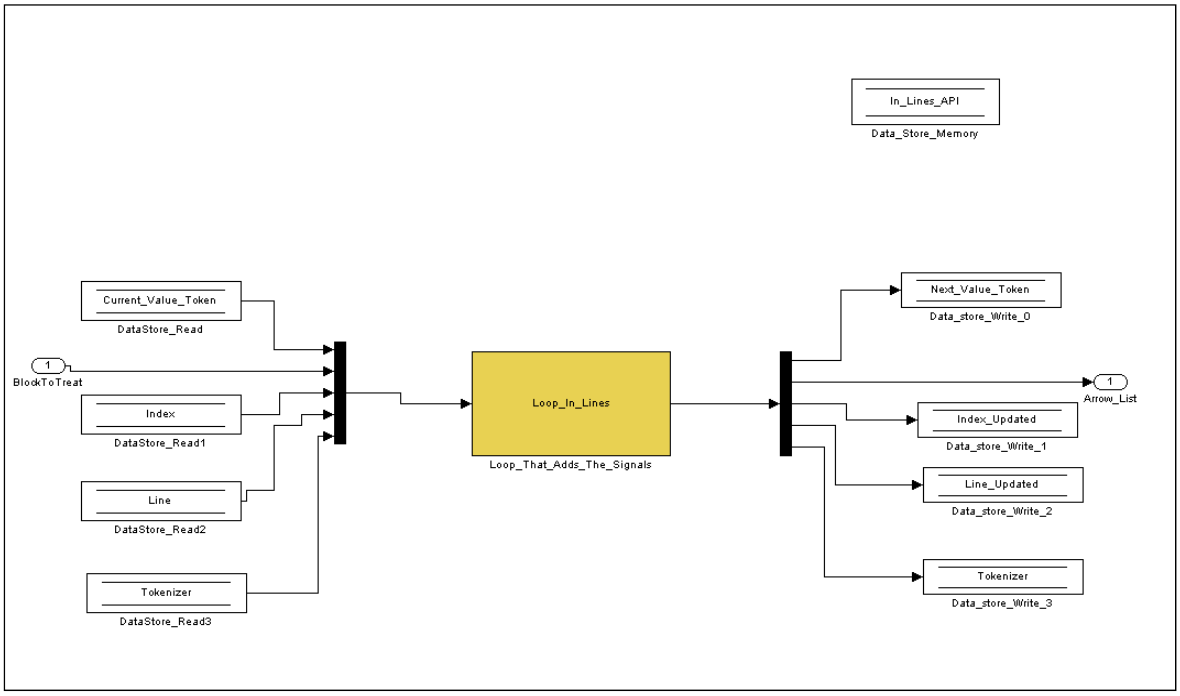


Figure 6.11 La fonction GetArrows

5. Une fonction spéciale (Special_Treatment) a été créée pour le traitement des blocs StateFlow (figure 6.12). Elle ajoute les éventuelles variables internes à la liste des sous-blocs de ces Processus Fonctionnels.

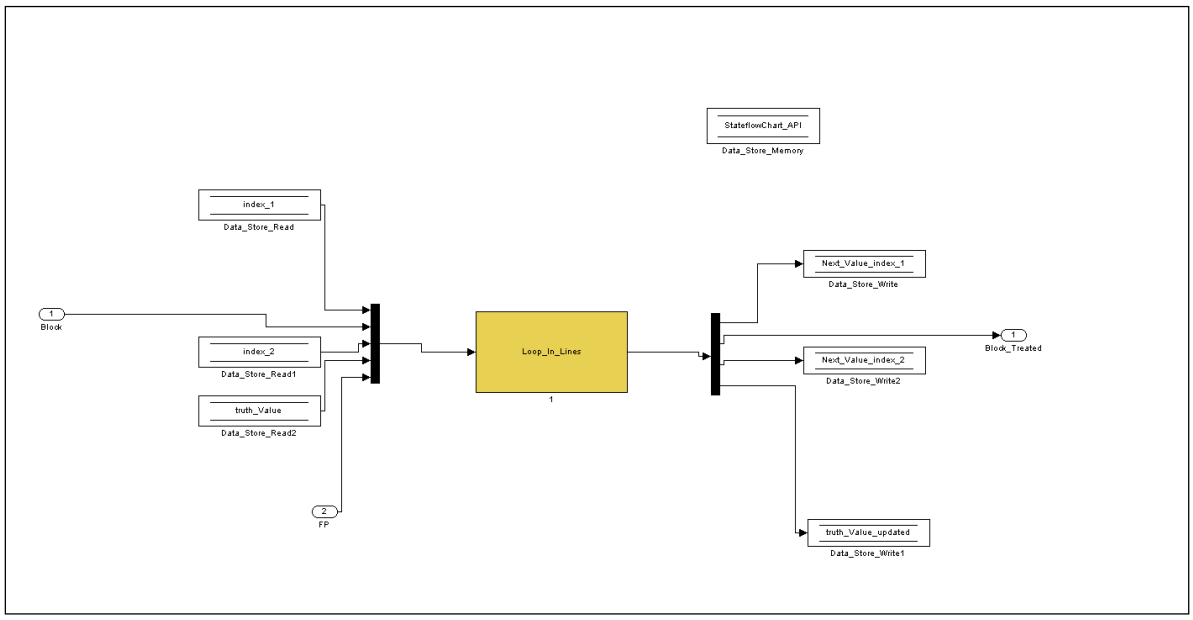


Figure 6.12 La fonction Special_Treatment

Donc, à la sortie du sous-module « Parseur », la liste des processus fonctionnels potentiels avec leurs listes de sous-blocs et signaux (lignes) est obtenue.

En sortie du parseur, les éléments suivant sont créés :

- Liste des Processus Fonctionnels candidats
- Pour chaque Processus Fonctionnel candidat: liste de tous les blocs et de l'ensemble (sous forme de liste aussi) des flèches par block.

6.3.3 Le sous-module ‘Générateur d’entrées’

Le sous-module ‘générateur d’entrées’ a comme entrée la liste sortie du parseur. Au niveau de ce module, la représentation semi-formelle est créée (sous forme de liste). Le « Générateur d’entrées » inclue plusieurs fonctions :

Les deux premières fonctions consistent à filtrer la liste des Processus Fonctionnels Candidats en entrée pour garder seulement les processus fonctionnels « réels » :

1. La première fonction (RIP_FP) ajoute une étiquette pour chaque Processus Fonctionnel pour savoir s’il s’agit d’un vrai Processus Fonctionnel et/ou d’un StateChart (figure 6.13).
2. La deuxième fonction (ReturnOnlyFPs) élimine les non-Processus Fonctionnels (figure 6.14).

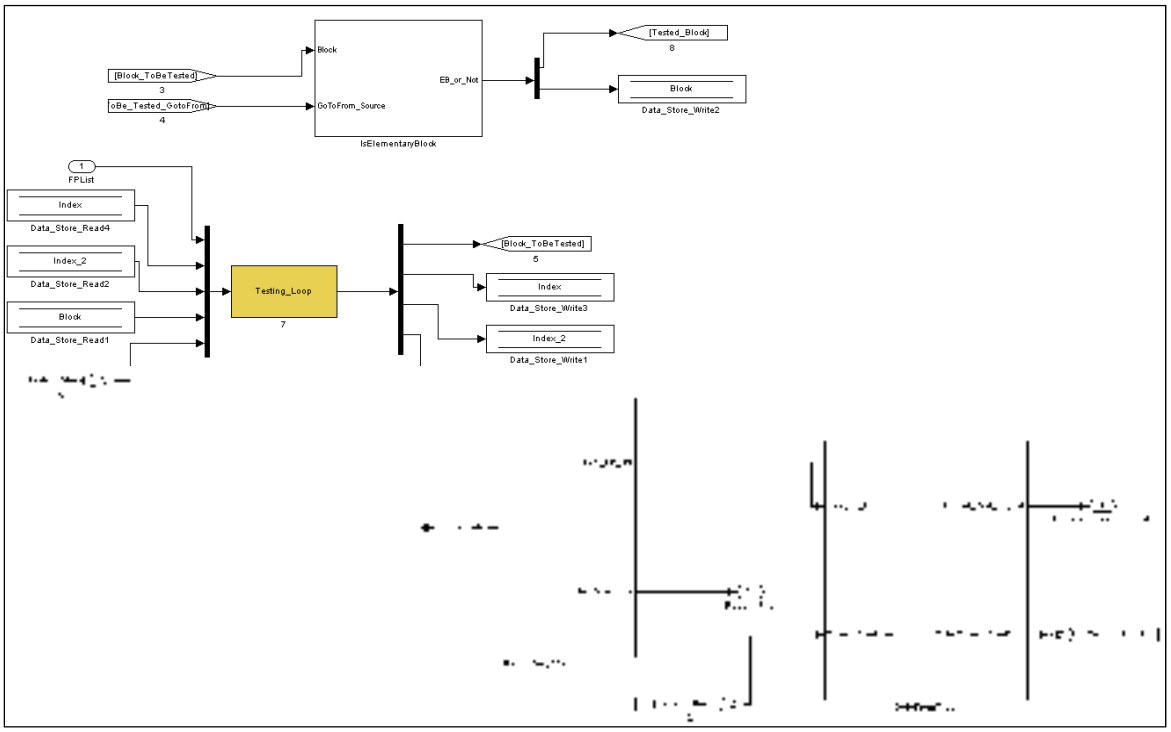


Figure 6.13 La fonction RIP_FP

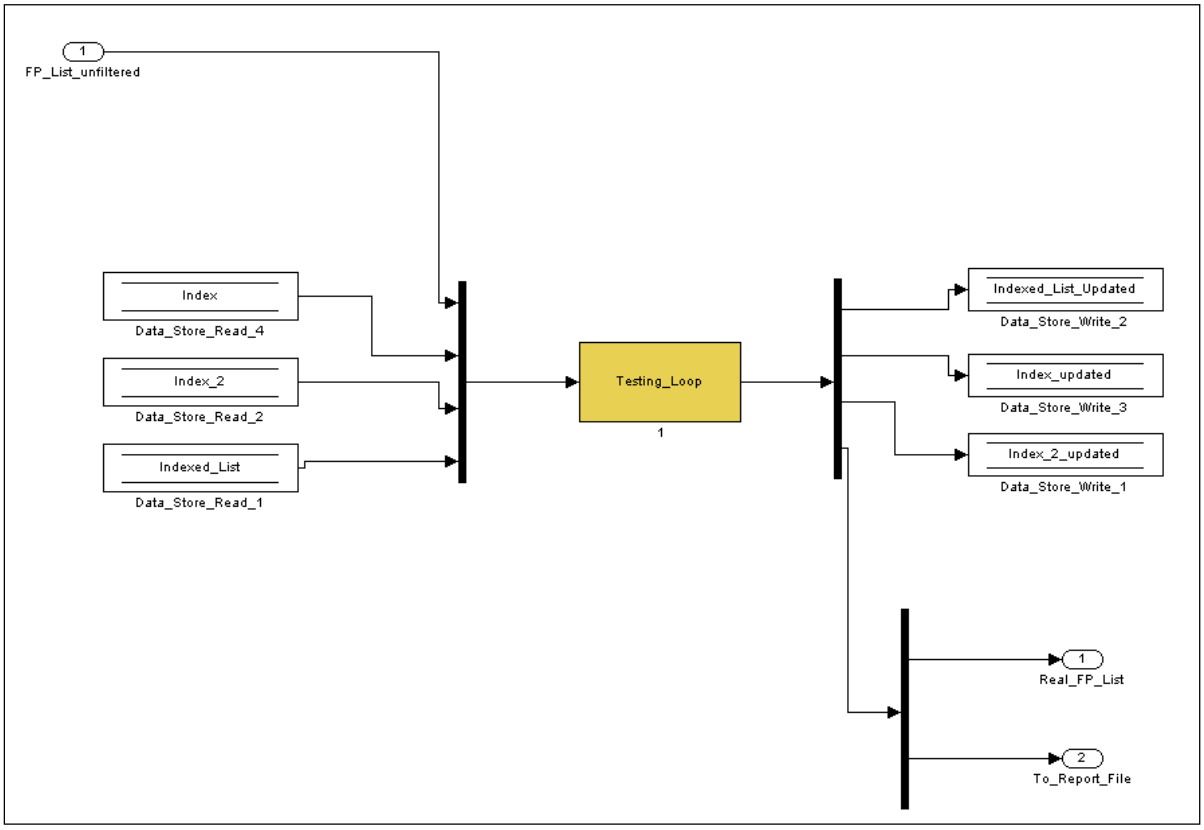


Figure 6.14 La fonction ReturnOnlyFPs

La première fonction décrite ci-dessus utilise une autre fonction qui identifie si le Processus Fonctionnel en question contient au moins un sous-block dit « block élémentaire ». Sont considérés comme blocs élémentaire tous les blocs appartenant aux catégories suivantes :
{Continuous; Discontinuities; Discrete; Logic And Bit Operations; Lookup Tables; Math Operations; Model Verification ; Model_Wide Utilities; Signal Attributes; Signal Routing; User Defined Functions; Additional Math And Discrete}

Une troisième fonction (GoToFromTreat) vient s'ajouter aux deux premières pour le traitement des cas des blocs spéciaux « GoTo » et « From ». Cette fonction cherche à trouver si les Goto des From correspondants sont liés ou pas à des blocs élémentaires (figure 6.15).

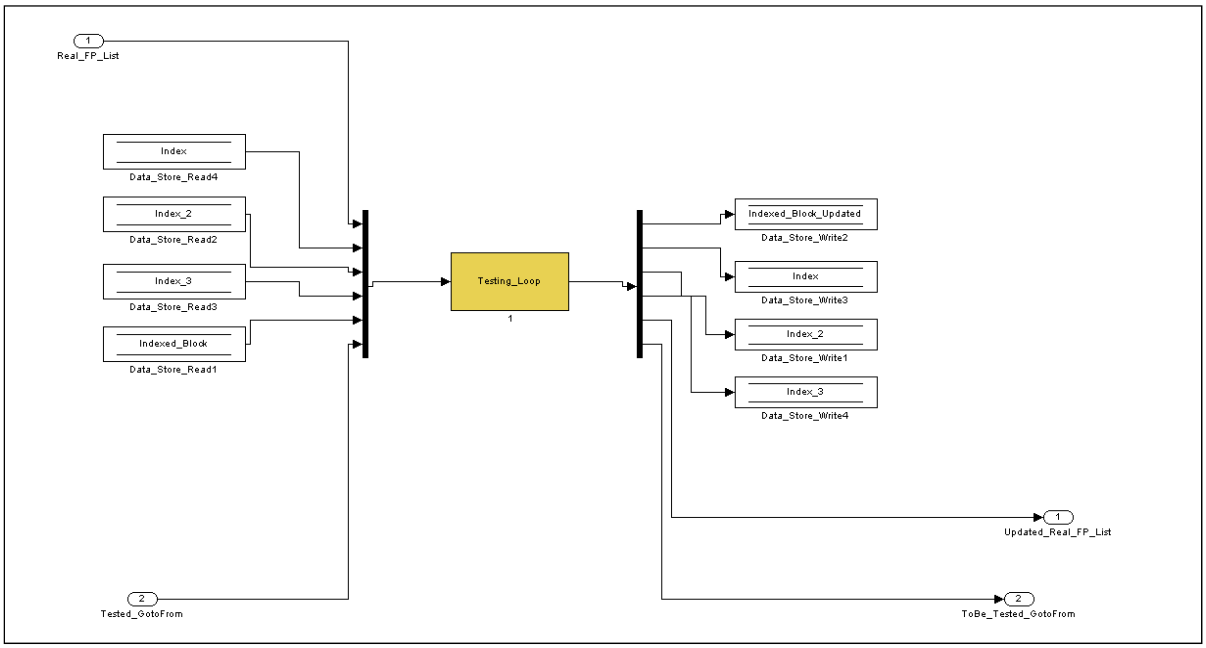


Figure 6.15 La fonction GoToFromTreat

Identification de la nature des mouvements des groupes de données : La fonction ‘GenerateSemiFormalModel’, présentée à la figure 6.16, de ce sous-module crée la représentation semi-formelle sous forme d’une liste de Mouvements de Données à partir de la liste des Processus Fonctionnels obtenus par les fonctions précédentes. Cette fonction utilise une autre fonction (AlreadyChecked, figure 6.17) pour éliminer la répétition éventuelle de certains mouvements de groupes de données.

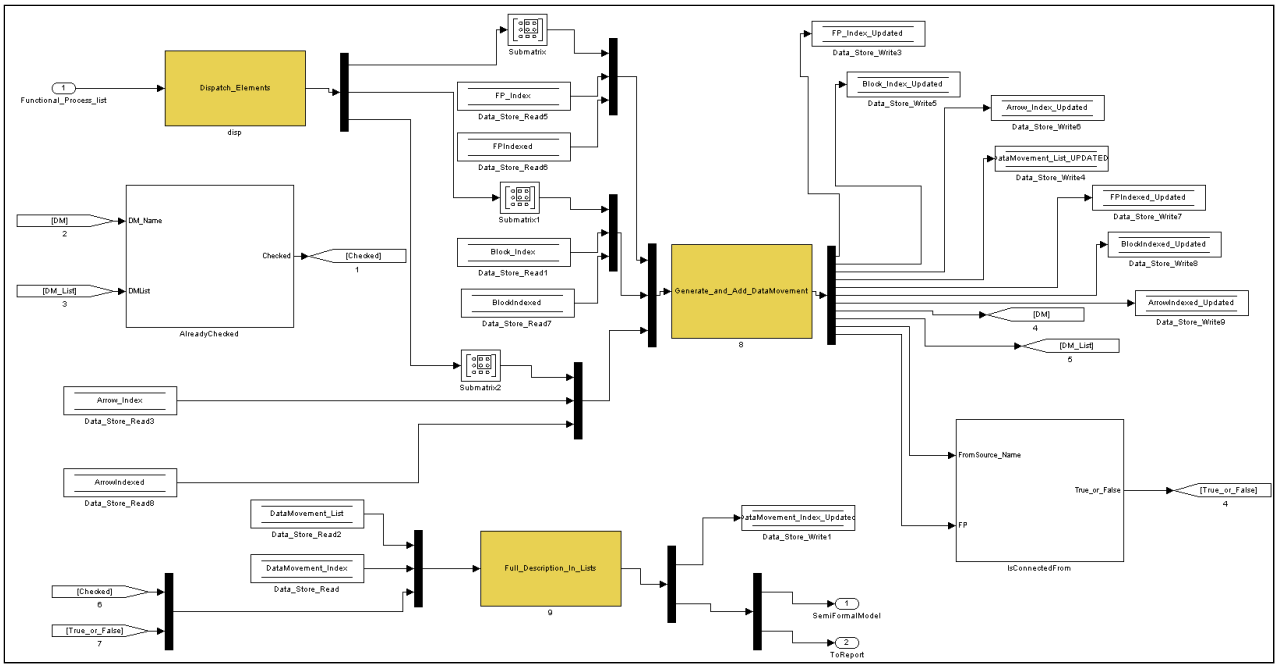


Figure 6.16 La fonction GenerateSemiFormalModel

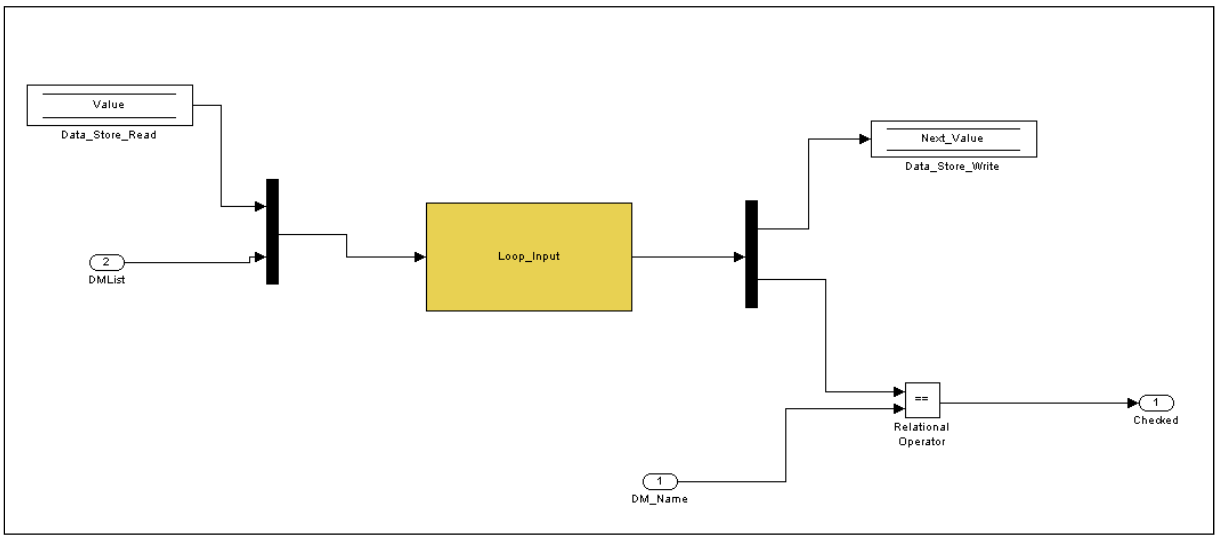


Figure 6.17 La fonction AlreadyChecked

Les éléments identifiés à ce niveau sont donc :

1. Les processus fonctionnels
2. Les mouvements de données et leurs types

A la sortie de ce module, on a le modèle semi-formel de mesure représenté sous forme d'une liste de mouvements de groupes de données relatifs au modèle de la spécification à mesurer.

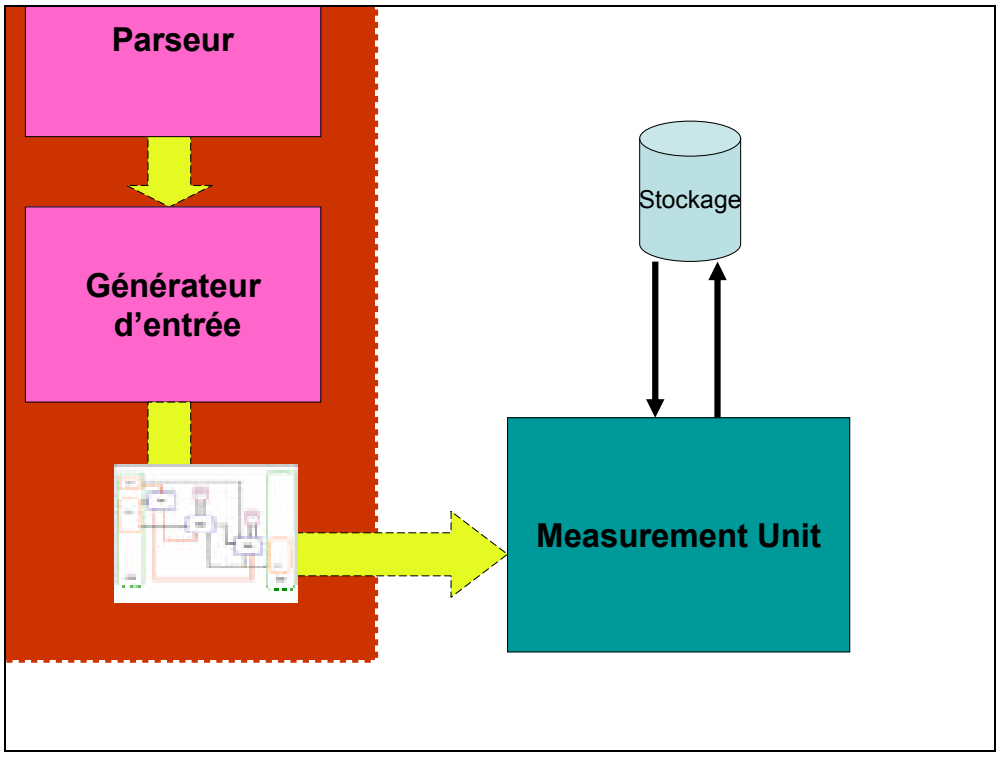


Figure 6.18 Le sous-module générateur d'entrée

6.3.4 Les objets du module 1 du prototype 1

1. **L'objet « PossibleFP» :** Le format de l'objet Processus Fonctionnel candidat (dénommé PossibleFP par la suite) a été défini sous le format suivant :

PossibleFP {Name, number, NumberOfBlocks, BlockList, IsFP, IsStateChart }

Où :

- *Name :* nom du processus fonctionnel
- *Number :* numéro identificateur du processus fonctionnel
- *NumberOfBlocks :* nombre de sous-block du processus fonctionnel
- *BlockList :* liste des sous-blocs que ce processus fonctionnel possède.
- *IsFP :* s'il s'agit d'un « vrai » PF ou non
- *IsStateChart :* s'il s'agit d'un StateChart ou non

2. L'objet « Block »

L'objet Bloc (dénommé Block par la suite) est défini suivant le format :

Block {Name, Type, Number, ArrowList}

Où :

- *Name :* nom du sous-block
- *Type :* type du sous-block
- *Number :* numéro identificateur du sous-block
- *ArrowList :* liste des signaux en relation avec le block
- *GotoTag :* attribut ajouté pour résoudre le problème des Goto/From
- *connected_to_EB :* attribut ajouté pour résoudre le problème des Goto/From

3. L'objet « Arrow » : Les lignes (dénommé Block par la suite) sont définies suivant le format :

Arrow {ID, Source, SourceType}

Où :

- *ID :* identifiant de l'Arrow
- *Source :* sous-block source de cette flèche
- *SourceType :* type du sous-block source de cette flèche

4. L'objet « Data movement»

Le format des mouvements de données (Data Movement) a été défini sous le format suivant :

DataMovement {ID, SourceFP, DestinationFP, Type}

Où :

- *ID :* numéro identificateur du mouvement de données
- *SourceFP :* processus fonctionnel (objet d'intérêt) source de ce mouvement de données
- *DestinationFP :* processus fonctionnel (objet d'intérêt) destination de ce mouvement de données
- *Type :* le type, selon la méthode COSMIC, de ce mouvement de données (i.e. Entrée/Sortie/Lecture/Ecriture).

6.3.5 Module 2 du prototype 1 : Unité de mesurage

Le module 2 du prototype 1 a comme entrée la description semi-formelle produite par le générateur d'entrée et qui contient la liste complète des mouvements de données retenus.

À ce niveau, sont effectués :

- l'attribution des valeurs numériques pour chaque mouvement.
- l'agrégation de la taille de tous les PFs.

Ces opérations sont effectuées grâce à une fonction qui examine la nature des mouvements présents dans la liste en entrée et sort les résultats vers la GUI. La figure 6.19 montre l'unité de mesurage modélisé en Simulink.

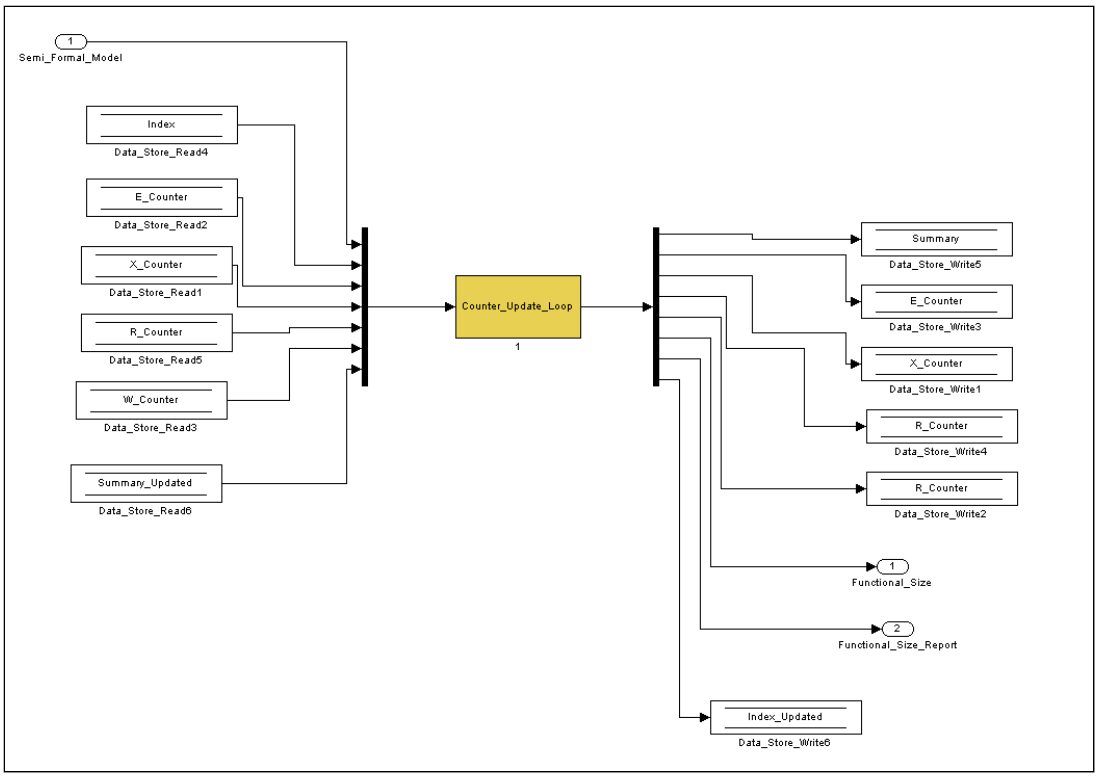


Figure 6.19 Le Module 2 du prototype 1 : Unité de mesurage

6.3.6 Module 3 du prototype 1 : Interface Graphique - GUI

Le module 'interface graphique – GUI' a comme rôle de permettre à l'utilisateur, d'une part, de choisir le fichier « .mdl » à mesurer. Et d'autre part les résultats de la mesure sont présentés, en partie, via cette interface (figure 6.20).

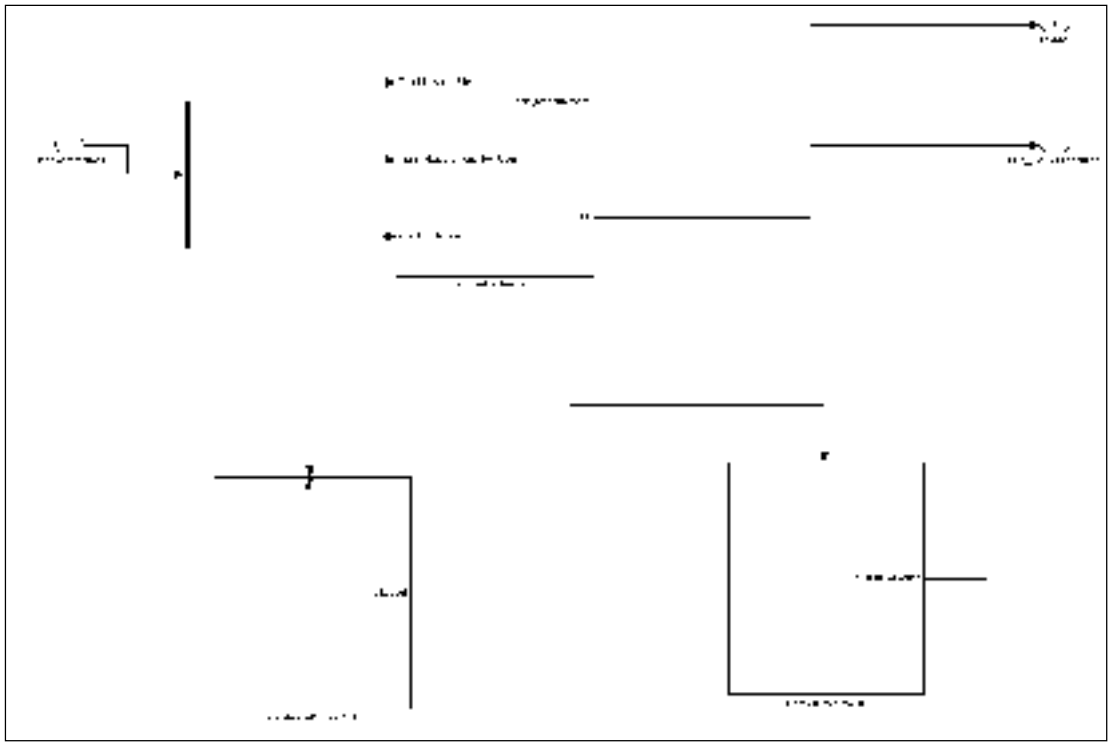


Figure 6.20 Le Module 2 du prototype 1: GUI

Ce module GUI est donc divisé en 3 sous-modules. Les deux premiers s'occupent de l'aspect visuel de l'interface graphique (createAndShowGUI et BorderLayoutGUI respectivement figures 6.21 et 6.22). Tandis que le troisième sous-module (ActionPerformed) reste en attente et exécute les commandes de l'utilisateur une fois déclenché (figure 6.23).

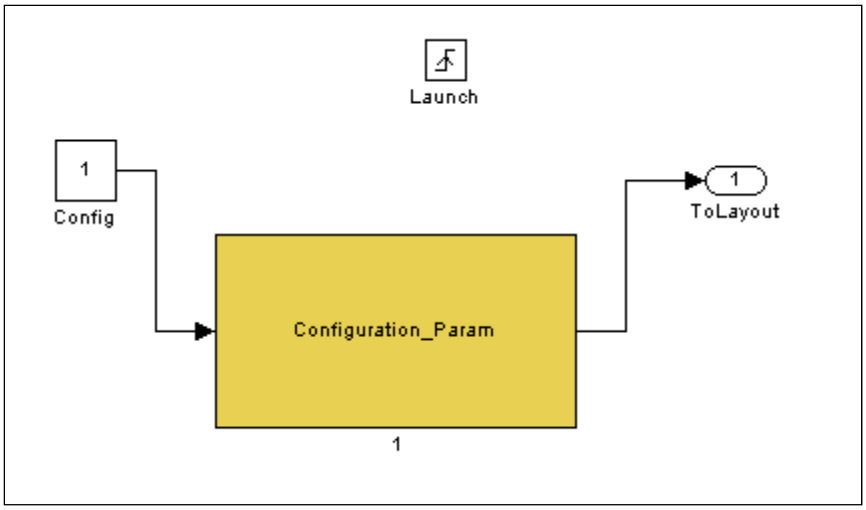


Figure 6.21 Le sous-module createAndShowGUI

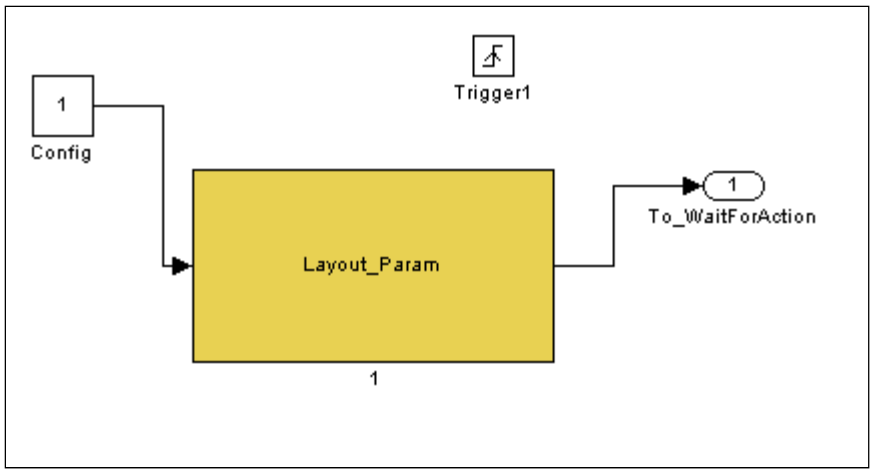


Figure 6.22 Le sous-module BorderLayoutGUI

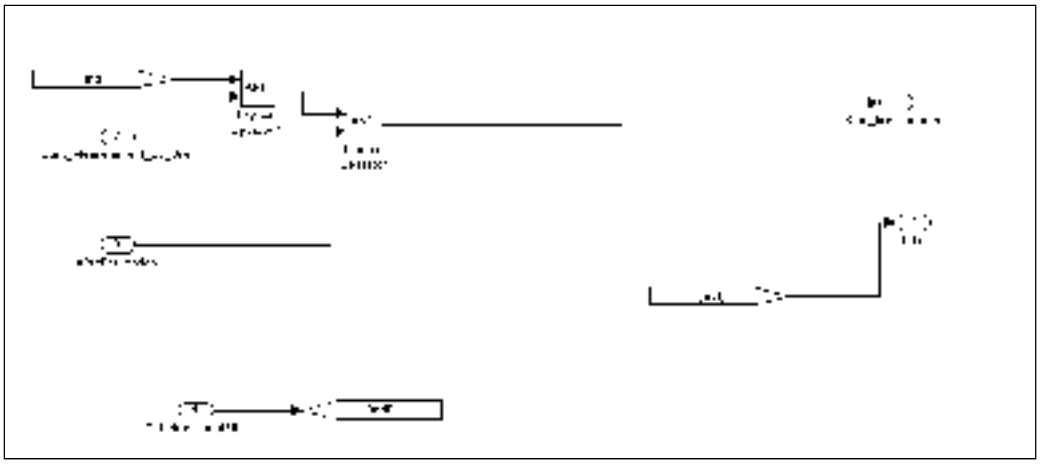


Figure 6.23 Le sous-module ActionPerformed

.

6.3.7 Les éléments COSMIC identifiés et couverts par le prototype

Le module 'interface graphique – GUI' a comme rôle de permettre à l'utilisateur, d'une part, de choisir le fichier « .mdl » à mesurer. Et d'autre part les résultats de la mesure sont présentés, en partie, via cette interface (figure 6.20).

Les éléments identifiés automatiquement, partiellement (cas réel et pratique) ou en intégralité (cas si les spécifications en entrée sont parfaites), par le prototype sont :

1. Les processus fonctionnels
2. Les mouvements des données

Les éléments non-identifiés à ce niveau sont les utilisateurs fonctionnels : la raison pour laquelle l'identification de cet élément n'est pas faisable, c'est parce qu'elle nécessite des informations complémentaires non-présentes dans les fichiers Simulink (.mdl) en entrée du prototype. Une partie de cette information ce trouve dans un document différent qui présente les origines des données d'une spécification (le MID - Module Interface Data). L'identification des utilisateurs fonctionnels est une étape de la phase 1 de la méthode COSMIC. Toutefois, cette identification n'a aucun impacte sur l'identification des

mouvements des groupes de données exprimés dans la spécification fonctionnelle en entrée, et donc elle n'a aucun impact sur le résultat final du mesurage.

6.3.8 Algorithme du prototype 1

L'ensemble des instructions du prototype 1 s'exécute comme suit :

1. Identifier tous les Processus Fonctionnels (PF) potentiels: l'ensemble des blocs du type 'Subsystem'.

2. Chercher tous les blocs du type 'Stateflow' : Chaque Bloc Stateflow est un PF.

3. Pour chaque bloc du type 'Subsystem', regarder l'ensemble de ses blocs pour savoir s'il contient au moins un bloc élémentaire. Pour faire ceci, chaque bloc doit être testé s'il appartient à la liste de la catégorie 'bloc élémentaire'. Si un bloc élémentaire se trouve dans ce sous-système, ce sous-système devient un PF.

4. Pour chaque PF identifié :

    a. Pour chaque bloc 'triggerport'/'enableport' identifier : une entrée (E)

    b. Pour chaque bloc 'source', vérifier les blocs auxquels il est connecté. S'il est relié à un bloc élémentaire, identifier : une entrée (E)

    c. Pour chaque bloc 'Sink', vérifier les blocs auxquels il est connecté. S'il est relié à un bloc élémentaire, identifier : une sortie (X)

    d. Pour chaque bloc du type « Subsystem », identifier :

        i. une entrée (E) pour chacune de ses sorties reliées à un block élémentaire

ii. une sortie (X) pour chacune de ses entrées reliées à un block élémentaire

e. Pour chaque bloc DataStoreRead, s'il est relié à un bloc élémentaire, identifier une lecture (R)

f. Pour chaque bloc DataStoreWrite, s'il est relié à un bloc élémentaire, identifier écriture (W)

5. Pour chaque Processus Fonctionnel identifié comme bloc Stateflow (selon l'étape 2):

   a. pour chaque événement déclencheur de ce processus fonctionnels identifier une entrée (E)

   b. Pour chaque 'Data Object' utilisées par une 'condition' dans ce Stateflow: tester si elle est une entrée pour ce 'Stateflow' à partir d'un bloc Simulink en dehors du bloc Stateflow:

      i. Si oui, identifier une entrée (E)
      ii. Sinon, identifier une lecture (R)

   c. Pour chaque 'Data Object' utilisé par une 'action' dans ce Stateflow: tester s'il est une sortie de ce 'Stateflow' vers un bloc Simulink en dehors de ce bloc Stateflow:

      i. Si oui, identifier une sortie (X)
      ii. Sinon, identifier une écriture (W)

6. Supprimer toute répétition éventuelle de mouvement de groupes de données.

7. Ajouter la taille de tous les mouvements de groupes de données (c.-à-d. les Sorties Entrées, Lectures et Ecriture identifiées à l'étape 3) à l'intérieur du PF.

8. Agréger des tailles de l'ensemble des PF identifiés dans les étapes 2 et 3.

6.3.9 Représentation semi-formelle et taille fonctionnelle du prototype 1

Dans la figure 6.24, la représentation semi-formelle des sous-modules du prototype 1 est présentée. Chacune des fonctions constitue un processus fonctionnel COSMIC.

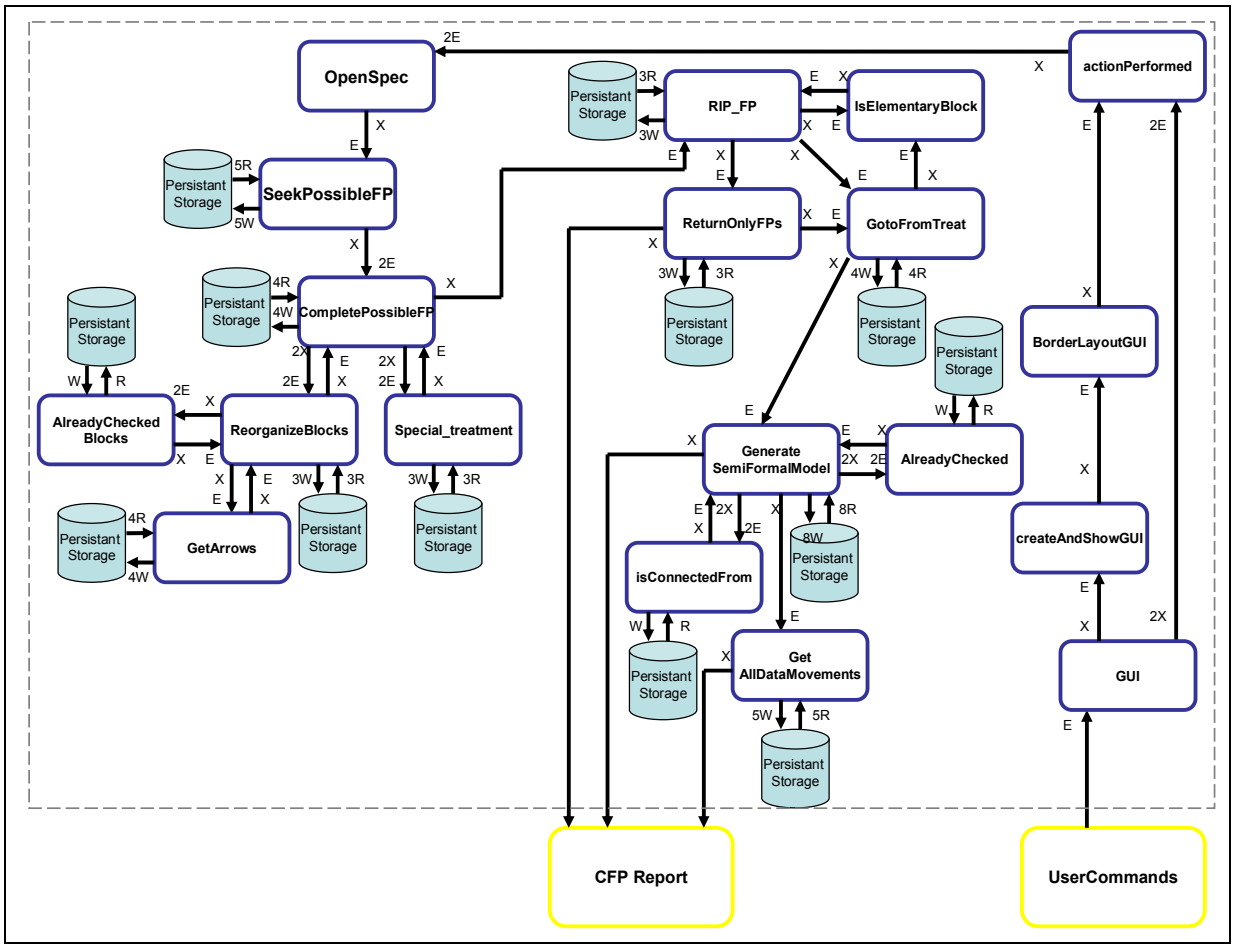


Figure 6.24 La représentation semi-formelle des sous-modules du prototype 1

Ainsi en utilisant les mouvements de données dans la représentation de la figure 6.24, la taille fonctionnelle du prototype 1 est détaillée dans le tableau 6.2.

Tableau 6.2 Taille fonctionnelle des différents modules du prototype 1

Sous-Module	**Fonction (Processus Fonctionnel)**	**E**	**X**	**R**	**W**	**TOTAL** (En manuel)	**Numéros des règles utilisées**
Parseur	OpenSpec(String)	2	1	0	0	3	4, 6, 7, 9
	SeekPossibleFP(SimulinkModel)	1	1	5	5	12	4, 7, 10, 11, 12
	CompletePossibleFP(lists, SimulinkModel)	4	5	4	4	17	4, 7, 10, 11, 12
	AlreadyCheckedBlocks (String, lists)	2	1	1	1	5	4, 7, 9, 11, 12
	ReorganizeBlox(String, SimulinkBlock)	4	3	3	3	13	4, 7, 9, 10, 11, 12
	Special_treatment(PossibleFP, SimulinkBlock)	2	1	3	3	9	4, 7, 9, 11, 12
	GetArrows(SimulinkBlock)	1	1	4	4	10	4, 7, 9, 11, 12
Générateur d'entrée	RIP_FP(lists)	2	3	3	3	11	4, 7, 8, 10,11, 12
	IsElementaryBlock(String)	2	1	0	0	3	4, 7, 9
	ReturnOnlyFPs(lists)	1	2	3	3	9	4, 7, 9, 11, 12
	GotoFromTreat (lists)	2	2	4	4	12	4, 7, 9, 11, 12
	GenerateSemiFormalModel(lists)	3	6	8	8	25	4, 7, 8, 9, 10, 11, 12
	isConnectedFrom	2	1	1	1	5	4, 7, 9, 11, 12
	AlreadyChecked(String, DataMovement[])	2	1	1	1	5	4, 7, 9, 11, 12
Unité de mesurage	GetAllDataMovements(DataMovement[])	1	2	6	6	15	4, 7, 9, 11, 12
GUI	GUI	1	3	0	0	4	4, 7, 8, 9, 10, 11, 12
	createAndShowGUI	2	1	0	0	3	4, 6, 7, 9
	BorderLayoutGUI	2	1	0	0	3	4, 6, 7, 9
	actionPerformed	3	1	0	0	4	4, 6, 7, 9
Totaux		39	37	46	46	168	

Ce qui donne 168 CFP comme taille fonctionnelle des modules du prototype. Pour pouvoir vérifier l'exactitude de la taille fonctionnelle obtenue, la spécification fonctionnelle du prototype 1 modélisée avec Simulink a été mesurée par le prototype lui-même. Le résultat du mesurage automatique a donné le même résultat de la mesure manuelle. Le détail des identifications des mouvements des groupes de donnée se trouve dans l'annexe III.

CHAPITRE 7

UN PROTOCOLE D'ÉVALUATION DES OUTILS DE MESURAGE AUTOMATIQUE DE LA TAILLE FONCTIONNELLE SUIVANT LA MÉTHODE DE MESURE COSMIC

7.1 Introduction

Ce chapitre présente la stratégie de test du prototype d'automatisation du mesurage de la taille fonctionnelle d'après COSMIC, présentée sous forme d'une démarche. Cette démarche s'inspire du protocole d'évaluation décrit par Mendes [43]. Elle s'appuie aussi sous la représentation semi-formelle pour COSMIC (modèle commun pour la mesure) qui, à son tour, s'inspire du modèle de Paton et al. [47].

7.2 Utilisation de la représentation semi-formelle pour les combinaisons de test

Le processus générique présenté dans [47] est constitué de trois 'types' d'éléments (processus/dépôts/flux) qui sont nécessaires pour pouvoir identifier les signatures (en termes de combinaisons valides de flux) des fonctions transactionnelles de la méthode IFPUG. La frontière est aussi présentée dans ce modèle générique.

Dans [47] les dépôts représentent les ILF (Internal Logical Files, fichiers situés à l'intérieur de la frontière) ou les ELF (External Logical Files) ou utilisateurs ou autres entités (situés à l'extérieur de la frontière). Donc le but de cette présentation 'semi-formelle' est de montrer qu'il y a des entités internes et des entités externes aux frontières sans faire la différence entre les « types » de ces entités. Cela présente des limitations : Mendes a présenté les limitations de ce modèle dans [43] et a proposé des extensions pour dépasser ces limitations. Inspirée des travaux de Mendes [43], la représentation semi-formelle basée sur la méthode COSMIC proposée dans ce projet de recherche est utilisée pour la création de combinaisons de tests et l'évaluation des outils de mesurage automatisés qui implémentent des procédures de mesurages qui sont basées sur le standard COSMIC ISO 19761.

7.2.1 Cas d'un processus isolé

Les éléments identifiés comme nécessaires pour la méthode COSMIC sont :

1. La « frontière » : l'élément-clef qui permet l'identification de deux « types » d'éléments : internes et externes (ici on se limitera à ces deux types). Toutefois, on doit préciser ici que le seul élément qui est considéré du côté du logiciel de la frontière est le *stockage persistant* (c.f. Méthode COSMIC Version 3.0 – Manuel de mesure 2007). Donc on peut le considérer à l'intérieur de la frontière à coté du processus-cible de mesure.

2. Le processus-cible est le processus fonctionnel qui est le composant élémentaire des FUR (Functional User Requirements).

La figure 7.1 montre les éléments identifiés pour le moment.

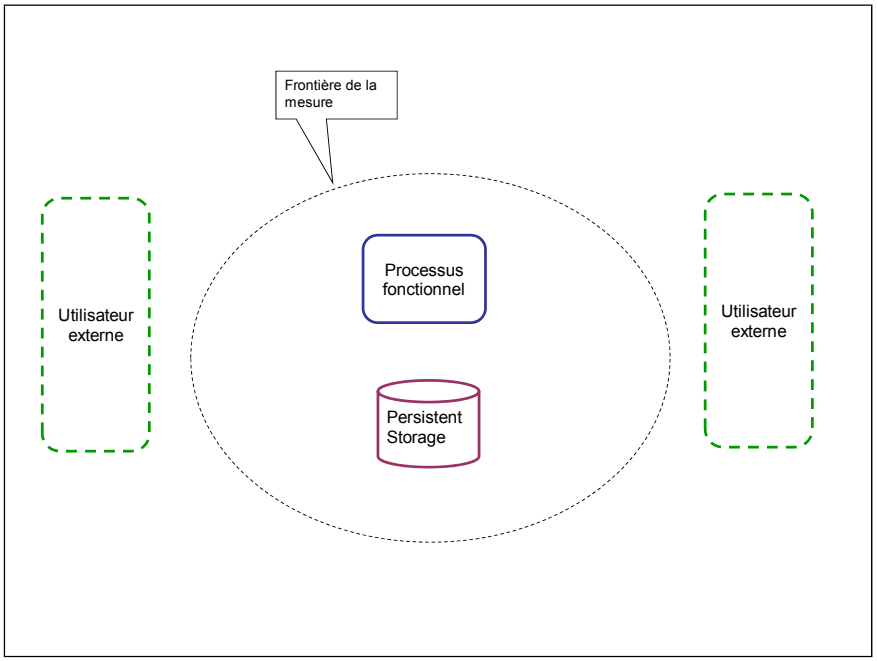


Figure 7.1 Représentation inspirée (sans les flux)

pour COSMIC du modèle de Paton et Abran

Maintenant qu'on a identifié les éléments de la représentation, la deuxième étape consiste à identifier les flux possibles. La présence d'un flux veut dire qu'il peut avoir un ou plusieurs mouvements de données entre la source et la destination de ce flux.

Il existe des flux entre chacun des éléments externes (sources des flux) et le processus-cible (destination des flux). Vice versa, il existe des flux entre le processus-cible (source des flux) et les éléments externes (destinations des flux). La raison pour laquelle les deux éléments externes sont présentés dans cette représentation est pour bien souligner qu'un processus fonctionnel peut interagir avec plusieurs éléments externes distincts, et que la réponse d'une sollicitation du Processus Fonctionnel (processus cible) ne va pas forcement vers le même élément solliciteur.

D'un autre coté, il existe un flux entre le Processus Fonctionnel (source) et le stockage persistant (destination) et un autre vice versa. Il ne peut avoir qu'un seul stockage persistant (vision logique—même si en physique le stockage persistant peut être réparti) pour un Processus Fonctionnel donné.

Il est impossible d'avoir un flux entre un élément externe et le stockage persistant. La réciproque est vraie aussi : il est impossible d'avoir un flux entre le stockage persistant et un élément externe. La figure 7.2 montre la représentation semi-formelle inspirée du modèle de Paton *et al.* [47].

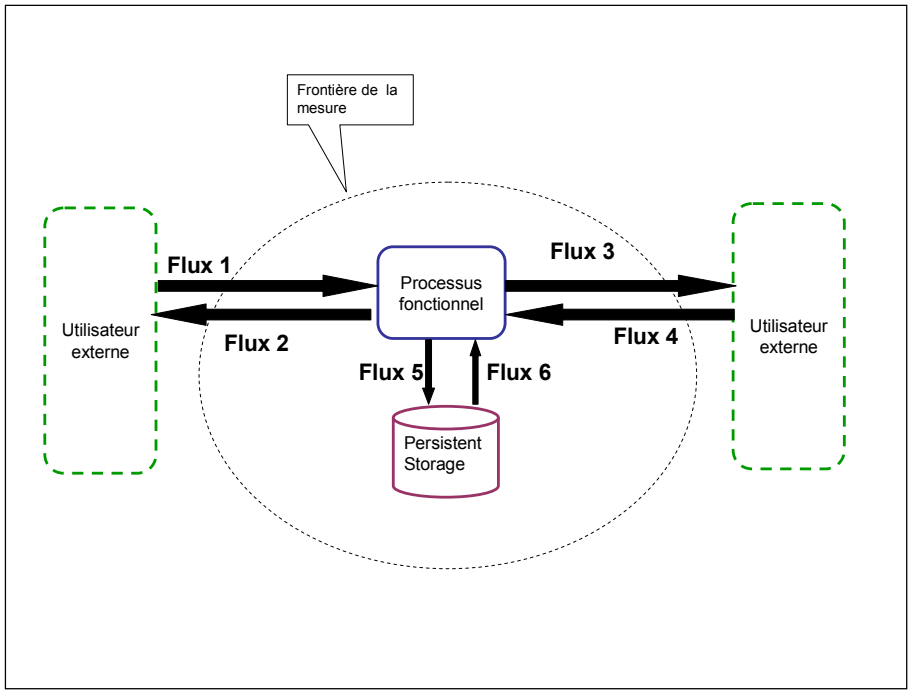


Figure 7.2 Représentation semi-formelle
inspirée du modèle de Paton *et al.*

Donc avec les ajustements apportés, le même genre de représentation du processus générique présenté à la section 6.1. de l'étude de Mendes, a été obtenu pour la méthode COSMIC.

Le Processus Fonctionnel ne peut s'exécuter que si au moins le flux 1 ou le flux 4 existe. La combinaison de flux minimale obligatoire pour le Processus Fonctionnel selon la méthode COSMIC (*règle d* : Méthode COSMIC Version 3.0 – Manuel de mesure 2007- page 37) est : (flux 1 OU flux 4) ET (flux 2 OU flux 3 OU flux 5). Toutes les autres combinaisons sont possibles tant que la combinaison minimale est assurée. Chacun des flux correspond à un type de mouvement de données. Le tableau 7.1 résume le mapping entre le flux et les quatre types de mouvement de données COSMIC (ENTRY soit E, EXIT soit X, READ soit R, WRITE soit W).

Tableau 7.1 Correspondance flux/mouvements de données

Flux	**Type de mouvement de groupe de données**
Flux 1	E
Flux 2	X
Flux 3	X
Flux 4	E
Flux 5	W
Flux 6	R

L'événement déclencheur (trigger, c'est une entrée E COSMIC) est fait ou bien par le flux 1 ou le flux 4.

La condition que le Processus Fonctionnel contient au moins 2 mouvements de données (1E + 1X/W) est assurée par la combinaison minimale obligatoire décrite ci-dessus. Le tableau 7.2 montre toutes les combinaisons possibles.

Tableau 7.2 Combinaisons de flux pour un processus isolé

Flux						**Statut* de la combinaison**
1	2	3	4	5	6	
Types de mouvements de données COSMIC						
E	X	X	E	W	R	
1	1	0/1	0/1	0/1	0/1	OK
1	0/1	1	0/1	0/1	0/1	OK
1	0/1	0/1	0/1	1	0/1	OK
0/1	1	0/1	1	0/1	0/1	OK
0/1	0/1	1	1	0/1	0/1	OK
0/1	0/1	0/1	1	1	0/1	OK
0	0/1	0/1	0	0/1	0/1	KO
0/1	0	0	0/1	0	0/1	KO

*selon la condition -5- décrite ci-dessus

Ce principe est applicable pour plusieurs Processus Fonctionnels. Ceci est présenté dans la section suivante.

7.2.2 Cas de plusieurs processus : extension de la représentation

Dans ce chapitre, une représentation semi-formelle ajustée pour la méthode COSMIC qui est basée sur le modèle de Paton et Abran a été présentée. Dans cette section, les extensions et les détails annoncés ci-dessus sont présentés.

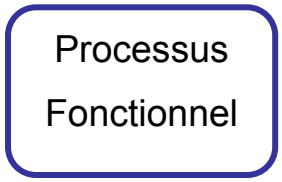
Au niveau des utilisateurs externes, 2 types de ceux-ci existent :

1. PF homologue : PF qui se trouve dans la même couche du PF-cible et donc qui est utilisateur fonctionnel de ce PF cible et qui donc se situe à l'extérieur de la frontière du PF cible.

2. Entité externe : entité qui se situe à l'extérieur du logiciel dans lequel se trouve le PF cible à mesurer. Cette entité peut se décomposer en plusieurs sous-entités (qui correspondent aux Objets d'intérêts de la méthode COSMIC).

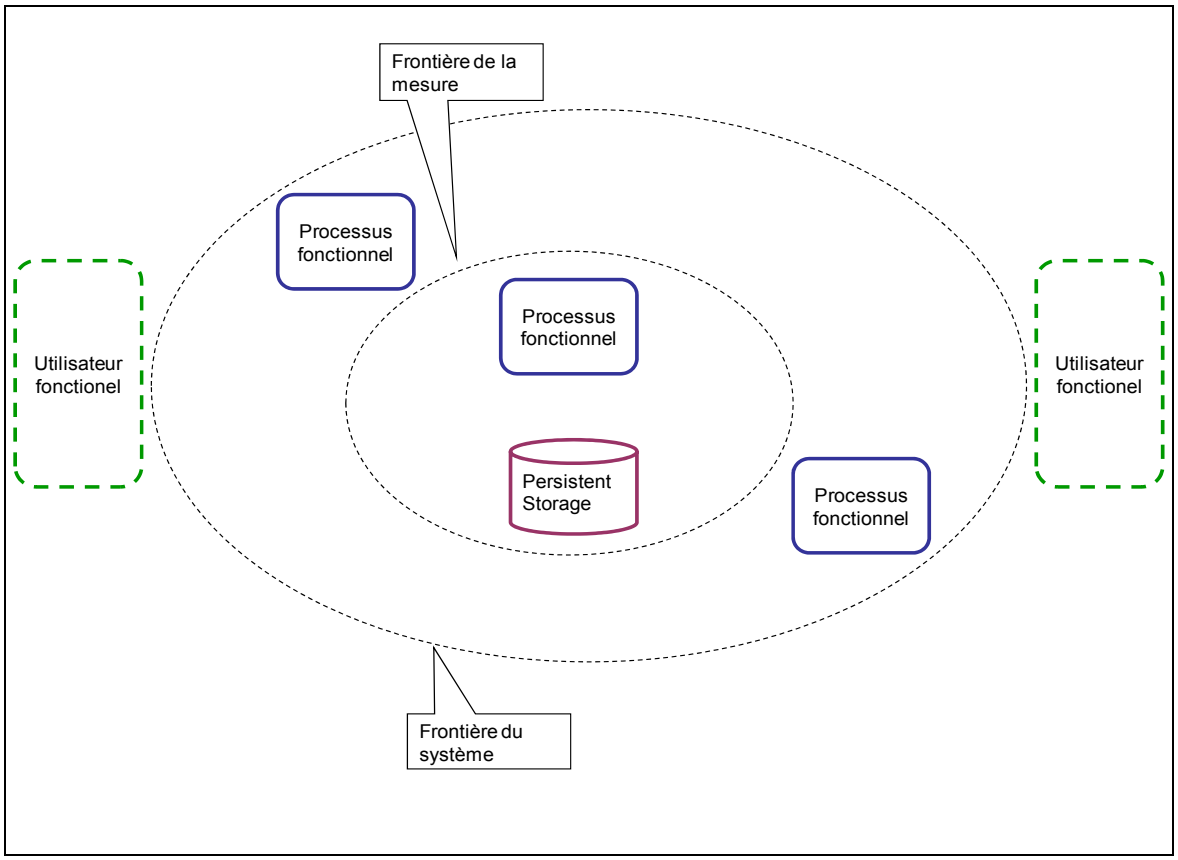
Le tableau 7.3 montre la notation de ces éléments externes.

Tableau 7.3 Types d'utilisateurs externes

Eléments	**Notation graphique**
PF homologue	
Utilisateur fonctionnel	

Le seul élément interne est le stockage persistant. Cet élément reste inchangé dans cette extension.

En utilisant ces notations, la représentation semi-formelle (sans flux) devient comme présentée dans la figure 7.3.



FFigure 7.3 Représentation étendue ajustée pour COSMIC (sans flux) du modèle de Paton et Abran

Pour illustrer ces éléments avec le modèle StateMate comme exemple, le PF cible correspond à une Activité de Contrôle (Control Activity) (voir chapitre 4) et les processus homologues sont les autres Activités de Contrôle qui sont présentes dans le système à mesurer. Les Entités externes correspondent aux Activités Externes (External Activities) et les sous-Entités externes correspondent aux sous-activités présentes dans les activités externes.

En suivant les mêmes hypothèses de la section 7.2.1, les flux possibles deviennent alors :

- il existe des flux entre chacun des PF homologues (sources des flux) et le processus-cible (destination des flux), soit les flux 1 et 3.
- vice versa, il existe des flux entre le processus-cible (source des flux) et les PF homologues (destinations des flux), soit les flux 2 et 4.

- il existe des flux entre chacune des entités externes (sources des flux) et le processus-cible (destination des flux), soit les flux 5 et 7.
- vice versa, il existe des flux entre le processus-cible (source des flux) et les entités externes (destinations des flux), soit les flux 6 et 8 - voir figure 7.4.

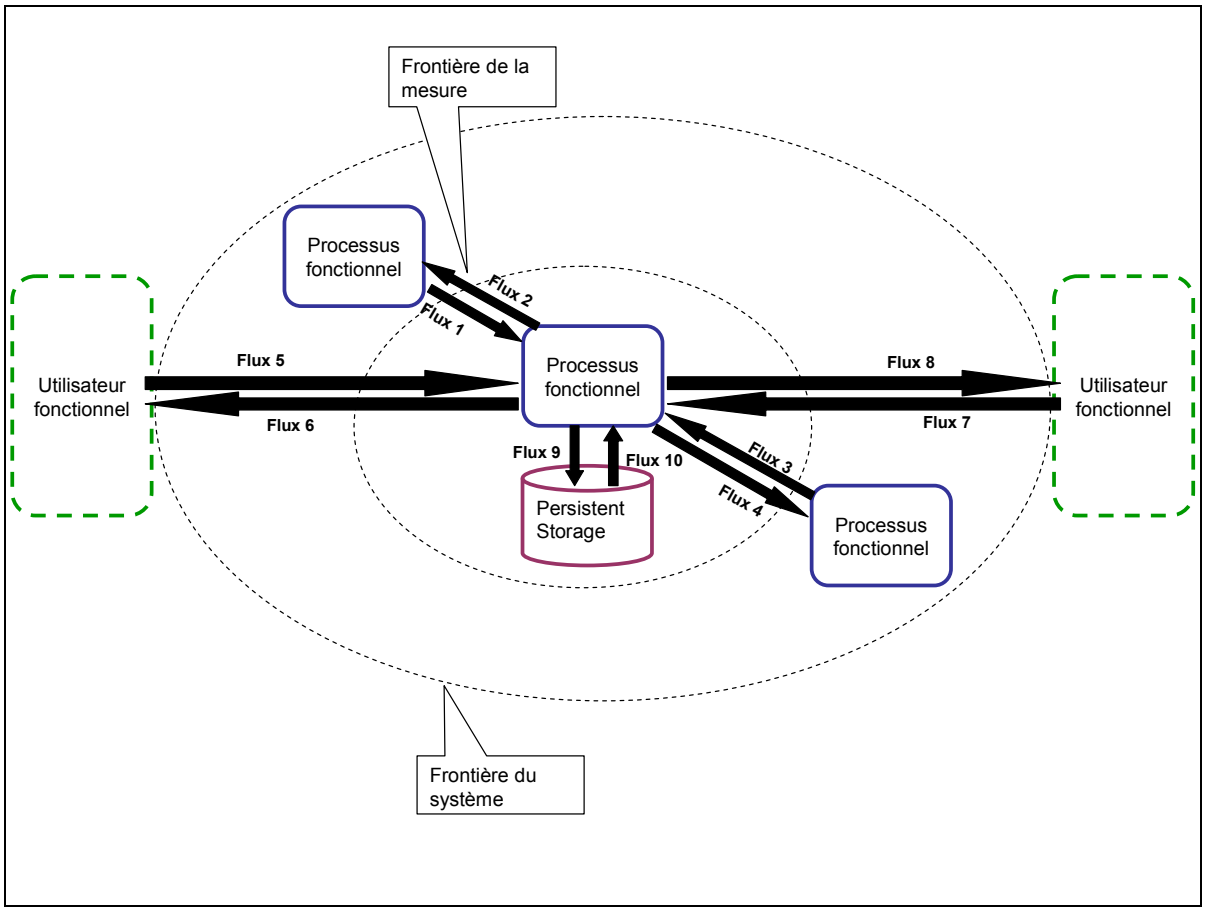


Figure 7.4 Représentation semi-formelle étendue ajustée
inspirée du modèle de Paton et Abran

Le tableau 7.4 montre le classement des flux selon leurs types de mouvements de données COSMIC est fait.

Tableau 7.4 Correspondance flux/type de mouvement

Flux	**Mouvement de données**
1, 3, 5, 7	E
2, 4, 6, 8	X
9	W
10	R

Pour respecter les règles COSMIC, on a besoin d'au moins une entrée E et une sortie X ou une écriture W. Donc, au moins un des flux 1, 3, 5, 7 doit impérativement être présent et au moins un des flux 2, 4, 6, 8, 9.

Le tableau 7.5 montre le statut des combinaisons de flux: les combinaisons de flux interdites sont statuées par KO. Les combinaisons permises sont statuées par OK.

Tableau 7.5 Combinaisons complètes de flux

Flux										**Statut* de la combinaison**
1	2	3	4	5	6	7	8	9	10	
Types de mouvements de données COSMIC										
E	X	E	X	E	X	E	X	W	R	
1	1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	OK
1	0/1	0/1	1	0/1	0/1	0/1	0/1	0/1	0/1	OK
1	0/1	0/1	0/1	0/1	1	0/1	0/1	0/1	0/1	OK
1	0/1	0/1	0/1	0/1	0/1	0/1	1	0/1	0/1	OK
1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	1	0/1	OK
0/1	1	1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	OK
0/1	0/1	1	1	0/1	0/1	0/1	0/1	0/1	0/1	OK
0/1	0/1	1	0/1	0/1	1	0/1	0/1	0/1	0/1	OK
0/1	0/1	1	0/1	0/1	0/1	0/1	1	0/1	0/1	OK
0/1	0/1	1	0/1	0/1	0/1	0/1	0/1	1	0/1	OK
0/1	1	0/1	0/1	1	0/1	0/1	0/1	0/1	0/1	OK
0/1	0/1	0/1	1	1	0/1	0/1	0/1	0/1	0/1	OK
0/1	0/1	0/1	0/1	1	1	0/1	0/1	0/1	0/1	OK
0/1	0/1	0/1	0/1	1	0/1	0/1	1	0/1	0/1	OK
0/1	0/1	0/1	0/1	1	0/1	0/1	0/1	1	0/1	OK
0/1	1	0/1	0/1	0/1	0/1	1	0/1	0/1	0/1	OK
0/1	0/1	0/1	1	0/1	0/1	1	0/1	0/1	0/1	OK
0/1	0/1	0/1	0/1	0/1	1	1	0/1	0/1	0/1	OK
0/1	0/1	0/1	0/1	0/1	0/1	1	1	0/1	0/1	OK
0/1	0/1	0/1	0/1	0/1	0/1	1	0/1	1	0/1	OK
0	0/1	0	0/1	0	0/1	0	0/1	0/1	0/1	KO
1	0	0/1	0	0/1	0	0/1	0	0	0/1	KO
0/1	0	1	0	0/1	0	0/1	0	0	0/1	KO
0/1	0	0/1	0	1	0	0/1	0	0	0/1	KO
0/1	0	0/1	0	0/1	0	1	0	0	0/1	KO

*selon la condition -5- décrite ci-dessus

Donc avec les ajustements et les extensions apportés, les combinaisons de flux présentées ci-dessus peuvent être utilisées pour représenter l'ensemble des entrées de tests à exécuter comme expliqué ci-après.

7.3 Test théorique de couverture

Ce test a comme but de démontrer que le prototype couvre toutes les possibilités de spécifications en entrée. Pour cela, il suffit de s'assurer que toutes les combinaisons des flux présentés ci-dessus sont testées. Toutefois, ceci est très difficile, voir impossible, de trouver en pratique tous les cas des combinaisons possibles, pour un nombre important de processus fonctionnels. Pour illustrer, en prenant le cas de 3 PF et de deux utilisateurs fonctionnels du paragraphe ci-dessus : le nombre de flux possibles s'élèvent à 2 flux entre 2 éléments, multiplié par le nombre d'éléments (5 dans ce cas), i.e. le nombre de flux= 2×5= 10. Le nombre d'entrées possibles s'élève à :

(Un 1 parmi les quatre flux 1, 3, 5, 7) ET (un 1 parmi les cinq flux 2, 4, 6, 8, 9)
En projetant dans la formule mathématique suivante : $C^n_p = n! / (p! (n-p)!)$;
Le nombre d'entrée est obtenu avec :

$$Nombre\ d'entrées = [C^4_1 = 4! / (1! (4-1)!)\ ] \times [C^5_1 = 5! / (1! (5-1)!)\ ]$$

Et,

$5! = 1 \times 2 \times 3 \times 4 \times 5 = 120$

$4! = 1 \times 2 \times 3 \times 4 = 24$

$3! = 1 \times 2 \times 3 = 6$

$1! = 1$

$$Nombre\ d'entrées = 24/6 \times 120/24 = 4 \times 5 = 20.$$

On retrouve le nombre d'entrées OK dans le tableau 7.5.

Le nombre de combinaisons se calcule en cherchant la combinaison de 10 éléments (2 flux parmi les dix doivent obligatoirement être à 1) où chaque élément peut prendre 2 valeurs différentes (0/1). En appliquant la formule p^n, avec n=10 et p=2. Mais comme 2 flux sont toujours fixés à 1 dans chaque entrée valide, le nombre de combinaison par entrée devient :

Avec :

$n=10-2=8$
$p=2$

$2^8 = 256$

Donc finalement, 256 combinaisons valides de flux par entrée sont possibles pour un PF qui interagit avec 2 autres PF homologues et deux utilisateurs externes. Pour l'ensemble des entrées valides possibles, on multiplie par 20, soit $256 \times 20 = 5120$ combinaisons possibles en global.

Donc pour les 3 PFs, le nombre de combinaisons devient : $3 \times 5120 = 15360$.

En généralisant, soit N le nombre de PFs possibles, le nombre de combinaisons de flux est alors, en fixant le nombre d'utilisateurs fonctionnels à deux : N×15360.

Cependant, si le nombre de PF tend à 100, le nombre de combinaison devient :

Lim N×15360=1536000.
N->1OO

Donc le nombre de combinaison de flux devient trop grand pour pouvoir tester tous les cas possibles.

A noter que chaque flux correspond au moins à un mouvement de données du type de mouvement qu'il représente. Soit M ce nombre. M théoriquement s'étend de 1 jusqu'à l'infini. En pratique et en moyenne, M se situe entre 1 et 25.

D’après l’explication précédente, une technique de tests plus réaliste en pratique devient une nécessité.

7.4 Technique de tests pratiques par « Analyseur de spectre »

Pour pouvoir tester le prototype, la technique de test s’inspirant du fonctionnement de l’analyseur de spectre [30] est utilisée.

Les analyseurs de spectre sont largement utilisés pour mesurer la réponse en fréquence, le bruit et les caractéristiques de distorsion de toutes sortes de circuits RF (Radiofréquence), en comparant les spectres d'entrée et de sortie. Ils peuvent être utilisés pour caractériser les signaux de test et mesurer la réponse de l'équipement à tester.

Cette technique consiste, dans le cadre du test du prototype d’automatisation, à utiliser une séquence de spécifications en entrée en partant de spécifications ne comportant qu’un seul processus fonctionnel et ayant la combinaison minimale obligatoire de mouvement de données, puis en progressant d’une manière méthodique et systématique pour arriver à la combinaison la plus complexe en terme de complétude de combinaisons de flux.

Le protocole utilise des échantillons au niveau de l'entrée de l'outil d'automatisation pour essayer de couvrir la plupart des types de cas d'entrée qui pourraient être rencontrés. Pour des fins de vérification, les échantillons doivent être mesurés manuellement, selon les procédures de mesure proposées. Ainsi, en plus de la taille fonctionnelle obtenue, tous les processus fonctionnels et les mouvements de groupes de données sont identifiés par la mesure manuelle. De plus, comme la source d'une erreur peut être le processus manuel de mesurage ou être causée par le prototype, l’application de ce protocole permet l’identification de la partie qui est responsable de tout genre d'erreur.

Un outil d'automatisation doit répondre à trois propriétés des procédures FSM:

- l'exactitude,

- la répétabilité et
- la reproductibilité.

Par exemple, la propriété d'exactitude est définie dans le VIM comme l'étroitesse de l'accord entre une quantité mesurée et la quantité vraie d'un mesurande [37]. La propriété de répétabilité peut être vérifiée et confirmée lorsque l'outil génère les mêmes résultats pour une spécification donnée en entrée dans les mêmes conditions. La reproductibilité est vérifiée si deux outils différents qui mettent en œuvre les mêmes procédures FSM, donnent le même résultat pour la même entrée.

Pour utiliser cette technique, les spécifications (à mesurer) en entrée du prototype sont divisée en 3 groupes dans lesquelles se trouvent les échantillons d'entrée du prototype. Ces groupes sont aussi divisés en sous-groupes en fonction du nombre de mouvements de données. Ces groupes sont définis comme présentés dans le tableau 7.6.

Tableau 7.6 Séparations des groupes d'échantillons des entrées

Nombre de PFs	Nombre de mouvements de données par PF	Groupes de tests
1 à 5	2 à 4	A
	5 à 10	B
	10 à 50	C
	50 à 100	D
	>100	E
6 à 10	2 à 4	F
	5 à 10	G
	10 à 50	H
	50 à 100	I
	>100	J
>10	2 à 4	K
	5 à 10	L
	10 à 50	M
	50 à 100	N
	>100	O

7.4.1 1ère phase : comparaison des résultats numériques finaux

Dans cette phase, les résultats (en termes de la CFP - points de fonction COSMIC) produits automatiquement par le prototype et ceux obtenus à partir du processus manuel de mesurage de la même entrée sont comparés. S'il ya un match entre ces deux séries de résultats, cela signifie qu'aucune différence n'a été détectée entre la mesure manuelle et la mesure automatisée. L'évaluation peut s'arrêter à ce stade. Toutefois, on peut continuer à vérifier, même s'il n'y a pas de différence au niveau du total (résultat final), qu'il n'y a pas des différences importantes au niveau des processus eux-mêmes. Ceci est très important si le nombre d'échantillons de spécifications utilisées en entrée est petit.

Tableau 7.7 Format de la présentation des résultats en boîte noire

Données	Résultat en manuel	Résultat en automatique	Match ?
Nombre des E	x	y	Oui/non
Nombre des X	x	y	Oui/non
Nombre des W	X	y	Oui/non
Nombre des R	X	y	Oui/non
Taille fonctionnelle totale obtenue	X	y	Oui/non

7.4.2 2ème phase : Comparaison détaillée

Si les résultats numériques finaux à la fin de la phase 1 ne correspondent pas, la phase 2 commence par la comparaison des résultats au niveau détaillé, ce qui signifie la vérification du nombre des processus fonctionnels obtenus automatiquement et manuellement sont égaux, et donc, une vérification du mesurage manuel au cas d'une erreur humaine en parallèle à la vérification détaillée du mesurage automatique :

1. Si aucune différence n'est observée dans le nombre de processus fonctionnels, chaque processus fonctionnel obtenu est automatiquement vérifié par rapport à ses «homologues» obtenus manuellement, afin de déterminer si oui ou non il ya une différence dans leurs noms (ou leurs identificateurs).

2. Si tous les processus fonctionnels obtenus automatiquement correspondent à leurs «homologues» obtenus manuellement, leurs tailles fonctionnelles sont comparées : s'il y a des différences, un ou plusieurs mouvements de données à l'intérieur du processus fonctionnel doivent être responsable.

À la fin de cette phase, les mouvements de données responsables de l'erreur sont isolés.

Le résultat des essais en boîte noire suit le format de la Tableau 7.8 :

Tableau 7.8 Format de la présentation des résultats en boîte grise

Données	**Résultat en manuel**	**Résultat en automatique**	**Match ?**
Nombre de PF	x	y	Oui/non
PF n°=1	x	y	Oui/non
	Mouvement de données n°=1	Mouvement de données n°=1	Oui/non
	Mouvement de données n°=2	Mouvement de données n°=2	Oui/non
	. . .	. . .	Oui/non
	Mouvement de données n°=n	Mouvement de données n°=n	Oui/non
PF n°=2	x	y	Oui/non
	Mouvement de données n°=1	Mouvement de données n°=1	Oui/non
. . .	Mouvement de données n°=2	Mouvement de données n°=2	Oui/non
PF n°=n	. . .	. . .	Oui/non
	Mouvement de données n°=n	Mouvement de données n°=n	Oui/non

7.4.3 3ème phase : l'inspection du prototype et de la spécification

Cette phase est déclenchée lorsque l'erreur humaine (au niveau du mesurage manuel) est écartée à la fin de la phase 2.

Puisqu'une erreur détectée peut avoir deux sources, cette phase consiste :

1. D'une part, à l'inspection du module (pour le prototype 1 développé pour Renault, les modules sont : le parseur, le générateur de données, ou l'unité de mesure) du prototype responsable de l'erreur, et

2. D'autre part et en parallèle, l’inspection de la spécification d'entrée pour trouver un éventuel défaut qui puisse être l’origine de l'erreur.

Une fois qu’un bug et/ou un défaut est détecté, il est enregistré. Si le problème peut être résolu s’il a été causé par le prototype, la spécification appropriée est réévaluée avec la nouvelle version du prototype. Si le problème a été dans la spécification d'entrée, le défaut responsable est enregistré pour d’éventuelles améliorations futures de la spécification fonctionnelle en entrée et/ou une implémentation spécifique pour résoudre ce type de problème rencontré dans les spécifications du même genre est implémentée dans l’outil.

Finalement, la nouvelle version corrigée de la spécification d'entrée est ensuite vérifiée par ce protocole pour s’assurer que les résultats des mesurages manuel et automatique sont égaux.

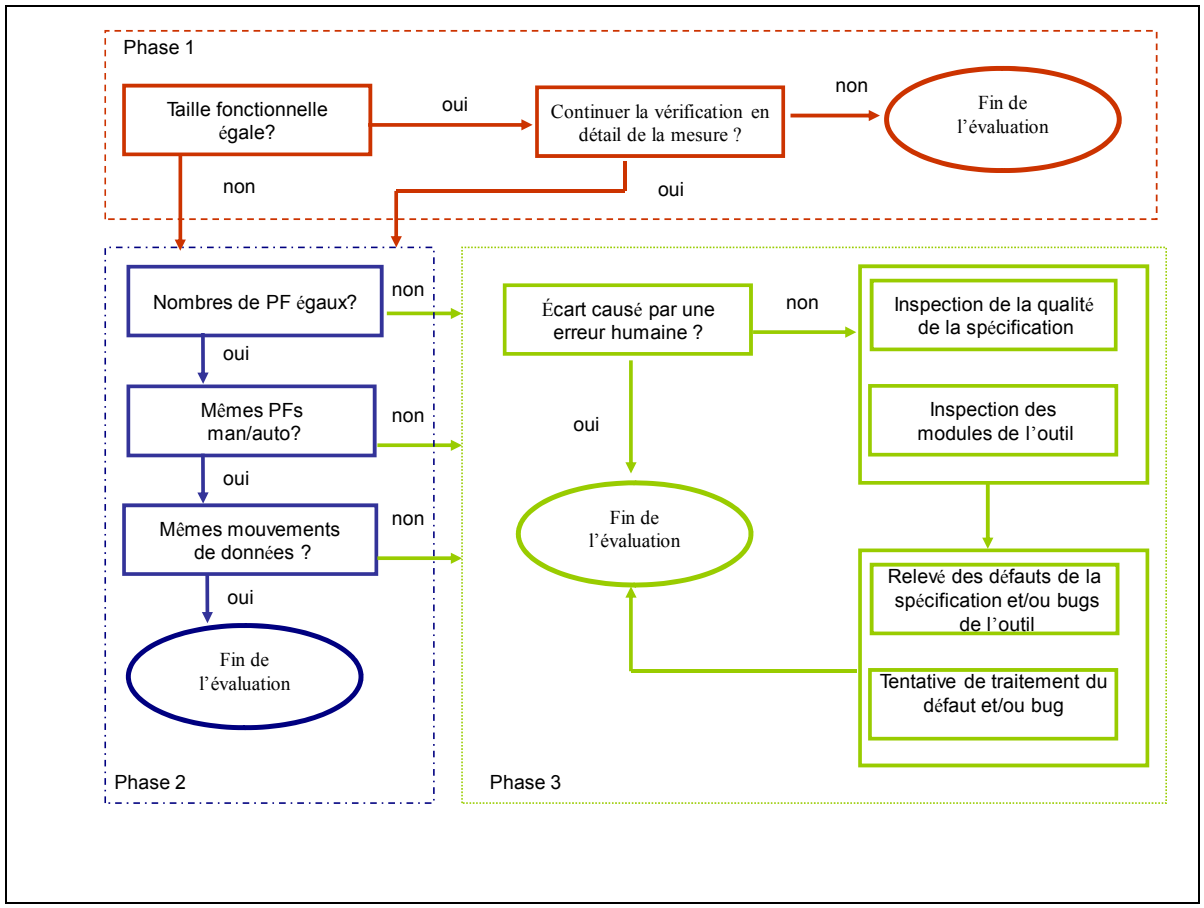


Figure 7.5 Phases du protocole d'évaluation

7.5 Application du protocole d'évaluation sur la spécification fonctionnelle du prototype 1

A titre d'exemple et en utilisant le résultat du processus manuel de mesurage obtenu en utilisant la procédure de mesurage décrite au chapitre 5 et celui du processus automatique du prototype 1, le résultat de l'application du protocole d'évaluation est présenté dans le tableau 7.9: il n'y a pas de différence dans les résultats entre les deux processus de mesurage. Le nombre de processus fonctionnels et leurs identificateurs étaient exactement les mêmes. Enfin, les mêmes mouvements de groupes de données ont été identifiés dans les deux résultats de mesures.

Tableau 7.9 Application du protocole d'évaluation sur le prototype 1

Phase numéro	**Sous-étape**	**Mesure manuelle**	**Mesure automatisée**	**Sortie de la sous-étape**
1	Vérification si les résultats finaux sont égaux	168 CFP	168 CFP	OUI, les résultats finaux sont égaux
1	Continuer la vérification en détail de la mesure ?			OUI: début phase 2 (évaluation détaillée)
2	Nombre égal de Processus Fonctionnels ?	19	19	OUI

Tableau 7.9 (Suite) Application du protocole d'évaluation sur le prototype 1

Phase numéro	**Sous-étape**	**Mesure manuelle**	**Mesure automatisée**	**Sortie de la sous-étape**
2	Mêmes PF identifiés dans les deux mesures?	1. OpenSpec 2. SeekPossibleFP 3. CompletePossibleFP 4. AlreadyCheckedBlocks 5. ReorganizeBlox 6. Special_treatment 7. GetArrows 8. RIP_FP 9. IsElementaryBlock 10. ReturnOnlyFPs 11. GotoFromTreat 12. GenerateSemiFormalModel 13. isConnectedFrom 14. AlreadyChecked 15. GetAllDataMovements 16. GUI 17. createAndShowGUI 18. BorderLayoutGUI 19. actionPerformed	1. OpenSpec 2. SeekPossibleFP 3. CompletePossibleFP 4. AlreadyCheckedBlocks 5. ReorganizeBlox 6. Special_treatment 7. GetArrows 8. RIP_FP 9. IsElementaryBlock 10. ReturnOnlyFPs 11. GotoFromTreat 12. GenerateSemiFormalModel 13. isConnectedFrom 14. AlreadyChecked 15. GetAllDataMovements 16. GUI 17. createAndShowGUI 18. BorderLayoutGUI 19. actionPerformed	OUI
2	Mêmes mouvements de données identifiés?	{ E:39 X:37 R:46 W:46 }	{ E:39 X:37 R:46 W:46 }	OUI: Fin de l'évaluation détaillée

CHAPITRE 8

APPLICATION DU PROTOCOLE SUR LA BATTERIE DE SPÉCIFICATIONS RENAULT

8.1 Introduction

L'application du protocole générique d'évaluation des outils de mesurage chez Renault a nécessité un échantillonnage des spécifications du secteur contrôle moteur thermique-DCMAP, Renault (utilisateurs prioritaire du prototype 1). Les spécifications du contrôle moteur sont distribuées sur plusieurs modules. Le prototype 1 a été testé sur 12 types de modules choisis de manière à mélanger des modules de différentes natures.

Les 77 spécifications utilisées ont été choisies parmi différents logiciels de contrôles moteurs Renault. Ces spécifications proviennent de modules de différentes natures, réalisées par des spécificateurs différents. L'intérêt de ce choix est de couvrir la diversité de modélisation des spécifications sous Simulink et de montrer par là, la pertinence du prototype de mesure dans un univers industriel.

Tableau 8.1 La liste des 77 différentes spécifications avec leurs nombres de PF

Numéro de la spécification testée	Nombre de processus Fonctionnels
1	1
2	10
3	1
4	2
5	19
6	6
7	2
8	2
9	0
10	4
11	17
12	1
13	11
14	2
15	2
16	2
17	2
18	1
19	3
20	7
21	5
22	1
23	3
24	1
25	1
26	1
27	1
28	4
29	2
30	1
31	4
32	1
33	1
34	1
35	4
36	4

Tableau 8.1 (Suite) La liste des 77 différentes spécifications
avec leurs nombres de PF

Numéro de la spécification testée	Nombre de processus Fonctionnels
37	4
38	1
39	1
40	1
41	1
42	1
43	1
44	5
45	5
46	3
47	10
48	6
49	1
50	1
51	2
52	1
53	11
54	5
55	1
56	5
57	5
58	1
59	3
60	3
61	1
62	3
63	2
64	1
65	7
66	1
67	2
68	1
69	1
70	1
71	5
72	5
73	1
74	3

Tableau 8.1 (Suite) La liste des 77 différentes spécifications
avec leurs nombres de PF

Numéro de la spécification testée	Nombre de processus Fonctionnels
75	3
76	1
77	1

Les spécifications choisies ont été échantillonnées pour obtenir différentes spécifications par module, de natures et de « tailles » différentes (taille en kilo-octet ou en nombre de pages). Ceci n'a pas un impact sur le protocole d'évaluation lui-même, mais ceci pouvait être très utile pour pouvoir démontrer la couverture des tests sur l'ensemble des types de spécifications Renault

Toutefois, on peut clairement classer les spécifications selon le nombre de processus fonctionnels (PF). Le tableau 8.2 montre ce classement : le nombre et les numéros des spécifications en fonctions du nombre de PF qu'elles contiennent. Le graphe de la figure 8.1 montre la répartition de ces spécifications en fonction du nombre de processus fonctionnels.

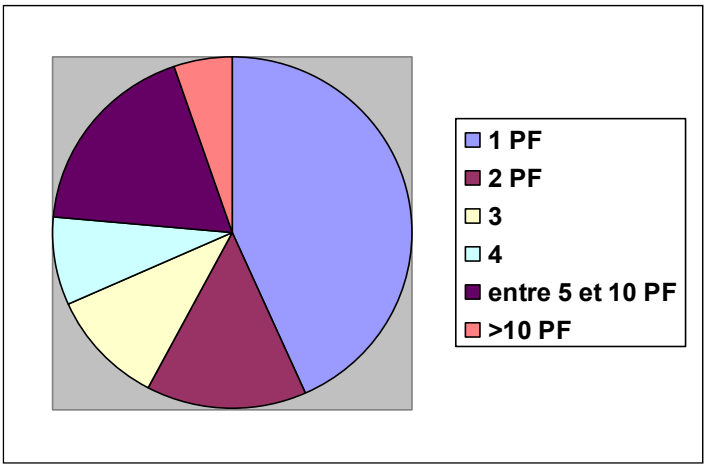


Figure 8.1 Répartition des spécifications en fonction
du nombre de processus fonctionnels

Tableau 8.2 Le nombre de Processus Fonctionnels (PF) dans les spécifications étudiées

Nombre de PF	Nombre de spécifications avec X PFs	Numéros des spécifications
0	1	9
1	34	1, 3, 12, 18, 22, 24, 25, 26, 27, 30, 32, 33, 34, 38, 39, 40, 41, 42, 43, 49, 50, 52, 55, 58, 61, 64, 66, 68, 69, 70, 73, 76, 77
2	11	4, 7, 8, 14, 15, 16, 17, 29, 51, 63, 67
3	8	19, 23, 46, 59, 60, 62, 74, 75
4	6	10, 28, 31, 35, 36, 37
5	8	21, 44, 45, 54, 56, 57, 71, 72
6	2	6, 48
7	2	20, 65
10	2	2, 47
11	2	13, 53
17	1	11
19	1	5

8.2 Résultats et analyse

Les différents tests réalisés du prototype 1 (prototype d'automatisation du mesurage de la taille fonctionnelle des nouveaux développements des logiciels en utilisant leurs spécifications fonctionnelles exprimées en Simulink) sont sur les spécifications dont des nombres de processus fonctionnels variant entre 1 et 19.

Les tests ont eu lieu en plusieurs phases à des dates différentes, mais avec le même lot du départ (18 spécifications) et en ajoutant de nouvelles spécifications à ce lot (nombre total de spécifications testées: 77).

8.2.1 1ère phase de tests

Le résultat des tests du lot initial est présenté dans le tableau 8.3. Ces tests ont été exécutés avec la 1ère version du prototype 1.

Tableau 8.3 Résultats du 1er lot de tests

Numéro de la spécification testée	Résultat de la mesure effectuée manuellement (en CFP)	Résultat d'après le Proto 1 (en CFP)	Différence en CFP
1	5	11	6
2	81	86	5
3	10	25	15
4	14	14	0
5	117	81	36
6	12	14	2
7	4	4	0
8	17	18	1
9	0	0	0
10	75	93	8
11	91	85	6
12	6	6	0
13	62	53	9
14	8	8	8
15	19	16	3
16	9	6	3
17	9	6	3
18	7	7	0

Au départ, sur les 18 spécifications ont été testées, les résultats sont identiques pour seulement 6 spécifications. La phase 2 du protocole a été exécutée. En vérifiant le processus manuel de mesurage, tout en se basant sur le rapport de mesure du prototype 1 pour pouvoir bien suivre l'identification des éléments (tâches en parallèle), les différences suivantes ont été éliminées :

1. Spécification n=° 2: le re-mesurage manuel a bien donné un résultat de mesure de 86 CFP.
2. Spécification n=° 6: le re-mesurage manuel a bien donné un résultat de mesure de 14 CFP.
3. Spécification n=° 8: le re-mesurage manuel a bien donné un résultat de mesure de 18 CFP.

D'autres différences sont causées par la manière de spécifier propre au secteur de spécification du contrôle moteur de Renault :

4. Pour la Spécification n=° 1 le prototype donnait 1 processus fonctionnel en plus : cela est dû à l'utilisation spécifique des blocs du type **Subsystem** comme librairies (Librairies RSA spécifique au département DCMAP de Renault et le mesureur en manuel n'a pas la possibilité de visualiser l'intérieur de ces blocs). Pour résoudre cette différence, on a été amené à implémenter une fonctionnalité qui oblige le prototype 1 à ne pas entrer dans les blocs Librairie RSA : i.e. lorsque le prototype trouve un **Subsystem** dont le nom commence avec RSA, il le considère comme un Block élémentaire et non-plus un PF potentiel.

5. Spécification n=° 3: dans cette spécification un block de type **Subsytem** porte le même nom que son type (i.e. « subsystem »). Puisque le prototype 1 parse la spécification en utilisant les noms et les types, ceci cause un problème : le prototype se perdait entre le type et le nom. Le problème a été résolu en changeant le nom du block dans la spécification en premier (version novembre 2010), et puis en implémentant une solution directement dans l'outil (version Janvier 2011) qui consiste à faire la différence entre le nom dont la première lettre doit être en majuscule et le type qui est en minuscule.

6. Spécification n=° 5: le prototype 'oublie' des entrées. Cela était dû à l'utilisation spécifique des blocs du type **RSA_demux_fc** spécifique au département DCMAP de

Renault. Pour résoudre ce problème, on a été amené à implémenter une fonctionnalité qui calibre le prototype 1 pour prendre en compte ces blocs en tant que blocs élémentaires.

7. Spécification n=° 10: Librairies RSA : problème résolu avec la solution du même problème que la Spécification n°= 1.

8. Spécification n=° 11: l'écart semblait provenir uniquement des librairies et blocs RSA, mais l'écart n'a pas été résolu. Ceci exige une analyse plus poussée.

9. Spécification n=° 13: nécessitait la MAJ de la liste des blocs élémentaires dans le prototype: i.e. dans la liste des blocs élémentaires que le prototype 1 utilise pour analyser si un block donné est ou non un block élémentaire (fonction dans le module générateur de données du prototype 1).

10. Spécification n=° 15, Spécification n=° 16, Spécification n=° 17: le mesurage manuel prend en compte la vue dite "root" du système. D'après la règle no= 4 ("Identify 1 functional process for each subsystem containing at least one elementary block") de la procédure de mesurage pour Simulink, seuls les blocs du type subsytem sont identifiables pour être comptabilisés, et non pas la vue 'root'. Pour prendre en compte les éléments présents dans la 'vue root' il faut l'enrober dans un Subsystem. Le mesurage manuel initial était donc faux. Les re-mesurage manuels ont bien donné respectivement 16, 6 et 6 CFP.

8.2.2 2ème phase de tests

Dans la deuxième phase de test, les mêmes spécifications ont été re-testées avec la nouvelle version du prototype 1, et les résultats sont présentés au tableau 8.4.

Tableau 8.4 Résultats des 1er et 2ème lots de tests

Numéro de la spécification testée	Résultat en manuel - en CFP (phase 1 de test)	Résultat d'après le Proto 1- en CFP (phase 1 de test)	Résultat en manuel – en CFP (phase 2 de test)	Résultat d'après le Proto 1 – en CFP (phase 2 de test)	Différence en CFP
1	5	11	5	5	0
2	81	86	87	87	0
3	10	25	10	10	0
4	14	14	14	14	0
5	117	81	117	115	2
6	12	14	14	14	0
7	4	4	4	4	0
8	17	18	18	18	0
9	0	0	0	0	0
10	75	93	75	75	0
11	91	85	91	97	6
12	6	6	6	6	0
13	62	53	62	71	9
14	8	8	8	8	0
15	19	16	16	16	0
16	9	6	6	6	0
17	9	6	6	6	0
18	7	7	7	7	0

Avec cette nouvelle série de tests, et donc après les modifications/implémentations au niveau du prototype 1, et la correction de certains résultats de mesure obtenus par l'application manuelle de la procédure de mesurage, il ne reste que trois spécifications où les résultats sont différents :

1. La Spécification n=° 5. Désormais le prototype prend en compte les blocs **RSA_demux_fc** en tant que block élémentaires. Mais un écart existe toujours.

2. La Spécification n=° 11. Le prototype prend des blocs spécifiques au secteur de spécification: **RSA_demux_fc et EpmHCrs_GetCurrPos** en tant que sorties de PFs lors de la création de la représentation semi-formelle et c'est tout à fait normal parce qu'ils sont du type subsystem (de la même nature que les PF pris en compte)**.** Une nouvelle fonctionnalité a été implémentée dans le prototype 1 pour résoudre ce problème: pour ces blocs spécifiques et récurrents, lorsque le prototype 1 les identifie, il les considère comme des blocs élémentaires.

3. La Spécification n=° 13. Une limitation actuelle du prototype impose que si les spécificateurs veulent utiliser plusieurs fois la même sortie, ils doivent utiliser les Goto/From à la place des fils (cas idéal et « propre » de spécifier). Pour le moment, les sorties 'tirées' par plusieurs fils sont comptées à chaque occurrence: une solution pouvait être proposée en fonction des noms des fils, mais les spécificateurs devraient fixer ces noms.

8.2.3 3ème phase de tests

Il faut noter qu'après avoir discuté avec les référents du métier chez Renault, il a été convenu de faire évoluer la manière d'identifier et d'interpréter des blocs « constantes » (**constant**) dont l'utilisation est spécifique dans les spécifications du contrôle moteur: désormais toute constante avec une valeur numérique doit être prise en compte même si elle est produite en plusieurs occurrences. Ceci est du au fait que le spécificateur ne peut pas exprimer explicitement si la constante utilisée est la même ou non : une constante de valeur '1' peut être une constante qui est l'entier '1' mais elle peut aussi être le 'vrai' logique. Tandis que les constantes non-numériques ne doivent pas être comptées qu'une seule fois, même si elles sont produites en plusieurs occurrences, parce que dans ce cas l'interprétation derrière l'utilisation de la constante n'existe pas.

Donc suite à cette modification et aux correction/implémentation apportées à l'issue des tests de novembre 2010, le lot de spécifications (initialement de 18) a été complété par l'ajout de

spécifications, jusqu'au total de 77 spécifications testées. Les résultats sont présentés dans le tableau 8.5.

Tableau 8.5 Résultats des 2ième et 3ième lots de tests

Numéro de la spécification testée	Résultat en manuel – en CFP (phase 2 de test)	Résultat d'après le Proto 1 – en CFP (phase 2 de test)	Résultat en manuel – en CFP (phase 3 de test)	Résultat d'après le Proto 1 – en CFP (phase 3 de test)	Différence en CFP
1	5	5	5	5	0
2	87	87	87	87	0
3	10	10	10	25*	15
4	14	14	14	13	0
5	117	115	117	112	5
6	14	14	14	14	0
7	4	4	4	4	0
8	18	18	18	18	0
9	0	0	0	0	0
10	75	75	75	75	0
11	91	97	91	97	6
12	6	6	6	6	0
13	62	71	62	71	9
14	8	8	8	8	0
15	16	16	16	16	0
16	6	6	6	6	0
17	6	6	6	6	0
18	7	7	7	7	0
19	N/A	N/A	37	35	2
20	N/A	N/A	53	53	0
21	N/A	N/A	35	35	0
22	N/A	N/A	12	12	0
23	N/A	N/A	27	27	0
24	N/A	N/A	3	3	0
25	N/A	N/A	27	27	0
26	N/A	N/A	12	12	0
27	N/A	N/A	14	14	0
28	N/A	N/A	66	68	2
29	N/A	N/A	5	5	0
30	N/A	N/A	1	1	0
31	N/A	N/A	19	19	0
32	N/A	N/A	1	1	0

Tableau 8.5 (Suite) Résultats des 2ième et 3ième lots de tests

Numéro de la spécification testée	Résultat en manuel – en CFP (phase 2 de test)	Résultat d'après le Proto 1 – en CFP (phase 2 de test)	Résultat en manuel – en CFP (phase 3 de test)	Résultat d'après le Proto 1 – en CFP (phase 3 de test)	Différence en CFP
33	N/A	N/A	3	3	0
34	N/A	N/A	28	28	0
35	N/A	N/A	22	22	0
36	N/A	N/A	38	38	0
37	N/A	N/A	23	23	0
38	N/A	N/A	1	1	0
39	N/A	N/A	25	25	0
40	N/A	N/A	1	1	0
41	N/A	N/A	1	1	0
42	N/A	N/A	10	10	0
43	N/A	N/A	1	1	0
44	N/A	N/A	48	48	0
45	N/A	N/A	34	34	0
46	N/A	N/A	15	15	0
47	N/A	N/A	124	124	0
48	N/A	N/A	65	65	0
49	N/A	N/A	4	4	0
50	N/A	N/A	4	4	0
51	N/A	N/A	15	15	0
52	N/A	N/A	3	3	0
53	N/A	N/A	52	53	1
54	N/A	N/A	47	46	1
55	N/A	N/A	3	3	0
56	N/A	N/A	22	22	0
57	N/A	N/A	51	51	0
58	N/A	N/A	5	5	0
59	N/A	N/A	18	18	0
60	N/A	N/A	13	13	0
61	N/A	N/A	3	3	0
62	N/A	N/A	29	51	22
63	N/A	N/A	27	27	0
64	N/A	N/A	3	3	0
65	N/A	N/A	32	32	0

Tableau 8.5 (Suite) Résultats des 2ième et 3ième lots de tests

Numéro de la spécification testée	Résultat en manuel – en CFP (phase 2 de test)	Résultat d'après le Proto 1 – en CFP (phase 2 de test)	Résultat en manuel – en CFP (phase 3 de test)	Résultat d'après le Proto 1 – en CFP (phase 3 de test)	Différence en CFP
66	N/A	N/A	3	3	0
67	N/A	N/A	11	11	0
68	N/A	N/A	14	14	0
69	N/A	N/A	8	8	0
70	N/A	N/A	16	17	1
71	N/A	N/A	18	18	0
72	N/A	N/A	16	16	0
73	N/A	N/A	1	1	0
74	N/A	N/A	10	10	0
75	N/A	N/A	46	46	0
76	N/A	N/A	5	5	0
77	N/A	N/A	2	2	0

*valeur obtenue sans le prétraitement (à l'extérieur du prototype 1) relatif à la résolution de l'écart

1. Pour la Spécification n=° 5, l'outil compte moins de CFP suite à un bug relatif à la nouvelle règle des constantes.

2. Pour les spécifications n=° 11 et n=° 13 : voir le paragraphe précédent.

3. Pour la Spécification n=° 4, suite à l'implémentation des nouvelles règles sur les blocs « constantes », la valeur d'une constante "constant" n'est pas définie. La constante « constant1 » a pour valeur "null" dans Simulink. Il s'agit ici d'une limitation de cette version du prototype 1.

4. Pour les spécifications n=° 19 et n=° 47, suite à l'implémentation des nouvelles règles sur les blocs « constantes », lorsque la valeur numérique d'une constante est différente de '0'

et'1' le prototype n'identifie pas les autres valeurs. La solution à ce problème a été implémentée: toute constante « numérique » est désormais prise en compte.

5. Pour les spécifications n=° 28, n=° 53 et n=° 70 : l'écart est dû au fait que le prototype compte chaque occurrence d'un bloc « From », même s'ils proviennent d'un même bloc « Goto ». C'est une limitation de la librairie du prototype 1 face à Simulink (quant à l'ID des blocs). Cette limitation sera éliminée dans la version industrielle du prototype.

6. Pour la spécification n=° 62, après discussion avec les référents du métier, la spécification est jugée non-propre : le spécificateur a utilisé un bloc de librairie, a nettoyé partiellement le masque, pour faire des modifications dedans. Les mesureurs en manuel n'ont pas regardé sous le masque le détail de ces blocs.

8.2.4 4ème phase de tests

Les 77 spécifications testées à la troisième phase ont été re-testées avec la nouvelle version du prototype 1, et les résultats sont présentés au tableau 8.6.

Tableau 8.6 Résultats des 3ème et 4ème lots de tests, ainsi que le pourcentage d'écart

Numéro de la spécification testée	Résultat en manuel – en CFP (phase 3 de test)	Résultat d'après le Proto 1 – en CFP (phase 3 de test)	Résultat en manuel – en CFP (phase 4 de test)	Résultat d'après le Proto 1 – en CFP (phase 4 de test)	Pourcentage d'écart
1	5	5	5	5	0%
2	87	87	87	87	0%
3	10	25	10	10	0%
4	14	13	14	14	0%
5	117	112	115	113	1,7%
6	14	14	14	14	0%
7	4	4	4	4	0%
8	18	18	18	18	0%
9	0	0	0	0	0%
10	75	75	75 / 57	57	0%
11	91	97	91	91	0%
12	6	6	6	6	0%
13	62	71	62	71	12,6%
14	8	8	8	8	0%
15	16	16	16	16	0%
16	6	6	6	6	0%
17	6	6	6	6	0%
18	7	7	7	7	0%
19	37	35	37	37	0%
20	53	53	53	53	0%
21	35	35	35	35	0%
22	12	12	12	12	0%
23	27	27	27	27	0%
24	3	3	3	3	0%
25	27	27	27	27	0%
26	12	12	12	12	0%
27	14	14	14	14	0%
28	66	68	66	68	2.9%
29	5	5	5	5	0%
30	1	1	1	1**	0%
31	19	19	19	19	0%

Tableau 8.6 (Suite) Résultats des 3ème et 4ème lots de tests, ainsi que le pourcentage d'écart

Numéro de la spécification testée	Résultat en manuel – en CFP (phase 3 de test)	Résultat d'après le Proto 1 – en CFP (phase 3 de test)	Résultat en manuel – en CFP (phase 4 de test)	Résultat d'après le Proto 1 – en CFP (phase 4 de test)	Pourcentage d'écart
32	1	1	1	1**	0%
33	3	3	3	3	0%
34	28	28	28	28	0%
35	22	22	22	22	0%
36	38	38	38	38	0%
37	23	23	23	23	0%
38	1	1	1	1**	0%
39	25	25	25	25	0%
40	1	1	1	1**	0%
41	1	1	1	1**	0%
42	10	10	10	10	0%
43	1	1	1	1**	0%
44	48	48	48	48	0%
45	34	34	34	34	0%
46	15	15	15	15	0%
47	124	124	124	124	0%
48	65	65	65	65	0%
49	4	4	4	4	0%
50	4	4	4	4	0%
51	15	15	15	15	0%
52	3	3	3	3	0%
53	52	53	52	53	1.8%
54	47	46	47	46	2.1%
55	3	3	3	3	0%
56	22	22	22	22	0%
57	51	51	51	51	0%
58	5	5	5	5	0%
59	18	18	18	18	0%
60	13	13	13	13	0%
61	3	3	3	3	0%
62	29	51	29	52	44.2%
63	27	27	27	27	0%
64	3	3	3	3	0%

Tableau 8.6 (Suite) Résultats des 3ème et 4ème lots de tests, ainsi que le pourcentage d'écart

Numéro de la spécification testée	Résultat en manuel – en CFP (phase 3 de test)	Résultat d'après le Proto 1 – en CFP (phase 3 de test)	Résultat en manuel – en CFP (phase 4 de test)	Résultat d'après le Proto 1 – en CFP (phase 4 de test)	Pourcentage d'écart
65	32	32	32	32	0%
66	3	3	3	3	0%
67	11	11	11	11	0%
68	14	14	14	14	0%
69	8	8	8	8	0%
70	16	17	16	17	5.8%
71	18	18	18	18	0%
72	16	16	16	16	0%
73	1	1	1	1	0%
74	10	10	10	10	0%
75	46	46	46	46	0%
76	5	5	5	5	0%
77	2	2	2	2	0%

Certaines spécifications Renault font une taille de 1 CFP parce qu'il s'agit de spécifications spéciales où certains types de blocs ont volontairement été ignorés (exemple bloc **Terminator).

1. La Spécification n=° 5. Avec la nouvelle version du proto on obtient 113 CFP. Le re-mesurage manuel a bien donné un résultat de mesure de 116 CFP (date 1/02/20011 par le mesureur 1 LF, 115 par les mesureurs 2 et 3 le 03/02/2011). Les écarts du point CFP sont causés : au sein du PF : *B1_Stopped_engine_computation* au niveau du trigger qui est écrit avec des espaces *Vbx_eng_pbt_stp or not Vbx_sens_pbt_v_fil_rdy*. L'écart est éliminé une fois que le nom dans la spécification est corrigé. Au sein du PF E11_Filtering (block *Unit delay* qui est un sub/librairie/mask)

2. La Spécification n=° 10. le re-mesurage manuel a bien donné un résultat de mesure de 57 CFP.

3. Les spécifications n=° 13, n=° 28, n=° 53, n=° 62, et n=° 70. Voir l'explication sur les problèmes identifiés avec ces spécifications dans le paragraphe précédent.

4. La spécification n=° 19. Problème de constantes multiples, résolu avec la toute nouvelle version du prototype.

5. La spécification n=° 54: au niveau du PF **Event_odtc_and_dgtc**. l'écart est dû au fait que le prototype compte chaque occurrence d'un FROM, même s'ils proviennent d'un même GOTO. C'est une limitation de la librairie du prototype 1 face à la limitation Simulink (quant à l'ID des blocs). Normalement, avec cette limitation, on devait se retrouver avec deux points CFP en plus (2 E), mais le fait que le spécificateur a ajouté un espace dans un des blocs 'From' (nom du block est : 'From1 3'), et le prototype s'est retrouvé avec deux blocs « From » qui portent le même nom (i.e. 'From1', due au parsing) et il n'a compté qu'un seul de ces blocs, d'où le seul point CFP d'écart.

8.3 Vérification des spécifications qui ne présentaient pas d'écart dans le résultat final

Comme l'indique le protocole d'évaluation des outils de mesurage automatique basé sur COSMIC, les spécifications qui ne présentent pas d'écart dans le résultat final, leurs mesurages peuvent être vérifiés pour s'assurer qu'il n'y a pas de différence dans le détail et qu'il n'y a pas eu compensation au niveau des erreurs du détail ce qui ce reflète sur le résultat final. Pour cette raison 50 spécifications ont été vérifiées en détails. Aucune spécification qui a présenté un résultat identique entre la mesure manuelle et automatique n'a présenté de différence dans le détail du mesurage. Seulement pour une spécification (n°= 10), au niveau de la documentation manuelle du résumé du détail de la mesure, le mesureur 1 avait noté 4 PF au lieu de 5 PF, bien qu'il ait trouvé 5 PF dans le détail de sa mesure. Le tableau 8.7 présente les spécifications et le résultat de la comparaison détaillée.

Tableau 8.7 Les spécifications testées et qui ne présentait pas d'écart dans le résultat final

Numéro de la spécification testée	Nombre de processus Fonctionnels	Résultat de la comparaison détaillée
1	1	OK
2	10	OK
3	1	OK
4	2	OK
6	6	OK
7	2	OK
8	2	OK
9	0	OK
10	4 (erreur de documentation)	OK
11	17	OK
19	3	OK
20	7	OK
21	5	OK
22	1	OK
23	3	OK
24	1	OK
25	1	OK
26	1	OK
27	1	OK
29	2	OK
30	1	OK
31	4	OK
32	1	OK
33	1	OK
34	1	OK
35	4	OK
36	4	OK
37	4	OK
38	1	OK
39	1	OK
40	1	OK
41	1	OK
42	1	OK
43	1	OK
44	5	OK
45	5	OK

Tableau 8.7 (Suite) Les spécifications testées et qui ne présentait pas d'écart dans le résultat final

Numéro de la spécification testée	Nombre de processus Fonctionnels	Résultat de la comparaison détaillée
46	3	OK
47	10	OK
48	6	OK
49	1	OK
50	1	OK
51	2	OK
52	1	OK
55	1	OK
56	5	OK
57	5	OK
58	1	OK
59	3	OK
60	3	OK
65	7	OK

8.4 Conclusion générale sur les tests

Certaines différences provenaient des erreurs causées par l'application manuelle de la procédure de mesurage: celles-ci ne présentaient aucune difficulté pour les résoudre. D'autres écarts ont été causés par un ou plusieurs « bugs » du prototype 1: après la correction des bugs du code et la vérification du prototype 1, ces écarts ont été enlevés.

Toutefois, les écarts qui sont fortement liés à des limitations avec les librairies Simulink utilisées dans le parseur du prototype 1 sont les plus difficiles à gérer :

- les sorties 'multiples' qui proviennent d'un seul block (fourchettes créées par des Signal-line Simulink) et qui sont comptabilisées plusieurs fois,
- la différence entre les noms et les étiquettes des blocs 'Goto' et 'From'.

Les écarts liés à la manière de spécifier propre au secteur de spécification considéré ont été éliminés par des implémentations dans le prototype 1 (librairies et blocs RSA).

Certains écarts relatifs à la propreté des spécifications ne peuvent pas être traités automatiquement.

À noter que toute « adaptation » ou modification des règles de mesurage, même mineure, peut entrainer l'apparition de nouveaux écarts. D'autre part, une solution à un problème peut entrainer l'apparition d'autres problèmes toujours à cause des limitations techniques/manière de spécifier, propreté de la spécification en entrée, etc. Par exemple, l'adoption de la nouvelle règle sur les constantes.

L'écart, lorsqu'il existe, en taille fonctionnelle à la fin de ces phases de test entre les résultats de mesure obtenus par l'application manuelle de la procédure de mesurage et celui du prototype varie entre 1.7% et 12.6% pour des écarts causés par la limitation des librairies utilisées dans le prototype 1 (excluant l'écart de 44.2% causé par une spécification jugée non propre par les experts du métier, et ceci pour 7 spécifications en entrée parmi les 77 : en résumé, 9.% des spécifications en entrée présentent un écart entre les résultats de mesure obtenus par l'application manuelle de la procédure de mesurage et ceux obtenus par l'application automatisée de cette procédure. Les problèmes identifiés et qui causent les écarts seront normalement résolus directement dans l'outil industrialisé. Le graphe 3 illustre l'évolution de l'écart sur les différentes phases.

Finalement, après l'application du protocole sur 50 spécifications parmi les 70 spécifications qui ne présentent pas d'écart dans le résultat final en CFP (soit donc ~71% des ces spécifications): les résultats sont identiques quant aux PF et mouvements de groupes de données identifiés.

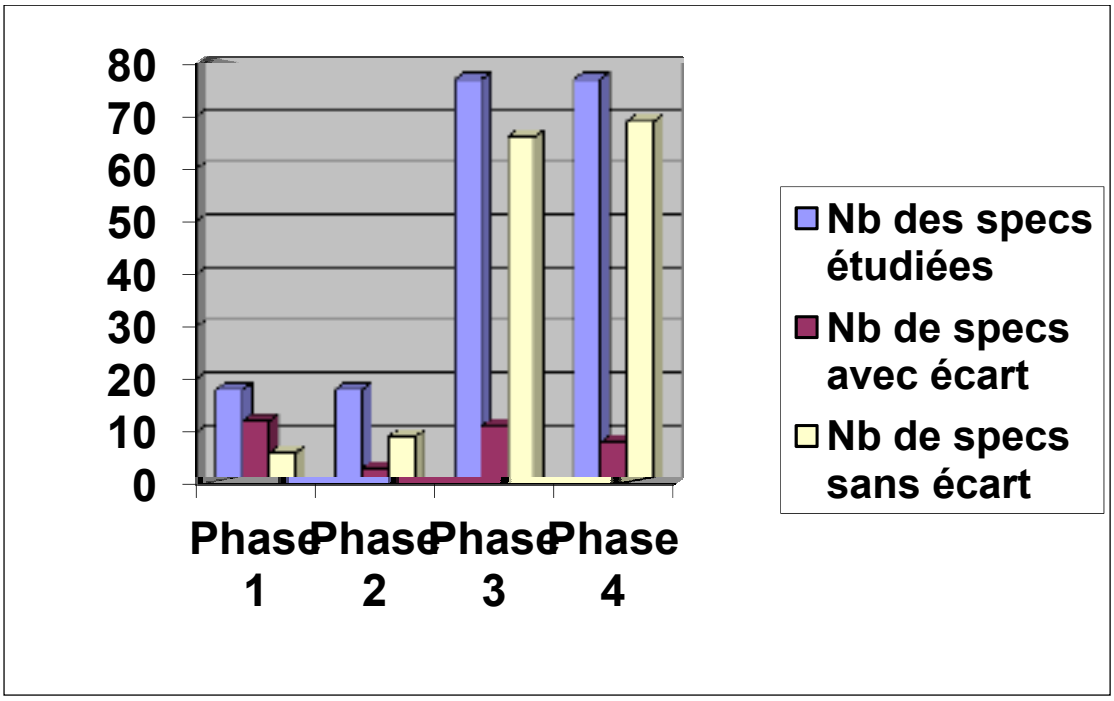


Figure 8.2 Les écarts sur les différentes dates de tests du Prototype 1

CONCLUSION

Le but principal de cette thèse est l'automatisation de la mesure de la taille fonctionnelle des logiciels des systèmes réactifs, temps-réel et embarqués en utilisant leurs spécifications fonctionnelles exprimées par les outils de modélisations temps-réel utilisés en industrie. Ce but de recherche a été traité en accomplissant deux objectifs de recherche qui sont:

1 Le design des procédures de mesurages basées sur la méthode COSMIC ISO 19761 pour chacun des outils de modélisations utilisés : Statemate et Simulink.

2 L'automatisation de ces procédures de mesurage et la réalisation d'un prototype de mesurage qui implémente ces procédures.

La méthodologie de recherche utilisée pour atteindre le but final de cette thèse et donc de chacun des objectifs a consisté en plusieurs phases.

La première phase consistait à analyser les documents et la littérature pour synthétiser les contenus des publications pour pouvoir trier et utiliser les travaux utiles pour ce projet de recherche. Dans cette phase aussi, et comme ce projet de recherche s'inscrit dans le cadre d'un projet industriel chez Renault SAS, les besoins spécifiques de Renault ont été analysés. Le premier et principal livrable produit à l'issue de cette phase 1 sont les objectifs de la recherche.

Les outils de modélisation et Simulink Statemate ont chacun leurs propres vocabulaires, éléments, architectures, et modèles. La deuxième phase était de comprendre les outils de modélisation utilisés chez Renault.

Il est essentiel de comprendre les modèles des outils de modélisation, afin de mapper leurs éléments avec les éléments de la méthode de mesure. Ce mapping permet la documentation des procédures de mesurage FSM. Ces procédures sont les règles de mesurage à appliquer

pour obtenir la taille fonctionnelle à partir de la spécification fonctionnelle des logiciels pour chacun des outils de modélisation. Chez Renault, deux outils de modélisation différents sont utilisés: Statemate et Simulink. Comme les deux outils sont utilisés pour la modélisation des exigences fonctionnelles du logiciel temps réel, leurs modèles reflètent les points de vue structure et comportement du logiciel à développer. Cependant, ils restent deux outils différents, chacun avec ses propres éléments distincts, vocabulaire, architecture du modèle, etc. Par exemple, certains des éléments Statemate sont: “control activity”, “external activity”, “activity”, “Statechart”, etc., tandis que les éléments trouvés dans Simulink sont les suivants: “subsystem”, “block”, “atomic block”, “line”, etc.

Avec Statemate, le comportement du logiciel se trouve au niveau de l'activité de contrôle des éléments du modèle. Une « activité de contrôle » contrôle la dynamique des activités et de leurs flux de données, incluant la capacité d'activer et de désactiver des activités, de causer des données à être écrite, modifiée, lue, d'envoyer des signaux, etc. En revanche, avec Simulink, le comportement du logiciel se trouve niveau du « Subsystem » du modèle Simulink.

Par conséquent, afin de bien mapper leurs éléments à la méthode de mesure COSMIC ISO 19761, il était important d'étudier et de comprendre chacun des outils de modélisation individuellement.

Selon le VIM, une méthode de mesure est une description générique d'une organisation logique des opérations utilisées dans une mesure. Quand il s'agit d'appliquer la méthode générique de mesure, tels que définie dans une norme, il faut créer des procédures de mesure spécifiques à chaque type de modèle d'entrée ces procédures définissent les étapes, les règles de dimensionnement, le niveau de précision et de vérification, etc. Si la même procédure de mesurage est appliquée sur les mêmes entrées, les mêmes résultats devraient être obtenus. Si aucune des procédures FSM spécifiques n’est définie et la méthode générique est utilisée seulement comme une référence, les résultats peuvent être différents.

Les procédures de mesurage sont définies en utilisant le vocabulaire spécifique et les éléments du modèle de l'outil de modélisation utilisé pour créer les spécifications fonctionnelles à mesurer. Il est important de comprendre pourquoi deux procédures différentes sont nécessaires: la raison se trouve dans les types d'entrées (c'est à dire les éléments et le vocabulaire de Simulink sont différents de ceux de Statemate), même si la méthode de mesure utilisée est la même, qui est COSMIC. Il est donc nécessaire de concevoir des procédures FSM spécifiques à chacun des outils de modélisation, ce qui a été réalisé dans les phases 3 et 4.

Chez Renault, une demande spécifique a été faite pour avoir des procédures de mesurage FSM pour couvrir à la fois les outils de modélisation Statemate et Simulink, utilisés dans leur contexte. Le but de procédures FSM est d'extraire les informations nécessaires à partir des modèles des outils de modélisation pour obtenir la taille fonctionnelle correcte du logiciel à-développer à partir de son modèle. Ces procédures contiennent les règles de mesurage à appliquer pour obtenir la taille fonctionnelle à partir des spécifications fonctionnelles des logiciels pour chacun des outils.

Ces procédures de mesurage sont utilisées et appliquées pour mesurer les différents logiciels des différentes livraisons de spécifications que Renault confie à ses fournisseurs. Les résultats ont été utilisés avec des données des efforts de développement pour produire des graphes et des modèles de simulation de la productivité. Ces modèles sont utilisés dans le processus d'estimation des coûts des développements, et entre donc dans la négociation entre Renault et ses fournisseurs.

La phase 5 constituait l'automatisation des procédures proposées.

La représentation semi-formelle « commune » est un des livrables importants de cette phase 5: elle constitue un modèle qui montre tous les flux de données du logiciel modélisé et à mesurer. Donc à l'aide de ce modèle de mesure, le mesureur identifie tous les mouvements

de données, étape nécessaire pour l'automatisation du mesurage de la taille fonctionnelle du logiciel en question.

Cette représentation semi-formelle est dite «commune » parce qu'elle peut être construite à partir de différents outils ou langages de modélisation utilisés pour la production des spécifications fonctionnelles des logiciels des systèmes temps-réel embarqués.

La priorité pour Renault était de d'automatiser le mesurage pour les spécifications Simulink. Donc, pour satisfaire à cette demande la stratégie des prototypes suivante a été adoptée :

Proto (n) = Proto (n-1) + X + Y
Où *Proto (n)* : la version *n* du prototype
Proto (n-1) : la version *n-1* du prototype
X : les éléments et les fonctionnalités ajoutés entre deux prototypes.
Y : les limitations éventuelles ou fonctionnalités qui ne peuvent par être développées pour des raisons diverses.

Le prototype 1 est celui pour le mesurage automatisé pour Simulink pour les nouveaux développements. Et une fois le prototype 1 testé, validé et déployé (phases 7 et 8), le prototype 2 devrait contenir les mêmes fonctionnalités du prototype 1 et en ajoutant, si possible, les fonctionnalités de mesurage pour les évolutions des spécifications Simulink (phase 9).

La spécification fonctionnelle et l'architecture du prototype ont été le deuxième livrable de cette phase, ainsi que le processus du choix de la technologie, des outils, environnement et autres détails qui sont utilisés dans la phase 6 qui ont joué un rôle important dans la réalisation du prototype 1.

La phase 6 a été la phase de développement du prototype 1 de mesurage des logiciels dont les spécifications fonctionnelles sont modélisées avec Simulink. En outre du prototype lui-

même, qui est développé en utilisant la technologie JAVA et l'environnement Eclipse, la documentation du prototype a été livrée à l'issue de cette phase.

Bien sûr, les outils d'automatisation FSM doivent être évalués avant leur utilisation afin de s'assurer que les résultats de mesure sont produites automatiquement sont utiles et conformes aux normes internationales. D'où, le livrable de la phase 7, un protocole d'évaluation des outils de mesurage automatique de la taille fonctionnelle suivant la méthode de mesure COSMIC ISO 19761 a été proposé et documenté pour évaluer la performance de ce type d'outil.

En utilisant le protocole proposé, le prototype 1 a été testé et évalué dans la phase 8. Un document de tests qui montre les étapes de développement du prototype 1 à partir des premières versions en divulguant les limitations et les fonctionnalités incluses et exclues dans chaque version a été livré.

Après avoir été validé, le prototype 1, a été déployé dans les différentes directions chez Renault. Une version industrielle a été réalisée. Une demande de brevet a été déposée.

Les nouveaux résultats de mesure obtenus (manuellement et surtout automatiquement) ont été utilisés pour des statistiques, pour la production des courbes de régression et donc pour produire des modèles de productivité. De nouvelles classes de modèles ont été proposées et/ou en cours de propositions avec la contribution active des responsables des métiers.

Donc avec ces résultats, le constructeur Renault SAS a eu des chiffres utiles pour ses processus d'estimations, d'évaluation et de négociation avec ses fournisseurs. Avant l'utilisation de ces données, le constructeur se basait uniquement sur les avis de ses experts.

Les seules limites identifiées à date du prototype 1 développé sont reliées à la librairie de l'interface de programmation JAVA pour Simulink. La version industrialisée corrige ses limitations. Bien entendu, les spécifications en entrée doivent être complètes et sans défauts.

Dans sa version actuelle, le prototype n'est pas capable de corriger « automatiquement » ces défauts. Toutefois, l'application du protocole dévaluation permet la détection de défauts ou d'imperfections dans les spécifications en entrée.

Un article intitulé « Automation of Functional Size Measurement of the Model-based Requirements of Real-time Embedded Software » a été soumis et présenté à l'IWSM/MetriKon/Mensura qui a eu lieu à Stuttgart, en Allemagne en 2010.
Un deuxième article intitulé « Design of a Functional Size Measurement Procedure for Real-Time Embedded Software Requirements Expressed using the Simulink Model» a été soumis et accepté à l'IWSM-Mensura qui aura lieu à Nara, au Japon en novembre 2011.

Un troisième article sur la procédure de mesurage des spécifications fonctionnelles modélisées avec l'outil Statemate est en attente d'approbation de soumission à une conférence.

Un abstract intitulé « Applying an assessment protocol to the COSMIC automation prototype tool at Renault SAS » a été soumis et accepté à l'International Conference on Software Metrics and Estimating, qui aura lieu à Londres, en Angleterre en octobre 2011.

Une demande de brevet intitulée « Procédé de détermination de la taille d'une partie d'un logiciel », FR1156318, au nom de Renault SAS a été déposée.

ANNEXE I

EXEMPLE DES OUTILS ET LANGAGES DE MODÉLISATION UTILISÉS EN INDUSTRIE

Plusieurs langages et outils de modélisations pour les logiciels temps réels existent et sont utilisés dans l'industrie. Pour citer quelques exemples :

1. Statemate : Rational ® Statemate par IBM ® est un outil graphique de conception, de simulation et de prototypage pour le développement rapide des systèmes embarqués complexes. La solution IBM Rational Statemate offre la possibilité de spécifier les systèmes sans ambiguïté, de trouver des erreurs au début du cycle de vie de la conception avant la phase de tests pour l'intégration. Statemate permet :

    - de créer des modèles graphiques des systèmes, basés sur des diagrammes d'ingénierie standards.
    - il montre les états actifs, les fonctions, et les scénarios pendant les simulations. d'offrir des maquettes pour la simulation des entrées et des sorties.
    - de simuler un modèle incomplet, ce qui permet la construction et la simulation des modèles dans une démarche itérative assez libre.
    - d'offrir tous les moyens de débogage visuels traditionnels.
    - d'analyser les spécifications afin de s'assurer que leur comportement est correct et il permet la capture des données des tests.

2. Simulink : Simulink® est un environnement de conception « Model-based design » et de simulation multi-domaines pour les systèmes réactifs, dynamique temps-réel et embarqués. Il fournit un environnement graphique interactif et ainsi qu'un ensemble de bibliothèques de « blocs » qui permettent à l'utilisateur de concevoir, de simuler, de mettre en œuvre, et de tester une variété de systèmes temps-réel : système de télécommunication, de contrôle, de traitement de signal, de traitement de vidéo, et de traitement d'image. Simulink est intégré avec MATLAB ®, offrant un accès immédiat

à une vaste gamme d'outils qui permettent de développer des algorithmes, d'analyser et de visualiser des simulations, de créer des scripts de traitement, de personnaliser l'environnement de modélisation, de définir des signaux, des paramètres et des données de tests. Les principales caractéristiques offertes avec Simulink sont:

- bibliothèques de nombreux blocs prédéfinis et extensibles.
- éditeur graphique interactif pour le montage et la gestion des schémas des blocs.
- Capacité de gérer des modèles complexes en les divisant dans des composants de conception hiérarchiques.
- explorateur de modèles qui permet de naviguer, de créer, de configurer et de rechercher les signaux, les paramètres, les propriétés, le code généré et associé au modèle.
- interfaces de programmation d'applications (API) qui permettent la l'interaction avec d'autres programmes de simulation.
- les blocs de fonctions MATLAB™ qui permettent l'utilisation des algorithmes MATLAB dans Simulink et de les mettre en œuvre de systèmes embarqués.
- plusieurs modes de simulation (Normal, Accelerator, et Rapid Accelerator) pour exécuter des simulations à des vitesses différentes.
- un débogueur et un « profileur » pour examiner les résultats des simulations et de diagnostiquer les performances ainsi que des comportements inattendus dans la conception.
- accès complet à MATLAB pour l'analyse et la visualisation des résultats, pour la personnalisation de l'environnement de la modélisation et pour la définition des signaux, paramètres et données de tests.
- outils d'analyse et de diagnostic des modèles pour s'assurer de la cohérence de modèle et d'identifier des erreurs éventuelles de modélisation.

3. RRRT (Rational Rose Real-Time) : Rational ® Rose ® Technical Developer par IBM ® est une solution solide de développement par modèle (Model-Driven Development (MDD)) expressément créée pour répondre aux défis du développement de systèmes complexes. Basé sur le standard Unified Modeling Language™ (UML ™), Rational Rose Technical Developer permet l'automatisation la plus robuste du développement par modèle, y compris des exécutions à partir des modèles et la génération de code exécutable complet.

4. SCADE : SCADE (Safety Critical Application Development Environment) est un environnement de développement graphique commercialisé par Esterel Technologies. L'outil SCADE a été conçu pour aider et assister au développement des systèmes embarqués critiques. Cet environnement est composé de plusieurs outils : un éditeur graphique, un simulateur, un contrôleur de modèle et un générateur automatique de code qui traduit spécifications graphiques en code C. Le langage SCADE est un langage de spécification graphique des flux de données basé sur Lustre (langage formel). Donc SCADE est basé sur le formalisme. Il est déterministe et fournit des solutions efficaces pour le développement des systèmes réactifs.

5. SysML : c'est un langage de modélisation d'applications des systèmes d'ingénierie, dont le nom exact est « OMG Systems Modeling Language (OMG SysML ™) ». SysML permet la spécification, l'analyse, la conception la vérification et la validation d'une vaste gamme de systèmes complexes: matériels, logiciels, systèmes d'information, processus... SysML réutilise un sous-ensemble d'UML 2 en ayant des extensions. SysML est adapté aux domaines des applications spécifiques, tels que les domaines de l'automobile, de l'aérospatiale, des télécommunications et des systèmes d'information.

ANNEXE II

APERÇU DE LA MÉTHODE COSMIC

La méthode COSMIC est une méthode normalisée de Mesure de la 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 ainsi que des hybrides de ces logiciels.

La méthode COSMIC a été acceptée par ISO en décembre 2002 en tant que norme internationale : 'COSMIC-FFP:2003 – A functional size measurement method'. La version la plus récente de la norme date de 2011: ISO/IEC 19761:2011 , « Software engineering -- COSMIC: a functional size measurement method ».

La norme ISO/IEC 19761 contient les définitions et les règles normatives fondamentales de la méthode.

La méthode de mesure COSMIC a été conçue pour être appliquée aux fonctionnalités des logiciels des domaines suivants : Logiciels d'application d'affaires, Logiciels de type temps réel et les logiciels hybrides (d'affaire & temps réel).

La méthode de mesure COSMIC n'a pas été conçue pour le moment, pour prendre en compte la taille fonctionnelle des logiciels ou morceaux de logiciels qui sont caractérisés par des algorithmes mathématiques complexes ou par d'autres règles spécialisées ou complexes.

La méthode de mesure 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 du logiciel.

La méthode de mesure COSMIC a été conçue pour être indépendante des décisions de réalisation et de mise en œuvre telles qu'implémentées dans les artéfacts opérationnels du logiciel à mesurer. La fonctionnalité est concernée par 'le traitement de l'information que le logiciel doit exécuter pour ses utilisateurs'.

Plus précisément, une déclaration de FUR décrit la fonctionnalité que le logiciel doit faire pour les utilisateurs fonctionnels, qui sont les émetteurs et les destinataires prévus des données qui vont et viennent du logiciel.

- ***Le modèle contextuel de logiciel COSMIC***

Un morceau de logiciel à mesurer, doit être défini avec soin (dans le périmètre de la mesure) et cette définition doit tenir compte du contexte de tous les logiciels et matériels avec lesquels il interagit. Ce modèle contextuel de logiciel introduit les principes et concepts nécessaires à cette définition :

i. Le logiciel est limité par le matériel ;
ii. Le logiciel est typiquement structuré en couches ;
iii. Une couche peut contenir un ou plusieurs morceaux de logiciel, de même niveau mais distincts, et tout morceau de logiciel peut aussi être construit de composants de même niveau mais distincts ;
iv. Tout morceau de logiciel à mesurer doit être défini par un périmètre de mesure, laquelle doit aussi être entièrement comprise à l'intérieur d'une seule couche de logiciel ;
v. La portée du morceau de logiciel à mesurer doit dépendre de la raison d'être de la mesure ;
vi. 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 ;

vii. Un morceau de logiciel interagit avec ses utilisateurs fonctionnels via les mouvements de données à travers la frontière. Ce morceau de logiciel peut déplacer les données vers et à partir d'un stockage persistant situé à l'intérieur de la frontière du logiciel ;
viii. La FUR d'un logiciel peut être exprimée à différents niveaux de granularité ;
ix. Le niveau de granularité auquel les mesures doivent normalement être effectuées, est celui du processus fonctionnel;
x. S'il n'est pas possible de mesurer à partir du niveau de granularité du processus fonctionnel, alors la FUR du logiciel doit être mesurée d'une manière approximative, et échelonnée selon le niveau de granularité des processus fonctionnels.

- ***Le modèle générique de logiciel COSMIC***

Après avoir interprété la FUR du logiciel à mesurer en fonction du modèle contextuel de logiciel, nous pouvons maintenant appliquer le modèle générique de logiciel à la FUR pour identifier les composants 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 :

i. Le logiciel reçoit des données en entrée de ses utilisateurs fonctionnels. Le logiciel produit aussi en retour d'autres types de résultats et/ou des données en sorties, à ses utilisateurs fonctionnels ;
ii. Chaque Fonctionnalité Utilisateur Requise (FUR) d'un morceau de logiciel à mesurer peut être arrimé à un seul processus fonctionnel ;
iii. Chaque processus fonctionnel est composé de plusieurs sous-processus ;
iv. Un sous-processus peut être un mouvement de données ou une manipulation de données ;
v. Chaque processus fonctionnel est déclenché par un mouvement de données en Entrée
vi. envoyé par un utilisateur fonctionnel, qui à son tour informe le processus fonctionnel qu'il a identifié un événement ;
vii. Un mouvement de données déplace qu'un seul groupe de données ;

viii. Un groupe de données est constitué d'un un seul ensemble d'attributs de données, qui lui, décrit un seul objet d'intérêt ;

ix. Il y a quatre types de mouvement de données. Une Entrée déplace un groupe de données vers le logiciel à partir d'un utilisateur fonctionnel. Une Sortie déplace un groupe de donnée du logiciel vers un utilisateur fonctionnel. Une Écriture déplace un groupe de données à partir du logiciel vers un stockage persistant. Une Lecture déplace un groupe de données à partir d'un stockage persistant vers le logiciel ;

x. Un processus fonctionnel doit inclure au minimum un mouvement de données de type Entrée et un autre mouvement de données, soit de type Écriture, soit de Sortie. Un processus fonctionnel doit donc comprendre au minimum deux mouvements de données ;

xi. Pour réaliser une estimation à des fins de mesure, les sous processus de manipulation de données ne sont pas mesurés séparément. Il faut tenir compte de la fonctionnalité des manipulations de données pour chaque mouvement de données associé.

- ***Le processus de mesure COSMIC***

Le processus général de mesure COSMIC comprend trois phases:

i. La phase de la stratégie de mesure (phase 1) : Dans laquelle le modèle contextuel de logiciel est appliqué au logiciel devant être mesuré (Chapitre 2) ;

ii. La phase d'arrimage (phase 2) : Dans laquelle le modèle générique de logiciel est appliqué au logiciel devant être mesuré;

iii. La phase de mesure (phase 3) : Dans laquelle les mesures de taille sont obtenues.

Les résultats obtenus après avoir appliqué le processus de mesure sur un morceau d'un logiciel est une mesure de taille fonctionnelle du logiciel, exprimée en Points de Fonction COSMIC (CFP').

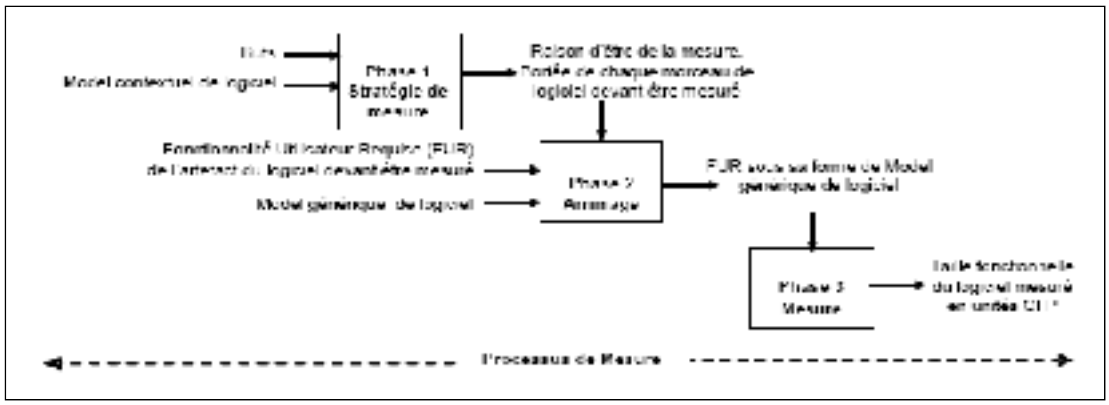


Figure A II-1 Structure de la Méthode COSMIC

ANNEXE III

DÉTAIL DU MESURAGE DU PROTOTYPE 1

Le détail du mesurage de la spécification fonctionnelle modélisée avec l'outil Simulink (fichier mdl) du prototype 1 faite par le prototype 1 lui-même.

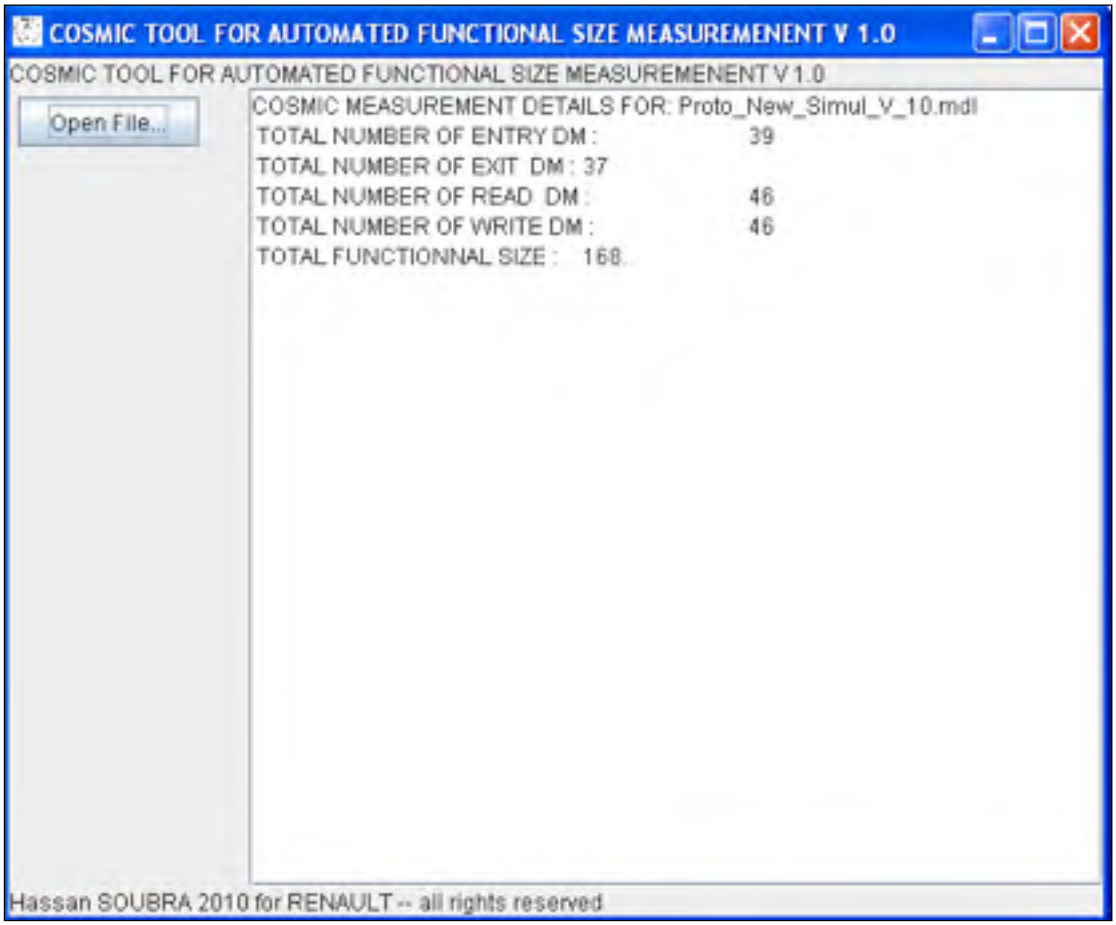


Figure A III-1 Résultat de la mesure en automatique de la spécification

Le détail de la mesure comme extrait du fichier « CFP Report » créé par le prototype :

##FP REPORT
##

NAME OF FUNCTIONAL PROCESS 1 IS: OpenSpec

NAME OF FUNCTIONAL PROCESS 2 IS: SeekPossibleFP

NAME OF FUNCTIONAL PROCESS 3 IS: CompletePossibleFP

NAME OF FUNCTIONAL PROCESS 4 IS: Special_Treatment

NAME OF FUNCTIONAL PROCESS 5 IS: ReorganizeBlocks

NAME OF FUNCTIONAL PROCESS 6 IS: GetArrows

NAME OF FUNCTIONAL PROCESS 7 IS: AlreadyCheckedBlocks

NAME OF FUNCTIONAL PROCESS 8 IS: RIP_FP

NAME OF FUNCTIONAL PROCESS 9 IS: ReturnOnlyFPs

NAME OF FUNCTIONAL PROCESS 10 IS: IsElementaryBlock

NAME OF FUNCTIONAL PROCESS 11 IS: GotoFromTreat

NAME OF FUNCTIONAL PROCESS 12 IS: GenerateSemiFormalModel

NAME OF FUNCTIONAL PROCESS 13 IS: IsConnectedFrom

NAME OF FUNCTIONAL PROCESS 14 IS: AlreadyChecked

NAME OF FUNCTIONAL PROCESS 15 IS: GetAllDataMovements

NAME OF FUNCTIONAL PROCESS 16 IS: GUI

NAME OF FUNCTIONAL PROCESS 17 IS: actionPerformed

NAME OF FUNCTIONAL PROCESS 18 IS: createAndShowGUI

NAME OF FUNCTIONAL PROCESS 19 IS: BorderLayoutGUI

##FP REPORT DONE!! ##

##DATA MOVEMENT DETAILS FOR FP: OpenSpec ###################

NUMBER:1 TYPE:E

DATA MOV: Trigger -->

OpenSpec

NUMBER:2 TYPE:X

DATA MOV: OpenSpec -->

SeekPossibleFP

NUMBER:3 TYPE:E

DATA MOV: External(New_Simulink_Model_Name) --> OpenSpec

##DATA MOVEMENT DETAILS FOR FP: SeekPossibleFP ####################

NUMBER:4 TYPE:W

DATA MOV: SeekPossibleFP --> Persistant_Strorage(Data_Store_Write4)

NUMBER:5 TYPE:W

DATA MOV: SeekPossibleFP --> Persistant_Strorage(Data_Store_Write5)

NUMBER:6 TYPE:W

DATA MOV: SeekPossibleFP --> Persistant_Strorage(Data_Store_Write6)

NUMBER:7 TYPE:W

DATA MOV: SeekPossibleFP --> Persistant_Strorage(Data_Store_Write1)

NUMBER:8 TYPE:W
DATA MOV: SeekPossibleFP -->
Persistant_Strorage(Data_Store_Write2)

NUMBER:9 TYPE:X
DATA MOV: SeekPossibleFP -->
CompletePossibleFP

NUMBER:10 TYPE:R
DATA MOV: Persistant_Strorage(Data_Store_Read) -->
SeekPossibleFP

NUMBER:11 TYPE:R
DATA MOV: Persistant_Strorage(Data_Store_Read4) -->
SeekPossibleFP

NUMBER:12 TYPE:E
DATA MOV: External(1) -->
SeekPossibleFP

NUMBER:13 TYPE:R
DATA MOV: Persistant_Strorage(Data_Store_Read2) -->
SeekPossibleFP

NUMBER:14 TYPE:R
DATA MOV: Persistant_Strorage(Data_Store_Read1) -->
SeekPossibleFP

NUMBER:15 TYPE:R

DATA MOV: Persistant_Strorage(Data_Store_Read3) -->
SeekPossibleFP

##DATA MOVEMENT DETAILS FOR FP: CompletePossibleFP ####################

NUMBER:16 TYPE:X

DATA MOV: CompletePossibleFP -->
External

NUMBER:17 TYPE:W

DATA MOV: CompletePossibleFP -->
Persistant_Strorage(Data_Store_Write1)

NUMBER:18 TYPE:W

DATA MOV: CompletePossibleFP -->
Persistant_Strorage(Data_Store_Write2)

NUMBER:19 TYPE:W

DATA MOV: CompletePossibleFP -->
Persistant_Strorage(Data_Store_Write3)

NUMBER:20 TYPE:X

DATA MOV: CompletePossibleFP -->
Special_Treatment

NUMBER:21 TYPE:X
DATA MOV: CompletePossibleFP -->
Special_Treatment

NUMBER:22 TYPE:W
DATA MOV: CompletePossibleFP -->
Persistant_Strorage(DataStoreWrite5)

NUMBER:23 TYPE:X
DATA MOV: CompletePossibleFP -->
ReorganizeBlocks

NUMBER:24 TYPE:X
DATA MOV: CompletePossibleFP -->
ReorganizeBlocks

NUMBER:25 TYPE:E
DATA MOV: ReorganizeBlocks -->
CompletePossibleFP

NUMBER:26 TYPE:E
DATA MOV: External(5) -->
CompletePossibleFP

NUMBER:27 TYPE:R
DATA MOV: Persistant_Strorage(Data_Store_Read1) -->
CompletePossibleFP

NUMBER:28 TYPE:R

DATA MOV: Persistant_Strorage(Data_Store_Read3) -->

CompletePossibleFP

NUMBER:29 TYPE:R

DATA MOV: Persistant_Strorage(Data_Store_Read) -->

CompletePossibleFP

NUMBER:30 TYPE:E

DATA MOV: External(FP_List) -->

CompletePossibleFP

NUMBER:31 TYPE:E

DATA MOV: External(Model) -->

CompletePossibleFP

NUMBER:32 TYPE:R

DATA MOV: Persistant_Strorage(Data_Store_Read2) -->

CompletePossibleFP

##DATA MOVEMENT DETAILS FOR FP: Special_Treatment ####################

NUMBER:33 TYPE:X

DATA MOV: Special_Treatment -->

External

NUMBER:34 TYPE:W
DATA MOV: Special_Treatment -->
Persistant_Strorage(Data_Store_Write)

NUMBER:35 TYPE:W
DATA MOV: Special_Treatment -->
Persistant_Strorage(Data_Store_Write1)

NUMBER:36 TYPE:W
DATA MOV: Special_Treatment -->
Persistant_Strorage(Data_Store_Write2)

NUMBER:37 TYPE:R
DATA MOV: Persistant_Strorage(Data_Store_Read2) -->
Special_Treatment

NUMBER:38 TYPE:E
DATA MOV: External(FP) -->
Special_Treatment

NUMBER:39 TYPE:R
DATA MOV: Persistant_Strorage(Data_Store_Read) -->
Special_Treatment

NUMBER:40 TYPE:E
DATA MOV: External(Block) -->
Special_Treatment

NUMBER:41 TYPE:R

DATA MOV: Persistant_Strorage(Data_Store_Read1) -->
Special_Treatment

##DATA MOVEMENT DETAILS FOR FP: ReorganizeBlocks ####################

NUMBER:42 TYPE:X

DATA MOV: ReorganizeBlocks -->
GetArrows

NUMBER:43 TYPE:W

DATA MOV: ReorganizeBlocks -->
Persistant_Strorage(Data_Store_Write1)

NUMBER:44 TYPE:W

DATA MOV: ReorganizeBlocks -->
Persistant_Strorage(Data_Store_Write2)

NUMBER:45 TYPE:W

DATA MOV: ReorganizeBlocks -->
Persistant_Strorage(Data_Store_Write3)

NUMBER:46 TYPE:X

DATA MOV: ReorganizeBlocks -->
AlreadyCheckedBlocks

NUMBER:47 TYPE:X
DATA MOV: ReorganizeBlocks -->
External

NUMBER:48 TYPE:E
DATA MOV: GetArrows -->
ReorganizeBlocks

NUMBER:49 TYPE:E
DATA MOV: External(Block_In) -->
ReorganizeBlocks

NUMBER:50 TYPE:E
DATA MOV: AlreadyCheckedBlocks -->
ReorganizeBlocks

NUMBER:51 TYPE:R
DATA MOV: Persistant_Strorage(Data_Store_Read) -->
ReorganizeBlocks

NUMBER:52 TYPE:E
DATA MOV: External(8) -->
ReorganizeBlocks

NUMBER:53 TYPE:R
DATA MOV: Persistant_Strorage(Data_Store_Read2) -->
ReorganizeBlocks

NUMBER:54 TYPE:R

DATA MOV: Persistant_Strorage(Data_Store_Read1) -->

ReorganizeBlocks

##DATA MOVEMENT DETAILS FOR FP: GetArrows ####################

NUMBER:55 TYPE:W

DATA MOV: GetArrows -->

Persistant_Strorage(Data_store_Write_0)

NUMBER:56 TYPE:W

DATA MOV: GetArrows -->

Persistant_Strorage(Data_store_Write_1)

NUMBER:57 TYPE:W

DATA MOV: GetArrows -->

Persistant_Strorage(Data_store_Write_2)

NUMBER:58 TYPE:X

DATA MOV: GetArrows -->

External

NUMBER:59 TYPE:W

DATA MOV: GetArrows -->

Persistant_Strorage(Data_store_Write_3)

NUMBER:60 TYPE:R
DATA MOV: Persistant_Strorage(DataStore_Read2) -->
GetArrows

NUMBER:61 TYPE:R
DATA MOV: Persistant_Strorage(DataStore_Read3) -->
GetArrows

NUMBER:62 TYPE:R
DATA MOV: Persistant_Strorage(DataStore_Read) -->
GetArrows

NUMBER:63 TYPE:E
DATA MOV: External(BlockToTreat) -->
GetArrows

NUMBER:64 TYPE:R
DATA MOV: Persistant_Strorage(DataStore_Read1) -->
GetArrows

##DATA MOVEMENT DETAILS FOR FP: AlreadyCheckedBlocks ##################

NUMBER:65 TYPE:R

DATA MOV: Persistant_Strorage(Data_Store_Read) --> AlreadyCheckedBlocks

NUMBER:66 TYPE:E

DATA MOV: External(Block_List) --> AlreadyCheckedBlocks

NUMBER:67 TYPE:X

DATA MOV: AlreadyCheckedBlocks --> External

NUMBER:68 TYPE:W

DATA MOV: AlreadyCheckedBlocks --> Persistant_Strorage(Data_Store_Write)

NUMBER:69 TYPE:E

DATA MOV: External(Block_Name) --> AlreadyCheckedBlocks

##DATA MOVEMENT DETAILS FOR FP: RIP_FP ##################

NUMBER:70 TYPE:E

DATA MOV: IsElementaryBlock --> RIP_FP

NUMBER:71 TYPE:W
DATA MOV: RIP_FP -->
Persistant_Strorage(Data_Store_Write1)

NUMBER:72 TYPE:W
DATA MOV: RIP_FP -->
Persistant_Strorage(Data_Store_Write2)

NUMBER:73 TYPE:W
DATA MOV: RIP_FP -->
Persistant_Strorage(Data_Store_Write3)

NUMBER:74 TYPE:X
DATA MOV: RIP_FP -->
ReturnOnlyFPs

NUMBER:75 TYPE:R
DATA MOV: Persistant_Strorage(Data_Store_Read1) -->
RIP_FP

NUMBER:76 TYPE:E
DATA MOV: External(FPList) -->
RIP_FP

NUMBER:77 TYPE:R
DATA MOV: Persistant_Strorage(Data_Store_Read4) -->
RIP_FP

NUMBER:78 TYPE:R
DATA MOV: Persistant_Strorage(Data_Store_Read2) -->
RIP_FP

NUMBER:79 TYPE:X
DATA MOV: RIP_FP -->
IsElementaryBlock

NUMBER:80 TYPE:X
DATA MOV: RIP_FP -->
GotoFromTreat

##DATA MOVEMENT DETAILS FOR FP: ReturnOnlyFPs ##################

NUMBER:81 TYPE:W
DATA MOV: ReturnOnlyFPs -->
Persistant_Strorage(Data_Store_Write_1)

NUMBER:82 TYPE:W
DATA MOV: ReturnOnlyFPs -->
Persistant_Strorage(Data_Store_Write_2)

NUMBER:83 TYPE:W
DATA MOV: ReturnOnlyFPs -->
Persistant_Strorage(Data_Store_Write_3)

NUMBER:84 TYPE:R

DATA MOV: Persistant_Strorage(Data_Store_Read_1) -->

ReturnOnlyFPs

NUMBER:85 TYPE:E

DATA MOV: External(FP_List_unfiltered) -->

ReturnOnlyFPs

NUMBER:86 TYPE:R

DATA MOV: Persistant_Strorage(Data_Store_Read_4) -->

ReturnOnlyFPs

NUMBER:87 TYPE:R

DATA MOV: Persistant_Strorage(Data_Store_Read_2) -->

ReturnOnlyFPs

NUMBER:88 TYPE:X

DATA MOV: ReturnOnlyFPs -->

External

NUMBER:89 TYPE:X

DATA MOV: ReturnOnlyFPs -->

External

##DATA MOVEMENT DETAILS FOR FP: IsElementaryBlock ##################

NUMBER:90 TYPE:X
DATA MOV: IsElementaryBlock -->
External

NUMBER:91 TYPE:E
DATA MOV: External(Block_To_Test) -->
IsElementaryBlock

NUMBER:92 TYPE:E
DATA MOV: External(GoToFrom_Source) -->
IsElementaryBlock

##DATA MOVEMENT DETAILS FOR FP: GotoFromTreat ##################

NUMBER:93 TYPE:W
DATA MOV: GotoFromTreat -->
Persistant_Strorage(Data_Store_Write4)

NUMBER:94 TYPE:X
DATA MOV: GotoFromTreat -->
External

NUMBER:95 TYPE:W
DATA MOV: GotoFromTreat -->
Persistant_Strorage(Data_Store_Write1)

NUMBER:96 TYPE:W
DATA MOV: GotoFromTreat -->
Persistant_Strorage(Data_Store_Write2)

NUMBER:97 TYPE:W
DATA MOV: GotoFromTreat -->
Persistant_Strorage(Data_Store_Write3)

NUMBER:98 TYPE:X
DATA MOV: GotoFromTreat -->
External

NUMBER:99 TYPE:R
DATA MOV: Persistant_Strorage(Data_Store_Read3) -->
GotoFromTreat

NUMBER:100 TYPE:R
DATA MOV: Persistant_Strorage(Data_Store_Read1) -->
GotoFromTreat

NUMBER:101 TYPE:E
DATA MOV: External(Tested_GotoFrom) -->
GotoFromTreat

NUMBER:102 TYPE:E
DATA MOV: External(Real_FP_List) -->
GotoFromTreat

NUMBER:103 TYPE:R
DATA MOV: Persistant_Strorage(Data_Store_Read4) -->
GotoFromTreat

NUMBER:104 TYPE:R
DATA MOV: Persistant_Strorage(Data_Store_Read2) -->
GotoFromTreat

##DATA MOVEMENT DETAILS FOR FP: GenerateSemiFormalModel ##################

NUMBER:105 TYPE:W
DATA MOV: GenerateSemiFormalModel -->
Persistant_Strorage(Data_Store_Write4)

NUMBER:106 TYPE:W
DATA MOV: GenerateSemiFormalModel -->
Persistant_Strorage(Data_Store_Write5)

NUMBER:107 TYPE:X
DATA MOV: GenerateSemiFormalModel -->
IsConnectedFrom

NUMBER:108 TYPE:X

DATA MOV: GenerateSemiFormalModel --> IsConnectedFrom

NUMBER:109 TYPE:W

DATA MOV: GenerateSemiFormalModel --> Persistant_Strorage(Data_Store_Write6)

NUMBER:110 TYPE:W

DATA MOV: GenerateSemiFormalModel --> Persistant_Strorage(Data_Store_Write7)

NUMBER:111 TYPE:E

DATA MOV: External(Functional_Process_list) --> GenerateSemiFormalModel

NUMBER:112 TYPE:W

DATA MOV: GenerateSemiFormalModel --> Persistant_Strorage(Data_Store_Write8)

NUMBER:113 TYPE:W

DATA MOV: GenerateSemiFormalModel --> Persistant_Strorage(Data_Store_Write9)

NUMBER:114 TYPE:W

DATA MOV: GenerateSemiFormalModel --> Persistant_Strorage(Data_Store_Write1)

NUMBER:115 TYPE:W

DATA MOV: GenerateSemiFormalModel --> Persistant_Strorage(Data_Store_Write3)

NUMBER:116 TYPE:R

DATA MOV: Persistant_Strorage(Data_Store_Read3) --> GenerateSemiFormalModel

NUMBER:117 TYPE:R

DATA MOV: Persistant_Strorage(Data_Store_Read8) --> GenerateSemiFormalModel

NUMBER:118 TYPE:R

DATA MOV: Persistant_Strorage(Data_Store_Read2) --> GenerateSemiFormalModel

NUMBER:119 TYPE:R

DATA MOV: Persistant_Strorage(Data_Store_Read) --> GenerateSemiFormalModel

NUMBER:120 TYPE:R

DATA MOV: Persistant_Strorage(Data_Store_Read5) --> GenerateSemiFormalModel

NUMBER:121 TYPE:R

DATA MOV: Persistant_Strorage(Data_Store_Read6) --> GenerateSemiFormalModel

NUMBER:122 TYPE:E
DATA MOV: External(6) -->
GenerateSemiFormalModel

NUMBER:123 TYPE:E
DATA MOV: External(7) -->
GenerateSemiFormalModel

NUMBER:124 TYPE:R
DATA MOV: Persistant_Strorage(Data_Store_Read1) -->
GenerateSemiFormalModel

NUMBER:125 TYPE:R
DATA MOV: Persistant_Strorage(Data_Store_Read7) -->
GenerateSemiFormalModel

NUMBER:126 TYPE:X
DATA MOV: GenerateSemiFormalModel -->
External

NUMBER:127 TYPE:X
DATA MOV: GenerateSemiFormalModel -->
External

NUMBER:128 TYPE:X
DATA MOV: GenerateSemiFormalModel -->
AlreadyChecked

NUMBER:129 TYPE:X

DATA MOV: GenerateSemiFormalModel -->

AlreadyChecked

##DATA MOVEMENT DETAILS FOR FP: IsConnectedFrom ####################

NUMBER:130 TYPE:R

DATA MOV: Persistant_Strorage(Data_Store_Read) -->

IsConnectedFrom

NUMBER:131 TYPE:E

DATA MOV: External(FP_) -->

IsConnectedFrom

NUMBER:132 TYPE:E

DATA MOV: External(FromSource_Name) -->

IsConnectedFrom

NUMBER:133 TYPE:X

DATA MOV: IsConnectedFrom -->

External

NUMBER:134 TYPE:W

DATA MOV: IsConnectedFrom -->

Persistant_Strorage(Data_Store_Write)

##DATA MOVEMENT DETAILS FOR FP: AlreadyChecked ##################

NUMBER:135 TYPE:R

DATA MOV: Persistant_Strorage(Data_Store_Read) --> AlreadyChecked

NUMBER:136 TYPE:E

DATA MOV: External(DMList) --> AlreadyChecked

NUMBER:137 TYPE:X

DATA MOV: AlreadyChecked --> External

NUMBER:138 TYPE:W

DATA MOV: AlreadyChecked --> Persistant_Strorage(Data_Store_Write)

NUMBER:139 TYPE:E

DATA MOV: External(DM_Name) --> AlreadyChecked

##DATA MOVEMENT DETAILS FOR FP: GetAllDataMovements ###################

NUMBER:140 TYPE:W
DATA MOV: GetAllDataMovements --> Persistant_Strorage(Data_Store_Write4)

NUMBER:141 TYPE:W
DATA MOV: GetAllDataMovements --> Persistant_Strorage(Data_Store_Write5)

NUMBER:142 TYPE:W
DATA MOV: GetAllDataMovements --> Persistant_Strorage(Data_Store_Write6)

NUMBER:143 TYPE:X
DATA MOV: GetAllDataMovements --> External

NUMBER:144 TYPE:X
DATA MOV: GetAllDataMovements --> External

NUMBER:145 TYPE:W
DATA MOV: GetAllDataMovements --> Persistant_Strorage(Data_Store_Write1)

NUMBER:146 TYPE:W
DATA MOV: GetAllDataMovements -->
Persistant_Strorage(Data_Store_Write2)

NUMBER:147 TYPE:W
DATA MOV: GetAllDataMovements -->
Persistant_Strorage(Data_Store_Write3)

NUMBER:148 TYPE:R
DATA MOV: Persistant_Strorage(Data_Store_Read1) -->
GetAllDataMovements

NUMBER:149 TYPE:R
DATA MOV: Persistant_Strorage(Data_Store_Read5) -->
GetAllDataMovements

NUMBER:150 TYPE:R
DATA MOV: Persistant_Strorage(Data_Store_Read3) -->
GetAllDataMovements

NUMBER:151 TYPE:R
DATA MOV: Persistant_Strorage(Data_Store_Read6) -->
GetAllDataMovements

NUMBER:152 TYPE:E
DATA MOV: External(Semi_Formal_Model) -->
GetAllDataMovements

NUMBER:153 TYPE:R
DATA MOV: Persistant_Strorage(Data_Store_Read4) -->
GetAllDataMovements

NUMBER:154 TYPE:R
DATA MOV: Persistant_Strorage(Data_Store_Read2) -->
GetAllDataMovements

##DATA MOVEMENT DETAILS FOR FP: GUI ##################

NUMBER:155 TYPE:X
DATA MOV: GUI -->
actionPerformed

NUMBER:156 TYPE:X
DATA MOV: GUI -->
actionPerformed

NUMBER:157 TYPE:X
DATA MOV: GUI -->
createAndShowGUI

NUMBER:158 TYPE:E
DATA MOV: External(UserCommands) -->
GUI

##DATA MOVEMENT DETAILS FOR FP: actionPerformed ##################

NUMBER:159 TYPE:X
DATA MOV: actionPerformed -->
External

NUMBER:160 TYPE:E
DATA MOV: External(WaitFor_Action) -->
actionPerformed

NUMBER:161 TYPE:E
DATA MOV: External(mdl_from1) -->
actionPerformed

NUMBER:162 TYPE:E
DATA MOV: External(Start_Measurement_By_User) -->
actionPerformed

##DATA MOVEMENT DETAILS FOR FP: createAndShowGUI ##################

NUMBER:163 TYPE:E
DATA MOV: Trigger -->
createAndShowGUI

NUMBER:164 TYPE:X
DATA MOV: createAndShowGUI -->
External

NUMBER:165 TYPE:E
DATA MOV: External(ConfigGUI) -->
createAndShowGUI

##DATA MOVEMENT DETAILS FOR FP: BorderLayoutGUI ##################

NUMBER:166 TYPE:E
DATA MOV: Trigger -->
BorderLayoutGUI

NUMBER:167 TYPE:X
DATA MOV: BorderLayoutGUI -->
External

NUMBER:168 TYPE:E

DATA MOV: External(Config_LayOut) --> BorderLayoutGUI

TOTAL FUNCTIONNAL SIZE: 168 CFP { E:39 X:37 R:46 W:46 }

LISTE DE RÉFÉRENCES BIBLIOGRAPHIQUES

1. Albrecht, A. J. 1979. *Measuring Application Development Productivity*. Proceedings of the Joint SHARE, GUIDE, and IBM Application Development Symposium, Monterey, California, October 14–17, IBM Corporation. pp. 83–92.

2. Abran, A. , Desharnais, JM., Lesterhuis, A. *et al*, 2009. *La Méthode de mesure COSMIC de la taille fonctionnelle Version 3.0.1, Manuel de mesure (Le Guide COSMIC d'implémentation pour ISO/IEC 19761: 2003*. The COSMIC Group. 86 pages.

3. April, A., Merlo, E. et Abran, A. 1997. "A reverse engineering approach to evaluate function point rules". In *Fourth Working Conference on Reverse Engineering*.

4. Azzouz, S. et Abran, A. 2004. "A proposed measurement role in the Rational Unified Process (RUP) and its implementation with ISO 19761: COSMIC FFP". In *Software Measurement European Forum*. (Rome).

5. Bévo, V., Lévesque, G., Abran, A. 1999. Application de la méthode FFP à partir d'une spécification selon la notation UML: compte rendu des premiers essais d'application et questions. In *9th International Workshop Software Measurement*. (Lac Supérieur, Canada). pp. 230-242.

6. Bévo, Valéry, Lévesque, Ghislain et Meunier, Jean-Guy. 2003. "Toward an Ontological Formalization for a Software Functional Size Measurement Method's Application Process: The COSMIC-FFP Case", in *13th International Workshop on Software Measurement*. (Montréal, Canada)

7. Bévo, Valéry. 2003. *Analyse et formalisation ontologique des procédures de mesure associées aux méthodes de mesure de la taille fonctionnelle des logiciels: de nouvelles perspectives pour l'automatisation du processus d'application d'une méthode de mesure de la taille fonctionnelle des logiciels à partir des spécifications. Exemple d'application : la méthode COSMIC-FFP et les spécifications UML*. Présentation du projet de recherche. UQAM.

8. Boehm, Barry. 1981. *Software Engineering Economics*. Englewood Cliffs, NJ: Prentice-Hall. ISBN 0-13-822122-7.

9. Boehm, Barry, Abts, Chris. Brown, A. Winsor. Chulani, Sunita, Clark, Bradford K., Horowitz, Ellis, Madachy, Ray, Reifer, Donald J. et Steece, Bert. 2000. *Software cost*

estimation with COCOMO II. Englewood Cliffs, NJ:Prentice-Hall. ISBN 0-13-026692-2

10. Coman ID, Silliti A, Succi G. 2009. "A case-study on using an Automated In-process Software Engineering Measurement and Analysis system in an industrial environment". In *IEEE 31st International Conference on Software Engineering*. (Vancouver, BC, May 16-24). pp. 89-99.

11. Condori-Fernández, N., Abrahão, S., Pastor, O. 2007. "On the Estimation of Software Functional Size from Requirements Specifications". *Journal of Computer Science and Technology 22(3)*. pp 358-370.

12. Diab, H., Frappier, M., and St-Denis, R. 2001. "Formalizing COSMIC-FFP Using ROOM," ACS/IEEE Int. Conf. on Computer Systems and Applications (AICCSA). (Beirut, Lebanon). pp. 312 – 318.

13. Diab, H., Frappier, M. et St-Denis, R. 2002. "A formal definition of function points for automated measurement of B specifications". In Formal Methods and Software Engineering, Lecture Notes in Computer Science vol. 2495, Springer, Berlin. C. George and H. Miao. pp 483-494.

14. Diab, H. Koukane, F. Frappier et St-Denis, R. 2005. "µcROSE: Automated Measurement of COSMIC-FFP for Rational Rose Real Time". In *Information and Software Technology 47(3)*, pp. 151-166.

15. Felfernig, A. et Salbrechter, A. 2004. "Applying function point analysis to effort estimation in configurator development". In *International Conference on Economic, Technical and organisational aspects of Product Configuration Systems*. (Kopenhagen, Denmark). pp. 109-119.

16. Fraternali, Piero, Tisi, Massimo et Bongio, Aldo. 2006. "Automating Function Point Analysis with Model driven Development". In *the 2006 conference of the Center for Advanced Studies on Collaborative research*. (Toronto, Canada, October 16-19).

17. Grau, G., Franch, X. 2007. "Using the PRiM method to Evaluate Requirements Model with COSMIC-FFP". In *Proceedings of the IWSM-MENSURA 2007*. (Mallorca). pp. 110-120.

18. Habela, P., Glowacki, E., Serafinski, T., Subieta, K. 2005. "Adapting Use Case Model for COSMIC-FFP Based Measurement". In *15th International Workshop on Software Measurement – IWSM 2005*. (Montréal). pp. 195--207.

19. Ho, V.T. et Abran, A. 1999. “A Framework for Automatic Function Point Counting from Source Code”. In *9th International Workshop Software Measurement*. (Lac Supérieur, Canada).

20. I-Logix. 2001. *Statemate MAGNUM Tutorial*. (Andover, USA). 62 pages.

21. International Organization for Standardization. 2002. Information Technology - Software Engineering - *Mk II Function Point Analysis - Counting Practices Manual*. MKII - ISO/IEC 20968. Geneva.

22. International Organization for Standardization. 2005. Information Technology – *Definitions And Counting Guidelines For The Application Of Function Point Analysis*. NESMA - ISO/IEC 24570. Geneva.

23. International Organization for Standardization. 2005. Information Technology - Software and systems engineering - *FiSMA 1.1 functional size measurement method*. ISO/IEC 29881:2008. Geneva.

24. International Organization for Standardization. 2006. Information technology – Software measurement -- Functional size measurement -- *Part 6: Guide for use of ISO/IEC 14143 series and related International Standards*. ISO/IEC 14143-6:2006. Geneva.

25. International Organization for Standardization. 2007. Software Engineering - *Software Measurement Process*. ISO/IEC 15939. ISBN: 9780580572500.

26. International Organization for Standardization. 2009. Software Engineering - Software and systems engineering -Software measurement - *IFPUG functional size measurement method*. IFPUG - ISO/IEC 20926:2009. Geneva.

27. International Organization for Standardization. 2011. Software Engineering. *COSMIC - A Functional Size Measurement Method*. COSMIC ISO/IEC 19761:2011. Geneva.

28. Jenner, M.S. 2001. “COSMIC-FFP and UML: Estimation of the Size of a System Specified in UML – Problems of Granularity”. In *4th European Conference on Software Measurement and ICT Control*. (Heidelberg, Allemagne). pp. 173-184.

29. Komi-Sirviö, Seija, Parviainen, Päivi et Ronkainen, Jussi. 2001. "Measurement Automation: Methodological Background and Practical Solutions-A Multiple Case Study," In *IEEE Seventh International Software Metrics Symposium (METRICS'01)*. pp. 306.

30. Lamma, Mello et Riguzzi. 2004. “A system for measuring Function Points from an ER-DFD specification”, *The Computer Journal*, 47(3):358-372.

31. Lemaitre, Jonathan, April, Alain et Desharnais, Jean-Marc. ‘Estimation de points de fonction d’un programme à partir de ses accès aux bases de données’, article non publié.

32. Levesque, G., Bevo, V., and Cao, D. T. 2008. “Estimating software size with UML models”. In *Proceedings of the 2008 C3S2E Conference*. (Montreal). pp.81-87.

33. Li, Zhen, Nonaka, Makoto, Kakurai, Akihiro et Azuma, Motoei. 2003. “Measuring Functional Size of Interactive Software: A Support System Based on XForms-Format User Interface Specifications”. In the *Third International Conference on Quality Software*. Page: 368. ISBN:0-7695-2015-4.

34. Marín, Beatriz, Condori-Fernández, Nelly et Pastor, Oscar. 2008. “Towards a Method for Evaluating the Precision of Software Measures”. In *Eighth International Conference on Quality Software QSIC '08*. pp. 305-310.

35. Marín, Beatriz, Pastor, Oscar, Giachetti, Giovanni. 2008. “Automating the Measurement of Functional Size of Conceptual Models in an MDA Environment”, *PROFES 2008*, pp. 215-229.

36. Marín, B., Condori-Fernández, N. et Pastor, O. 2008. “Design of a functional size measurement procedure for a model-driven software development method” in *Proceedings of the 3rd Workshop on Quality in Modeling of MODELS (QiM '08)*. J.-L. Sourrouille, M. Staron, L. Kuzniarz, P. Mohagheghi, and L. Pareto, Eds. Toulouse, France. pp. 1–15.

37. Marín, B., Pastor, O. et Abran, A. 2010. “Towards an accurate functional size measurement procedure for conceptual models in an MDA environment”. *Data & Knowledge Engineering*, vol. 69, no. 5, pp. 472–490.

38. Marín, B., Condori-Fernández, N., Pastor, O., Abran, A. 2008. “Measuring the Functional Size of Conceptual Models in a MDA Environment”. In *20th*

International Conference on Advanced Information Systems Engineering Forum. (Montpellier). pp. 33-36.

39. Marín, Beatriz, Giachetti, Giovanni et Pastor, Oscar. 2008. "Measurement of Functional Size in Conceptual Models: A Survey of Measurement Procedures based on COSMIC". In *IWSM/Metrikon/Mensura '08 International Conferences on Software Process and Product Measurement*. (Munich, Germany). pp. 170-183.

40. Mathworks. http://www.mathworks.com/

41. McGarry, John, Card, David, Jones, Cheryl, Layman, Beth, Clark, Elizabeth, Dean, Joseph, et Hall, Fred, 2002. *Practical Software Measurement: Objective Information for Decision Makers*. Addison-Wesley. 277 p.

42. Meli, R. et Santillo, L. 1999. "Function Point Estimation. Methods: a Comparative Overview", *FESMA '99 Conference proceedings*. (Amsterdam, 4-8 Octobre).

43. Mendes O. 1996. « Développement d'un protocole d'évaluation pour les outils informatisé de comptage automatique de points de fonction ». Mémoire de la maitrise en informatique de gestion. UQAM. Canada.

44. Mendes, O., Abran, A., Bourque, P. 1996. "An FP Tool Classification Framework and Market Survey". In *IFPUG International Function Point Users Group Fall Conference*. (Dallas).

45. Nagano, S., Ajisaka, T. 2003. "Functional metrics using COSMIC-FFP for object-oriented realtime systems", *13th International Workshop on Software Measurement*. (Montreal)

46. Oligny, Serge et Abran, Alain. 1999. "On The Compatibility Between Full Function Points And IFPUG Function Points". *10th European Software Control and Metric Conference (ESCOM SCOPE 99)*. (Herstmonceux Castle, England, April 27-29). pp. 1 - 9.

47. Paton, K. et Abran, A. 1995. *A Formal Notation For The Rules Of Function Point Analysis*. Research Report 247, Département de mathématique et d'informatique. (UQAM, Canada), 44 pages.

48. Poels, G. 2002. “A Functional Size Measurement Method for Event-Based Object-Oriented Enterprise Models”. *4th International Conference on Enterprise Information Systems –ICEIS*. (Ciudad Real). pp. 667-675.

49. Schreiber, G., Akkermans, H., Anjewierden, A., De Hoog, R., Shadbolt, N., Van de Velde, W., and Wielinga, B. 1999. *Knowledge engineering and management: The CommonKADS methodology*, A Bradford Book. The MIT Press, Cambridge, Massachusetts,

50. Stambollian, A., Abran, A. 2006. “Survey of Automation Tools Supporting COSMIC-FFP – ISO 19761”. *International Workshop on Software Measurement – IWSM-Metrikom*, (Postdam, Germany, Nov. 2-3). Shaker Verlag. pp. 435-454

51. Uemura, Takuya, Kusumoto, Shinji et Katsuro, Inoue, "Function Point Measurement Tool for UML Design Specification", Sixth International Software Metrics Symposium (METRICS'99), 1999. pp. 62.

52. Xunmei, G., Guoxin S., Hong Z. 2006. “The Comparison between FPA and COSMIC-FFP”. *Software Measurement European Forum - SMEF'2006*. (May 10-12, Rome, Italy).

53. Zelkowitz, Marvin V. 2009. *Advances in Computers: Computer Performance Issues*. Academic Press. 360 p.

54. Zuse, Horst. 1998. *A framework of software measurement*. Walter De Gruyter Inc. ISBN-10: 3110155877. ISBN-13: 978-3110155877. 755 p.