

UNIVERSITÉ DU QUÉBEC À MONTRÉAL

CONCEPTION ET MISE EN ŒUVRE D'UNE TECHNIQUE DE
FORMALISATION PAR DES TRIPLETS POUR L'AUTOMATISATION DE
LA MESURE DE LA TAILLE FONCTIONNELLE À PARTIR D'EXIGENCES
LOGICIELLES ÉCRITES EN LANGAGE NATUREL.

THÈSE

PRÉSENTÉE

COMME EXIGENCE PARTIELLE

DU DOCTORAT EN INFORMATIQUE COGNITIVE

PAR

BRUEL GÉRANÇON

JUIN 2022

UNIVERSITÉ DU QUÉBEC À MONTRÉAL
Service des bibliothèques

Avertissement

La diffusion de cette thèse se fait dans le respect des droits de son auteur, qui a signé le formulaire *Autorisation de reproduire et de diffuser un travail de recherche de cycles supérieurs* (SDU-522 – Rév.04-2020). Cette autorisation stipule que «conformément à l'article 11 du Règlement no 8 des études de cycles supérieurs, [l'auteur] concède à l'Université du Québec à Montréal une licence non exclusive d'utilisation et de publication de la totalité ou d'une partie importante de [son] travail de recherche pour des fins pédagogiques et non commerciales. Plus précisément, [l'auteur] autorise l'Université du Québec à Montréal à reproduire, diffuser, prêter, distribuer ou vendre des copies de [son] travail de recherche à des fins non commerciales sur quelque support que ce soit, y compris l'Internet. Cette licence et cette autorisation n'entraînent pas une renonciation de [la] part [de l'auteur] à [ses] droits moraux ni à [ses] droits de propriété intellectuelle. Sauf entente contraire, [l'auteur] conserve la liberté de diffuser et de commercialiser ou non ce travail dont [il] possède un exemplaire.»

REMERCIEMENTS

Heureux la personne qui a trouvé la sagesse et qui possède l'intelligence. Car le gain qu'elle procure est préférable à celui de l'argent, et le profit qu'on en tire vaut mieux que l'or. C'est par la sagesse que l'Éternel a fondé la terre, c'est par l'intelligence qu'il a affermi les cieux. C'est par sa science que les abîmes se sont ouverts, et que les nuages distillent la rosée.

Tout d'abord, je tiens à exprimer mes plus vifs remerciements et ma gratitude au créateur de l'univers qui m'a accordé de la sagesse et de l'intelligence pour mener à bien ce projet de recherche.

Je veux ensuite remercier spécialement mes trois directeurs de recherche, en l'occurrence, professeurs Roger Nkambou, Sylvie Trudel et Serge Robert pour leur encadrement, leurs innombrables heures de lectures, leurs commentaires et leurs remarques constructives. Je les remercie également pour la confiance qu'ils m'ont accordée. Outre de ces apports scientifiques, je les remercie pour leurs qualités humaines et leur soutien qui m'ont permis de mener à bien ma thèse de doctorat.

Je remercie spécialement la professeure Sylvie Trudel pour ses attentives relectures, ses suggestions et ses qualités humaines. De plus, elle est la première experte que j'ai consultée dans le domaine du processus de mesure de la taille fonctionnelle des logiciels, et qui m'a ouvert les yeux sur ce sujet de recherche.

Par ailleurs, je tiens à souligner l'apport des professeurs qui enseignent au programme de doctorat en informatique cognitive qui, à travers leur cours et séminaire, m'ont ouvert les yeux dans le domaine de l'intelligence artificielle.

Je tiens à remercier le professeur Hakim Lounis pour avoir accepté de présider le jury. Je voudrais également remercier le professeur Stevan Harnard d'avoir accepté d'être membre du jury interne. Je tiens à remercier la professeure Asma Sellami d'avoir accepté d'être membre du jury externe. Par ailleurs, je remercie le professeur Petko Valtchev pour ses échanges scientifiques enrichissants.

Je remercie l'agente à la gestion des études du programme de doctorat en informatique cognitive, en l'occurrence, Madame Mylène Dagenais, ainsi que Mme Élisabeth Linday pour leur écoute attentive à nos requêtes.

Je profite de remercier particulièrement Anie de m'avoir accompagné à la fin de ce travail de recherche.

Finalement, je remercie mes amis, ainsi que mes frères et mes sœurs qui ont beaucoup investi dans mes études universitaires et dans mon parcours professionnel.

TABLE DES MATIÈRES

LISTE DES TABLEAUX	xi
LISTE DES FIGURES	xiii
RÉSUMÉ	xv
CHAPITRE I INTRODUCTION	1
1.1 Contexte de la recherche	1
1.2 Problématique	2
1.3 Questions de recherche et approche proposée	5
1.4 Hypothèses de recherche	6
1.5 Objectifs de recherche	7
1.6 Plan de la thèse	8
CHAPITRE II REVUE DE LA LITTÉRATURE	10
2.1 Introduction	10
2.2 Mesures du logiciel	11
2.3 Mesure de la taille fonctionnelle	12
2.4 Corrélation entre la taille fonctionnelle et l'effort	13
2.5 Méthodes de mesures de la taille fonctionnelle des logiciels	15
2.5.1 Méthode IFPUG	15
2.5.2 Méthode COSMIC –ISO 19761 : 2019	16
2.5.3 Compatibilité entre IFPUG et COSMIC	18
2.5.4 Lacunes de IFPUG absentes dans COSMIC	19
2.6 Automatisation de la mesure de la taille fonctionnelle des logiciels	20
2.6.1 Outils d'automatisation de la mesure de la taille fonctionnelle des logiciels	21
2.6.2 Automatisation à partir du processus unifié	21

2.6.3	Automatisation à partir des diagrammes UML (diagrammes de Cas d'utilisation, diagrammes de séquence)	22
2.6.4	Automatisation à partir des règles formelles (équations mathématiques)	23
2.6.5	Automatisation à partir des modèles conceptuels	24
2.6.6	Automatisation à partir des modèles entités-relations et d'un diagramme de flux de données	24
2.6.7	Automatisation à partir du code source	25
2.6.8	Automatisation de la mesure avec IFPUG	26
2.6.9	Cadre référentiel pour les outils d'automatisation de la mesure de la taille fonctionnelle des logiciels	26
2.6.10	Synthèse	27
2.7	Mesure des points de fonction	28
2.7.1	Utilisation des points de fonction dans l'industrie	30
2.7.2	Limites de la méthode de point de fonction	30
2.8	Techniques et méthodes de rédaction d'exigences du logiciel	31
2.8.1	Les Cas d'utilisation (<i>Use Cases</i>)	32
2.8.2	Les récits d'utilisation (<i>User Stories</i>)	33
2.8.3	Développement piloté par les tests d'acceptation (ATDD)	34
2.9	Outils de tests fonctionnels et de tests d'acceptation pour la rédaction des exigences logicielles	35
2.9.1	Outil Cucumber/Gherkin	36
2.9.2	Syntaxes de Gherkin	36
2.9.3	Outil FitNesse	38
2.10	Limites des approches, méthodes ou techniques de rédaction d'exigences du logiciel	38
2.11	Synthèse	41
	CHAPITRE III CADRE CONCEPTUEL DE LA THÈSE	43
3.1	Problématique de la recherche et volet cognitif	43

3.2	Mémoire épisodique, sémantique et procédurale	44
3.2.1	Mémoire épisodique	44
3.2.2	Mémoire sémantique	45
3.2.3	Mémoire procédurale	46
3.3	Rôle des mémoires épisodique, sémantique et procédurale dans notre recherche	47
3.4	Ontologie et mémoire sémantique formalisée	49
3.5	Outils de construction/d'extraction d'ontologies	50
3.6	Approche à base de règles en intelligence artificielle et logiques de description	51
3.7	Représentation des connaissances et logiques de description	52
3.8	Logiques de description et prédicats du premier ordre	53
3.8.1	Logiques de description	53
3.8.2	Logique des prédicats du premier ordre	55
3.9	Synthèse	55
	CHAPITRE IV MÉTHODOLOGIE DE RECHERCHE	57
4.1	Introduction	57
4.2	Notre approche méthodologique	58
4.2.1	Évaluation des techniques de rédaction des exigences du logiciel	58
4.2.2	Conception d'un triplestore pour la rédaction d'exigences du logiciel	59
4.2.3	Développement d'un outil d'automatisation de la mesure de la taille fonctionnelle des logiciels, selon la méthode COSMIC ISO 19761	59
4.2.4	Évaluation et validation des résultats	59
4.2.5	Cadre référentiel pour la validation des outils d'automatisation de la mesure de la taille fonctionnelle	60
4.2.6	Validation par les experts en COSMIC ISO 19761	61
4.3	Originalité de la recherche	61

4.4 Synthèse	61
CHAPITRE V ÉVALUATION DES TECHNIQUES ET MÉTHODES DE RÉDACTION D'EXIGENCES DU LOGICIEL ET PROPOSITION D'UNE APPROCHE PAR TRIPLETS POUVANT FACILITER L'AUTOMATISATION DE LA MESURE	63
5.1 Introduction	63
5.2 Évaluation des techniques et méthodes de rédaction d'exigences du logiciel	64
5.3 Spécification des exigences logicielles avec l'approche par triplets	65
5.4 Correspondance entre les concepts du triplet et la méthode COSMIC	67
5.5 Approche par triplets, une méthodologie pour le développement des exigences logicielles	67
5.6 Modèle de triplets (triplestore) développé	70
5.7 Modèle de triplets et logique des prédicats du premier ordre	72
5.8 Glossaire : Concepts du modèle de triplets (triplestore)	74
5.9 Synthèse	75
CHAPITRE VI OUTIL DE GÉNÉRATION AUTOMATIQUE DES TRIPLETS À PARTIR DES USE CASES ET DE MESURE DE LA TAILLE FONCTIONNELLE DES LOGICIELS	76
6.1 Introduction	76
6.2 Génération automatique des textes	77
6.3 Outil de génération des triplets et d'automatisation de la taille fonctionnelle	77
6.3.1 Algorithme de génération des triplets et logique des prédicats du 1er ordre	79
6.3.2 Fonctionnement de l'algorithme de génération des triplets	81
6.3.3 Algorithme de génération du sujet	81
6.3.4 Algorithme de génération du prédicat (verbe)	82
6.3.5 Règle ou algorithme de génération de l'objet (complément)	83
6.3.6 Algorithme de génération des triplets (sujet, prédicat, objet)	84

6.3.7	Génération des triplets par ellipse grammaticale	85
6.3.8	Algorithme du découpage par ellipse grammaticale	87
6.3.9	Génération des triplets par découpage multiple	87
6.3.10	Algorithme de découpage multiple avec plusieurs objets	89
6.3.11	Critères de génération des triplets	89
6.3.12	Algorithme de découpage multiple avec des connecteurs logiques	90
6.4	Résultats et validation des Cas d'utilisation générés en triplets	92
6.5	Méthode d'évaluation des Cas d'utilisation générés en triplets	92
6.6	Module d'obtention de la taille fonctionnelle à partir des triplets	94
6.6.1	Processus de mesurage de la méthode COSMIC	94
6.6.2	Règle ou algorithme d'obtention de la taille fonctionnelle	95
6.6.3	Règles d'identification des mouvements de données	96
6.6.4	Règle d'identification des mouvements de données de type En- trée (E/E)	97
6.6.5	Règle d'identification des mouvements de données de type Sor- tie (S/X)	98
6.6.6	Règle d'identification des mouvements de données de type Lec- ture (L/R)	98
6.6.7	Règle d'identification des mouvements de données de type éCri- ture (C/W)	98
6.7	Description d'un Cas d'utilisation et sa taille fonctionnelle manuelle	100
6.8	Description d'un Cas d'utilisation et sa taille fonctionnelle automatique	101
6.9	Capacité d'apprentissage automatique de l'outil	101
6.10	Algorithme d'heuristique (exclusion des triplets)	103
6.11	Choix des technologies	104
6.12	Synthèse	105
CHAPITRE VII ÉVALUATION ET ANALYSE DES RÉSULTATS		106
7.1	Introduction	106

7.2	Résultats de la taille fonctionnelle des cinq (5) projets en anglais . . .	107
7.2.1	Projet : « Resto Sys »	107
7.2.2	Résultat de la mesure manuelle et automatisée du projet #1 « Resto Sys»	108
7.2.3	Projet #2 : « ACME Car Hire »	109
7.2.4	Résultat de la mesure manuelle et automatisée du projet #2 « ACME Car Hire »	110
7.2.5	Projet #3 : « Rice Cooker »	111
7.2.6	Résultat de la mesure manuelle et automatisée du projet « Rice Cooker»	112
7.2.7	Projet #4 : « uObserve »	112
7.3	Résultat de la mesure manuelle et automatisée du projet «uObserve»	114
7.3.1	Projet #5 : « C-Reg »	114
7.3.2	Résultat de la mesure manuelle et automatisée du projet « C-Reg »	115
7.4	Résultats de la taille fonctionnelle des cinq (5) projets en français . .	117
7.4.1	Projet #6 : « Système de gestion de réservation » (SGR) . . .	117
7.4.2	Résultat de mesure manuelle et automatisée du projet « SGR »	118
7.4.3	Projet #7 : « Système de gestion du commerce électronique » (SGCE)	119
7.4.4	Résultat de mesure manuelle et automatisée	119
7.4.5	Projet #8 : « Système de guichet automatique » (SGA) . . .	120
7.4.6	Projet #9 : « Application de contrôle d'accès à un bâtiment » (ACAB)	121
7.4.7	Résultat de mesure manuelle et automatisée	121
7.4.8	Projet #10	121
7.5	Synthèse des résultats	123
7.6	Réponses aux sous-questions de recherche	124
7.6.1	Première sous-question : difficulté d'automatisation	124

7.6.2	Deuxième sous-question : avantages à automatiser la mesure	125
7.6.3	Troisième sous-question : approche par triplets comme méthode de rédaction	128
7.6.4	Question principale de recherche : Approche par triplets comme technique facilitant l'automatisation	130
7.7	Atteinte des objectifs de recherche	131
7.8	Évaluation et validation des résultats	131
7.9	Utilité du processus d'automatisation de la mesure de la taille fonctionnelle des logiciels	132
7.10	Synthèse	134
CHAPITRE VIII CONCLUSION : CONTRIBUTION, PERSPECTIVES ET LIMITES DE RECHERCHE		136
8.1	Conclusion	136
8.2	Contribution /originalité de la thèse	137
8.2.1	Contribution informatique	137
8.2.2	Contribution cognitive	139
8.3	Perspectives et limites de recherche	143
APPENDICE A PUBLICATIONS DE L'AUTEUR LIÉES À CETTE THÈSE		146
APPENDICE B PROTOTYPE : QUELQUES EXEMPLES DE CLASSES D'IMPLEMENTATION		147
APPENDICE C RÉSULTATS DÉTAILLÉS DE MESURE MANUELLE ET AUTOMATISÉE DES PROJETS EN ANGLAIS		155
RÉFÉRENCES		189

LISTE DES TABLEAUX

Tableau	Page
2.1 Poids de la taille fonctionnelle FPA par niveau.	29
2.2 Poids de la taille fonctionnelle FPA par niveau.	30
5.1 Faiblesses des approches de rédaction d'exigences du logiciel. . . .	65
5.2 Correspondance entre les concepts du triplet et la méthode COSMIC. . .	67
5.3 Correspondance entre les concepts du triplet,UML et la méthode COSMIC.	70
6.1 Critères de découpage des Cas d'utilisation ou Récits utilisateurs. . . .	90
6.2 Triplets générés à partir des Use Cases ou User Stories.	92
6.3 Évaluation des triplets générés à partir des Use Cases ou User Stories. . .	94
6.4 Règles de correspondance des mouvements de données.	99
6.5 Exemple de mesure manuelle d'un processus fonctionnel.	100
6.6 Outils technologiques.	105
7.1 Résultats de mesure manuelle et automatisée de l'application « Resto Sys ».	109
7.2 Résumé des résultats de mesurage manuel et automatisé du projet « ACME Car Hire ».	110
7.3 Résultats de mesure manuelle et automatisée du projet « Rice Co- oker ».	112
7.4 Résultats de mesure manuelle et automatisée du projet «uObserve». . . .	114
7.5 Résultats de mesure manuelle et automatisée du projet «C-Reg».	117
7.6 Résultats de mesure manuelle et automatisée du projet SGR.	118

7.7	Résultats de mesure manuelle et automatisée du projet SGCE. . .	119
7.8	Résultats de mesurage manuel et automatisé du système « SGA ». . .	120
7.9	Résultats de mesurage manuel et automatisé du projet « ACAB ». . .	122
7.10	Résultats de mesure manuelle et automatisée du projet #10. . . .	122
7.11	Synthèse des résultats de mesure manuelle et automatisée.	124
7.12	Effort de mesurage manuel et automatisé.	135

LISTE DES FIGURES

Figure	Page
2.1	Corrélation entre la taille et l'effort. 14
2.2	Vue d'ensemble de la méthode COSMIC. 18
2.3	Format de description d'un User Story 33
2.4	Syntaxe en Gherkin 37
2.5	Exemple de scénario en Gherkin 37
3.1	Rôle des différentes mémoires dans la thèse 48
4.1	Phases de la méthodologie de la recherche. 60
5.1	Modèle de triplets de rédaction des exigences logicielles. 71
5.2	Exemple de Cas d'utilisation sous forme de triplets. 73
6.1	Algorithme de génération du sujet. 82
6.2	Algorithme de génération du prédicat (verbe). 83
6.3	Algorithme de génération de l'objet (complément). 84
6.4	Algorithme de génération des triplets. 85
6.5	Algorithme de découpage par ellipse grammatical. 87
6.6	Algorithme de découpage multiple avec plusieurs objets (compléments). 89
6.7	Algorithme de découpage multiple avec des connecteurs logiques. 91
6.8	Processus de mesure de la méthode COSMIC (Abran et al, 2020). 95
6.9	Exemple de mesure automatisée de la taille fonctionnelle. 101
6.10	Algorithme d'exclusion des triplets redondants par l'outil. 104

7.1	Diagramme des cas d'utilisation de « Resto Sys » (Selami <i>et al.</i> , 2019)	108
7.2	Utilisateurs fonctionnels du « Rice Cooker»	111
7.3	Diagramme de Cas d'utilisation de uObserve	113
7.4	Liste des fonctionnalités du système « C-Reg » (Symons et al; 2018).115	
7.5	Cas d'utilisation du système du SGR (Levesque, 2015).	117

RÉSUMÉ

Le domaine de l'automatisation de la mesure de la taille fonctionnelle des logiciels, à partir des documents de spécifications logicielles, reste et demeure peu développé jusqu'à présent. En effet, la littérature consultée a montré que les tentatives d'automatisation du processus de la mesure de la taille fonctionnelle des logiciels ont eu peu de succès au niveau de l'industrie. Plusieurs outils d'automatisation de la mesure de la taille fonctionnelle des logiciels ont été développés en fonction de la méthode COSMIC ISO 19761 et de celle de IFPUG. Toutefois, ces outils présentent de nombreux défauts, contraintes et limitations. De plus, les méthodes, techniques et outils de rédaction des documents de spécifications logicielles utilisés dans l'industrie sont loin de permettre l'automatisation de la mesure de la taille fonctionnelle des logiciels. Dans l'industrie, les exigences du logiciel sont généralement écrites en langage naturel, et aucun détail technique n'est précisé. Ainsi, les exigences du logiciel sont incomplètes, incohérentes et propices aux ambiguïtés, et donc des erreurs d'interprétation peuvent facilement être commises par les analystes. De ce fait, l'automatisation de la mesure de la taille fonctionnelle des logiciels n'est pas une tâche évidente.

Cette thèse propose une nouvelle technique de rédaction d'exigences du logiciel permettant de favoriser l'automatisation du processus de mesure de la taille fonctionnelle des logiciels à partir des documents de spécifications logicielles écrites en langage naturel. De ce fait, nous avons exploré les techniques, outils et méthodes de rédaction d'exigences logicielles existantes, ainsi que leurs limitations. Par la suite, nous avons examiné les possibilités offertes par un modèle de spécification des exigences fondé sur la logique des prédicats du 1er ordre. En effet, nos contributions portent principalement sur trois (3) propositions qui sont notamment : a) un modèle de triplets fondé sur la logique des prédicats du 1er ordre, comme modèle de spécification, de rédaction et de représentation des exigences logicielles pour permettre l'automatisation de la taille fonctionnelle des logiciels à partir des exigences écrites sous forme de structure de triplet ; b) un outil permettant de générer des triplets à partir des exigences logicielles écrites en langage naturel, en vue de l'automatisation du processus de mesure de la taille fonctionnelle des logiciels ; c) l'implémentation des règles de la méthode COSMIC permettant d'identifier les types de mouvements de données (Entré, Sortie, Lecture, éCriture) de chaque processus fonctionnel.

Les résultats de cette recherche démontrent que la rédaction d'exigences fonctionnelles sous forme de structure de triplets permet d'automatiser le processus de mesure de la taille fonctionnelle des logiciels à partir des documents de spécifications. Notre technique a été éprouvée, testée et validée par l'outil d'automatisation proposé, tel que défini par la méthode COSMIC ISO 19761. Nous avons testé l'outil avec dix (10) documents de spécifications d'exigences logicielles, en raison de cinq (5) projets anglais et de cinq (5) projets français. Les résultats présentés ont été comparés à ceux d'experts humains certifiés avec la méthode COSMIC. Les résultats mesurés manuellement des projets écrits en anglais sont publiés et disponibles sur le site Internet de COSMIC. En fait, l'outil d'automatisation que nous proposons pour valider notre technique de rédaction d'exigences présente des résultats automatisés cohérents avec les résultats manuels validés et publiés par des experts, avec une précision moyenne de 96,96%, variant entre 93,14% et 100%.

Mots clés : Méthode COSMIC, Automatisation, Taille fonctionnelle, Triplet, Traitement automatique des langues naturelles.

ABSTRACT

The domain of software functional size measurement automation, from software specification documents, has been a research topic over the last. The literature consulted shows that attempts to automate the process of measuring the software functional size has obtained little success at the industry level. Several tools for automating the measurement of software functional size have been developed according to the Common Software Measurement International Consortium (COSMIC) method (ISO 19761) website and that of International Function Point User Group (IFPUG). However, these tools encountered many flaws, constraints, and limitations. Moreover, the methods, techniques and tools for writing software specification documents used in the industry are far from allowing easily the automation of the measurement of software functional size. In industry, software requirements are often written in natural language, and no technical details are specified. Thus, software requirements are usually incomplete, inconsistent, and prone to ambiguities, and therefore, the analysts can easily make errors of interpretation. Therefore, automating the software functional size measurement is not an easy task.

This thesis proposes a new technique for writing software requirements (SR) to promote the automation of the software functional size measurement process from software requirements specifications documents written in natural language. Indeed, our contributions mainly relate to three (3) proposals which are in particular : a) A triplet model based on the logic of first order predicates, as a model for specifying, drafting, and representing SR to allow the automation of the software functional size from requirements written as triplets; b) A tool for generating

triplets from software requirements specifications written in natural language, in order to automate the software functional size measurement process ;c) The implementation of the rules of the COSMIC method which allows identifying the types of data movements (Entry, eXit, Read, Write).

The results of our research showed that writing functional requirements as a triplet structure allows automating the software functional measurement process from SRS documents. Our technique has been proven, tested, and validated by the proposed automation tool, as defined by the COSMIC ISO 19761 method. We have tested the tool with ten (10) SRS documents. The results presented were compared with those of human experts certified with the COSMIC method. In fact, the automation tool proposed presents automated results consistent with the manual results validated, and published by experts, with an average of 96,96%, where the precision varies between 93,14%, and 100%.

Keywords : COSMIC method, Automation, Functional size, Triplet, Automatic natural language processing.

ACRONYMES

ATDD	Acceptance Test-Driven Development
BDD	Behavior Driven Development
COCOMO	Constructive COst Model
COSMIC	Common Software Measurement International Consortium
DL	Descriptive logical
E	Entry
EA	Effort automatisé
EM	Effort manuel
FISMA	Finnish Software Measurement Association
FPA	Functional Point Analysis
GAB	Guichet automatique bancaire
IA	Intelligence artificielle
IDE	Integrated Development Environment
IFPUG	International Function Point Users Group
ISO	International Standardization Organization
LD	Logique de description
LOC	Lines of code
MTF	Mesure de la taille fonctionnelle
NESMA	Netherlands Software Measurement Association

NTIC	Nouvelles technologies de l'information et de la communication
OGT	Outils de génération des triplets
OWL	Web Ontology Language
PF	Points de fonction
PFC	Points de fonction COSMIC
PSO	Prédicat, sujet, objet
RC	Règles de correspondance
RUP	Rational Unified Process
SFSM	Software Functional Size Measurement
SAB	Système automatique bancaire
SBC	Système à base de connaissances
SGCE	Système de gestion de commerce électronique
SGR	Système de gestion de réservation
SIG	Système d'information de Gestion
SOP	Sujet, objet, prédicat
SPO	Sujet, prédicat, objet
SR	Software Requirements
SRS	Software Requirements Specifications
TCF	<i>Technical Complexity Factor</i>
TF	Taille fonctionnelle
UFP	Unadjusted Function Point
UML	Unified Modeling Language
W	Write
X	eXit

XML eXtensible Markup Language

XP Extreme Programming

CHAPITRE I

INTRODUCTION

1.1 Contexte de la recherche

La mesure de la taille fonctionnelle (MTF) des logiciels joue un rôle important dans le domaine du génie logiciel et des nouvelles technologies de l'information et de la communication (NTIC). Elle est un facteur clé qui permet d'estimer l'effort et le coût de réalisation de produits logiciels. Plusieurs méthodes et approches d'estimation ont été proposées. À titre d'exemple, (Boehm *et al.*, 1981) a proposé la méthode COCOMO (CONstructive COst MOdel) pour estimer le coût et la durée des projets logiciels. COCOMO s'appuie sur l'estimation du nombre de lignes de code devant être écrites pour un logiciel. Ainsi, le nombre de lignes de code correspond à la taille physique du logiciel. (Albrecht, 1979) a proposé une méthode s'appuyant sur le nombre de points de fonction, le principe étant d'identifier et de quantifier les fonctionnalités utilisateurs, donnant ainsi naissance à la notion de taille fonctionnelle (TF). Plusieurs méthodes de mesurage des logiciels, telles que COSMIC, IFPUG, NESMA, Mark-II et FISMA ont été proposées et approuvées par l'Organisation Internationale de Normalisation (ISO). Parmi les différentes méthodes et outils existants, COSMIC est une récente méthode de mesure, développée dans le but de pallier les faiblesses des méthodes antérieures. Une particularité de la méthode COSMIC est qu'elle peut s'appliquer tôt dans le cycle de vie du

logiciel et sur un ensemble de composants logiciels, dont les logiciels embarqués et en temps-réel¹. Bien que plusieurs méthodes de mesurage des logiciels aient été proposées dans la littérature. Cependant, l'application de ces méthodes reste et demeure jusqu'à présent une tâche difficile (Abran, 2010) (Fenton et Bieman, 2019). De ce fait, l'industrie du génie logiciel a besoin d'outils d'automatisation du processus de la MTF des logiciels (Black et Wigg, 1999)(Bevo, 2005)(Quesada-López *et al.*, 2019). D'après la littérature consultée, l'une des principales pistes ou approches de recherche, en matière d'automatisation du processus de la MTF des logiciels part des spécifications (Bevo, 2005) (Levesque *et al.*, 2008) (Soubra, 2011) (Wentzlaff, 2017). Dans le cadre d'une telle approche, la TF des logiciels est mesurée à partir des documents de spécifications. Toutefois, les techniques de rédaction d'exigences du logiciel facilitent-elles l'automatisation de la MTF des logiciels? Ce travail de recherche s'inscrit dans le contexte d'automatisation du processus de mesure de la TF des logiciels à partir des spécifications fonctionnelles écrites en langage naturel.

1.2 Problématique

L'industrie du génie logiciel reconnaît que le coût et la productivité sont deux facteurs clés dans la réussite des projets logiciels. L'un des moyens que les responsables du projet peuvent utiliser pour faire une bonne estimation du coût, de l'effort et de la productivité d'un projet logiciel est de quantifier fonctionnellement le logiciel, c'est-à-dire qu'ils procèdent à la MTF des logiciels. Les méthodes de MTF des logiciels sont difficiles à appliquer dans l'industrie (Bevo, 2005)(Levesque *et al.*, 2008)(Fenton et Bieman, 2019) et, de ce fait, la conception des outils d'automatisation de la MTF des logiciels est une préoccupation pour les chercheurs,

1. www.cosmic-sizing.org

gestionnaires et professionnels qui travaillent dans le domaine de production de logiciels. La plupart des outils d'automatisation développés possèdent de nombreuses faiblesses et lacunes (Soubra, 2011). À titre d'exemple, (Diab *et al.*, 2005) ont proposé un outil appelé McRose qui permet de mesurer automatiquement la TF des logiciels, tel que défini par la méthode COSMIC, à partir du code source en C++ du logiciel, et non pas à partir des exigences fonctionnelles. Toutefois, cet outil contient de nombreux défauts et contraintes au niveau de l'efficacité et de la performance. On entend par efficacité et performance la capacité d'un logiciel de produire le maximum des résultats avec le minimum d'effort à un temps requis (Laporte et April, 2018). Par exemple, McRose a été utilisé pour mesurer la TF d'un logiciel et les résultats présentés par cet outil ont été comparés avec les résultats d'un expert COSMIC indépendant. On a ainsi pu constater que les tailles fonctionnelles obtenues étaient nettement différentes. De plus, les résultats de la mesure présentés par McRose ne sont pas cohérents avec ceux validés par des experts. Cette limitation réside dans le fait que le module qui analyse le code C++ ne peut examiner que certains opérateurs de base, puisque cet outil utilise des segments de code C++ pour effectuer la mesure. (Azzouz et Abran, 2004) ont proposé un prototype d'automatisation de la MTF du logiciel en fonction du *Rational Unified Process* (RUP). La conception de cet outil était basée sur une mise en correspondance directe entre COSMIC et les notations UML. Cependant, cet outil n'a pas été testé et validé, il reste donc jusqu'à ce jour exploratoire. Ainsi, cet outil est limité, puisqu'un ensemble d'étapes doivent se faire manuellement et il est loin de permettre l'automatisation totale de la MTF des logiciels. Le domaine d'automatisation de la MTF des logiciels revêt une importance capitale pour les gestionnaires et chercheurs. (Lamma *et al.*, 2004) ont proposé un outil de mesure de points de fonction du logiciel, en fonction de la méthode IFPUG et à partir des exigences logicielles exprimées sous la forme de modèles entités-relations et de flux de données. Le problème avec la méthode IFPUG est surtout lié à ses

limitations et contraintes. IFPUG a été conçue pour des applications de gestion et ne s'applique pas à des logiciels embarqués ou à des applications industrielles et scientifiques (Abran, 2010). Les résultats de mesure présentés par cet outil sont, d'après les chercheurs, en conformité avec ceux des experts humains. Cependant, les résultats de mesures manuelles effectuées par les experts humains en IFPUG ne sont pas mentionnés dans l'article. Ainsi, de futurs travaux ont été annoncés par les chercheurs en vue de tester l'outil en utilisant d'autres applications logicielles, pour comparer par la suite les résultats avec ceux des experts humains (Lamma *et al.*, 2004). Selon la littérature consultée, la majorité des outils d'automatisation de la MTF des logiciels conçus ne sont pas efficaces (Laird et Brennan, 2006)(Soubra, 2011)(Singh, 2017). En effet, très souvent, les résultats de mesure présentés par l'outil ne facilitent pas les analyses et ne sont pas conformes à des normes. De ce fait, il serait intéressant de développer un nouvel outil selon un cadre référentiel normalisé qui permettrait de mesurer automatiquement et efficacement la TF des logiciels. Dans l'industrie, les exigences logicielles sont normalement rédigées dans un langage naturel ou techniquement neutre (Wieggers et Beatty, 2013)(Sadoun, 2014). Cela sous-entend que l'analyste fonctionnel ou d'affaires décrit les exigences du logiciel du point de vue utilisateur. Les outils de rédaction utilisés peuvent être les Cas d'utilisation (Use Cases) ou les récits d'utilisation (User Stories), ou encore des formes textuelles plus classiques. Les Use Cases et User Stories sont des formes textuelles largement utilisées pour identifier et consigner les besoins des utilisateurs. Ils sont rédigés typiquement en langage naturel et, par conséquent, ne comportent pas de terminologie technique. La TF des logiciels est généralement mesurée dans l'industrie à partir des documents de spécifications logicielles². Cependant, les techniques et méthodes de rédaction d'exigences du logiciel existantes, autrement dit les techniques de rédaction d'exigences du logi-

2. <https://cosmic-sizing.org/measurement-manual/>

ciel utilisées dans l'industrie, ne favorisent pas l'automatisation de la MTF des logiciels. Car les spécifications logicielles sont souvent ambiguës, incomplètes et incohérentes (Trudel, 2012). L'idée de parvenir à automatiser le processus de la MTF des logiciels paraît donc difficile. La problématique principale de ce projet de recherche réside au niveau des faiblesses des méthodes ou techniques de rédaction d'exigences du logiciel qui rendent difficile l'automatisation du processus de MTF à partir des spécifications fonctionnelles. De ce constat, le but de cette thèse est d'améliorer les méthodes de rédaction d'exigences du logiciel par une approche par triplets, qui facilitera l'automatisation de la MTF des logiciels. En d'autres termes, l'idée serait de proposer de rédiger les exigences sous forme de triplets afin de faciliter l'automatisation du processus de mesure de la TF des logiciels.

1.3 Questions de recherche et approche proposée

L'automatisation du processus de la MTF des logiciels intéresse de nombreux chercheurs et professionnels évoluant dans l'industrie. Les chercheurs estiment que la taille du logiciel est l'un des paramètres les plus importants pour l'estimation du coût et de l'effort des projets logiciels (Abran, 2015). De ce fait, la construction d'outils automatisant la TF des logiciels serait la bienvenue au niveau de l'industrie (Bevo, 2005)(Quesada-López *et al.*, 2019). La conception et la construction d'outils d'automatisation efficaces de la MTF des logiciels seraient facilitées par une nouvelle méthode de rédaction d'exigences du logiciel. En effet, les difficultés qu'a rencontrées l'industrie dans le cadre de la MTF des logiciels nous amènent à soulever la question principale de recherche suivante :

- Dans quelle mesure une technique appropriée de rédaction d'exigences du logiciel pourrait-elle faciliter une meilleure automatisation de la mesure de la taille fonctionnelle des logiciels ?

Pour répondre à la question principale de cette recherche, on a relevé les questions

de recherche secondaires suivantes :

- Pourquoi les méthodes de rédaction d'exigences du logiciel existantes ne facilitent-elles pas, au point de vue pratique, l'automatisation de la mesure de la taille fonctionnelle des logiciels ?
- Quels seraient les avantages d'automatiser le processus de mesure de la taille fonctionnelle des logiciels, à partir des spécifications écrites sous forme de triplets ?
- Dans quelle mesure la spécification des exigences par triplets pourrait servir de méthodologie pour les concepteurs dans le domaine de développement des exigences logicielles ?

1.4 Hypothèses de recherche

L'hypothèse principale sur laquelle se base notre projet de recherche est que la spécification des exigences logicielles par triplets permet d'améliorer les techniques de rédaction d'exigences du logiciel, en vue de favoriser l'automatisation du processus de mesure de la taille fonctionnelle des logiciels. Cette hypothèse principale est divisée en deux hypothèses secondaires :

Hypothèse H1 : Les exigences logicielles écrites sous forme de structure de triplets permettent d'automatiser le processus de mesure de la taille fonctionnelle des logiciels.

Hypothèse H2 : La spécification par triplets peut servir de méthodologie pour les analystes et concepteurs dans le développement des exigences logicielles

En effet, conscient de la problématique relative à l'automatisation de la MTF des logiciels, en utilisant les spécifications fonctionnelles, l'on se donne pour objectif de développer une passerelle de spécification d'exigences du logiciel avec les

techniques de rédaction d'exigences du logiciel existantes afin de faciliter l'automatisation de la MTF des logiciels. Cette passerelle de spécification d'exigences du logiciel est une technique de rédaction d'exigences qui pourrait permettre l'automatisation de la MTF des logiciels à partir des spécifications exécutables. Elle découle des Cas d'utilisation (*Use Cases*), des récits d'utilisateurs (*User Stories*) ou de toute autre forme textuelle d'exigences logicielles. On a développé aussi un outil d'automatisation de la mesure de la taille fonctionnelle des logiciels, selon la méthode COSMIC ISO 19761. Le développement de cet outil permet de valider l'approche par triplets proposée pour la rédaction d'exigences du logiciel.

1.5 Objectifs de recherche

L'objectif général de notre recherche est de :

Concevoir et de mettre en œuvre une nouvelle technique de rédaction d'exigences du logiciel à partir d'une approche par triplets, qui favorise l'automatisation de la MTF des logiciels.

Ainsi, cette recherche contribue à améliorer les techniques de rédaction du logiciel à partir d'une approche par triplets. En d'autres termes, nous avons développé un modèle de triplets et un outil d'automatisation du processus de mesure de la TF des logiciels. Cette technique fournit des indications simples et efficaces pour faciliter l'automatisation de la TF des logiciels. Elle est éprouvée, testée et validée par la conception d'un nouvel outil d'automatisation de la MTF des logiciels, conformément à la méthode COSMIC ISO 19761.

1.6 Plan de la thèse

Nous présentons au chapitre I une introduction de la mesure de la taille fonctionnelle des logiciels. Ensuite, nous décrivons les problématiques liées au processus d'automatisation de la MTF des logiciels, ainsi que les questions, hypothèses et objectifs de la recherche. Le chapitre II présente une revue de littérature portant essentiellement sur la MTF des logiciels, les différentes méthodes de mesure et outils d'automatisation de la TF des logiciels. Par la suite, nous passons en revue les techniques, outils et méthodes de rédaction d'exigences du logiciel, ainsi que leurs limitations. Le chapitre III présente le cadre conceptuel et le volet cognitif de la thèse, particulièrement les logiques de description et la logique des prédicats du 1er ordre. Ainsi, nous décrivons le rôle de la mémoire épisodique, sémantique et procédurale dans cette thèse. Dans le chapitre IV, nous présentons la méthodologie de la recherche adoptée dans le cadre de cette thèse. Dans le chapitre V, nous évaluons les différentes approches, méthodes et techniques de rédaction d'exigences du logiciel. Par la suite, nous décrivons notre position par rapport à ces méthodes et proposons notre approche. En outre, nous décrivons au chapitre V notre approche de spécification et de rédaction d'exigences proposée. Cette approche consiste à spécifier et rédiger les exigences logicielles sous forme de structure de triplets, en vue d'automatiser le processus de mesure de la taille fonctionnelle des logiciels. Dans le chapitre VI, nous présentons l'outil d'automatisation développé, qui permet d'extraire des triplets (sujet, prédicat, objet) à partir des documents de spécifications d'exigences logicielles écrites en langage naturel. Par la suite, nous présentons le module d'obtention de la taille fonctionnelle des logiciels à partir des exigences écrites sous forme de structure de triplets. Dans le chapitre VII, nous présentons les résultats de mesure de la taille fonctionnelle des logiciels de dix (10) projets, ainsi que les écarts de mesurage manuel et automatisé observés. Finalement, le document se termine par une conclusion globale, en

rappelant la contribution informatique et cognitive de la thèse et la présentation des perspectives et limitations.

CHAPITRE II

REVUE DE LA LITTÉRATURE

2.1 Introduction

La mesure du logiciel est un sujet de recherche dont les premiers travaux remontent aux années 1970. Elle joue un rôle important dans le domaine du génie logiciel et elle est utilisée pour surveiller certains paramètres de projets de développement (c'est-à-dire la portée, la durée relative, les coûts et l'effort) et comme intrant à l'estimation des coûts, de l'effort et de la durée d'un projet logiciel (Abran, 2015). D'après la littérature, plusieurs méthodes de mesurage des logiciels ont été proposées, toutefois, l'automatisation du processus de la mesure de la taille fonctionnelle des logiciels reste encore peu développée. Le but de cette recherche est de concevoir et de mettre en œuvre une nouvelle technique de rédaction d'exigences du logiciel favorisant l'automatisation du processus de la mesure de la taille fonctionnelle des logiciels en fonction d'une approche par triplets. Donc, dans le cadre de ce chapitre, nous ferons ressortir les éléments pertinents de la recherche en lien avec cette thématique. Dans les sections 2.2, 2.3 et 2.4, nous abordons les notions de mesure, la mesure de la taille fonctionnelle et les méthodes de mesurages de logiciels. Dans la section 2.5, nous présentons les outils d'automatisation de la mesure de la taille fonctionnelle des logiciels, ainsi que les forces et faiblesses de ces outils en relation avec notre problématique. Dans la section 2.6, nous présentons le

modèle d'estimation avec des points de fonction. Nous présentons dans la section 2.7, les principales techniques et méthodes de rédaction d'exigences du logiciel. Dans la section 2.8, nous introduisons les outils de tests fonctionnels pour la rédaction des exigences logicielles. Finalement, nous présentons dans la section 2.9, les limitations des techniques de rédaction d'exigences du logiciel et une synthèse résumant ces sections.

2.2 Mesures du logiciel

L'estimation des coûts et de l'effort dans les projets logiciels revêt aujourd'hui une assez grande importance, qu'il s'agisse de projets de développement ou de maintenance des logiciels. Il est nécessaire, en termes de ressources humaines, matérielles et financières, d'avoir une idée de l'effort requis pour la réalisation du projet. La mesure, qu'il s'agisse de l'estimation du coût du projet, de son effort et des activités de productivité et de qualité liées au développement et à la maintenance des produits logiciels, reste et demeure une activité incontournable (Jones, 2007)(Abran, 2015). Les gestionnaires reconnaissent l'importance de la mesure de la taille des logiciels au sein des organisations (Galorath et Evans, 2006)(Abran, 2015)(Fenton et Bieman, 2019). Par définition, la mesure du logiciel désigne un processus qui permet d'attribuer une valeur numérique à un produit logiciel. En génie logiciel, les mesures sont utilisées pour contrôler ensuite la qualité du produit logiciel et mieux gérer les projets de développement de logiciels. En d'autres termes, les mesures du logiciel constituent les paramètres fondamentaux pour estimer l'effort de développement ou de maintenance du logiciel, et pour contrôler ensuite le processus de développement du logiciel ainsi que le budget et la portée. La mesure du logiciel se définit dans le glossaire normalisé d'IEEE comme « *a quantitative assessment of the degree to which a software product or process has an attribute* » (IEEE standard board, 2009). (Fenton et Bieman, 2019), une

mesure du logiciel est un processus par lequel les symboles sont attribués aux propriétés ou aux attributs des entités du monde réel. Une mesure du logiciel identifie en général l'entité et l'attribut que l'on veut mesurer. Selon (Fenton et Bieman, 2019), l'entité d'une mesure du logiciel peut être un processus, c'est-à-dire l'ensemble des composants du cycle de vie du logiciel; un produit, c'est-à-dire les extrants ou résultats finaux du processus; les ressources, c'est-à-dire le temps et le budget alloués pour les processus. Bon nombre de chercheurs affirme que la mesure de la taille du logiciel est l'un des paramètres les plus importants en matière d'estimation du coût ou de l'effort de projets (Clark, 1997)(Laird et Brennan, 2006)(Abran, 2015). Le processus de mesure du logiciel est défini dans la norme ISO/IEC 15939, en termes de processus, d'activités et de tâches, et est mis en œuvre par la méthode Practical Software Measurement (PSM). La norme ISO/IEC 15939 propose un processus de mesure applicable aux disciplines de gestion et aux produits logiciels. Ce processus de mesure se décrit à l'aide d'un modèle d'information et fournit des définitions des termes utilisés dans l'industrie du génie logiciel (ISO/IEC 15939). En effet, le processus de mesure du logiciel peut être divisé en deux types, qui sont notamment la mesure de la taille fonctionnelle (MTF) et celle de la taille physique. Dans le cadre de cette recherche, nous nous concentrons spécifiquement sur la mesure de la taille fonctionnelle (MTF) des logiciels.

2.3 Mesure de la taille fonctionnelle

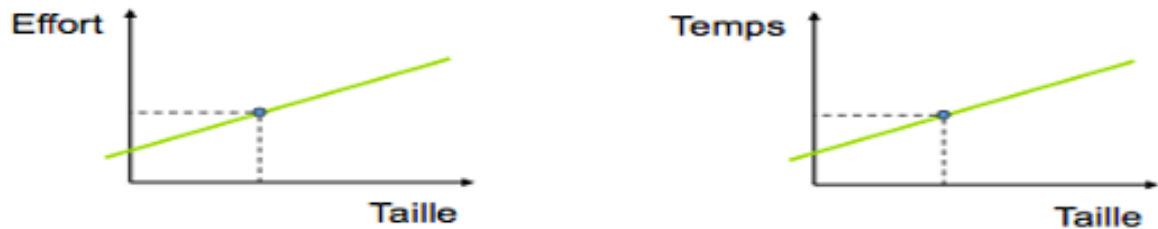
La mesure de la taille fonctionnelle (MTF) des logiciels peut s'avérer une tâche difficile à entreprendre. Le logiciel étant un produit intellectuel et un objet abstrait, la mesure de sa TF est une chose non évidente (Bevo, 2005)(Singh, 2017). La MTF des logiciels s'utilise pour estimer le coût des projets de développement de logiciels, effectuer des analyses de productivité et de qualité, ainsi que pour le

suivi des changements fonctionnels tout au long du cycle de vie du logiciel (Laird et Brennan, 2006)(Abran, 2015). Le coût des projets de développement de logiciels revêt un caractère important pour les gestionnaires de projets (Huijgens *et al.*, 2017). Dans l'industrie du génie logiciel, la TF des logiciels est exprimée en points de fonction, tandis que la taille physique des logiciels est exprimée en lignes de code (Abran, 2010). Le processus de mesure de la taille des logiciels basé sur le nombre de lignes de code reste à ce jour le plus populaire. L'approche de mesure basée sur les lignes de code vise à estimer le volume de code source associé à une application logicielle et la mesure se fait à l'aide d'un compteur automatique de lignes de code. L'approche par points de fonction (PF) a été introduite comme alternative aux lignes de code, puisque l'approche de mesure basée sur les lignes de code s'avère peu pertinente pour une bonne estimation de coût ou d'effort, parce qu'elle est connue trop tard pour servir de base aux estimations. Aussi, étant donné que les applications modernes sont souvent développées avec différents langages de programmation, une mesure en lignes de code devient incohérente. La MTF des logiciels est basée sur des fonctionnalités. L'idée est de déterminer la quantité de fonctionnalités fournies à un utilisateur pour un produit logiciel donné, sans égard aux technologies qui seront utilisées pour son développement.

2.4 Corrélation entre la taille fonctionnelle et l'effort

L'un des atouts majeurs de la MTF des logiciels à partir des spécifications logicielles réside dans le fait que la mesure peut se faire tôt dans le processus de développement du logiciel. Cela permettra au gestionnaire de projet d'estimer l'effort et le coût de réalisation de produits logiciels. Selon (Abran, 2015), l'avantage de la taille fonctionnelle est qu'elle permet de prédire la quantité de logiciel qu'une équipe devrait livrer en moyenne par bloc de temps, ce qui rend prévisible la date de livraison. En effet, il existe une forte corrélation entre la taille fonctionnelle,

l'effort et le temps de livraison de logiciels. La mesure de la taille fonctionnelle permet aux gestionnaires de réaliser une analyse de la performance de l'organisation ou de l'équipe de développement de logiciels, tels que le coût de développement, la productivité et le taux de livraison (Abran, 2015)(Trudel et Boisvert, 2011). En effet, selon une étude menée par(Commeyne *et al.*, 2016), la taille fonctionnelle permet de produire un référentiel de productivité. Ils ont estimé l'effort de seize (16) tâches d'un projet à partir des points de fonction COSMIC, qui représente entre autre la taille fonctionnelle, ainsi qu'à partir des « *User Point* ». Les résultats de cette étude ont montré que la mesure en point de fonction COSMIC ou mesure de la taille fonctionnelle fournit des preuves objectives pour de meilleures estimations de l'effort, du coût de développement et de la performance de l'équipe de réalisation de produits logiciels. La figure 2.1 montre la corrélation qui existe entre la taille fonctionnelle et l'effort, ainsi que le temps de livraison et la taille fonctionnelle.



- **Coût = Effort/Taille --->heures/PF**
- **Productivité = Taille /Effort ---> PF/mois-personne**
- **Taux de livraison= Taille / Temps ---> PF/mois**

FIGURE 2.1 Corrélation entre la taille et l'effort.

2.5 Méthodes de mesures de la taille fonctionnelle des logiciels

Plusieurs méthodes de MTF des logiciels ont été proposées dans la littérature. Selon une étude menée par (Albrecht, 1979), la fonctionnalité du logiciel est l'une des caractéristiques mesurables dans le processus de MTF des logiciels. La MTF des logiciels nécessite, entre autres, que les mesureurs aient une expertise propre pour chacune des méthodes de mesure (Singh, 2017). Il existe différents outils et méthodes de mesure de la TF des logiciels approuvés par l'Organisation internationale de normalisation (ISO) tels que COSMIC, IFPUG, NESMA, Mark-II et FiSMA. Les méthodes de MTF des logiciels, dites méthodes de première génération, sont apparues dès 1979. Elles sont basées sur les modèles de données (entités et leurs attributs). L'un des inconvénients majeurs de ces méthodes est le fait que la taille obtenue dépend fortement des outils technologiques et des langages de programmation utilisés (Soubra, 2011). La méthode COSMIC, la seule méthode de deuxième génération, se base sur les artefacts produits dans le processus de développement de logiciel, à l'instar de ceux produits durant la phase d'analyse et de conception, comme par exemple les exigences fonctionnelles des utilisateurs.

2.5.1 Méthode IFPUG

La méthode IFPUG est la première méthode de MTF des logiciels ayant été publiée dans la littérature de recherche. IFPUG, présentement connue sous la norme ISO/IEC 20926 : 2009, décrit de façon claire et détaillée les calculs de points de fonction. Cette norme fournit une base pour vérifier que les calculs de points de fonction sont cohérents et permet aussi de faire le comptage des points de fonction des exigences des utilisateurs fonctionnels à partir des livrables et des méthodologies de développement de logiciels. Il est à souligner que la norme ISO/IEC 20926 : 2009 présente un cadre de référence qui permet l'automatisation du calcul

des points de fonction. Toutefois, aucun outil fondé sur la méthode IFPUG n'a été encore publié. Un article de (Diab *et al.*, 2002) présente une formalisation des règles d'application de la méthode IFPUG pour mesurer des logiciels spécifiés avec la méthode de spécification formelle B. D'après ces chercheurs, cette formalisation pourrait s'utiliser pour l'automatisation de la méthode IFPUG à partir des spécifications conçues avec la méthode B. Étant donné certaines imperfections existant dans la définition de la méthode IFPUG, l'étude menée par Diab et al. (2002) n'a pas été concluante. Il est à noter que la méthode B est une méthode de spécification formelle capable de retranscrire de manière rigoureuse les exigences d'un cahier des charges, au moyen des preuves mathématiques. Une particularité de la méthode IFPUG est qu'elle est applicable à la mesure des applications de gestion de l'information (Management Information System). Cependant, elle n'a pas été conçue pour mesurer des applications de types embarquées ou temps-réel.

2.5.2 Méthode COSMIC –ISO 19761 : 2019

La méthode COSMIC –ISO 19761 : 2019 est une méthode de MTF des logiciels approuvée par l'organisation internationale de normalisation (Abran *et al.*, 2019). Elle a été conçue pour être appliquée aux fonctionnalités de logiciels d'application d'affaires, de logiciels de type temps réel et de logiciels hybrides (affaires et temps réel). Les logiciels d'application d'affaires sont des systèmes d'information de gestion, tels que les logiciels des domaines bancaires, des assurances, du personnel, des achats et de la comptabilité. Les logiciels de type temps réel sont des applications ayant pour tâche de suivre le contrôle d'événements en temps réel, tels que des logiciels embarqués, des logiciels d'échanges téléphoniques. Les logiciels hybrides sont à la fois des applications d'affaires et des logiciels en temps réel, par exemple, des logiciels de réservations de vols ou d'hôtels en temps réel. La méthode COSMIC a évolué depuis sa création en 1999, et la plus récente version

du manuel de mesurage est la v5.0. Il existe dans la littérature divers outils ou prototypes basés sur la méthode COSMIC. À titre d'exemple, Diab et al. (2002) proposent un outil appelé McRose qui permet de mesurer la TF des logiciels, telle que définie par la méthode COSMIC, à partir de règles formelles (équations mathématiques). Marin et al. (2008) présentent une procédure de mesurage de la TF d'application orientée objets générés à partir de modèles conceptuels. Lavazza et al. (2018) proposent de mesurer la TF à partir des métriques orientées objet basées sur des modèles d'exigences UML.

En résumé, l'objectif de la méthode de mesurage COSMIC est d'appliquer un ensemble de modèles, de principes, de règles et de processus aux fonctionnalités afin de déterminer la TF du logiciel. COSMIC mesure donc des mouvements de données appliqués à chaque groupe de données manipulé par chacune des fonctionnalités ou des processus fonctionnels (Trudel, 2012). Il existe quatre (4) types de mouvements de données et qui sont respectivement : Entrées (E), Sorties (S), Lecture (L) et écriture (C)¹. Un mouvement de données de type Entrée (E) déplace un groupe de données depuis un utilisateur fonctionnel dans le processus fonctionnel qui le requiert. Quant au mouvement de données de type Sortie (S), il déplace un groupe de données d'un processus fonctionnel vers un utilisateur fonctionnel. Un mouvement de données de type Lecture (L) déplace un groupe de données depuis le stockage persistant vers l'utilisateur fonctionnel. Par contre, un mouvement de données de type écriture (C) déplace un groupe de données du processus fonctionnel vers le stockage persistant. Le stockage persistant peut prendre plusieurs formes, telles que bases de données, fichiers indexés, fichiers de données structurées (ex. Json, XML), fichiers plats (ex. CSV, ASCII), blocs de mémoire pré-alloués. La figure 2.2 donne une vue d'ensemble de la méthode COSMIC.

1. <https://cosmic-sizing.org/wp-content/uploads/2020/09/Manuel-de-mesurage-COSMIC-v5-Partie-1-FR.pdf>

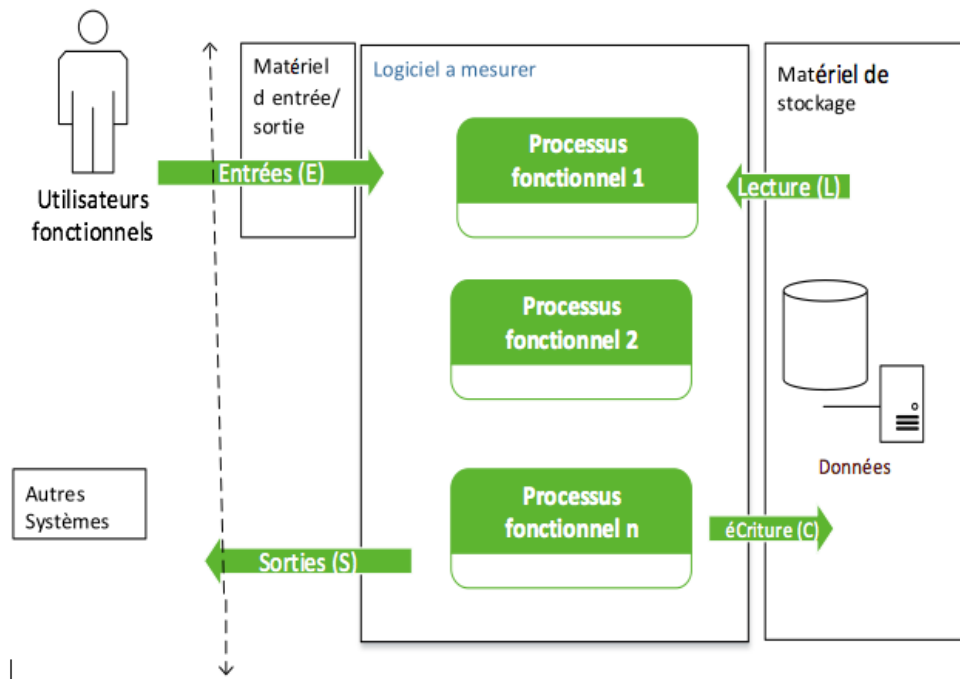


FIGURE 2.2 Vue d'ensemble de la méthode COSMIC.

2.5.3 Compatibilité entre IFPUG et COSMIC

Plusieurs chercheurs ont tenté d'étudier la compatibilité entre les méthodes COSMIC et IFPUG, notamment pour faciliter la conversion vers COSMIC de données IFPUG existantes et pour permettre l'étalonnage de performance de projets dont certains auraient été mesurés en IFPUG et d'autres en COSMIC. D'après un article publié par Abran et al. (2003), la compatibilité entre les deux méthodes est utile pour passer de IFPUG à COSMIC. En effet, la compatibilité entre deux méthodes est fondamentale pour pouvoir additionner les résultats des deux méthodes dans une seule mesure de taille fonctionnelle (Abualkishik, 2019) (Soubra, 2011). La compatibilité doit donc être analysée, d'une part, entre les méthodes de mesure et d'autre part entre les processus de mesure pour les deux méthodes (Abualkishik

et Lavazza, 2018)(Soubra, 2011). La similitude ou compatibilité entre IFPUG et COSMIC vient du fait que ces deux méthodes modélisent le logiciel comme une hiérarchie de fonctions et qu'un poids direct en termes de « points » peut être attribué à ces fonctions (Abran, 2010)(Abualkishik et Lavazza, 2018). La différence entre ces deux méthodes est que IFPUG mesure les processus élémentaires qui apportent des données à l'application logicielle. Tandis que les mouvements de données mesurés par COSMIC sont des composants d'un processus fonctionnel (Abran, 2010). De plus, la règle de comptage de points de fonction dans IFPUG est différente pour chaque type de processus (différentes règles de détermination de complexité). Tandis qu'on utilise la même règle de comptage de points de fonction en COSMIC. Ainsi, l'unité de mesure, contrairement dans la méthode IFPUG, est bien définie dans la méthode COSMIC (Abran, 2010). Quant à (Gencel et Bideau, 2012), ils soutiennent que la précision de la convertibilité entre IFPUG et COSMIC devient un problème. En outre, le type de conversion le plus couramment utilisé ne fournit pas des résultats précis. Par conséquent, il est donc important de tester plusieurs types de conversions afin d'obtenir des résultats précis.

2.5.4 Lacunes de IFPUG absentes dans COSMIC

Selon (Xunmei *et al.*, 2006), la méthode IFPUG FPA a été conçue avant la norme ISO 14143, alors que la méthode COSMIC a été conçue après. (Xunmei *et al.*, 2006) ont relaté que la méthode COSMIC a été conçue pour assurer la conformité avec la norme ISO 14143, tandis que la méthode IFPUG n'a pas été complètement adoptée par l'organisation internationale de normalisation. La norme ISO 14143-6 : 2006 présente un résumé des normes internationales en relation avec les mesures de la TF, en précisant les définitions et concepts pour les procédures de mesurage. (Xunmei *et al.*, 2006) ont essayé de montrer dans leur article que les domaines d'application pouvant se mesurer avec la méthode COSMIC sont nettement plus

larges que ceux couverts par la méthode IFPUG FPA. Compte tenu de ces lacunes, nous avons choisi COSMIC comme méthode de MTF pour notre recherche.

2.6 Automatisation de la mesure de la taille fonctionnelle des logiciels

L'automatisation de la MTF des logiciels joue un rôle important, face à la collecte manuelle qui consomme du temps et est sujette à l'erreur et à des biais (Coman *et al.*, 2009)(Singh, 2017). On entend par automatisation du mesurage, l'automatisation de la collecte des données, du calcul et de la production des graphiques d'analyse, en suivant les données stockées dans une ou plusieurs sources de données (Andersson, 2017). L'automatisation de la MTF des logiciels devient de nos jours une priorité pour les chercheurs et gestionnaires.

Plusieurs études et recherches ont été effectuées sur les outils et prototypes permettant l'automatisation de la TF. D'après la littérature consultée, ces outils et prototypes ne favorisent pas nécessairement l'automatisation du processus de la MTF des logiciels à partir des spécifications fonctionnelles, car on recense davantage d'outils qui automatise la mesure à partir des autres artefacts tels que les diagrammes de séquences UML et le code source, etc. De ce fait, il y a un besoin de concevoir et construire des outils ou prototypes qui pourraient aider à automatiser le processus de MTF des logiciels à partir des spécifications. Toutefois, les mécanismes utilisés pour décrire les exigences du logiciel ne permettent pas l'automatisation de la TF des logiciels à partir des documents de spécifications du logiciel. Autrement dit, les méthodes de rédaction d'exigences du logiciel existantes ne facilitent pas, du point de vue pratique, l'automatisation de la TF des logiciels.

2.6.1 Outils d'automatisation de la mesure de la taille fonctionnelle des logiciels

La revue de la littérature de cette section se concentre sur les principaux prototypes, outils et procédures d'automatisation de la MTF des logiciels basés sur les méthodes COSMIC et IFPUG. L'automatisation de la MTF à partir de documents de spécifications logicielles reste jusqu'à présent dans une phase exploratoire. La plupart des outils développés ne permettent pas d'automatiser complètement le processus d'obtention de la TF des logiciels. En effet, plusieurs approches et outils ont été proposés par des chercheurs se spécialisant dans le domaine du processus de mesure, cependant, ces approches possèdent plusieurs limitations.

2.6.2 Automatisation à partir du processus unifié

(Azzouz et Abran, 2004) proposent un prototype qui permet d'automatiser la MTF de logiciels développés en utilisant le Rational Unified Process (RUP) pour les systèmes d'information de gestion (SIG). La conception de cet outil était basée sur une mise en correspondance directe entre la méthode COSMIC et les notations UML, notamment les diagrammes de séquence. L'idée était de permettre d'obtenir la TF une fois les spécifications logicielles complétées. Toutefois, cet outil nécessitait des interventions manuelles pour l'obtention des résultats. Ce prototype reste, jusqu'à présent, de type exploratoire et n'a pas été non plus soumis à des tests. En d'autres termes, il n'a pas été testé et validé pour s'assurer que les résultats fournis soient précis.

2.6.3 Automatisation à partir des diagrammes UML (diagrammes de Cas d'utilisation, diagrammes de séquence)

(Bévo *et al.*, 1999), dans l'objectif de favoriser l'automatisation du processus de mesure de la TF des logiciels, ont effectué une étude sur une mise en correspondance entre les concepts des diagrammes UML de la version 1.0 (cas d'utilisation, scénarios) et les concepts de la méthode COSMIC. L'idée de (Bévo *et al.*, 1999) était d'évaluer la possibilité de déterminer la TF des logiciels à partir des documents de spécification en UML, et à l'aide de la méthode de mesure COSMIC. Il est à souligner que cette étude concerne uniquement les systèmes d'information de gestion (SIG). Néanmoins, cette étude ne facilite pas le processus d'automatisation de la TF des logiciels, puisque le problème d'automatisation réside dans le fait que les documents de spécification du logiciel sont écrits en langage naturel. Donc il serait d'abord nécessaire d'améliorer les mécanismes de rédaction des documents de spécification du logiciel pour pouvoir faciliter l'automatisation de la TF des logiciels. (Levesque *et al.*, 2008) ont proposé d'utiliser les diagrammes de Cas d'utilisation et de séquence pour l'obtention de la TF des logiciels. L'idée de cette procédure consiste à rapprocher COSMIC avec UML. Cela sous-entend que les mouvements de données en COSMIC correspondraient aux messages du diagramme de séquence. Les processus fonctionnels correspondraient aux diagrammes de Cas d'utilisation et de séquence. Quant aux utilisateurs fonctionnels, ils correspondraient aux acteurs des Cas d'utilisation. Cependant, (Levesque *et al.*, 2008) ne proposent aucun outil pour supporter leur procédure.

(Barkallah, 2012) a proposé d'utiliser des modèles UML pour supporter la MTF des logiciels. L'objectif de cette recherche vise à étudier le processus d'automatisation de la MTF avec COSMIC, à partir des spécifications écrites à l'aide du langage UML (Barkallah, 2012). Cette recherche a montré que l'utilisation des modèles UML ne suffit pas pour faciliter l'automatisation de la taille fonctionnelle

des logiciels (Barkallah, 2012). Par conséquent, il a proposé de définir un profil UML spécifique à la méthode COSMIC qui pourrait rendre moins fastidieuse la tâche de mesurage, puisque ce profil spécifique permettra d'obtenir les informations nécessaires pour établir la mesure (Barkallah, 2012). Cependant, ce profil UML ne permet pas d'automatiser totalement la procédure de mesure de la TF avec COSMIC. Certains chercheurs ont suggéré de mesurer la TF des logiciels à l'aide des diagrammes de séquence (Sellami *et al.*, 2015)(Karim *et al.*, 2017). À titre d'exemple, (Sellami *et al.*, 2015) ont proposé une méthode de mesure basée sur la structure des diagrammes de séquences UML. Cette méthode vise à concevoir une mesure améliorée de la taille des diagrammes de séquences UML en prenant en compte la structure du diagramme de séquence. Les résultats de cette recherche ont montré que la taille des diagrammes de séquence peut être mesurée selon les perspectives fonctionnelles et structurelles (Sellami *et al.*, 2015). (Karim *et al.*, 2017) ont proposé un outil permettant d'exporter les diagrammes de séquence sous format XML (eXtensible Markup Language) en vue d'obtention de la TF. Toutefois, la principale limitation de cette approche vient du fait que les diagrammes de séquence ne sont jamais complets et ne couvrent que certaines fonctionnalités ou portions du logiciel.

2.6.4 Automatisation à partir des règles formelles (équations mathématiques)

(Diab *et al.*, 2002) fournissent un ensemble de règles formelles, sous forme d'équations mathématiques, permettant de mesurer la TF des logiciels, en utilisant une mise en correspondance basée sur la méthode COSMIC version 2.0. Cette procédure est mise en œuvre par un outil nommé McRose. En termes d'avantages, cet outil permet de simplifier et d'organiser le processus de MTF des logiciels, tout en favorisant la réduction des coûts de mesure. À noter que McRose a été le premier outil à adresser l'automatisation de la MTF, telle que définie par la mé-

thode COSMIC. Ainsi, il peut être intégré dans l’outil Rational Rose Real Time. Cependant, cet outil possède de nombreuses contraintes. Il a été expérimenté pour mesurer la TF d’un logiciel et les résultats fournis par l’outil ont été comparés avec ceux d’un expert COSMIC indépendant. Les tailles fonctionnelles obtenues ont été largement différentes. D’après (Diab *et al.*, 2005), la principale limitation de cet outil concerne le module effectuant l’analyse du code C++ pour estimer la mesure. Cette limitation réside dans le fait que ce module peut examiner uniquement certains concepts de base. De plus, les résultats de mesure présentés par l’outil n’ont pas été précis (Diab *et al.*, 2002)(Diab *et al.*, 2005).

2.6.5 Automatisation à partir des modèles conceptuels

(Marín *et al.*, 2008) proposent un prototype qui permet de mesurer la TF d’applications orientées-objets, basée sur la version 3.0 de la méthode COSMIC et qui peut s’appliquer aux modèles conceptuels. En fait, le prototype permet d’automatiser une bonne partie de la méthode COSMIC via la notation UML. Il offre aux chercheurs un aperçu global de l’état actuel des procédures de mesure de la taille fonctionnelle des logiciels, basée sur la méthode COSMIC. Ce prototype a été développé avec le langage de programmation Java et utilise les diagrammes de cas d’utilisation, les diagrammes de séquence tracés à partir de la description des cas d’utilisation (Use Case) et les modèles objets d’analyse. Toutefois, (Marín *et al.*, 2008) ne fournissent pas de chiffres pour les résultats de validation ainsi que pour les analyses des résultats.

2.6.6 Automatisation à partir des modèles entités-relations et d’un diagramme de flux de données

(Lamma *et al.*, 2004) présentent un outil de mesure de points de fonction à partir des exigences du logiciel exprimées sous forme de modèles entités-relations (mo-

dèle de données) et d'un diagramme de flux de données. Cet outil a été conçu pour mesurer des applications de gestion et ne peut pas s'appliquer à des systèmes embarqués. Ainsi, il existe présentement dans l'industrie peu de processus de développement de logiciels qui documentent les modèles de données sous forme d'entités-relations et qui documentent leurs modèles de flux de données.

2.6.7 Automatisation à partir du code source

D'autres chercheurs ont proposé des outils d'automatisation à partir du code source. (Chamkha *et al.*, 2018) ont proposé un outil d'automatisation permettant de déterminer la mesure de la TF des logiciels à l'aide des interfaces graphiques Swing. Cet outil fournit des résultats identiques par rapport aux résultats de mesure manuelle. (Tarhan et Sağ, 2018) ont proposé une nouvelle version de l'outil dénommée « COSMIC Solver » en vue d'automatiser le processus de mesure de la TF des logiciels à partir du code source Java. Cet outil fournit une précision avec une moyenne de 77% par rapport aux résultats de mesurage manuel. (Sahab et Trudel, 2020), ils ont proposé un outil qui permet de mesurer la TF d'applications Web Java à partir du code source en utilisant le cadre Spring WEB MVC. La précision est supérieure à 96% et s'est avérée plus précise que la mesure manuelle parce que des exigences étaient incomplètes alors que le code était complet. Cet outil a été testé et validé avec trois (3) applications Open Source (Sahab et Trudel, 2020). Par ailleurs, les mesures basées sur le code source ou nombre des lignes de code s'obtiennent après la phase de développement de logiciels. Il est donc fondamental de concevoir des outils qui pourraient permettre de mesurer la TF des logiciels dès le début du cycle de vie de développement à partir des spécifications écrites en langage naturel.

2.6.8 Automatisation de la mesure avec IFPUG

D'autres chercheurs proposent des outils et approches d'automatisation de la MTF des logiciels, telle que définie par la méthode IFPUG. Par exemple, (Uemura *et al.*, 1999) ont développé un outil de mesure de points de fonction utilisant en entrée les spécifications de conception. Le langage de programmation Visual C++ sous Windows 98 a été utilisé pour la conception et mise en œuvre de cet outil. Les entrées sont respectivement des diagrammes de séquences et de classes, tandis que les sorties représentent les résultats de la mesure. Cet outil a été testé avec des données réelles et les résultats présentés par l'outil ont été comparés avec ceux d'un expert humain. Toutefois, les chercheurs ne fournissent aucune donnée qui pourrait montrer que les résultats obtenus de l'outil se rapprochent de ceux des experts humains.

2.6.9 Cadre référentiel pour les outils d'automatisation de la mesure de la taille fonctionnelle des logiciels

La conception des outils d'automatisation de la TF en fonction des normes ou des cadres référentiels reconnus est une priorité pour les chercheurs et gestionnaires. (Abran et Paton, 1997) ont identifié un ensemble de caractéristiques souhaitables pour les outils d'automatisation de la mesure de la TF des logiciels. Ces principales caractéristiques se résument ainsi :

- Les outils d'automatisation doivent être associés à des normes ou à des standards reconnus.
- L'architecture des outils doit être ouverte, cela sous-entend que les outils doivent offrir, par exemple, la possibilité d'interagir avec les outils de rédaction d'exigences du logiciel.
- Les outils doivent fournir une présentation des résultats de mesure facilitant

les analyses.

- Les outils doivent pouvoir gérer des incertitudes liées à l’identification des concepts dues parfois au manque de qualité des documents de spécification du logiciel.
- Les outils doivent être multiplateformes.

Dans le cadre de cette recherche, nous avons développé un outil d’automatisation selon ce cadre référentiel afin de valider l’approche par triplets proposée pour la rédaction d’exigences du logiciel.

2.6.10 Synthèse

En somme, il n’existe pas à ce jour pas de techniques de rédaction du logiciel qui pourraient faciliter l’automatisation du processus de mesure de la TF des logiciels. Plusieurs chercheurs proposent des approches, prototypes et outils. Toutefois, ces propositions possèdent de nombreuses contraintes, faiblesses et limitations. Certains outils et prototypes développés nécessitent des interventions manuelles pour l’obtention des résultats du processus de mesure. De ce fait, il y a un besoin de concevoir et construire des nouveaux outils à partir des normes, qui pourraient faciliter l’automatisation de la TF des logiciels à partir des documents de spécifications. Cependant, les mécanismes de rédaction d’exigences du logiciel utilisés dans l’industrie ne favorisent pas l’automatisation du processus de MTF des logiciels à partir des documents de spécification. Par conséquent, il est souhaitable, pour pallier ces difficultés, d’améliorer les méthodes et techniques de rédaction du logiciel.

2.7 Mesure des points de fonction

Les Points de Fonction (PF) mesurent la taille d'un logiciel en quantifiant les fonctionnalités offertes aux utilisateurs, en se basant respectivement sur la modélisation logique des données et les spécifications fonctionnelles du logiciel (Abran, 2010). La première version de cette méthode a été proposée par (Albrecht, 1979) et aide à prévoir la taille d'un projet, l'effort de développement ainsi que la productivité du projet de développement. Les Points de Fonction, communément appelés « analyse des points de fonction » sont connus en anglais sous le terme « *Function Points Analysis* » (FPA). D'après (Albrecht, 1979), les fonctions du logiciel s'identifient à partir des spécifications du logiciel. Dans le cadre d'une telle approche, chaque fonction est évaluée pour sa complexité et un poids lui est assigné. La somme totale des poids est connue comme le nombre de points de fonction et représente la TF du logiciel. Les points de fonction ont été développés avec l'objectif de répondre à l'analyse de productivité et d'estimation. L'idée visée par (Albrecht, 1979) était de définir une méthode qui permet d'évaluer la productivité de 24 projets de développement réalisés chez IBM de 1974 à 1978. De nombreux auteurs ont effectué une étude approfondie sur la méthode de points de fonction (St-Pierre *et al.*, 1997) (Dumke et Abran, 2016)(Dumke et Abran, 2016) (Abran *et al.*, 2016). En effet, (Albrecht, 1979) relate que les points de fonction mesurent une application logicielle en quantifiant la fonctionnalité du traitement de l'information. Ce traitement d'information est associé avec les principaux intrants des données externes, les sorties et les types de fichiers (Albrecht, 1979). Une version améliorée de la méthode de points de fonction a été publiée en 1983 et est devenue à cette époque la méthode de facto utilisée pour mesurer les points de fonction, soit la méthode IFPUG. Il est à mentionner que la méthode des points de fonction est réputée pour être applicable aux systèmes d'information de gestion (SIG). L'idée principale de cette méthode est que la TF du logiciel se quantifie à partir de

cinq types d'éléments de base qui sont respectivement : entrées (*external inputs*) , sorties (*external outputs*), requêtes (*external queries*) , fichiers externes (*external interface files*) et fichiers internes (*internal logical files*) . Dans le cadre d'une telle approche, on identifie les items du logiciel. À chaque item, on assigne subjectivement un poids qui correspond au niveau de sa complexité sur une échelle de trois niveaux (simple, moyenne, complexe). La somme des poids de tous les items est appelée en anglais *Unadjusted Function Point* (UFP). Les points de fonction non ajustés représentent la taille brute du logiciel. (Albrecht et Gaffney, 1983) a établi dans son article un modèle de prédiction d'effort à partir des points de fonction du logiciel. Il est à souligner que la méthode proposée par Albrecht permet de quantifier la TF du logiciel indépendamment du langage de programmation utilisé. Cela sous-entend que les points de fonction sont interprétés du point de vue de l'utilisateur. Le tableau 2.1 présente les types des items de base, ainsi que les poids associés à chacun des niveaux de complexité (Albrecht et Gaffney, 1983).

Item	Poids		
	Simple	Moyen	Complexe
Entrées	3	4	6
Sorties	4	5	7
Requêtes	3	4	6
Fichiers externes	5	7	10
Fichiers internes	7	10	15

TABLEAU 2.1 Poids de la taille fonctionnelle FPA par niveau.

À la lumière des données du tableau 2.1, on constate que si on devait mesurer une fonctionnalité très complexe, le nombre de points maximum sera rapidement atteint et la taille fonctionnelle ne serait pas adéquatement reflétée. C'est là une des lacunes de IFPUG que COSMIC n'a pas, car il n'y a pas de nombre maximum.

2.7.1 Utilisation des points de fonction dans l'industrie

Les points de fonction sont utilisés dans l'industrie comme un indice indiquant la complexité fonctionnelle du logiciel. Dans son article, Garmus a présenté un bilan de l'utilisation des points de fonction dans l'industrie (Garmus et Herron, 1996). Il a souligné que la productivité indique l'efficacité du programmeur, l'efficacité de l'équipe de développement et de la production. Le tableau 2.2 présente un exemple d'utilisation des points de fonction dans l'industrie (Garmus et Herron, 1996).

Types	Mesures	Description
Productivité	Taille fonctionnelle / effort	Points de fonction par heure-personne, ce qui permet de prédire par exemple la quantité de logiciel qu'une équipe devrait livrer en moyenne par bloc d'effort.
Capacité de développement	Taille fonctionnelle / durée	Points de fonction par mois de calendrier, ce qui permet de prédire dans combien de temps une quantité donnée de taille fonctionnelle devrait pouvoir être livrée par l'équipe.
Qualité	Nombre de défauts / Points de fonction	Nombre de défauts par point de fonction, ce qui permet de comparer les densités relatives de défauts de différents logiciels.
Affaires	Coût / taille fonctionnelle	Coût unitaire d'un point de fonction, ce qui permet par exemple de facturer le client d'un développement logiciel en dollars par point de fonction, ou de connaître et d'améliorer son coût unitaire en optimisant le processus de développement.

TABLEAU 2.2 Poids de la taille fonctionnelle FPA par niveau.

2.7.2 Limites de la méthode de point de fonction

La méthode des points de fonction possède plusieurs limitations. Tout d'abord, elle a été conçue spécifiquement pour les systèmes d'information de gestion (SIG) et représente une méthode de facto pour mesurer la taille des logiciels de type

SIG. Elle n'est donc pas applicable aux logiciels de type temps réel et aux logiciels hybrides (Fenton et Bieman, 2019). (Kitchenham et Stell, 1997) estiment que la taille fonctionnelle du logiciel mesurée par les points de fonction, combinant le *Unadjusted Function Point* (UFP) et combinant le *Technical Complexity Factor* (TCF) ne permet pas d'améliorer considérablement la mesure de taille pour prévoir l'effort. De plus, Symons (1998) montre que la méthode d'Albrecht fait un double comptage de complexité en utilisant les facteurs UFP (Unadjusted Function Point) et TCF (Technical Complexity Factor).

2.8 Techniques et méthodes de rédaction d'exigences du logiciel

L'automatisation du processus de la MTF des logiciels dépend nécessairement des outils, techniques et méthodes utilisés pour rédiger les spécifications. Plusieurs techniques et méthodes d'écriture d'exigences ont été proposées dans la littérature et ces techniques sont utilisées dans l'industrie. Par exemple, (Jacobson, 2003) propose la technique des « *Use Cases* » pour rédiger les exigences du logiciel. (Beck et West, 2014) pour leur part, proposent les (Jacobson, 2003) propose la technique des « *User Stories* » comme technique de rédaction d'exigences du logiciel. Ces techniques sont des textes largement utilisés pour identifier et consigner les besoins fonctionnels du logiciel. D'autres auteurs proposent la méthode « ATDD », une méthode associée plus particulièrement à l'approche Agile, souvent appliquée avec les méthodes Scrum et *Extreme Programming* (XP). Dans le cadre de cette section, nous avons passé en revue les principales techniques et méthodes de rédaction des exigences du logiciel. Par la suite, nous avons procédé à l'évaluation de ces techniques pour voir dans quelle mesure celles-ci pourraient favoriser l'automatisation du processus de mesure de la taille fonctionnelle des logiciels à partir des documents de spécifications.

2.8.1 Les Cas d'utilisation (*Use Cases*)

Les Cas d'utilisation (*Use Cases*) sont des descriptions textuelles, utilisées pour documenter les spécifications logicielles. Ils influencent l'ensemble des artefacts liés au processus de développement du logiciel, notamment l'analyse, la conception, l'implémentation et les tests. Les *Use Cases* racontent sous forme de texte la façon dont un acteur ou un utilisateur utilisera un système logiciel pour atteindre un but. La finalité des *Use Cases* est d'identifier, de décrire et de documenter les fonctionnalités du logiciel en précisant de quelle manière le système pourra s'utiliser pour permettre aux différentes parties prenantes et aux utilisateurs d'atteindre leurs objectifs. À noter que les *Use Cases* sont exprimés dans un langage naturel ou techniquement neutre, sans préciser aucune terminologie technique. Plusieurs formats ont été proposés pour la description des *Use Cases*, ils sont notamment les formats « abrégé », « informel » et « détaillé » :

- Format « abrégé » : c'est un résumé succinct présentant généralement le scénario de succès dans un paragraphe. Par définition, un scénario est une instance d'un cas d'utilisation, une histoire particulière décrivant la façon dont on utilise un système.
- Format « informel » : on décrit les différents scénarios dans plusieurs paragraphes.
- Format « détaillé » : on décrit avec beaucoup de détails les *Use Cases*. Les *Use Cases* détaillés sont alors des descriptions approfondies et structurées.

(Cockburn, 2001) propose plusieurs principes de rédaction des *Use Cases*, en fonction de la phase du processus de développement du logiciel. Ils sont respectivement le style « essentiel », le style « concret » et le style « en boîte noire ». Dans le cadre du principe en style « essentiel », on rédige le scénario d'un cas d'utilisation au niveau des intentions des utilisateurs et des responsabilités du sys-

tème, en laissant de côté l'interface utilisateur (Cockburn, 2001). Par contraste, la description d'un cas d'utilisation (*Use Cases*) en style « concret » nécessite qu'on mette l'accent sur les interactions avec les utilisateurs. Les cas d'utilisation « en boîte noire » décrivent le système en termes de responsabilité. Dans une telle approche, on définit le comportement ou les besoins fonctionnels du système, sans préciser comment il le fera.

2.8.2 Les récits d'utilisation (*User Stories*)

Les récits d'utilisation (*User Stories*) consistent en quelques lignes de textes qui décrivent une fonctionnalité qu'un logiciel doit offrir pour permettre à un acteur ou un utilisateur donné de réaliser un objectif précis. Les User Stories sont généralement écrits dans un langage naturel et ne comportent pas de termes techniques. L'un des intérêts majeurs de cette approche est qu'elle est centrée sur l'utilisateur du système. Cela sous-entend que le besoin utilisateur est le point de départ de la fonctionnalité. Un format de description d'un *User Story* souvent utilisé est le schéma suivant :

<p>En tant que « rôle d'utilisateur » Je veux « cette fonctionnalité ou caractéristique » Afin de « bénéfice ou valeur d'affaire, raison d'affaires »</p>
--

FIGURE 2.3 Format de description d'un User Story

Les récits d'utilisation sont fréquemment utilisés dans les environnements agiles et Lean (Scrum, Extreme Programming, Kanban). Ils décrivent des fonctionnalités pouvant être développées rapidement, c'est-à-dire, à l'intérieur d'un sprint, soit d'un court cycle de développement, tel qu'on les retrouve dans les méthodes agiles. Les *User Stories*, tout comme les *Use Cases*, sont rédigés dans un langage

naturel. Les *User Stories* sont susceptibles d'être modifiés au cours du processus de développement de logiciel, afin de pouvoir mieux répondre à la valeur d'affaires du *Product Owner* (client) pouvant évoluer dans le temps. Les *User Stories* sont écrites dans un langage naturel afin d'être compris par toutes les parties prenantes du projet de développement du logiciel, notamment le *Product Owner* (client ou représentant du client) qui ne possède pas nécessairement de connaissances techniques. Tandis que les *Use Cases* sont représentées sous des formes textuelles et d'un diagramme UML, ils se basent principalement sur les interactions entre les utilisateurs et le système.

2.8.3 Développement piloté par les tests d'acceptation (ATDD)

L'ATDD, traduit en anglais par *Acceptance Test Driven Development*, est une technique de développement logiciel itérative utilisée surtout en mode agile, basée sur la communication entre le *Product Owner* (client), les développeurs et les testeurs. Plus spécifiquement, il s'agit d'une approche collaborative dans laquelle les utilisateurs, les développeurs et les testeurs définissent des critères d'acceptation automatisés. L'ATDD englobe un ensemble de pratiques aidant les développeurs et testeurs à bien comprendre la valeur d'affaires du *Product Owner* (besoins du client) avant sa mise en œuvre et permet au *Product Owner* d'être en mesure de converser dans le vocabulaire propre à son domaine d'affaires. L'ATDD est étroitement lié au développement piloté par les tests (TDD) et met en évidence l'écriture des tests d'acceptation avant que les développeurs commencent le codage. L'approche ATDD est basée sur un principe fondamental où les tests sont au cœur du projet de développement logiciel. Le principe fondamental de l'ATDD se résume ainsi² :

2. <http://www.assertselenium.com/atdd/difference-between-tdd-bdd-atdd/>

- Les tests d'acceptation servent de spécifications du logiciel ;
- Les tests sont créés en amont du projet de développement logiciel ;
- Le développement du logiciel est réalisé en fonction des tests et ces tests s'exécutent automatiquement au fur et à mesure du développement, assurant ainsi une non-régression des acquis.

2.9 Outils de tests fonctionnels et de tests d'acceptation pour la rédaction des exigences logicielles

Le test est l'une des phases la plus importante du cycle de vie du logiciel, en d'autres termes, les tests jouent un rôle clé dans le développement de logiciels fiables et de qualité. Par définition, le test est un processus formel conduit par des spécialistes, effectué avec l'objectif d'évaluer un système par des moyens automatiques ou manuels, pour vérifier s'il répond à ses spécifications. D'après le dictionnaire normalisé de l'IEEE, le test se définit ainsi : « l'exécution et l'évaluation d'un système ou d'un composant pour établir les différences ou les écarts entre les résultats attendus et les résultats obtenus » (IEEE standard board, 2009). Le test fonctionnel, communément appelé "test boîte noire", est effectué en général par des analystes d'affaires ou fonctionnels, il vise à vérifier la conformité du logiciel par rapport à ses spécifications. Tandis que le test d'acceptation est un test formel, effectué pour permettre à un client (Product Owner) de déterminer s'il convient d'accepter un produit ou un composant. Pour savoir si le composant ou le produit livré correspond à son besoin, il est nécessaire de s'assurer que les *Use Cases* ou les *User Stories* sont respectées. Ces *Use Cases* ou *User Stories* sont regroupés dans les tests d'acceptation. Il existe dans la littérature quelques outils qui permettent de travailler avec les tests d'acceptation, dont les plus connus sont Cucumber (avec le langage Gherkin) et FitNesse.

2.9.1 Outil Cucumber/Gherkin

Cucumber est un outil logiciel utilisé par les développeurs pour tester un logiciel. Il exécute automatiquement les tests d'acceptation écrits en style BDD (*Behavior Driven Development*), il joue un rôle central dans l'approche de développement de BDD et ATDD. Cucumber est écrit dans le langage de programmation Ruby. À l'origine, il a été utilisé exclusivement pour le test Ruby, en complément du Framework BDD. Présentement, Cucumber supporte une plusieurs langages de programmation, il utilise le langage Gherkin pour définir les cas de tests. Gherkin a été conçu pour être non technique et lisible par un non-informaticien. En d'autres termes, Gherkin utilise un langage naturel, bien que structuré, pour décrire les scénarios de test relatifs à un logiciel. La syntaxe de Gherkin promeut les pratiques de développement axées sur le BDD et l'ATDD. Le BDD est une technique qui permet d'écrire des scénarios de tests décrivant le comportement attendu du système³. Ces scénarios sont écrits dans un langage simple de façon à ce qu'ils soient compris par toutes les parties prenantes du projet. Le BDD s'applique à différents niveaux de tests (tests d'acceptation, tests unitaires, tests systèmes, etc.). À noter que Gherkin définit un script pour les tests automatiques, autrement dit, la syntaxe du langage naturel de Gherkin a été conçue pour fournir une documentation simple du code sous test (Wynne *et al.*, 2017).

2.9.2 Syntaxes de Gherkin

La syntaxe de Gherkin est centrée autour d'une conception orientée ligne, la structure d'un fichier est définie en utilisant des caractères de contrôle et des espaces (Wynne *et al.*, 2017). En Gherkin, les instructions sont une ligne non vide et ne

3. <https://dannorth.net/introducing-bdd/>

contiennent pas de commentaires. Les fichiers Gherkin ont l'extension «. Feature », ils constituent un fichier de test exécutable.

- Feature : représente le nom de la fonctionnalité.
- Scenario : c'est un Use Case ou un User Story qui décrit une fonction spécifique du logiciel testé.
- Scenario : c'est un Use Case ou un User Story qui décrit une fonction spécifique du logiciel testé.
- Given : décrit les préconditions et l'état initial avant le début d'un test
- When : décrit les actions prises par un utilisateur ou des évènements qui surviennent lors d'un test.
- Then : décrit la sortie résultant des actions prises dans la clause When.

La syntaxe de Gherkin se résume ainsi :

<i>Feature:</i> Fonctionnalité A <i>Scenario:</i> Scenario 1 <i>Given:</i> préconditions <i>When:</i> actions <i>Then:</i> résultats

FIGURE 2.4 Syntaxe en Gherkin

Voici un exemple de scénario en Gherkin où un utilisateur désire retirer 300 \$ sur son compte bancaire via un guichet automatique :

<i>Feature:</i> retirer de l'argent sur mon compte <i>Scenario :</i> le solde de mon compte dépasse 300 \$ <i>Given:</i> le solde de mon compte est de 500 \$ <i>When:</i> je retire 300 \$ <i>Then:</i> le guichet me remet 300 \$ <i>Then:</i> le solde de mon compte devient 200 \$

FIGURE 2.5 Exemple de scénario en Gherkin

2.9.3 Outil FitNesse

FitNesse est un outil collaboratif qui permet d'écrire et d'exécuter des tests d'acceptation. Les tests sont écrits sous forme de pages wiki en langage « Wiki Markup », ce qui permet aux non-informaticiens de pouvoir écrire des tests en FitNesse. L'un des avantages majeurs de cet outil est qu'il permet aux analystes fonctionnels et aux clients de pouvoir écrire la spécification du logiciel à l'aide du code wiki markup, en d'autres termes, FitNesse favorise la collaboration entre client et fournisseur. Tandis que les développeurs et testeurs peuvent écrire les Custom Fixtures ou codes de liaison afin de faire la liaison entre les spécifications et le logiciel à tester. Il est à noter que FitNesse est un serveur Web qui ne demande aucune configuration, on n'a qu'à le lancer, écrire et exécuter les tests. En effet, FitNesse peut fonctionner à deux systèmes de tests différents, les *Uses Stories* et les *Use Cases* sont écrits en langage wiki markup, par la suite, ils seront interprétés par l'un des deux systèmes de tests⁴. FitNesse est un outil Open-Source (donc libre de droits) qui paraît assez intéressant, il permet de sortir les tests du code. De plus, les tests sont lisibles par des non-informaticiens, ce qui pourrait permettre à ces derniers de pouvoir contribuer à l'écriture des tests. Toutefois, on n'est pas loin de faire de la programmation quand on écrit les tests.

2.10 Limites des approches, méthodes ou techniques de rédaction d'exigences du logiciel

L'ingénierie d'exigences est une phase importante dans le domaine de production ou de développement de logiciels. Par définition, une exigence logicielle est une condition à laquelle un logiciel ou un système doit pouvoir satisfaire. En d'autres mots, une exigence logicielle est une capacité que doit présenter un système pour

4. <https://mermet.users.greyc.fr/Enseignement/CoursPDF/fitnesse.pdf>

satisfaire un contrat entre un client et un fournisseur (IEEE 729-1990). Dans le domaine du génie logiciel, le processus d'ingénierie d'exigences, plus précisément les activités d'élicitation, d'analyse, de spécification, de vérification et validation des exigences s'avèrent importantes pour les ingénieurs logiciels. Dans leur ouvrage intitulé *Software requirements*, (Wiegers et Beatty, 2013) définissent l'élicitation des exigences comme un processus d'exploration, de découverte et d'invention. Cela permet de découvrir les exigences d'un système logiciel en communiquant avec les clients, utilisateurs et autres parties prenantes ayant un intérêt dans le développement du système (Wiegers et Beatty, 2013). Les exigences, une fois découvertes, seront analysées et décrites dans un document ou un outil de spécification d'exigences logicielles. Ce document, ou les informations saisies dans l'outil, constituera, après la vérification et validation du client, la base contractuelle entre les ingénieurs logiciels et le client. L'ingénierie d'exigences est une activité interdisciplinaire servant d'intermédiaire entre le fournisseur et le client afin de pouvoir spécifier et gérer les exigences qui doivent être satisfaites par le système. Elle consiste donc à identifier les buts et la portée du logiciel et connaître le contexte dans lequel le logiciel sera utilisé. Quant à (Boehm *et al.*, 2000), le processus d'ingénierie d'exigences représente la partie amont du processus de développement de logiciels. Il permet, entre autres, le passage des besoins informels exprimés par les parties prenantes en des exigences abstraites jusqu'à parvenir à l'obtention d'une spécification d'exigences du logiciel (Boehm *et al.*, 2000). Les exigences en amont, une fois produites, seront décrites dans un cahier des charges. Ce document (cahier des charges) constitue le point d'entrée pour les phases de développement de logiciel entre le client et les développeurs. Cependant, les techniques de rédaction d'exigences logicielles adoptées dans l'industrie possèdent plusieurs limites. En effet, l'une des limites de l'approche classique ou traditionnelle de rédaction d'exigences du logiciel vient du fait que les techniques, notamment les Use Case et Use Stories, utilisés pour spécifier et décrire les exigences du logiciel sont en

langage naturel sous forme de texte avec des détails superflus. Ces techniques ne facilitent pas par conséquent l'automatisation du processus de mesure de la taille fonctionnelle des logiciels. D'après (Ambler, 2003) et (Laporte et April, 2018), ces techniques augmentent le risque d'échec des projets de développement de logiciels, puisqu'elles décrivent un pourcentage important de spécifications logicielles qui ne sont jamais implémentées. De plus, l'approche classique de rédaction d'exigences ne parvient pas à résoudre le problème de la sémantique des exigences, puisque le langage naturel est intrinsèquement ambigu. Les exigences logicielles sont souvent incomplètes, incohérentes et propices aux ambiguïtés, et donc des erreurs d'interprétation peuvent facilement être commises par les ingénieurs logiciels (Trudel, 2012). L'ambiguïté et l'incohérence des exigences constituent un problème central au niveau de l'approche actuelle de l'ingénierie d'exigences. De plus, l'approche traditionnelle exige d'écrire les exigences détaillées tôt dans le cycle de vie d'un projet et que la plupart de ces exigences ne reflètent pas souvent certains besoins réels des utilisateurs qui seront découverts plus tard dans le cycle de vie. De ce fait, cela augmente le risque d'échec de produire des logiciels qui ne répondent pas à la valeur d'affaires du client. D'autant plus, la limite de l'approche classique pourrait se situer au niveau de la traçabilité des exigences tout au long de l'ensemble des phases du cycle de vie du logiciel (analyse, conception, codage, test). On constate fort souvent à la phase de test que le logiciel final ne correspond pas aux exigences du logiciel spécifié en analyse et en conception (Laporte et April, 2018). Par conséquent, ne serait-il pas nécessaire de trouver des nouvelles méthodes, techniques ou approches pour la rédaction d'exigences du logiciel? En effet, outre de ces limitations, les exigences du logiciel sont complexes, incohérentes, non concises et sujettes à diverses interprétations. De ce fait, ces facteurs influent sur la lecture, la compréhension, l'interprétation et la gestion des exigences. L'utilisation d'une approche par triplet dans le domaine d'ingénierie d'exigences ne pourrait-elle pas résoudre ces limitations et réduire l'ambiguïté des

exigences du logiciel? Enfin, selon (Nuseibeh et Easterbrook, 2000), les limitations de l’approche classique de rédaction d’exigences se situent premièrement au niveau de la non-spécification des liens sémantiques entre les artefacts de développement durant les premières phases de développement. Les exigences logicielles sont potentiellement incohérentes, puisqu’elles sont décrites par des parties prenantes ayant chacune un point de vue différent, de divergences conceptuelles et des intérêts divergents. Deuxièmement, la limitation vient de la difficulté pour les parties prenantes de produire une spécification formelle des exigences (Sadoun, 2014)(Fotso, 2019)(Nuseibeh et Easterbrook, 2000). Troisièmement, la limitation de l’approche classique de spécification et rédaction d’exigences du logiciel réside dans le fait que les liens de traçabilité entre les exigences et les artefacts de développement sont rarement spécifiés (El Ghazi, 2009)(Cleland-Huang *et al.*, 2012)(Wiegiers et Beatty, 2013)(Mäder *et al.*, 2017). Ainsi, il est difficile, dans le cadre de l’approche actuelle d’ingénierie d’exigences, de réaliser des traitements automatiques sur les exigences. Par conséquent, les techniques de rédaction d’exigence du logiciel existantes sont loin de permettre l’automatisation du processus de mesure de la taille fonctionnelle des logiciels. Il est donc souhaitable de proposer une nouvelle technique de rédaction d’exigences du logiciel qui pourrait pallier ces difficultés et faciliter l’automatisation de la mesure de la taille des logiciels. Pour réaliser cette automatisation de la mesure de la taille fonctionnelle, nous allons procéder par une approche par triplets, à savoir l’utilisation de la logique de description.

2.11 Synthèse

Dans ce chapitre, nous avons effectué une revue de littérature basée sur les mesures, les méthodes de mesurage, la TF des logiciels, les outils d’automatisation, ainsi que les outils et méthodes de rédaction d’exigences du logiciel. Plusieurs mé-

thodes de mesure de la TF des logiciels ont été proposées. Parmi les différentes méthodes, COSMIC est une récente méthode de mesure, développée dans le but de pallier les lacunes des autres méthodes. Par la suite, nous avons présenté un survol des outils d'automatisation du processus de mesure de la TF des logiciels. La littérature consultée a montré que la majorité des outils développés sont peu efficaces et possèdent plusieurs limitations et contraintes. De plus, nous avons passé en revue les principales méthodes et techniques de rédaction d'exigences du logiciel utilisées dans l'industrie. La plupart de ces méthodes permettent d'exprimer les exigences du logiciel dans un langage techniquement neutre. Par conséquent, il est préférable d'exprimer ou de rédiger les exigences du logiciel selon une approche par triplets. Cette nouvelle technique, favorise le processus d'automatisation de la TF des logiciels.

CHAPITRE III

CADRE CONCEPTUEL DE LA THÈSE

3.1 Problématique de la recherche et volet cognitif

La problématique de ce projet de recherche telle qu'indiquée au chapitre I réside dans le fait que les techniques et méthodes de rédaction d'exigences du logiciel existantes, autrement dit les techniques de rédaction d'exigences du logiciel utilisées dans l'industrie, ne favorisent pas l'automatisation de la MTF des logiciels. Pour résoudre ce problème, nous proposons de concevoir et de mettre en œuvre une nouvelle technique de rédaction d'exigences du logiciel, à partir d'une approche par triplets. L'idée est de créer une ontologie ou un triplestore de représentation des exigences du logiciel sous forme d'un modèle de triplets RDF (*Resource Description Framework*). Cette ontologie sert de la mémoire sémantique dans le cadre de notre recherche. Chaque triplet, étant une phrase simple (sujet, prédicat, objet), est extrait à partir de Cas d'utilisation ou récits utilisateurs écrits en langage naturel. Les prédicats sont, entre autres, un ensemble de mouvement de données générés à partir des triplets. Nous avons développé donc un outil qui permet de générer la TF du logiciel à mesurer, en puisant dans le triplestore. Une logique de description est utilisée pour décrire la dimension subatomique des prédicats. La logique de description, la mémoire sémantique et la mémoire procédurale constituent les éléments fondamentaux d'un système de production à base

de règles pour la mesure. Dans la suite de ce chapitre, nous décrivons succinctement quelques éléments fondamentaux qui constituent le volet cognitif de notre cadre méthodologique.

3.2 Mémoire épisodique, sémantique et procédurale

Au cours des dernières décennies, les chercheurs en sciences cognitives et en intelligence artificielle ont découvert le rôle central de la mémoire dans la cognition. La mémoire est utile pour nous aider à raisonner et faire des inférences. En effet, dans les nouvelles découvertes en sciences cognitives, le fonctionnement humain contient plusieurs mémoires différentes qui sont la mémoire sémantique, la mémoire procédurale, la mémoire épisodique, la mémoire de travail ou à court terme, etc. Selon le modèle de (Tulving, 1972), les mémoires sémantique, procédurale et épisodique sont des mémoires à long terme distinctes. Dans le cadre de cette recherche, nous insisterons sur les mémoires épisodique, sémantique et procédurale. Par la suite, nous déterminerons leur nature dans le processus d'automatisation de la mesure de la taille fonctionnelle des logiciels.

3.2.1 Mémoire épisodique

Par définition, la mémoire épisodique désigne le processus par lequel l'humain se souvient des événements antérieurs qui arrivent dans sa vie (Tulving, 1972). Elle est une mémoire déclarative et personnelle qui définit notre identité. Lorsque la mémoire épisodique d'une personne se décompose (ex. par la maladie d'Alzheimer), l'identité personnelle de la personne disparaît (Tacconnat, 2012). Par exemple, selon une observation faite dans certaines pathologies, des troubles de la mémoire autobiographique épisodique peuvent aboutir à des troubles de l'identité. En effet, le fonctionnement de la mémoire épisodique est basé sur trois étapes

principales qui sont respectivement l'encodage, le stockage et la récupération (Taconnat, 2012). On entend par l'encodage, le fait d'entrer des informations dans notre mémoire, à partir des différents organes sensoriels et leur modification du tissu nerveux ou la transformation en trace mnésique dans le cerveau. Le stockage permet de refléter la conservation de la trace mnésique. Lorsqu'on recherche de façon volontaire et consciente de l'information stockée dans la mémoire épisodique, on parle de récupération.

3.2.2 Mémoire sémantique

La mémoire sémantique est un système par lequel on regroupe les connaissances générales sur le monde et les définitions des concepts (Taconnat, 2012). Les connaissances sémantiques font référence à la signification des mots, des objets, aux connaissances et aux croyances sur les événements historiques (Tulving, 1972). Au cours des dernières années, certains auteurs ont essayé de montrer que les connaissances sémantiques sont représentées dans un système nommé de mémoire sémantique, indépendamment de la mémoire procédurale et épisodique (Desgranges et Eustache, 2011)(Cohen et Squire, 1980)(Tulving, 1972). La mémoire sémantique est un type de mémoire déclarative et se rapporte à des connaissances que le sujet a apprises. Elle donne du sens à nos perceptions et est indispensable pour la réalisation de certaines activités cognitives et linguistiques, telles que la compréhension des mots et l'identification des concepts (Taconnat, 2012). En effet, la mémoire sémantique constitue une base de connaissances que l'humain possède, et dont une grande partie de ces connaissances nous est accessible rapidement.

3.2.3 Mémoire procédurale

La mémoire procédurale est la mémoire des actions, procédures et relations stimulus-réponse (Tulving, 1972). Elle inclut les procédures et le savoir-faire de nature perceptuelle, sensorimotrice et cognitive (Tremblay, 2017). Selon (Squire et Knowlton, 2000), la mémoire procédurale fait référence à la mémoire implicite. Cela vient du fait que les informations qu'elle contient sont difficilement verbalisables et ne sont pas toujours accessibles à la conscience (Squire et Knowlton, 2000). Par ailleurs, la mémoire sémantique fait référence à la mémoire explicite, puisque les informations qu'elle contient sont verbalisables et accessibles à la conscience (Squire et Knowlton, 2000). La mémoire procédurale est une sorte de mémoire non déclarative et fonctionne grâce à différentes zones du cerveau et ces zones sont reliées entre elles par des synapses qui fonctionnent avec des neurotransmetteurs ; ce qui permet donc l'apprentissage des procédures.

En somme, la mémoire sémantique est plus spécifiquement humaine, notamment parce qu'elle est déclarative et liée au langage articulé. Ainsi, les humains développent leur mémoire procédurale en premier, avant d'accéder au langage, et ce n'est qu'en accédant au langage qu'ils ajoutent une mémoire sémantique à la mémoire procédurale (Squire et Knowlton, 2000). Ensuite, ils peuvent développer de nouvelles expertises. Mais aussi, contrairement aux autres animaux, les humains peuvent développer des expertises plus sophistiquées, en rendant automatique du traitement d'information symbolique, par le moyen de règles explicites. Il s'agit alors d'un traitement procédural de contenus de la mémoire sémantique. Dans notre recherche, on veut automatiser la mesure de la taille fonctionnelle des logiciels, pour faire une telle contribution : rendre procédurale la mesure du processus fonctionnel qui était auparavant sémantique, sans procédure automatique. La procéduralisation se fait par l'application automatique de règles, grâce au système de

production que nous mettons en place.

3.3 Rôle des mémoires épisodique, sémantique et procédurale dans notre recherche

Dans le domaine du développement de logiciels, le client a recours en général à sa mémoire épisodique pour exprimer ses besoins avec plein de détails superflus, mais vécus par ce dernier. Ensuite, les analystes d'affaires traduisent les besoins ou récits racontés par le client issus de la mémoire épisodique en des informations plus ou moins sémantiques (mémoire sémantique non formalisée) en un format normalisé sous forme de Cas d'utilisation ou de récits utilisateurs, mais aussi avec des détails superflus et souvent ambigus. Dans le but d'automatiser la MTF, nous avons traduit les Use Cases ou User Stories en informations plus structurées ou formelles (mémoire sémantique formalisée) pour pouvoir construire des triplets. En d'autres termes, nous avons construit une ontologie en utilisant la logique de description pour construire des triplets. Par la suite, nous parvenons à automatiser la MTF par la conception d'un outil. Finalement, le client va recourir à sa mémoire sémantique, et progressivement à sa mémoire procédurale pour mettre en œuvre l'outil en vue d'obtention de la TF des logiciels. Dans une telle perspective, on rend en partie procédural ce qui était auparavant sémantique et ce qui était épisodique avant d'être sémantique. Donc, la mémoire sémantique formalisée se construit à partir des Cas d'utilisation et des récits utilisateurs et qui sont ensuite transformés en triplets. En outre, nous transformons la mémoire sémantique non formalisée en une ontologie formelle qui utilise la logique de description. Par ailleurs, l'obtention des triplets permet de traiter le contenu de la mémoire sémantique formalisée par des règles, donnant lieu à l'automatisation de la mesure des points de fonction. Dans le cadre de cette recherche, la mémoire sémantique formalisée est un modèle de triplets, qui sert à stocker des données de type de triplets. En d'autres termes,

nous transformons les exigences fonctionnelles écrites en langage naturel en un modèle de triplets (SPO). L'automatisation permet de traiter une information sémantique de manière procédurale plutôt que sémantique ; ce qui est plus simple, plus rapide et plus efficace. Par contre, la mémoire procédurale est l'outil de génération automatique des triplets à partir des Cas d'utilisation ou Récits utilisateurs que nous avons développé. Ces deux mémoires sont connectées entre elles, ce qui permet au module de l'outil d'automatisation de puiser dans ces deux mémoires pour en tirer des inférences, c'est-à-dire de déterminer la TF du logiciel à mesurer et d'identifier les mouvements de données. Dans plusieurs expertises humaines, il y a une activité où on peut puiser des choses dans la mémoire sémantique et dans la mémoire procédurale pour générer de nouvelles informations. La figure 3.1 donne une vue sur le rôle de la mémoire épisodique, de la mémoire sémantique et de la mémoire procédurale dans cette thèse.

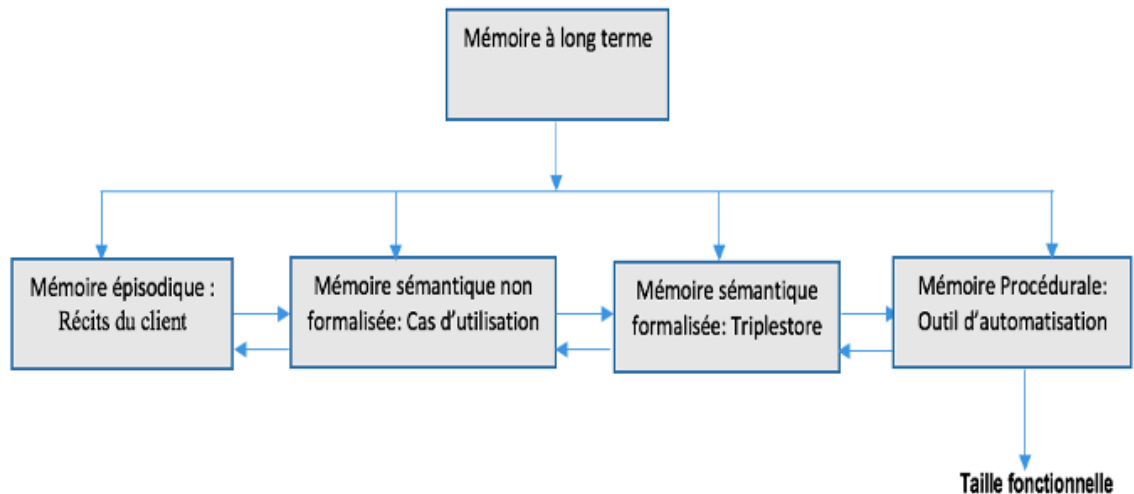


FIGURE 3.1 Rôle des différentes mémoires dans la thèse

3.4 Ontologie et mémoire sémantique formalisée

L'ontologie est un concept du monde informatique, qui a été emprunté à la philosophie et est principalement utilisé en intelligence artificielle, en Web sémantique, en traitement des langues naturelles, en représentation des connaissances et en ingénierie logicielle (Breitman *et al.*, 2007). Dans le domaine de l'informatique, une ontologie est un ensemble de connaissances structuré à un domaine particulier qui représente le sens des informations (Lehmann et Voelker, 2014). En effet, il existe plusieurs définitions pour une ontologie. Elle peut être un ensemble de concepts et de termes structurés qui représentent le sens d'un domaine spécifique de connaissances. D'après (Berners-Lee *et al.*, 2001), une ontologie est un modèle de représentation des propriétés du monde réel dans un formalisme permettant un traitement informatique. Quant à (Gruber, 1993), « An ontology is an explicit specification of a conceptualization, i.e., an abstract, simplified view of the world that we wish to represent for some purpose ». Elle est ainsi employée pour raisonner et tirer des inférences à partir des concepts du domaine en question. Par ailleurs, (Uschold et Jasper, 1999) définissent une ontologie comme un artefact conceptuel, qui rassemble un vocabulaire de termes et de spécifications et décrit la façon dont ils sont interreliés les uns aux autres. Les termes correspondent aux classes ou aux concepts, tandis que les propriétés correspondent aux propriétés ou aux attributs. La finalité d'une approche par ontologie est de faciliter la communication entre les humains et d'éviter l'ambiguïté (Uschold et Jasper, 1999). Dans le domaine du Web sémantique, les ontologies consistent à expliciter la sémantique ou signification des ressources informationnelles décrites par les données RDF. OWL (Web Ontology Language) est utilisé pour modéliser les ontologies. Ce langage de modélisation d'ontologie, qui utilise RDF et RDFS (Resource Description Framework Schema) comme moyens d'expression, fournit des opérateurs des logiques de description pour la formalisation des connaissances descriptives. Ces

connaissances, une fois formalisées, pourront permettre aux machines d'effectuer des traitements sur les informations. Dans cette recherche, l'ontologie qui est créée, est un triplestore et constitue une mémoire sémantique formalisée de manière telle qu'on pourra générer des inférences de façon automatique. Elle consiste à stocker les concepts et connaissances des processus fonctionnels du logiciel sous forme d'un triplet, et aussi les données pour appliquer le processus de mesurage de COSMIC. L'idée est de transformer les exigences logicielles écrites en langage naturel en une représentation plus simple, qui facilite l'automatisation. Ainsi, cette ontologie est centrale pour notre recherche.

3.5 Outils de construction/d'extraction d'ontologies

La construction d'ontologies à partir de textes constitue un sous-domaine à part entière de l'ingénierie des ontologies. Selon (Mondary *et al.*, 2008), ces ontologies servent essentiellement, dans le contexte du Web sémantique, à l'annotation sémantique de ressources et à la structuration de bases de connaissances. À titre d'exemples, Text2Onto est un outil ayant été conçu par (Cimiano et Völker, 2005) pour construire des ontologies de manière complètement automatique à partir des textes. Cet outil a été développé en Java et est composé de modules qui permettent d'extraire des concepts, des relations entre ces concepts et des instances de concepts, à partir des textes (Cimiano et Völker, 2005). OntoGen est un outil implémentant une approche semi-automatique pour la construction d'ontologies de thèmes à partir de collections de documents (Fortuna *et al.*, 2006). Il est à noter que c'est un outil interactif codé en .Net, suggérant à l'expert du domaine des concepts sous la forme de classes de documents. Il propose une dénotation sous la forme de mots clés et leur associe automatiquement des instances (les documents). Une particularité de l'outil OntoGen vient du fait qu'il permet de visualiser une ontologie en cours de construction. Quant à la méthode Terminae, elle

est supportée par un logiciel et sert à guider l'expert du domaine (l'ingénieur des connaissances) dans la conception de l'ontologie (Aussenac-Gilles *et al.*, 2008). Elle s'appuie sur des résultats des outils de traitement automatique des langues. (Zouaq et Nkambou, 2009) ont développé un outil nommé TextCommon, qui permet d'extraire une ontologie formelle à partir des textes. TextCommon est un outil d'apprentissage d'ontologies de puzzles de connaissances, qui permet d'extraire des cartes conceptuelles à partir de textes (Zouaq et Nkambou, 2008)(Zouaq et Nkambou, 2009). En somme, il existe dans la littérature plusieurs outils permettant de construire ou d'extraire automatiquement des ontologies à partir des textes écrits en langage naturel. Dans le cadre de notre projet de recherche, l'ontologie que nous avons développée est un triplestore. De ce fait, nous avons développé un outil qui permet de générer automatiquement des triplets à partir des exigences fonctionnelles écrites en langage naturel. La structure ciblée par l'outil est un triplet présenté sous forme de (SPO).

3.6 Approche à base de règles en intelligence artificielle et logiques de description

Au début du XXI^e siècle, la pensée était attribuée à la manipulation des symboles ou des règles et la connaissance consistait à se représenter adéquatement l'état du monde par des symboles (Sagaut, 2008). Cette façon de penser relève de l'approche à base de règles, qui est une des formes de l'approche symboliste. L'approche symboliste en intelligence artificielle (IA) est vue comme une mécanisation de règles, en d'autres termes, l'IA est comme un ensemble de règles de manipulation de symboles que l'on peut injecter dans un ordinateur. Dans cet ordre d'idées, les règles peuvent être automatisées dans un système expert à base de règles. Par conséquent, les systèmes experts constituent un modèle paradigmatique de l'IA symboliste. En outre, un système expert à base de règles est un ensemble de règles

de format (if... then) et contient dans sa base de données des connaissances codées sous forme de règles. Ainsi, le système expert imite le raisonnement humain en fonction des règles et des actions associées. Ce raisonnement, fondé sur des règles, utilise donc un moteur d'inférences capable de traiter des règles logiques. Par ailleurs, la logique est un langage applicable dans de nombreux domaines, et elle permet de représenter des connaissances à l'aide des symboles. Dans le cadre de notre recherche, nous avons développé une base de règles pour être capable de construire des triplets et de les extraire à partir des Cas d'utilisation ou Récits utilisateurs écrits en langage naturel. En effet, cette base de règles se réalise à trois niveaux. Dans le premier niveau, nous avons appliqué des règles pour pouvoir associer des prédicats à des sujets et objets, c'est-à-dire de construire les triplets. Par la suite, des règles sont appliquées dans le deuxième niveau pour permettre à l'outil d'extraire ces triplets à partir des textes écrits en langage naturel, d'identifier les mouvements de données (Entrée, Sortie, Lecture, écriture) et de déterminer la taille fonctionnelle. Finalement, nous avons appliqué des règles pour permettre à l'outil d'apprendre des nouvelles règles afin d'améliorer l'extraction des règles du deuxième niveau. Nous avons utilisé les formules de la logique des prédicats du 1er ordre pour la décomposition des prédicats moléculaires en prédicats atomiques, tout en conservant la sémantique des prédicats. L'idée principale est de diviser le prédicat moléculaire, qui n'est autre le processus fonctionnel, en des prédicats atomiques. Ces derniers (prédicats atomiques) correspondent aux mouvements de données du processus fonctionnel à mesurer.

3.7 Représentation des connaissances et logiques de description

La représentation des connaissances est l'un des facteurs clés en intelligence artificielle (IA), plus précisément dans le domaine des systèmes à base de connaissances (SBC). Une connaissance est une collection d'informations appropriées et relatives

à un domaine particulier. Dans le domaine de l'IA, les connaissances doivent être représentées de façon à ce que les programmes informatiques puissent les traiter et en tirer des inférences. Un système de représentation de base de connaissance nécessite un formalisme ou un langage spécifique pour coder les connaissances. Donc, les formalismes ou langages de représentation basés sur la logique utilisent souvent la logique de description pour capturer les faits au sujet du monde réel (Baader *et al.*, 2003)(Baader *et al.*, 2007). Quant au système à base de connaissances, il contient un ensemble de données écrites sous forme de règles, qui décrivent la connaissance de façon logique. Des opérateurs logiques tels que : la conjonction (et), la disjonction (ou) peuvent être utilisés pour établir des connaissances. En effet, la logique de description est un formalisme efficace pour le traitement automatique des connaissances. Dans le cadre de cette recherche, le terme connaissance correspond aux exigences logicielles, desquelles découlent les processus fonctionnels en COSMIC. L'objectif vise à représenter les exigences logicielles sous la forme d'un modèle de triplets (SPO). Par ailleurs, on a utilisé la logique de description pour la décomposition des prédicats moléculaires en prédicats atomiques, tout en conservant la sémantique des prédicats. L'idée principale est de diviser le prédicat moléculaire, qui n'est autre que le processus fonctionnel, en des prédicats atomiques. Nous choisissons le formalisme de la logique de description (DL) à cause de sa capacité de représentation et de raisonnement sur les connaissances.

3.8 Logiques de description et prédicats du premier ordre

3.8.1 Logiques de description

Les logiques de description (DL) est une famille de langages de représentations de connaissances pouvant être utilisés pour représenter la connaissance terminologique d'un domaine d'application de façon formelle et structurée. Ils s'appuient

sur la logique des prédicats, les schémas et réseaux sémantiques (Minsky, 2019). En effet, les logiques de descriptions (LD) constituent un sous-ensemble de la logique des prédicats du premier ordre. Elles permettent de représenter spécifiquement les concepts ou classes et les relations qui peuvent être établies entre les instances de ces classes. On distingue deux composants dans une base de connaissances en logique de description (LD), qui sont respectivement TBox et ABox. La TBox, ou niveau terminologique, contient les axiomes qui définissent les concepts du domaine, tandis que l'ABox, appelé également niveau factuel, spécifie les assertions qui apparaissent dans le domaine ou dans la description du monde réel. Les logiques de description sont utilisées dans de nombreux domaines tels que : le Web sémantique, le traitement automatique des langues et l'ingénierie logicielle. Il est à signaler que dans le cadre du Web sémantique, les logiques de description sont utilisées pour la représentation des ontologies (Baader *et al.*, 2005). Tandis qu'elles sont utilisées en ingénierie logicielle pour la représentation de la sémantique des diagrammes de classes UML (Berardi *et al.*, 2005). Dans le cadre de cette recherche, il est question de représenter les exigences logicielles sous forme d'un triplet. Une logique de description est utilisée pour décomposer les prédicats du triplestore en un ensemble de prédicats atomiques. Ces derniers (prédicats atomiques) reposeront sur un niveau subatomique, qui renvoie à des attributs. Ce qui garantit la non-décomposition du prédicat. Il est à souligner que dans le domaine de la MTF, les mouvements de données sont comptés sur un groupe de données et non sur des attributs. Dans le cas où le prédicat atomique repose sur un niveau subatomique, cela renvoie automatiquement aux attributs, ce qui signifie que le prédicat devient absolument non décomposable.

3.8.2 Logique des prédicats du premier ordre

La logique des prédicats du premier ordre est issue d'une formalisation du langage des mathématiques proposées par Frege (Goldfarb, 1979). Elle possède une syntaxe et une sémantique formelle. La syntaxe correspond à la forme des expressions qu'on peut écrire dans le calcul des prédicats, tandis que la sémantique sert à déterminer si une expression est logiquement correcte. En outre, la logique du 1er ordre est à l'origine de nombreuses applications informatiques, plus particulièrement de langages à base de règles. Dans le cadre du calcul des prédicats, une expression est une formule, tandis que les formules sont un ensemble de prédicats qui sont combinés à l'aide des connecteurs logiques et quantificateurs. Par ailleurs, les prédicats décrivent des faits ou des relations entre les objets. Quant aux objets, ils sont décrits par des termes qui peuvent être des fonctions, des variables ou des constantes. Grosso modo, les termes, connecteurs logiques, quantificateurs et prédicats sont décrits par des symboles. Dans le cadre de cette recherche, les données de type de triplets sont converties ou transformées en formules des prédicats du premier ordre exploitables par des programmes informatiques.

3.9 Synthèse

Dans ce chapitre, nous avons effectué une revue de littérature portant principalement sur les mémoires épisodique, sémantique et procédurale, l'ontologie, la logique de description, les triplets qui en découlent et l'automatisation que cela va permettre et qui va rendre le traitement procédural. La mémoire sémantique est un type de mémoire déclarative et constitue une base de connaissances que l'humain possède. Tandis que la mémoire procédurale est la mémoire des actions et procédures (Tulving, 1972). Dans le cadre de notre recherche, la mémoire sémantique formalisée constitue le modèle de triplets. Par ailleurs, la mémoire procédu-

rale est l'outil d'automatisation de la TF. Autrement dit, la mémoire procédurale est l'outil de génération automatique des triplets à partir des exigences fonctionnelles écrites en langage naturel et d'obtention de la TF. Ces deux mémoires sont connectées entre elles afin de permettre au module de l'outil d'automatisation d'y puiser pour déterminer la TF du logiciel à mesurer et identifier les mouvements de données.

CHAPITRE IV

MÉTHODOLOGIE DE RECHERCHE

4.1 Introduction

L'objectif principal de notre recherche est de concevoir et de mettre en œuvre une nouvelle technique de rédaction de logiciels, favorisant l'automatisation du processus de mesure de la TF des logiciels. Pour atteindre cet objectif, nous nous appuyons sur une démarche de recherche de type « *Design Science Research* » proposé par (Hevner *et al.*, 2004). La science du design est une méthodologie de recherche en technologie de l'information basée sur les résultats et offre des directives spécifiques pour l'évaluation des projets de recherche (Hevner *et al.*, 2004). En effet, (Hevner *et al.*, 2004) ont décrit un ensemble de caractéristiques et de lignes directrices qu'un projet de recherche en technologie de l'information doit pouvoir suivre. Ces lignes directrices pourraient se résumer ainsi :

- La création d'un artefact novateur ciblant un domaine spécifique.
- L'artefact proposé doit pouvoir résoudre un problème qui n'a pas été déjà résolu et doit contribuer à l'avancement de la recherche.
- L'artefact doit être évalué, démontré et validé par des méthodes reconnues afin de garantir son utilité pour le problème spécifié.
- L'objectif principal de la recherche en « *Design Science* » est de proposer

des solutions basées sur la technologie de l'information, capables de résoudre des problématiques utiles et pertinentes.

4.2 Notre approche méthodologique

Dans le cadre de cette recherche, nous nous inspirons de la démarche de recherche de type « *Design Science Research* » pour traiter notre problématique de recherche choisie. Notre méthodologie de recherche adoptée est divisée en quatre étapes principales. Dans la première étape, nous avons évalué les principales techniques de rédaction des exigences du logiciel. Dans la deuxième étape, nous avons conçu un triplestore dans l'objectif d'améliorer les techniques de rédaction d'exigences du logiciel. Dans la troisième étape, nous avons développé un outil, qui sert à stocker les données du triplestore et déterminer la taille fonctionnelle des logiciels. Finalement, dans la quatrième étape, nous avons évalué les résultats produits par l'outil d'automatisation, en comparaison aux résultats manuels des processus logiciels ou des systèmes mesurés par des experts humains certifiés en COSMIC.

4.2.1 Évaluation des techniques de rédaction des exigences du logiciel

Dans la littérature, il existe plusieurs méthodes, outils et techniques qui permettent de rédiger les exigences fonctionnelles du logiciel. Cette étape vise à évaluer ces outils et techniques afin de voir dans quelle mesure l'on pourrait concevoir et mettre en œuvre une technique qui pourrait permettre de rédiger les exigences fonctionnelles à partir d'une approche par triplets.

4.2.2 Conception d'un triplestore pour la rédaction d'exigences du logiciel

Le principal défi de ce projet de recherche est l'automatisation du processus de mesure de la taille fonctionnelle des logiciels. Dans cette phase, on se fixe l'objectif de concevoir et mettre en œuvre une technique de rédaction d'exigences du logiciel qui pourrait faciliter l'automatisation de la MTF des logiciels à partir des spécifications fonctionnelles. De ce fait, notre approche vise à concevoir un triplestore pour la rédaction des exigences logicielles sous forme d'un triplet. L'objectif est de décrire les exigences logicielles sous forme d'un modèle de triplets RDF.

4.2.3 Développement d'un outil d'automatisation de la mesure de la taille fonctionnelle des logiciels, selon la méthode COSMIC ISO 19761

Dans cette étape, nous avons développé un nouvel outil qui automatise la TF du processus de mesurage. Cet outil, comme indiqué dans la section 4.1, comporte deux modules. Le premier module sert à générer des triplets à partir des Use Cases ou User Stories écrits en langage naturel. Quant au second module, il sert à déterminer automatiquement la taille fonctionnelle du logiciel selon la méthode COSMIC ISO 19761.

4.2.4 Évaluation et validation des résultats

Dans le cadre de cette recherche, nous avons utilisé, d'une part, un cadre référentiel pour la validation de l'outil d'automatisation développé. D'autre part, nous avons fait une expérimentation en utilisant une base de cas de documents d'exigences mesurés et publiés par des experts humains en COSMIC. Les résultats de mesurage de la TF de ces documents de spécifications logicielles sont publiés et disponibles sur le site Web de COSMIC. Nous avons comparé les résultats de mesure obtenus de l'outil par rapport aux résultats trouvés par les experts hu-

maines afin de s'assurer que notre outil se rapproche des résultats publiés par les experts. Nos expérimentations sont donc effectuées avec des cas réels afin de tester l'interaction entre la technique de rédaction d'exigences du logiciel proposée et l'outil d'automatisation, en vue de la MTF des logiciels. La figure 4.1 présente les différentes phases de la méthodologie de la recherche. Chaque phase contient des entrées et sorties avec des chapitres correspondants.

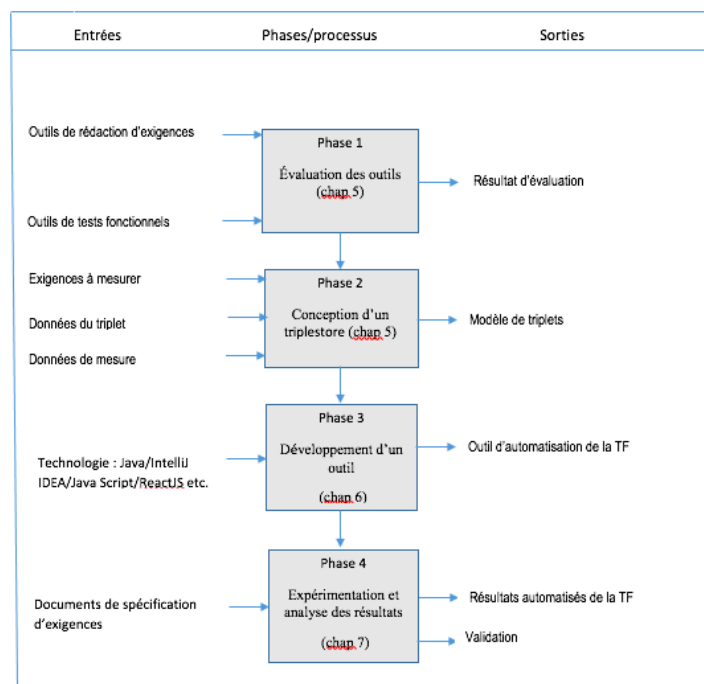


FIGURE 4.1 Phases de la méthodologie de la recherche.

4.2.5 Cadre référentiel pour la validation des outils d'automatisation de la mesure de la taille fonctionnelle

Dans cette étape, nous avons testé et évalué l'outil développé afin de vérifier la validité des résultats générés par ce dernier. Dans une telle perspective, nous avons utilisé le cadre référentiel pour les outils d'automatisation de la MTF des logiciels,

proposé par (Abran et Paton, 1997) afin de s'assurer que l'outil d'automatisation peut interagir avec la technique proposée de rédaction d'exigences du logiciel. Ce cadre référentiel décrit un ensemble de caractéristiques souhaitables pour les outils d'automatisation de la mesure de la TF des logiciels. Les principales caractéristiques du cadre référentiel recommandé par (Abran et Paton, 1997) indiquées dans la section 2.5.8.

4.2.6 Validation par les experts en COSMIC ISO 19761

Dans le cadre notre recherche, nous avons mis en œuvre l'outil avec dix (10) projets et les résultats sont comparés à ceux des experts certifiés en COSMIC ISO 19761. Cela permet de vérifier que les résultats obtenus de l'outil se rapprochent de ceux des experts dans le domaine de MTF des logiciels.

4.3 Originalité de la recherche

Dans le cadre de cette recherche, nous avons abordé les éléments non couverts dans les travaux antérieurs. D'une part, nous avons amélioré les techniques de rédaction d'exigences du logiciel, en vue de l'automatisation de la MTF des logiciels à partir des spécifications fonctionnelles, par la conception d'un triplestore. En termes de contribution cognitive, cette thèse vise à automatiser la MTF des logiciels, en rendant procédurale la mesure du processus fonctionnel qui était auparavant sémantique, sans procédure automatique.

4.4 Synthèse

Notre méthodologie de recherche adoptée est basée sur une démarche de recherche de type « Design Science Research ». L'idée est créer une technique novatrice de

réduction d'exigences logiciel facilitant l'automatisation de la TF des logiciels.

CHAPITRE V

ÉVALUATION DES TECHNIQUES ET MÉTHODES DE RÉDACTION D'EXIGENCES DU LOGICIEL ET PROPOSITION D'UNE APPROCHE PAR TRIPLETS POUVANT FACILITER L'AUTOMATISATION DE LA MESURE

5.1 Introduction

Ce chapitre vise à évaluer les principaux outils, méthodes, techniques et approches permettant de rédiger les exigences du logiciel. Il est divisé en quatre sections principales. Dans la première section, nous procéderons à l'évaluation de ces outils et approches pour voir dans quelle mesure ceux-ci pourraient-ils aider à améliorer les techniques de rédaction d'exigences du logiciel pour faciliter la MTF des logiciels. Dans la deuxième section, nous présenterons les limites de ces outils et approches en matière de rédaction d'exigences du logiciel. Dans la troisième section, nous passerons en revue l'approche par triplets proposée pour la rédaction d'exigences du logiciel. Par la suite, nous décrivons l'utilité de cette approche dans le domaine d'ingénierie ou de spécification des exigences et du processus d'automatisation de la TF des logiciels. Finalement, dans la quatrième et dernière section, nous présenterons une synthèse résumant ce chapitre.

5.2 Évaluation des techniques et méthodes de rédaction d'exigences du logiciel

Les principaux défis, problèmes et besoins en matière d'automatisation de la MTF des logiciels à partir des spécifications résident dans le fait que, d'une part, les outils d'automatisation possèdent plusieurs limitations et contraintes, d'autre part, les outils, méthodes et techniques de rédaction des spécifications logicielles ne précisent pas de détails techniques qui pourraient favoriser l'automatisation du processus de mesure de la TF. De ce fait, il y a un besoin d'améliorer les techniques et méthodes de rédaction d'exigences du logiciel existantes à partir d'une approche par tripletS, en vue de faciliter l'automatisation de la MTF des logiciels à partir des documents de spécifications du logiciel. Quels seraient donc les avantages de rédiger les exigences du logiciel sous forme de triplets ? En effet, (Wiegiers et Beatty, 2013) présente un ensemble de caractéristiques et d'indicateurs décrivant les faiblesses et lacunes des approches existantes de rédaction d'exigences du logiciel. Ces faiblesses se situent au niveau principalement de l'incohérence, de l'ambiguïté et de la non-atOMICITÉ des exigences. Les exigences sont donc souvent en contradiction avec d'autres exigences, possèdent plusieurs interprétations possibles et n'expriment pas un seul besoin (Wiegiers et Beatty, 2013)(Trudel, 2012). Quant à (Trudel et Boisvert, 2011), ils soutiennent que les exigences logicielles rédigées dans l'industrie sont parfois peu utiles (lorsqu'elles n'apportent pas de valeur d'affaires), non utilisables (dans leur forme et fonction, souvent parce qu'elles sont ambiguës ou incomplètes) et peu utilisées (quand ce sont des exigences superflues). Cela sous-entend qu'une grande partie des exigences logicielles rédigées dans l'industrie ne sont pas pertinentes pour les consommateurs ou clients, et ne sont pas par conséquent utilisées par ces derniers. Il y a donc un besoin d'améliorer les approches existantes de rédaction d'exigences du logiciel. En outre, (Gilb et Graham, 1993) proposent un ensemble de critères et de règles génériques que devrait respecter un document d'exigences logicielles. Ces princi-

Les critères sont axés sur l'unicité, la clarté et la brièveté des exigences logicielles. Les Cas d'utilisation ou Récits utilisateurs doivent être uniques, c'est-à-dire que les informations doivent être énoncées une seule fois ; ils doivent être clairs, c'est-à-dire que les informations doivent être claires et compréhensibles pour toutes les parties prenantes ; les Cas d'utilisation doivent être finalement énoncés succinctement. Le tableau 5.1 présente un ensemble d'indicateurs décrivant les faiblesses et lacunes des approches existantes de rédaction d'exigences du logiciel (Gilb et Graham, 1993)(Lévesque, 1998)(Wiegiers et Beatty, 2013)(Trudel, 2012)(Sadoun, 2014)(Bennaceur *et al.*, 2019).

Caractéristiques des exigences logicielles	Description
Incomplètes	Les exigences ne décrivent pas complètement les besoins ou la fonction du client.
Ambiguës	Les parties prenantes n'obtiennent pas la même interprétation des exigences.
Incohérentes	Les exigences sont en contradiction avec d'autres exigences.
Non atomiques	Les exigences n'expriment pas l'unique besoin de l'utilisateur.
Non correctes	Les exigences ne correspondent aux besoins réels des utilisateurs
Non traçables	Les exigences recueillies en analyse sont différentes en conception, en codage et en test.
Absence de brièveté	Les exigences logicielles ne sont pas décrites succinctement.

TABLEAU 5.1 Faiblesses des approches de rédaction d'exigences du logiciel.

En se basant sur les faiblesses, lacunes et limitations des approches de rédaction d'exigences, il est souhaitable de proposer une nouvelle technique ou approche pouvant aider à pallier ces difficultés.

5.3 Spécification des exigences logicielles avec l'approche par triplets

La technique de rédaction des exigences logicielles que nous proposons pour faciliter le processus d'automatisation de la TF des logiciels est une approche par triplets. En d'autres termes, cette approche est un modèle qui permet de spéci-

fier et rédiger les exigences logicielles sous forme de structure de triplets. Chaque triplet est une phrase simple, composé d'un trio de concepts (sujet, prédicat, objet). Le sujet représente l'acteur (c'est-à-dire l'utilisateur fonctionnel) qui interagit avec le système. Un prédicat peut être un prédicat composite ou atomique. Un prédicat composite représente le scénario de Cas d'utilisation; un prédicat atomique représente les méthodes, transactions ou événements déclenchés par l'acteur (utilisateur fonctionnel, autres systèmes, dispositif matériel). L'objet représente un composant ou une entité logicielle. L'objectif de l'approche par triplets est de permettre aux analystes d'écrire les besoins des utilisateurs de façon simple et efficace avec seulement des informations essentielles (Gérançon *et al.*, 2021). Cela signifie que la structure sous forme de triplets fournit une description atomique et succincte des exigences logicielles avec peu ou sans détails superflus. Elle s'accroche sur la clarté et la brièveté des exigences logicielles. En effet, le principal problème avec l'automatisation de la TF des logiciels est que les exigences écrites en langage naturel sont difficiles d'être interprétées par la machine. L'approche par triplets permet de résoudre ces difficultés. En outre, elle facilite la réalisation des traitements automatiques sur les exigences logicielles. Ce qui permet d'automatiser le processus de mesure de la TF des logiciels à partir des documents de spécifications écrites en langage naturel. En conséquence, l'approche par triplets est une technique de rédaction des exigences logicielles qui complète les techniques des Use Cases et User Stories. Dans une telle perspective, nous avons développé un outil qui permet d'extraire des triplets à partir des Use Cases ou User Stories écrits en langage naturel. Notre outil permet également de détecter des détails superflus dans le document de spécification des exigences logicielles.

5.4 Correspondance entre les concepts du triplet et la méthode COSMIC

La structure de triplets est un trio de concepts où le sujet correspond à l'utilisateur fonctionnel en COSMIC ; le prédicat composite correspond au processus fonctionnel et le prédicat atomique correspond aux mouvements de données. Quant à l'objet, il correspond à un groupe de données manipulables par le processus fonctionnel. Finalement, le triplet représente une partie du processus fonctionnel du logiciel à mesurer. Le tableau 5.2 décrit la correspondance entre les concepts du triplet et la méthode COSMIC.

Triplet	COSMIC
Sujet	Utilisateur fonctionnel
Prédicat composite	Processus fonctionnel
Prédicat atomique	Mouvements de données
Objet	Groupes de données

TABLEAU 5.2 Correspondance entre les concepts du triplet et la méthode COSMIC.

5.5 Approche par triplets, une méthodologie pour le développement des exigences logicielles

Dans le domaine de développement de logiciel, il existe souvent un manque de traçabilité entre les différentes phases du cycle de vie du logiciel (El Ghazi, 2009)(Cleland-Huang *et al.*, 2012)(Wieggers et Beatty, 2013)(Mäder *et al.*, 2017). Les spécifications logicielles en analyse ne sont pas toutes implantées dans le logiciel final (Lévesque, 1998)(Ambler, 2003)(Wieggers et Beatty, 2013)(Mäder *et al.*, 2017). En arrivant à la phase de test d'acceptation, on a souvent constaté que le logiciel, à la fin, ne correspond pas toujours aux documents de spécifications logi-

cielles. Plusieurs chercheurs affirment que les techniques de rédaction d'exigences logicielles existantes augmentent le risque d'échec des projets de développement de logiciels (Ambler, 2003)(Mäder *et al.*, 2017)(Laporte et April, 2018). Cela provient du fait que la description des exigences logicielles contient des détails superflus, incohérents et propices aux ambiguïtés. Dans une telle optique, l'approche par triplets pourrait servir de méthodologie pour les concepteurs et développeurs dans le domaine de spécification des exigences logicielles pour plusieurs raisons fondamentales. Premièrement, elle pourrait résoudre et faciliter les liens de traçabilité entre les exigences et les artefacts de développement de logiciels, notamment dans la phase d'analyse, de conception, de codage et de tests logiciels. Cela vient du fait que l'approche par triplets propose une description atomique et succincte des exigences logicielles exprimant l'unique besoin des utilisateurs sans détails superflus. Cela faciliterait la conception et l'implémentation des exigences logicielles spécifiées dans la phase d'analyse. Deuxièmement, l'approche par triplets pourrait résoudre les problèmes d'ambiguïté et d'incohérence des exigences logicielles. La spécification des exigences sous forme de triplets s'accroît sur l'unicité, la clarté et la brièveté du document d'exigences logicielles. En d'autres termes, les exigences logicielles seraient décrites de façon qu'elles ne soient pas en contradiction avec d'autres exigences. Ce qui permettrait de décrire la sémantique des exigences et d'éviter de développer des exigences ambiguës. Cette approche faciliterait également aux différentes parties prenantes (gestionnaire, analyste, concepteurs, programmeurs, testeurs) d'obtenir une compréhension commune des exigences logicielles. Troisièmement, la spécification des exigences par triplets permet de définir des exigences de bonne qualité, respectant les critères génériques que devrait respecter un document d'exigences logicielles de qualité. En effet, l'une des principales limitations de l'approche classique ou traditionnelle de rédaction des exigences logicielles vient du fait que les techniques, notamment, les *Use Cases* et *User Stories* utilisées pour spécifier et décrire les exigences logicielles, sont en

langage naturel, avec des détails inutiles. Ainsi, ces techniques ne facilitent pas l'automatisation du processus de MTF du logiciel. L'approche par triplets proposée permet surmonter ces difficultés, puisque la structure de triplets est simple et efficace avec peu ou sans détails superflus. Dans l'approche classique d'ingénierie d'exigences, les exigences proviennent de différentes sources et sont par conséquent hétérogènes et incohérentes. Une approche par triplets peut aider à joindre ces exigences provenant de diverses sources. Dans cet ordre d'idées, nous proposons un modèle de triplets, c'est-à-dire une ontologie qui permet de représenter et d'intégrer les exigences logicielles. La finalité est de construire un langage commun qui pourrait créer une compréhension commune entre les différentes parties prenantes et de réduire les problèmes d'incohérence et d'ambiguïté d'exigences. Il est à noter qu'on cherche à proposer une approche applicable dans l'industrie pour réduire l'ambiguïté et l'incohérence dans le processus de spécification des exigences logicielle en vue de faciliter l'automatisation de la TF des logiciels. On ne peut pas complètement outrepasser la langue naturelle pour décrire les exigences, car le langage naturel est pratiquement le seul moyen que les parties prenantes aient en commun. L'idée de rédiger les exigences sous forme de triplets est une méthode complémentaire à la technique des Cas d'utilisation. En outre, dans le processus de développement de logiciels, le passage de l'analyse et de la conception au codage paraît difficile (Happel et Seedorf, 2006)(Happel *et al.*, 2010)(Cleland-Huang *et al.*, 2012)(Laporte et April, 2018). Cela vient du fait que les opérations ou transactions du système, correspondant aux mouvements de données dans le domaine de la mesure, ne sont pas assez explicites dans la description des exigences en langage naturel. La représentation des exigences sous forme de structure de triplets permet également d'explicitier ces opérations du système, qui sont entre autres, des méthodes dans la phase de codage ou d'implémentation. Dans une telle approche, une exigence logicielle présentée sous forme d'un modèle de triplets, sera décomposée en plusieurs prédicats atomiques. Ces derniers représentent donc les

méthodes qui seront implémentées. Finalement, l'approche par triplets facilitera la réalisation des diagrammes de classes UML. Un diagramme de classes est un modèle utilisé en génie logiciel pour représenter les classes, interfaces des systèmes et leurs relations. Dans le domaine de développement des exigences logicielles, les classes constituent les composants logiciels qui seront implémentés. La structure de triplets est constituée d'un trio de concepts où le sujet correspond à l'utilisateur fonctionnel (acteur); les prédicats composites correspondent aux scénarios des Cas d'utilisation. Les prédicats atomiques correspondent aux méthodes ou transactions du système et aux noms des associations dans les diagrammes de classes. Quant aux objets, ils correspondront aux classes logicielles dans les diagrammes de classes. Le tableau 5.3 décrit les rapprochements entre les concepts du triplet, diagrammes de classes UML et la méthode COSMIC.

Triplet	UML	COSMIC
Sujet	Acteur du système	Utilisateur fonctionnel
Prédicat composite	Scénarios des Cas d'utilisation	Processus fonctionnel
Prédicat atomique	Événements/méthodes	Mouvements de données
Objet	Classe/Objets/interfaces	Groupes de données

TABLEAU 5.3 Correspondance entre les concepts du triplet,UML et la méthode COSMIC.

5.6 Modèle de triplets (triplestore) développé

Le modèle de triplets ou triplestore développé est un modèle qui permet de représenter les exigences fonctionnelles des logiciels sous forme de triplets, en vue de faciliter l'automatisation de la TF des logiciels. Comme indiqué dans la section 3.4, notre modèle de triplets proposé stocke les concepts et connaissances des processus fonctionnels du logiciel sous forme d'un triplet, ainsi que les données pour

appliquer le processus de mesurage de COSMIC. Par la suite, on a développé un outil qui permet de générer automatiquement des triplets à partir des exigences fonctionnelles écrites en format de Cas d'utilisation ou de Récit utilisateur. La structure ciblée par l'outil est représentée sous forme de (SPO). L'objectif est de représenter les exigences fonctionnelles sous la forme d'un modèle de triplets constitué d'un sujet, d'un prédicat et d'un objet. La figure 5.1 présente le modèle de triplets proposé pour la rédaction des exigences logicielles.

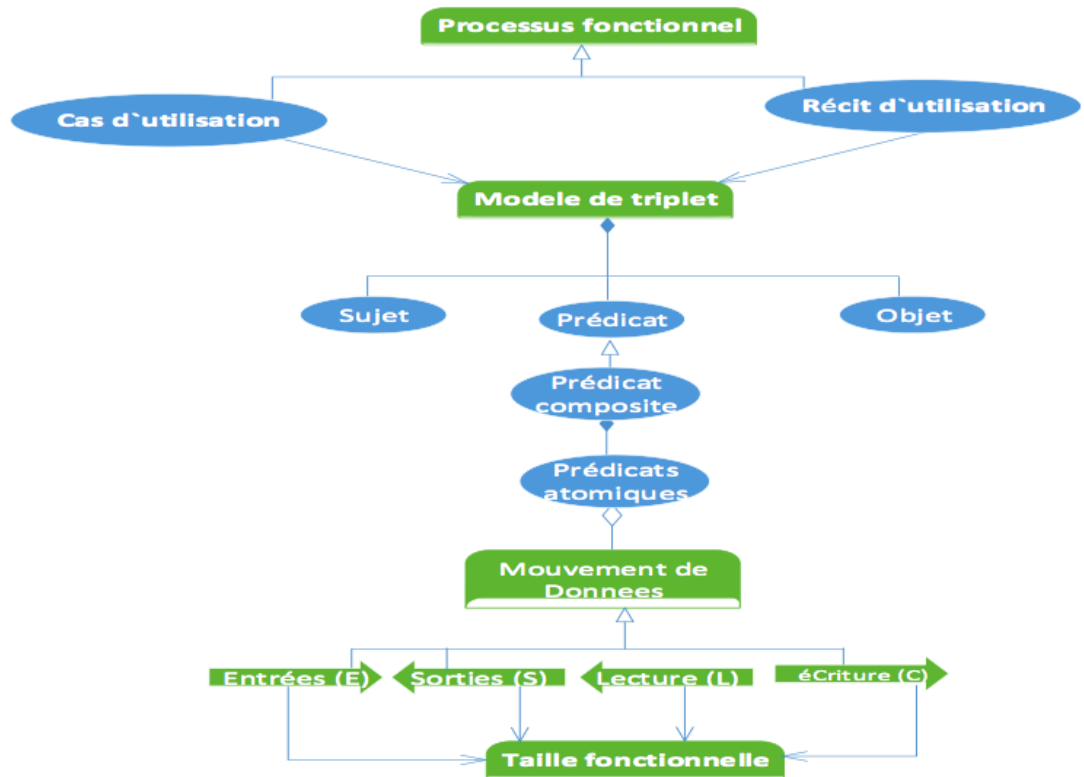


FIGURE 5.1 Modèle de triplets de rédaction des exigences logicielles.

5.7 Modèle de triplets et logique des prédicats du premier ordre

Le modèle de triplets proposé pour la rédaction des exigences fonctionnelles des logiciels exploite la logique des prédicats du premier ordre. Dans le cadre d'une exigence fonctionnelle, le titre du processus fonctionnel se décrit par un prédicat composite. Tandis que l'ensemble des événements des scénarios des Cas d'utilisation se décrivent par des prédicats atomiques. La décomposition d'un prédicat composite nécessite qu'on identifie les objets qui sont connectés avec ce prédicat. Nous avons utilisé la logique des prédicats du 1er ordre pour la décomposition des prédicats composites ou moléculaires en prédicats atomiques. La représentation des exigences fonctionnelles se traduit en des formules des prédicats du 1er ordre. Dans une telle perspective, on présuppose que la rédaction des Cas d'utilisation ou Récits utilisateurs se fait avec des prédicats de deux arguments $f(x, y)$. Le prédicat s'exprime par un verbe. Les variables « x » et « y » correspondent respectivement aux utilisateurs fonctionnels (acteurs) et aux objets. On utilise donc les prédicats pour établir les relations entre le sujet et l'objet. Par ailleurs, des connecteurs logiques tels que la conjonction « et » et la disjonction « ou » sont utilisés pour découper en triplets les phrases contenant des opérateurs logiques, ainsi que les quantificateurs universels et existentiels. La figure 5.2 illustre l'exemple d'un Cas d'utilisation représenté en des formules logiques du premier ordre et son découpage en triplets.

<p>Description du Cas d'utilisation : « L'utilisateur demande de renouveler un permis de conduire ». Le système demande à l'utilisateur de fournir le numéro de l'ancien permis afin de vérifier l'existence du permis. L'utilisateur fournit le numéro de permis. Le système vérifie les données du permis et affiche les informations du permis à renouveler.</p> <p><u>Variables pour dénoter les sujets :</u></p> <p>{x1} = utilisateur {x2} = système</p> <p><u>Variables pour dénoter les objets :</u></p> <p>{y1} = permis {y2} = numéro {y3} = données du permis {y4} = informations</p> <p><u>Variables pour dénoter les prédicats :</u></p> <p>{f1}= renouveler {f2}= demande de fournir {f3}= fournir {f4}= vérifier {f5}= afficher</p> <p><u>Représentation en des formules logiques du premier ordre :</u></p> <p>\exists sujet, \exists objet tel que prédicat (sujet, objet) ou $\exists y, \exists x$ tel que $f(x, y)$: $\exists x1, x2, y1, y2, y3, y4 [f1(x1, y1) \wedge f2(x2, y2) \wedge f3(x1, y3) \wedge f4(x2, y4) \wedge f5(x2, y4)]$</p> <p><u>Découpage en triplets :</u></p> <ul style="list-style-type: none"> - L'utilisateur, demande de renouveler, un permis - Le système, demande de fournir, numéro de l'ancien permis -L'utilisateur, fournit, numéro de permis -Le système, vérifie, données du permis - Le système, affiche, informations du permis

FIGURE 5.2 Exemple de Cas d'utilisation sous forme de triplets.

5.8 Glossaire : Concepts du modèle de triplets (triplestore)

Les principaux éléments du modèle de triplet développé sont respectivement : processus fonctionnel, Cas d'utilisation, triplet (sujet, prédicat, objet), prédicats composites, prédicats atomiques, mouvements de données (Entrée, Sortie, Lecture, écriture) et taille fonctionnelle.

- Processus fonctionnel : Le processus fonctionnel représente les exigences fonctionnelles à mesurer. Ces exigences peuvent s'écrire sous forme de Cas d'utilisation ou de récit utilisateur.
- Cas d'utilisation : Les Cas d'utilisation (Use Cases) sont des descriptions textuelles, utilisés pour documenter les spécifications logicielles. Ils racontent sous forme de texte la façon dont un acteur ou un utilisateur fonctionnel utilise un système logiciel pour atteindre un but.
- Récit utilisateur : Les Récits utilisateurs (User Stories) consistent en quelques lignes de textes qui décrivent une fonctionnalité qu'un logiciel doit offrir pour permettre à un acteur ou un utilisateur fonctionnel donné de réaliser un objectif précis. Les User Stories sont généralement écrits dans un langage naturel et ne comportent pas de termes techniques.
- Triplet : C'est une structure de représentation des exigences logicielles en un trio de concepts tels que : sujet, prédicat, objet.
- Sujet : représente l'utilisateur fonctionnel du logiciel.
- Prédicat : ensemble de mouvements de données applicables au processus fonctionnel du logiciel. Un prédicat pourrait être moléculaire (composite) ou atomique.
- Prédicats composites : appelé également prédicat moléculaire, un prédicat composite est composé de prédicats atomiques. Le prédicat composite correspond à un processus fonctionnel à mesurer.

- Prédicats atomiques : ce sont les mouvements de données déclenchés par un acteur humain, un dispositif matériel ou un autre système.
- Objet : un groupe de données manipulables par un processus fonctionnel.
- Mouvement de données : c'est un « composant de base qui déplace un seul groupe de données. Un mouvement de données peut être de type (Entrée, Sortie, Lecture, écriture) » [ISO 19761].
- Entré : mouvement de données qui déplace des groupes de données à travers la frontière d'un utilisateur fonctionnel vers un processus fonctionnel [ISO 19761].
- Sortie : mouvement de données qui déplace un groupe de données d'un processus fonctionnel vers l'utilisateur fonctionnel [ISO 19761].
- Lecture : mouvement de données qui déplace un groupe de données à partir du stockage permanent vers un processus fonctionnel [ISO 19761].
- écriture : mouvement de données qui déplace un groupe de données à partir d'un processus fonctionnel vers le stockage permanent [ISO 19761].
- Taille fonctionnelle : Taille du logiciel dérivée en quantifiant les exigences fonctionnelles des utilisateurs.
- Point de fonction COSMIC (PFC) : Unité de mesure de la méthode COSMIC.

5.9 Synthèse

Dans ce chapitre, nous avons présenté les limites des techniques et méthodes de rédaction d'exigences du logiciel. Par la suite, nous avons décrit notre approche proposée, qui est entre autres, un modèle de triplets fondé sur la logique des prédicats du 1er ordre, comme modèle de spécification, de rédaction et de représentation des exigences logicielles.

CHAPITRE VI

OUTIL DE GÉNÉRATION AUTOMATIQUE DES TRIPLETS À PARTIR DES USE CASES ET DE MESURE DE LA TAILLE FONCTIONNELLE DES LOGICIELS

6.1 Introduction

Dans ce chapitre, nous décrivons un outil de génération automatique des triplets, d'obtention de la taille fonctionnelle (TF) et ses algorithmes, que nous avons développés dans le cadre de cette recherche. Cet outil permet le découpage des phrases sous forme de sujet, prédicat, objet. Notre objectif vise à découper des Use Cases ou User Stories écrits en langage naturel en des triplets en vue de l'obtention de la taille fonctionnelle des logiciels. Dans une telle perspective, nous évaluons dans la littérature la façon dont on implémente les méthodes de génération automatique des textes (GAT). Par la suite, nous proposons notre propre algorithme ou base de règles et implémentation de génération des triplets à partir des Cas d'utilisation ou Récits utilisateurs. Il est à souligner que notre méthode permet d'extraire des triplets pour les langues françaises et anglaises. Cela vient du fait que ces langues adoptent une structure générale d'ordre (sujet, prédicat, objet). Dans le domaine de rédaction des Cas d'utilisation, le sujet représente l'acteur ou l'utilisateur fonctionnel du système et se place devant l'action. L'action représente les mouvements de données dans le domaine de la mesure. Quant au prédicat, il

se situe entre le sujet et l'objet. L'approche par triplet se retrouve donc dans la logique des prédicats du premier ordre : prédicat (sujet, objet).

6.2 Génération automatique des textes

La génération automatique des textes (GAT) est inscrite dans le cadre de traitement automatique du langage naturel (TALN). Son but est de produire des énoncés en langage naturel à partir des représentations formelles ou informatisées. Ces énoncés doivent être syntaxiquement et sémantiquement corrects. Dans le cadre de cette recherche, nous nous intéressons à générer des triplets à partir des exigences fonctionnelles, c'est-à-dire des Cas d'utilisation ou Récits utilisateurs écrits en langage naturel. En effet, la description des Cas d'utilisation n'est pas toujours précise, elle est souvent ambiguë et sa structure linguistique ne facilite pas l'automatisation du processus de mesure de la taille fonctionnelle des logiciels. Nous voulons donc proposer dans cette recherche une structure linguistique de rédaction d'exigences logicielles favorisant l'automatisation de la taille fonctionnelle. La structure ciblée est un triplet présenté sous forme de (SPO).

6.3 Outil de génération des triplets et d'automatisation de la taille fonctionnelle

L'outil que nous avons développé comporte deux modules. Le premier module permet de générer automatiquement des triplets à partir des Use Cases ou des User Stories écrits en langage naturel. En outre, nous présumons que la rédaction des exigences fonctionnelles du logiciel se fait avec des prédicats dyadiques, c'est-à-dire des prédicats de deux arguments $f(x, y)$. Le prédicat s'exprime par un verbe (une action) et correspond à un mouvement de données. La variable « x » est le sujet de l'action. Tandis que la variable « y » est l'objet de l'action ou le groupe de données dans le domaine de la mesure. Nous présumons que dans un

système à base de règles, les règles sont basées sur l'idée que la rédaction des exigences logicielles est l'élaboration de fonctions dyadiques, c'est-à-dire des $f(x, y)$. Dans une telle perspective, nous associons la fonction « f » aux variables « x » et « y ». Ensuite, nous établissons des règles de traitement de relation entre $f(x, y)$. Nous utilisons des formules de prédicat du 1er ordre pour représenter les phrases à découper en triplet. En effet, pour parvenir à identifier sémantiquement les sujets, verbes et objets d'un texte ou d'un Cas d'utilisation écrit en langage naturel, nous nous inspirons de l'analyseur de langage naturel proposé par Stanford NLP Group Software¹. Cet analyseur de langage est un ensemble de bibliothèques (libraries) dans le domaine du traitement automatique du langage naturel (TALN) qui permettent de déterminer la structure syntaxique et sémantique des phrases. Cet analyseur de langage contient une classe nommée « TagWord », qui identifie sémantiquement les mots d'un texte écrit en langage naturel tels que sujets, verbes et objets. En s'inspirant de ce programme, nous construisons et appliquons un ensemble de règles ou d'algorithmes permettant de mettre en association le sujet, le prédicat et l'objet, pour générer par la suite des triplets à partir des Cas d'utilisation ou des Récits utilisateurs écrits en langage naturel. Le deuxième module de l'outil permet l'obtention de la taille fonctionnelle du logiciel à mesurer. En fait, les triplets générés sont vus comme des règles de traitement pour pouvoir inférer ou déterminer la taille fonctionnelle. Ce module quantifie donc le nombre de prédicats (verbes) atomiques de chaque triplet correspondant à des mouvements de données. Ensuite, nous implémentons un ensemble de règles de la méthode COSMIC pour faire correspondre à chaque prédicat le type de mouvement de données (Entrée, Sortie, Lecture, écriture). Ainsi, nous utilisons un cadre référentiel proposé par Abran et Paton (1997) pour valider l'outil d'automatisation de mesure de la taille fonctionnelle proposé et pour s'assurer que ce dernier puisse interagir

1. <https://nlp.stanford.edu/software/>

avec les techniques de rédaction d'exigences logicielles et fournir une présentation des résultats facilitant des analyses.

6.3.1 Algorithme de génération des triplets et logique des prédicats du 1er ordre

Nous avons appliqué, dans le cadre de notre outil de génération des triplets (OGT), un ensemble de règles de processus de recherche qui permet de localiser dans chaque Cas d'utilisation ou Récit utilisateur les sujets, prédicats et objets. Par la suite, nous avons appliqué un ensemble de règles permettant de découper chaque Cas d'utilisation ou User Story en un groupe de phrases de structure (SPO). Finalement, l'outil génère les triplets associés aux Use Case ou aux User Stories. Dans la littérature de traitement automatique du langage naturel, il existe des algorithmes permettant de produire des phrases complexes par agrégation de proto-phrases de structures (SPO). Le processus d'agrégation vise à assembler des phrases élémentaires de structure (SPO) pour parvenir par la suite à produire une phrase complexe (Charton *et al.*, 2008)(Barone *et al.*, 2017). Dans le cadre de cette recherche, nous procédons par désagrégation. Nous cherchons plutôt à extraire des triplets à partir des exigences fonctionnelles, qui sont entre autres des phrases complexes et souvent ambiguës. Dans le domaine de linguistique cognitive, les langues anglaises, françaises et espagnoles adoptent une règle générale de structure (SPO) (Delbecque, 2006). La structure SPO exploite donc la logique des prédicats du premier ordre. De ce fait, nous utilisons le formalisme de la logique des prédicats du premier ordre pour la génération des triplets à partir des exigences logicielles (Cas d'utilisation, Récits utilisateurs) écrites en langage naturel. Nous illustrons à l'aide d'un exemple la description d'un Cas d'utilisation pour lequel nous utilisons des formules de la logique des prédicats du 1er ordre, en vue de la génération des triplets : « Le directeur de vente demande au système d'ajou-

ter un nouveau produit. Le système enregistre le produit et affiche un message de confirmation ». Nous présumons que la rédaction de ce Cas d'utilisation se fait avec des prédicats de deux arguments $f(x, y)$. Le prédicat s'exprime par un verbe, le sujet de l'action est représenté par la variable « x ». Tandis que l'objet de l'action est représenté par la variable « y ». Nous associons la fonction « f » avec les variables « x » et « y ». La logique des prédicats du premier ordre pourrait être formulée ainsi : $\exists \text{objet}, \exists \text{sujet tel que } \text{predicat}(\text{objet}, \text{sujet})$.

En remplaçant le sujet par la variable «x», l'objet par la variable «y» et le prédicat par f, la formule devient $\exists y, \exists x \text{ tel que } f(x, y)$.

Les sujets :

- x1 = Le directeur de vente
- x2 = Le système

Les objets :

- y1 = Nouveau produit
- y2 = Produit
- y3 = Message de confirmation

Les prédicats :

- f1 =Demande d'ajouter
- f2 =Enregistre
- f3 =Affiche

La formule logique correspondant à la description de ce Cas d'utilisation se présente ainsi : $\exists x1, x2, y1, y2, y3[f1(x1, y1) \wedge f2(x2, y2) \wedge f3(x2, y3)]$.

Les triplets générés à partir de la description de ce Cas d'utilisation sont respectivement :

- Directeur de vente, demande d'ajouter, nouveau produit
- Système, enregistre, produit
- Système, affiche, message de confirmation

6.3.2 Fonctionnement de l'algorithme de génération des triplets

Notre algorithme ou base de règles que l'on construit permet de découper des Cas d'utilisation et Récit utilisateurs en un trio de concepts (sujet, prédicat, objet). Par la suite, l'outil détermine la taille fonctionnelle en fonction du nombre de triplets et de prédicats atomiques générés. Quant aux prédicats composites, ils sont transformés en prédicats atomiques, en utilisant une logique des prédicats du 1er ordre. Nous expliquons dans cette section le fonctionnement de notre algorithme, les méthodes adoptées pour générer les sujets, les prédicats (verbes), les compléments (objets), ainsi que les triplets à partir des Cas d'utilisation et des Récits utilisateurs écrits en langage naturel.

6.3.3 Algorithme de génération du sujet

Nous considérons une fonction qui retourne un sujet à partir d'un Cas d'utilisation ou d'un Récit utilisateur passé en argument. Nous déclarons des variables telles que nom, verbe. Par la suite, on stocke dans un tableau l'ensemble des tags d'une phrase ou d'un Cas d'utilisation. L'algorithme parcourt le tableau contenant les phrases. Il considère le nom comme le tag du nom, en vérifiant si le tag obtenu équivaut au tag d'un nom commun, d'un nom pronom personnel ou non. Par la suite, il considère le verbe comme le tag du nom, en vérifiant si le tag obtenu correspond au tag du verbe. En termes de résultats, l'algorithme retourne un nom, dans le cas où le tag obtenu correspond au nom. Dans le cas contraire, il détermine l'ensemble des mots contenant le verbe. Si aucune de ces conditions

ne sont satisfaites, l'algorithme retourne des caractères vides. Le pseudo code pourrait être résumé ainsi :

```

Text subject (RichSentence richSentence) then
  Text firstNoun ← null;
  Text verb ← null;
  For each (TaggedWord taggedWord:
    richSentence.getTaggedWords()) do
    text tag ← taggedWord.tag ();
    if (Tag.NC.name () =tag OR Tag.NPP.name () =tag OR
      Tag.N.name () =tag)
    then
      firstNoun ← taggedWord.word();
      break;
    end
    if (Tag.V.name () =tag)
    then
      verb ← taggedWord.word();
      break;
    end
  end
  if (firstNoun <> null)
  then
    return firstNoun;
  else
    text words [ ] ← verb <> null? verb.split("\\s+") :
      new text[0];
    if (words.length > 1)
    then
      return words [0];
    end
  end
end
return null;
end

```

FIGURE 6.1 Algorithme de génération du sujet.

6.3.4 Algorithme de génération du prédicat (verbe)

Dans cet algorithme, nous définissons une fonction qui retourne la liste des prédicats ou verbes à partir des Cas d'utilisation fournis. Chaque verbe identifié dans une phrase est stocké dans une variable. On stocke dans un tableau l'ensemble des verbes trouvés de la phrase. Ensuite, on utilise un indice afin d'identifier l'ordre

des verbes. L'ensemble des tags d'une phrase seront stockés dans un tableau. En parcourant ce tableau, on vérifie si le tag obtenu est un verbe pour par la suite le stocker dans une variable. La variable de l'indice sera utilisée pour stocker la position du tag. Enfin, on fait appel à une fonction pour identifier la position des verbes, ainsi que leurs groupes. Le pseudocode se résume ainsi :

```

List<text> verb (RichSentence richSentence) then
  Text verb ← «";
  List<text> verbs;
  int firstVerbIndex ← 0;
  for each (int i ← 0; i < richSentence.getTaggedWords().
size
    (); i++) do
      TaggedWord taggedWord ← richSentence.getTaggedWords().
        get(i);
      if (Tag.V.name () =taggedWord.tag ())
      then
        then
          verb ← taggedWord.word();
          firstVerbIndex ← i;
          break;
        end
      end
    end

  verbHelper (verb, verbs, firstVerbIndex, richSentence);
  return verbs;
end

```

FIGURE 6.2 Algorithme de génération du prédicat (verbe).

6.3.5 Règle ou algorithme de génération de l'objet (complément)

Nous définissons une fonction qui retourne la liste des objets (compléments) à partir des Cas d'utilisation fournis par l'utilisateur. Ensuite, nous déclarons une variable qui sert à identifier si le tag est un verbe, un tableau contenant les compléments et un tableau contenant les noms. L'ensemble des tags d'une ou plusieurs phrases sont stockés dans un tableau. En parcourant le tableau, on vérifie si le tag est un nom et un complément pour ensuite l'ajouter au tableau contenant le nom

et le complément. Une fois qu'on ajoute chaque nom au tableau du complément, on fait vider le tableau. Ce pseudocode présente les étapes permettant de générer des objets à partir des phrases fournies écrites en langage naturel.

```

List<text> complement (RichSentence richSentence) then
  boolean verbFound ← false;
  List<text> complements;
  List<text> nouns;
  For each (TaggedWord taggedWord:
    richSentence.getTaggedWords()) do
    if (Tag.V.name () =taggedWord.tag ()) then
      verbFound ← true;
    end

    if (verbFound AND taggedWord.tag () =Tag.NC.name ())
    then
      nouns.add (taggedWord.word());
    end
    if (Tag.CC.name () =taggedWord.tag ()) AND!
      nouns.isEmpty ())
    then
      complements.add (text. join (" ", nouns));
      nouns.clear ();
    end
  end
  complements.add (text. join (" ", nouns));
  return complements;
end

```

FIGURE 6.3 Algorithme de génération de l'objet (complément).

6.3.6 Algorithme de génération des triplets (sujet, prédicat, objet)

L'objectif de cet algorithme est d'extraire des triplets à partir des Cas d'utilisation ou Récits utilisateurs écrits en langage naturel. Nous partons de l'hypothèse que l'écriture des exigences logicielles (Cas d'utilisation, Récits utilisateurs) se fait sous forme de triplets. Nous définissons une fonction qui retourne une liste de triplets à partir des phrases fournies. Par la suite, on invoque la fonction qui permet de localiser les sujets, verbes et compléments. Les triplets générés sont stockés dans

un tableau. Le pseudocode qui suit présente l'ensemble des étapes conduisant à la construction et génération des triplets.

```

List<Triplet> emit (RichSentence richSentence) then
  text subject ← subject(richSentence);
  List<text> verbs ← verb(richSentence);
  List<text> complements ← complement(richSentence);
  List<Triplet> results;
  if ((NOT isEmpty(subject)) AND (NOT verbs.isEmpty ()) AND
      (NOT complements.isEmpty ()))
  then
    richSentence.setStatus(SentenceStatus.PROCESSED);
  end
  for each (text verb: verbs) do
    results.add (new Triplet (subject, verb,
                             complements.get (0)));
  end
  if (complements.size () > 1)
  then
    for each (String verb: verbs) do
      results.add (new Triplet (subject, verb,
                               complements.get (1)));
    end
  end
  return results;
end
return Collections.emptyList ();
end

```

FIGURE 6.4 Algorithme de génération des triplets.

6.3.7 Génération des triplets par ellipse grammaticale

L'ellipse est une figure de style rhétorique visant à omettre un ou plusieurs éléments afin de rendre plus court une phrase, tout en favorisant la compréhension. Dans le cadre de l'ellipse grammaticale, on a tendance à omettre par exemple un verbe ou un objet. En effet, dans le domaine de rédaction des exigences logicielles, les analystes, dans le but d'éviter la répétition, omettent un prédicat (verbe) ou un nom (objet). Illustrons par un exemple un Cas d'utilisation où il y a une omission d'un objet : « Le système vérifie et enregistre les données ». Dans ce Cas

d'utilisation, il est question que le système vérifie les données. Par la suite, il va procéder à l'enregistrement des données. L'objet « données » a été donc omis dans la première partie de la phrase. Pour que l'outil découpe ce Cas d'utilisation en triplets, on procède par découpage par ellipse grammaticale. Nous présumons que la description des Cas d'utilisation se fait avec des prédicats de deux arguments $f(x, y)$. La description en logique des prédicats du 1er ordre pourrait être : $\exists y, \exists x \text{ tel que } f(x, y)$.

En d'autres termes, on pourrait écrire la formule ainsi : $\exists \text{objet}, \exists \text{sujet tel que : } \text{predicat}(\text{objet}, \text{sujet})$.

Sujet :

- x = le système

Objet :

- y = données

Prédicats :

- f1 = vérifie
- f2 = enregistre

La formule logique correspondant est la suivante : $\exists x, y[f1(x, y) \wedge f2(x, y)]$.

La description du Cas d'utilisation est découpée en deux triplets qui sont respectivement :

- Système, vérifie, données
- Système, enregistre, données

6.3.8 Algorithme du découpage par ellipse grammaticale

Cet algorithme décrit l'ensemble des règles qui permet de découper des Cas d'utilisation ou des Récits utilisateurs écrits en langage naturel en des triplets dans le cas où il y a une omission d'objets avec plusieurs prédicats (verbes).

```

text verbInfHelper (text verbInf) then
  text verb;
  int verbInfLength ← verbInf.length();
  if (verbInf.substring(verbInfLength - 3) = "yer")
  then
    verb ← verbInf.substring(0, verbInfLength - 3). concat("ie");
  else if (verbInf.substring(verbInfLength - 2) = "ir")
  then
    verb ← verbInf.substring(0, verbInfLength - 1);
  concat("t");
  end
  else if (verbInf.substring(verbInfLength - 2) = "re") then
    verb ← verbInf.substring(0, verbInfLength - 2);
  end
  else if (verbInf.substring(verbInfLength - 2) = "er") then
    verb ← verbInf.substring(0, verbInfLength - 1);
  end
  else then
    verb ← verbInf;
  end
  return verb;
end

```

FIGURE 6.5 Algorithme de découpage par ellipse grammaticale.

6.3.9 Génération des triplets par découpage multiple

Dans la description des Cas d'utilisation, il existe des phrases contenant plusieurs objets (compléments) et qui sont connectés par des connecteurs logiques (ET, OU : en français; AND, OR : en anglais), par des conjonctions de coordination (afin, lorsque : en français) ou par des points de ponctuation (,). Dans la littérature de génération automatique des textes, il existe des méthodes qui permettent d'agréger des phrases de structure (SPO) en utilisant les connecteurs logiques. Dans le cadre de notre outil, nous nous inspirons de ces méthodes pour procéder par

désagrégation. Illustrons par l'exemple le Cas d'utilisation suivant : « Le système vérifie les informations, enregistre les données ou retourne un message d'erreur ». Nous transformons chacun de ces scénarios des Cas d'utilisation en une suite de prédicats de deux arguments de la forme : objet, sujet tel que prédicat (objet, sujet). Autrement dit, $\exists y, \exists x \text{ tel que } f(x, y)$.

Nous transformons donc la description des scénarios de ce Cas d'utilisation en logique formelle par des variables pour représenter les sujets, prédicats et objets.

Sujet :

— x = Le système

Objets :

— y_1 = informations

— y_2 = données

— y_3 = Message d'erreur

Prédicats :

— f_1 =vérifie

— f_2 =enregistre

— f_3 =retourne

Ce qui nous donne la formule logique suivante : $\exists x[f_1(x, y_1) \wedge f_2(x, y_2) \wedge f_3(x, y_3)]$.

La description du Cas d'utilisation est découpée en trois triplets qui sont respectivement :

— Système, vérifie, informations,

— Système, enregistre, données

— Système, retourne, message d'erreur

6.3.10 Algorithme de découpage multiple avec plusieurs objets

Ce pseudocode présente les règles permettant de générer des triplets à partir des Cas d'utilisation ou récits utilisateurs par découpage multiple dans le cas où la description des scénarios des Cas d'utilisation contient plusieurs objets (compléments).

```

List<Triplet> emit (RichSentence richSentence) then
  text subject ← subject(richSentence);
  List<text> verbs ← verb(richSentence);
  List<text> complements ← complement(richSentence);
  List<Triplet> results;
  if ((NOT isEmpty(subject)) AND (NOT verbs.isEmpty ()) AND
      (NOT complements.isEmpty ()))
  then
    richSentence.setStatus(SentenceStatus.PROCESSED);
  end
  for each (text verb: verbs) do
    results.add (new Triplet (subject, verb, complements.get
(0)));
  end
  if (complements.size () > 1)
  then
    for each (text verb: verbs) do
      results.add (new Triplet (subject, verb, complements.get
(1)));
    end
  end
  return results;
end
return Collections.emptyList();
end

```

FIGURE 6.6 Algorithme de découpage multiple avec plusieurs objets (compléments).

6.3.11 Critères de génération des triplets

Nous définissons dans le tableau 6.1 un ensemble de critères nécessitant de générer ou découper des Cas d'utilisation ou Récits utilisateurs en triplets.

Critères	Langues		Explication
Connecteur logique	ET, OU	AND, OR	Découpage du CU en triplet
Conjonction de subordination	Afin		Découpage du CU en triplet
Ponctuation	,	,	Découpage du CU en triplet

TABLEAU 6.1 Critères de découpage des Cas d'utilisation ou Récits utilisateurs.

6.3.12 Algorithme de découpage multiple avec des connecteurs logiques

Ce pseudocode présente les règles permettant de générer des triplets à partir des Cas d'utilisation ou récits utilisateurs par découpage multiple dans le cas où la description des scénarios des Cas d'utilisation contient des connecteurs logiques et / ou des ponctuations.


```

List<text> verbHelper (text verb, List<text> verbs, int
firstVerbIndex,
    RichSentence richSentence)
{
    int firstIndex = firstVerbIndex + 1;
    int nextIndex = firstVerbIndex + 2;
    if (firstIndex < richSentence.getTaggedWords(). size ())
    then
        TaggedWord taggedWord =
            richSentence.getTaggedWords(). get(firstIndex);
        text tag = taggedWord.tag ();
        if (tag=FrenchTag.V.name () OR tag=FrenchTag.VINF.name ()
OR
            tag=FrenchTag.VPP.name ())
        then
            verb = verbInfHelper (richSentence.getTaggedWords()
                . get(firstIndex). word ());
        end
    else if (punchHelper(taggedWord) AND (NOT verb. isEmpty ()))
    then
        verbs.add(verb);
        verb = «";
        verbHelper (verb, verbs, firstIndex, richSentence);
        nextIndex++;
    end
end
if (nextIndex < richSentence.getTaggedWords(). size ())
then
    TaggedWord taggedWord =
        richSentence.getTaggedWords(). get(nextIndex);
    text tag = taggedWord.tag ();
    if (tag=FrenchTag.V.name () OR tag=FrenchTag.VINF.name ()
OR
        tag=FrenchTag.VPP.name ())
    then
        verb = verbInfHelper (richSentence.getTaggedWords()
            . get(nextIndex). word ());
        end
    else if (punchHelper(taggedWord) AND (NOT verb. isEmpty ()))
    then
        verbs.add(verb);
        verb = «";
        verbHelper (verb, verbs, nextIndex, richSentence);
    end
end
if ((NOT verb. isEmpty ()))
then
    verbs.add(verb);
end
return verbs;
end

```

FIGURE 6.7 Algorithme de découpage multiple avec des connecteurs logiques.

6.4 Résultats et validation des Cas d'utilisation générés en triplets

Notre outil décompose 104 Use Cases et User Stories en des triplets, à raison de 65 Use Cases en anglais et 39 Use Cases en français. L'outil génère environ 359 triplets en anglais, à raison de 8 triplets qui sont sémantiquement erronés, ambigus ou syntaxiquement incorrects. Cela vient du fait que l'outil retourne le sujet pour l'objet, c'est-à-dire que le sujet est à la fois sujet et objet (exemple : uSleuth, displays, uSleuth). Dans d'autres cas, l'outil identifie un nom pour un verbe. Tandis que l'outil génère environ 334 triplets en français, à raison de 6 triplets qui sont syntaxiquement et sémantiquement erronés et 8 triplets qui sont ambigus. Ils sont analysés et évalués en fonction d'un ensemble de critères présentés au tableau 6.2 pour vérifier leur qualité syntaxique et sémantique. Les résultats de cette expérience sont en français et en anglais. La majorité des triplets générés sont syntaxiquement et sémantiquement corrects, à l'exception de certaines phrases pour lesquelles il existe certaines erreurs et ambiguïtés.

Langues	Syntaxe		Sémantique		
	Correcte	Erroné	Correcte	Ambigu	Erroné
Anglais	355	4	347	8	4
Français	328	6	320	8	6

TABLEAU 6.2 Triplets générés à partir des Use Cases ou User Stories.

6.5 Méthode d'évaluation des Cas d'utilisation générés en triplets

D'après la littérature consultée, il n'existe pas de méthode particulière pour évaluer l'efficacité, ainsi que la performance des Cas d'utilisation découpés en triplets. En d'autres termes, il n'existe pas jusqu'à présent une approche ou mesure par-

ticulière qui pourrait permettre de mesurer la performance des outils agrégateurs et désagrégateurs. Dans le cadre de notre recherche, nous avons testé notre implémentation à partir d'un ensemble de Cas d'utilisation découpés manuellement par des humains en triplets. Par la suite, nous avons comparé les résultats produits manuellement avec les résultats générés par l'outil. De plus, les Cas d'utilisation découpés en triplets par l'outil sont analysés et évalués par le jugement humain à partir d'un ensemble de critères prédéfinis à la section. Ces critères d'évaluation sont basés sur la qualité syntaxique et sémantique de l'ensemble des triplets générés par l'outil. Nous attribuons un score compris entre 0 et 1 pour décrire la qualité syntaxique des triplets générés par l'outil. Nous avons attribué une note de 0 lorsque le triplet ne respecte pas syntaxiquement la structure sujet, prédicat, objet ; une note de 1 lors que le triplet est correct, c'est-à-dire qu'il respecte la structure sujet, prédicat, objet. Dans le cadre de la description de la qualité sémantique du triplet, nous nous accentuons sur le degré d'agrégation ou de restitution du texte original pour s'assurer que les Cas d'utilisation ont été décomposés sans perte d'information essentielle. Nous attribuons un score de 0 lorsque le triplet est sémantiquement erroné ; un score de 1 lorsque le triplet est sémantiquement compréhensible, mais contenant certaines erreurs. Une note de 2 est attribuée lorsque le triplet est sémantiquement correct et identique, avec la possibilité de restituer le texte original sans perte d'information essentielle. Le tableau 6.3 présente, en termes de rappel, les résultats de cette expérience en français et en anglais. La majorité des triplets générés sont syntaxiquement et sémantiquement corrects.

Note	Syntaxe		Sémantique		
	1	0	2	1	0
Langues	Correcte	Erroné	Correcte	Ambigu	Erroné
Anglais	355	4	347	8	4
Français	328	6	320	8	6

TABLEAU 6.3 Évaluation des triplets générés à partir des Use Cases ou User Stories.

6.6 Module d'obtention de la taille fonctionnelle à partir des triplets

Dans cette section, nous présentons le deuxième module de l'outil, qui permet l'obtention de la taille fonctionnelle des logiciels à partir des triplets. Ce module détermine le nombre de prédicats atomiques de chaque triplet constituant des mouvements de données. Par la suite, il fait correspondre à chaque prédicat le type de mouvement de données (Entrée, Sortie, Lecture, écriture).

6.6.1 Processus de mesurage de la méthode COSMIC

La taille fonctionnelle (TF) est basée sur des fonctionnalités du logiciel. L'idée est de déterminer la quantité de fonctionnalités fournies à un utilisateur pour un produit logiciel donné. Cela sous-entend que la TF représente la taille du logiciel dérivée en quantifiant les fonctionnalités utilisateurs requises [ISO 14143]. Il existe différentes méthodes de mesure de la taille fonctionnelle. Dans le cadre de cette thèse, nous avons adopté la méthode COSMIC. Elle consiste à appliquer un ensemble de principes, de règles et de processus pour mesurer les exigences fonctionnelles des utilisateurs d'un logiciel donné. Le résultat est une valeur numérique telle que définie par l'ISO et représente la TF du logiciel [ISO 19761].

En COSMIC, on mesure les mouvements de données applicables par groupes de données dont chacun est manipulé par des processus fonctionnels du logiciel. Les mouvements de données représentent les prédicats atomiques de chaque triplet, c'est à-dire, Entrée (Entry), Sortie (Exit), Lecture (Read), écriture (Write). Les groupes de données sont les objets des triplets, tandis que les scénarios des Cas d'utilisation ou Récits utilisateurs sont entre autres les processus fonctionnels qui sont découpés en triplets (sujet, prédicat, objet). La figure 6.8 résume le processus de mesure de la méthode COSMIC.

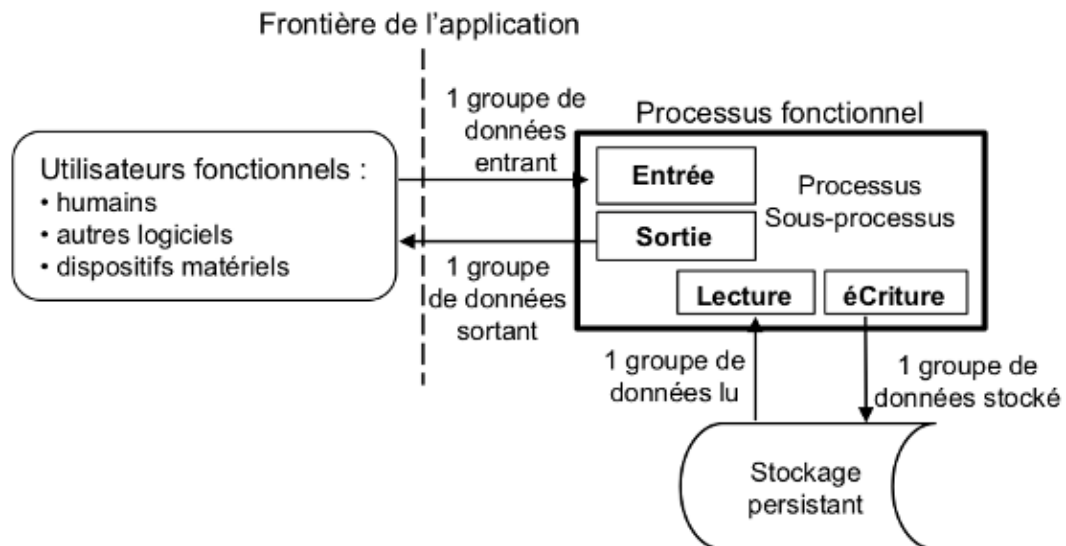


FIGURE 6.8 Processus de mesure de la méthode COSMIC (Abran et al, 2020).

6.6.2 Règle ou algorithme d'obtention de la taille fonctionnelle

Nous avons appliqué, dans le cadre de cette recherche, trois (3) niveaux de règles. Le premier niveau consiste à appliquer des règles pour mettre en association le sujet, le prédicat et l'objet. Le deuxième niveau de règles permet d'extraire les triplets à partir des Cas d'utilisation écrits en langage naturel. Par la suite, nous

avons appliqué un ensemble de règles pour inférer la TF et identifier les types de mouvements de données. La taille d'un processus fonctionnel est égale au nombre total de ses mouvements de données. Chaque mouvement de données correspond à un prédicat atomique de chaque triplet et a une taille de 1 point de fonction COSMIC (PFC). L'obtention de la TF des logiciels résulte donc de la somme des prédicats atomiques des processus fonctionnels. Dans le troisième niveau, l'outil apprend de nouvelles règles pour améliorer l'extraction des règles du deuxième niveau. En d'autres termes, l'outil que nous avons développé ne fait pas seulement que mettre en œuvre des règles. Par la mise en œuvre de ses règles, il se donne des moyens d'apprendre de nouvelles règles complémentaires par rapport aux premières règles qui lui sont imposées. Ce troisième niveau de règles est décrit dans la section « capacité d'apprentissage automatique de l'outil ». Grosso modo, pour mesurer la TF d'un logiciel on applique les règles suivantes :

1. On affecte une unité de mesure, 1 PFC, à chaque mouvement de données (Entrée, Sortie, Lecture, écriture) identifié dans chaque processus fonctionnel.
2. On multiplie le nombre de mouvements de données (prédicats atomiques) de chaque type par sa taille unitaire.
3. On additionne les tailles de l'étape 2 pour chacun des types de mouvements de données dans le processus fonctionnel. La formule de la TF des logiciels en COSMIC se résume ainsi : Taille (processus fonctionnel) = taille (Entrées) + taille (Sorties) + taille (Lectures) + taille (écritures).

6.6.3 Règles d'identification des mouvements de données

Un mouvement de données est une opération système, un événement ou un composant fonctionnel de base qui déplace un groupe de données (ISO 19761). Il existe en COSMIC quatre (4) types de mouvements de données qui sont respectivement :

Entrée, Sortie, Lecture, écriture. Dans le cadre de cette recherche, l'outil développé identifie les types de mouvements de données des logiciels à mesurer, après avoir généré les triplets.

6.6.4 Règle d'identification des mouvements de données de type Entrée (E/E)

Un mouvement de données de type Entrée (E) déplace un groupe de données à travers la frontière d'un utilisateur fonctionnel vers le processus fonctionnel (ISO 19761). On applique les règles suivantes afin de permettre à l'outil d'identifier les mouvements de données de type Entrée (E) : Une entrée (E) doit être :

1. Un mouvement de données provenant du côté d'un utilisateur fonctionnel (humain, autres logiciels, dispositifs matériels) est traité comme un mouvement de donnée de type Entrée (E).
2. Toute fonctionnalité de demande de réception des données d'entrée doit être incluse dans le type de données de type Entrée (E). Il est à remarquer qu'il y a en annexe un ensemble de règles et de verbes qui correspondent à des mouvements de données de type Entrée (E). Ainsi, on dispose d'une base de règles avec des règles associées aux différents verbes pour lesquels l'addition est complexe. Par exemple, le verbe « enregistrer » correspond en COSMIC à un mouvement de données de type écriture (C). Dans cet exemple, c'est le système qui invoque la méthode ou l'opération « enregistrer ». Par ailleurs, lorsque c'est l'utilisateur qui demande d'enregistrer au système, le prédicat ou verbe « enregistrer » correspond à un mouvement de données de type Entrée (E).

6.6.5 Règle d'identification des mouvements de données de type Sortie (S/X)

Un mouvement de données de type Sortie (S) déplace un groupe de données d'un processus fonctionnel vers un utilisateur fonctionnel (ISO 19761). Une Sortie (S) doit :

1. Envoyer des données d'un seul groupe de données à partir d'un processus fonctionnel vers l'utilisateur fonctionnel.
2. Prendre en compte toutes les manipulations de formatage et de présentation des données requises, y compris le traitement requis pour envoyer les attributs de données à l'utilisateur fonctionnel. Toutefois, ces manipulations ne doivent pas impliquer un autre type de mouvement de données.

6.6.6 Règle d'identification des mouvements de données de type Lecture (L/R)

Les mouvements de données de type Lecture (L) déplacent des groupes de données à partir du stockage persistant vers un processus fonctionnel (ISO 19761). Une lecture doit :

1. Chercher un seul groupe de données dans le stockage persistant.
2. Prendre en compte les traitements logiques et / ou calculs mathématique nécessaire à la lecture des données.
3. Faire partie d'une fonctionnalité qui requiert une lecture.

6.6.7 Règle d'identification des mouvements de données de type écriture (C/W)

Les mouvements de données de type écriture (C) déplacent des groupes de données à partir d'un processus fonctionnel vers le stockage persistant (ISO 19761). Une écriture doit :

1. Déplacer un unique groupe de données vers un stockage persistant.

2. Tenir compte tout le traitement logique et/ou calcul mathématique nécessaire pour créer les données à sauvegarder.

En résumé, nous avons implémenté et appliqué un ensemble de règles de COSMIC pour déterminer la TF et identifier les types de mouvements de données. La taille d'un processus fonctionnel est égale au nombre total de ses mouvements de données. Chaque mouvement de données correspond à un prédicat atomique de chaque triplet et a une taille d'un point de fonction COSMIC (PFC). Il est à souligner ces règles sont supportées par une base de données en mémoire, c'est-à-dire une liste de verbes qui correspondent à des types de mouvements de données. Le tableau 6.4 résume la liste des règles de correspondance qui sont implémentées pour identifier les types de mouvements de données de chaque triplet généré par l'outil.

ID	Définition des règles de correspondance (RC)
RC01	Tout mouvement de données provenant de l'utilisateur fonctionnel (humain, autres logiciels, dispositifs matériel) est considéré comme un mouvement de données de type Entrée (E).
RC02	Toute fonctionnalité de demande de réception des données d'entrée doit être incluse dans le mouvement de données de type Entrée (E).
RC03	Tout mouvement de données provenant d'un processus fonctionnel vers l'utilisateur fonctionnel est considéré comme un mouvement de données de type Sortie (S).
RC04	Toutes les manipulations de formatage et de présentation des données requises, pour envoyer les attributs de données à l'utilisateur fonctionnel est inclus dans le mouvement de type Sortie (S).
RC05	La recherche d'un groupe de données dans le stockage persistant est considérée comme une Lecture (L).
RC06	Les traitements logiques et / ou calculs mathématique nécessaire à la lecture des données sont inclus dans le mouvement de type Lecture (L).
RC07	Toute fonctionnalité de demande de lecture est incluse dans le mouvement de données de type Lecture (L).
RC08	Le déplacement d'un unique groupe de données vers un stockage persistant est considéré comme une écriture.
RC09	Les traitements logiques et / ou calcul mathématique nécessaires pour créer données à sauvegarder sont inclus dans un mouvement de type écriture.
RC10	Une exigence pour supprimer un groupe de données du stockage persistant est considéré comme une écriture.

TABLEAU 6.4 Règles de correspondance des mouvements de données.

6.7 Description d'un Cas d'utilisation et sa taille fonctionnelle manuelle

Cet exemple décrit un Cas d'utilisation en anglais. Il est découpé manuellement en plusieurs triplets. Par la suite, nous identifions les groupes de données, les mouvements de données (Entrée, Sortie, Lecture, écriture) et calcule la taille fonctionnelle manuelle. Cas d'utilisation : Add product Utilisateur fonctionnel : Sales manager Description du processus fonctionnel : The sales manager asks to add a new product. The system verifies the sales manager credentials and displays the new product form or displays a credential error message. The sales manager enters the new product information and asks the system to save the new product. The system verifies the data, records the product, and returns a confirmation message for the addition of the new product or an error message if the product already exists. Le tableau 6.5 présente les différents processus fonctionnels, les groupes de données, les mouvements de données identifiés et la taille fonctionnelle manuelle obtenue.

Functional process elements	Data Groups	E	X	R	W	Sum of CFP
Asks to add a new product	Credentials	1				1
Verifies the sales manager credentials	Credentials			1		1
Displays the new product form	[New product form]					-
Displays a credential error message	Error message		1			1
Enters the new product information	New product	1				1
Verifies the data	Data			1		1
Records the product	Product				1	1
Returns a confirmation message	Message		1			1
Total		2	2	2	1	7

TABLEAU 6.5 Exemple de mesure manuelle d'un processus fonctionnel.

6.8 Description d'un Cas d'utilisation et sa taille fonctionnelle automatique

L'outil décompose le Cas d'utilisation en plusieurs triplets. Par la suite, il identifie les mouvements de données Entré (E), Sortie (S), Lecture (L) et écriture (E) et détermine la taille fonctionnelle. Il est à noter que la taille fonctionnelle obtenue de l'outil est identique à celle obtenue par les experts humains. La figure 6.9 présente le résultat de la taille fonctionnelle automatisée obtenue par l'outil (Gérançon *et al.*, 2021)².

English ▾ Use word document ▾ Use case is in English ▾ Report version 2 ▾

Select file Choisir un fichier Example-Add product.docx Submit

Download

Triplets list and other statistics

Functional Process	Triplets	Entry	Exit	Read	Write	Sum CFP	Include
Add product	sales manager,asks,product	1	0	0	0	1	<input checked="" type="checkbox"/>
Add product	system,verifies,sales	0	0	1	0	1	<input checked="" type="checkbox"/>
Add product	system,displays,form	0	0	0	0	0	<input type="checkbox"/>
Add product	system,displays,error	0	1	0	0	1	<input checked="" type="checkbox"/>
Add product	sales manager,enters,product	1	0	0	0	1	<input checked="" type="checkbox"/>
Add product	system,verifies,data	0	0	1	0	1	<input checked="" type="checkbox"/>
Add product	system,records,product	0	0	0	1	1	<input checked="" type="checkbox"/>
Add product	system,returns,confirmation	0	1	0	0	1	<input checked="" type="checkbox"/>
Total Of Sum CFP		2	2	2	1	7	

FIGURE 6.9 Exemple de mesure automatisée de la taille fonctionnelle.

6.9 Capacité d'apprentissage automatique de l'outil

La faculté d'apprendre à partir des données ou expériences passées et de s'adapter est une des caractéristiques fondamentales de la vie humaine. Cette faculté est

2. https://www.thinkmind.org/index.php?view=articlearticleid=icsea2021_14018004

fondamentale à l'humain dans les étapes élémentaires de sa vie, telles que la reconnaissance de la voix d'une personne, la capacité d'apprendre à parler ou à marcher, etc. L'apprentissage automatique (Machine Learning) est une tentative de reproduire avec une machine les capacités cognitives de l'humain par le moyen de l'apprentissage. L'idée est de permettre aux machines ou aux ordinateurs d'apprendre à partir des données sans être explicitement programmés (Mitchell, 1997)(LeCun *et al.*, 2015)(Goodfellow *et al.*, 2016). L'apprentissage automatique est donc un ensemble de techniques qui permettent de donner aux machines l'habilité d'apprendre à partir des données en vue de résoudre un problème ou d'accomplir une tâche cognitive (Domingos, 2012)(LeCun *et al.*, 2015)(Goodfellow *et al.*, 2016). Dans le cadre de cette recherche, nous avons implémenté préliminairement une composante de gestion des heuristiques qui fait en sorte que l'outil développé apprend à exclure les triplets redondants et superflus. En effet, une fois généré la liste des triplets, l'outil identifie les mouvements de données et détermine la taille fonctionnelle. Puisque la base de données de l'outil fait une relation entre les prédicats atomiques et les mouvements de données, les triplets sont en général tous inclus, une fois qu'ils soient générés par l'outil. Cependant, les prédicats qui ne correspondent pas à des mouvements de données (événements ou opérations du système) ne sont pas comptés comme des points de fonction. Par ailleurs, nous pouvons permettre aux experts ou aux utilisateurs d'exclure des triplets qui paraissent redondants ou superflus. À partir du moment où on exclut un triplet, nous permettons à ce que la base de données soit mise à jour automatiquement. Dans le cas où le mesureur/l'utilisateur exclut à plusieurs reprises (deux ou trois fois) un triplet, nous faisons en sorte que l'outil exclut systématiquement ce triplet les prochaines fois qu'on fasse la mesure. L'outil apprend donc une nouvelle règle par induction sur une succession de cas. Nous avons encodé des heuristiques ou d'autres expertises des experts mesureurs humains de manière explicite sous forme de règles. L'idée est de permettre à l'outil d'apprendre à résoudre un pro-

blème par lui-même par induction. En d'autres termes, l'outil se met à acquérir une expertise par sa mise en œuvre, de sorte qu'il se donne des règles de gestion de données par l'usage ou par l'apprentissage par renforcement. Il est à signaler que le module d'apprentissage automatique de l'outil développé pourrait faire l'objet de nos recherches ultérieures.

6.10 Algorithme d'heuristique (exclusion des triplets)

Cet algorithme résume la méthode utilisée pour permettre à l'outil d'être capable d'exclure systématiquement des triplets redondants dans le cas où l'utilisateur les exclut à deux ou trois reprises. Chaque triplet possède une signature unique. À chaque fois que le mesureur ou l'utilisateur exclut un triplet, on enregistre sa signature dans la base de données temporaire. Lorsque le mesureur soumet par exemple un Cas d'utilisation, l'outil vérifie dans sa base de données temporaire les triplets qui ont été exclus. Si l'utilisateur exclut à nouveau ces triplets, l'outil les supprime de la base de données temporaire et ne compte pas les prédicats atomiques (verbes) de ces triplets comme des mouvements de données pour une prochaine mesure. Dans le cas où le mesureur inclut ces triplets, l'outil les inclura systématiquement pour une prochaine mesure. Les détails de l'implémentation de cet algorithme se trouvent à l'annexe B.

```
HttpStatus excludeTriplet (Triplet triplet)
{
    Text textBuilder = triplet. getSubject ()
    {
        triplet. getVerb () +
        triplet.getComplement();
        HttpStatus excludedTriplets.add(textBuilder);
        return HttpStatus.OK;
    }

    HttpStatus includeTriplet (Triplet triplet)
    {
        Text textBuilder = triplet. getSubject () +
        triplet.getVerb() +
        triplet.getComplement ();
        excludedTriplets.remove(textBuilder);
        return HttpStatus.OK;
    }
}
```

FIGURE 6.10 Algorithme d'exclusion des triplets redondants par l'outil.

6.11 Choix des technologies

Pour des raisons techniques, nous n'avons pas implémenté notre système comme un système à base de règles. Nous avons développé un prototype en Java comme étant une preuve de concepts, sans solliciter une librairie à base de règles. Par ailleurs, la spécification de notre système est une solution cognitive fondée sur un système basé sur des mémoires (mémoire sémantique, mémoire procédurale) qui se connectent ensemble pour permettre de faire des inférences en termes d'estimation de la taille fonctionnelle des logiciels. Le tableau 6.6 décrit les outils technologiques utilisés pour le développement de notre outil d'automatisation de

la taille fonctionnelle des logiciels.

Outils technologiques	Description
Java 8	Langage de programmation utilisé pour implémenter les classes de notre outil d'automatisation.
IntelliJ IDEA	Environnement de développement intégré utilisé pour le développement de notre outil d'automatisation.
JavaScript ES6	Langage de programmation Web utilisé pour implémenter les mécanismes de la création d'interface utilisateur des résultats de mesure.
ReactJS 16.13.1	Bibliothèque de JavaScript utilisée pour la création d'interface utilisateur des résultats de mesure.
Git 2.33.1	Outil utilisé pour faire la gestion des versions.
GitHub	Outil utilisé pour le stockage du code source.

TABLEAU 6.6 Outils technologiques.

6.12 Synthèse

Nous avons présenté dans ce chapitre le premier module de notre d'outil d'automatisation de la TF des logiciels à partir des documents de spécifications. L'outil génère automatiquement des triplets à partir textes écrits en langage naturel. Ensuite, il détermine la Taille fonctionnelle. L'idée est que nous présumons que la rédaction des exigences fonctionnelles du logiciel se fait avec des prédicats dyadiques. Nous avons ajusté le module NLP en appliquant un ensemble de règles qui permettent de construire et d'extraire des triplets à partir des textes écrits en langage naturel. De plus, nous avons montré que l'outil proposé du processus d'automatisation de la TF est capable d'apprendre automatiquement de nouvelles règles en vue d'améliorer l'extraction des premières règles qui lui sont imposées. En somme, notre outil intègre un ensemble de techniques en intelligence artificielle complexe permettant de soutenir la tâche de mesure des points de fonction COSMIC à partir des spécifications fonctionnelles.

CHAPITRE VII

ÉVALUATION ET ANALYSE DES RÉSULTATS

7.1 Introduction

Dans ce chapitre, nous présentons les résultats de mesure de la taille fonctionnelle de dix (10) projets, à raison de cinq (5) documents de spécification d'exigences logicielles écrites en anglais et de cinq (5) documents de spécification d'exigences logicielles écrites en français. Premièrement, les experts certifiés en COSMIC ont déterminé manuellement la taille fonctionnelle de chacun de ces projets selon la procédure de mesure indiquée au chapitre VI. Les résultats de mesure manuelle des cinq documents de spécification d'exigences logicielles écrites en anglais sont publiés et disponibles sur le site Internet de COSMIC. Par la suite, nous utilisons l'outil développé pour déterminer automatiquement la taille fonctionnelle de ces projets. Nous comparons les résultats générés par l'outil par rapport aux résultats obtenus et publiés par les experts mesureurs. Nous décrivons les écarts observés, suite aux comparaisons des résultats automatisés et manuels. Ensuite, pour les cinq (5) projets dont les exigences étaient rédigées en français, ma codirectrice de recherche a effectué la mesure manuelle. Elle est reconnue comme experte de la méthode COSMIC. Le reste du processus est le même que pour les projets dont les exigences étaient écrites en anglais.

7.2 Résultats de la taille fonctionnelle des cinq (5) projets en anglais

Les dix documents de spécification d'exigences logicielles mesurées sont des Use Case et des User Stories qui sont généralement décrits sous forme de texte. L'outil cible dans ces variantes de cas la structure (sujet, prédicat, objet). Ensuite, il génère l'ensemble des triplets découlant de chacun des Use Case ou des User Stories. Finalement, il détermine la taille fonctionnelle en additionnant la liste des prédicats atomiques, qui sont entre autres, des mouvements de données, identifie et totalise les différents types de mouvements de données (Entré, Sortie, Lecture, écriture) en fonction d'un ensemble de règles de correspondance prédéfinies.

7.2.1 Projet : « Resto Sys »

Le projet #1, dénommé « Resto Sys »¹, est un système de gestion des restaurants. Il est composé de deux modules qui sont respectivement une application Web et une application mobile. L'application Web contient vingt-sept (27) Cas d'utilisation, tandis que, l'application mobile contient trois (3) Cas d'utilisation. La figure 7.1 représente le diagramme global des Cas d'utilisation de « Resto Sys ».

1. <https://cosmic-sizing.org/publications/restosys-case-v1-2/>

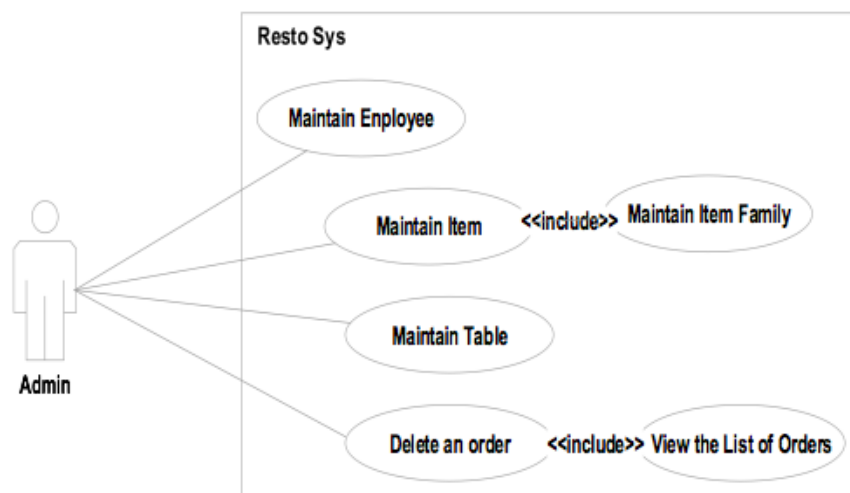


FIGURE 7.1 Diagramme des cas d'utilisation de « Resto Sys » (Selami *et al.*, 2019)

7.2.2 Résultat de la mesure manuelle et automatisée du projet #1 « Resto Sys»

On a constaté que les tailles fonctionnelles du mesurage manuel et automatisé ont enregistré un écart de deux (2) points de fonction COSMIC (2 PFC), soit 1,68%. Cela signifie que l'outil que nous proposons présente pour ce projet des résultats automatisés cohérents avec les résultats manuels validés et publiés par des experts, avec une moyenne de 98,32%, comme montre le tableau 7.1. Les mouvements de données en Entrée (E), en Lecture (L) et en écriture (C) n'ont enregistré aucun écart. Par contre, un écart de +2 PFC a été enregistré pour les mouvements de données en Entrée. Pour identifier cet écart, un examen de la documentation de spécification des exigences logicielles a été effectué et nous avons constaté que l'outil n'est pas capable de générer des triplets pour le scénario suivant : « The mobile sends the modified items to the web app ». Il est à mentionner que les

résultats de mesurage manuel publiés par les experts et les résultats de mesurage automatisé obtenus par l'outil sont présentés à l'annexe C. Le tableau 7.1 résume les résultats de la taille fonctionnelle manuelle et automatisée du projet « Resto Sys ».

Mouvements de données	Mesure manuelle (PFC)	Mesure automatisée (PFC)	Écart	Écart (%)	Précision (%)
Entré (E)	42	42	0	0,00%	100,00%
Sortie (S)	46	44	+2	4,35%	95,65%
Lecture (L)	16	16	0	0%	100,00%
écriture (C)	15	15	0	0,00%	100,00%
Total	119	117	2	1,68%	98,32%

TABLEAU 7.1 Résultats de mesure manuelle et automatisée de l'application « Resto Sys ».

7.2.3 Projet #2 : « ACME Car Hire »

Le projet #2, dénommé « ACME Car Hire »², est une application d'affaires (business application) orientée Web qui permet à des utilisateurs de faire des réservations de location de véhicule en ligne. Le document de spécification d'exigences fonctionnelles de cette application comporte huit (8) Cas d'utilisation (Use Cases).

2. <https://cosmic-sizing.org/wp-content/uploads/2018/08/ACME-Car-Hire-Case-Study-v1.0-1-1.pdf>

7.2.4 Résultat de la mesure manuelle et automatisée du projet #2 « ACME Car Hire »

On a remarqué que la taille fonctionnelle obtenue par notre outil présente des résultats automatisés cohérents en comparaison avec les résultats manuels validés et publiés par des experts, avec une moyenne de 96,97%, soit 32 PFC sur 33 PFC. Un faible écart d'un point de fonction COSMIC (1 PFC), soit 3,03% a été enregistré. Cela provient du fait que l'outil identifie le verbe « Update » dans ce scénario de Cas d'utilisation : « The system updates customer record » comme un nom. Cependant, le verbe « Update » correspond à un mouvement de données de type « écriture ». Quant aux mouvements de données en Entrée (E) et en écriture (C), ils ont respectivement enregistré un écart de +1CFP et -1CFP. Par ailleurs, aucun écart en Lecture (L) n'a été enregistré. Le tableau 7.2 résume les résultats de taille fonctionnelle manuelle et automatique, ainsi que les écarts enregistrés. Il est à noter que les résultats du mesurage manuel validés et publiés par les experts, ainsi que les résultats du mesurage automatisé obtenus par l'outil sont décrits en détails en annexe.

Mouvements de données	Mesure manuelle (PFC)	Mesure automatisée (PFC)	Écart	Écart (%)	Précision (%)
Entré (E)	8	9	-1	-12,50%	87,50%
Sortie (S)	14	14	0	0,00%	100,00%
Lecture (L)	9	8	+1	11,11%	88,89%
<u>écriture</u> (C)	2	1	-1	-50,00%	50,00%
Total	33	32	1	3,03%	96,97%

TABLEAU 7.2 Résumé des résultats de mesurage manuel et automatisé du projet « ACME Car Hire ».

7.2.5 Projet #3 : « Rice Cooker »

Le projet #3 est une étude de cas présentant le premier prototype de l'application «RiceCooker»³. C'est un logiciel d'application à temps réel qui contient trois (3) Cas d'utilisation (appelés « functional processes » dans le document). En ce qui a trait à la première version du prototype de cuiseur de riz, plus deux (2) descriptions de modifications appliquées à la deuxième version du prototype du « Rice Cooker ». La figure 7.2 montre les différents utilisateurs fonctionnels du système « Rice Cooker ». Étant donné que le logiciel interagit avec des composants matériels, cette figure illustre les interactions en Entrée (à gauche), en Sortie (à droite) et en Lecture/écriture (en dessous).

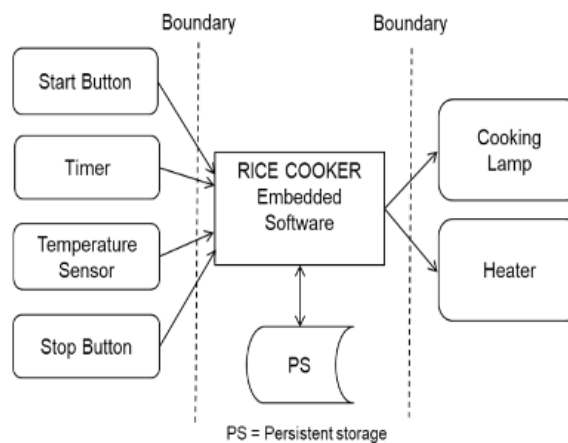


FIGURE 7.2 Utilisateurs fonctionnels du « Rice Cooker»

(Abran *et al.*, 2018)

3. <https://cosmic-sizing.org/wp-content/uploads/2018/08/Rice-Cooker-2.0.1-GoogleDocs.pdf>

7.2.6 Résultat de la mesure manuelle et automatisée du projet « Rice Cooker »

La mesure de la taille fonctionnelle automatisée obtenue par l'outil se rapproche à celle des experts humains avec une moyenne de 100%. Par ailleurs, les mouvements de données en Entrée (E) et Lecture (L) ont enregistré respectivement un écart de +2 CFP et -2 CFP. Les mouvements de données en Sortie (S) et en écriture (C) n'ont enregistré aucun écart. Le tableau 7.3 résume les résultats de la taille fonctionnelle obtenus par les experts humains, ceux de la taille fonctionnelle obtenus de l'outil et les écarts entre le mesurage manuel et le mesurage automatisé. Par ailleurs, les résultats du mesurage manuel des experts, ainsi que les résultats du mesurage automatisé obtenus par l'outil sont présentés en annexe.

Mouvements de données	Mesure manuelle (PFC)	Mesure automatisée (PFC)	Écart	Écart (%)	Précision (%)
Entré (E)	42	42	0	0,00%	100,00%
Sortie (S)	46	44	+2	4,35%	95,65%
Lecture (L)	16	16	0	0%	100,00%
<u>écriture</u> (C)	15	15	0	0,00%	100,00%
Total	119	117	2	1,68%	98,32%

TABLEAU 7.3 Résultats de mesure manuelle et automatisée du projet « Rice Cooker ».

7.2.7 Projet #4 : « uObserve »

uObserve est un outil qui permet de prendre en charge les tests d'utilisabilité des interfaces utilisateurs. Il enregistre des données audio et vidéo d'un utilisateur qui exécute une application d'espionnage. uObserve est composé de deux (2) sous-systèmes qui sont respectivement uSleuth et uSpy. uSleuth est une application

serveur qui s'exécute sur un poste de travail. Tandis que uSpy est une application client qui envoie un journal d'événements à uSleuth via un réseau local. Quant à uSleuth, il enregistre et lit les fichiers audio/vidéo (Trudel et Lavoie, 2007). Le document de spécifications d'exigences logicielles de cet outil contient huit (8) Cas d'utilisation (Use Cases) écrits en langage naturel. La figure 7.3 présente le diagramme de Cas d'utilisation global de « uObserve ».

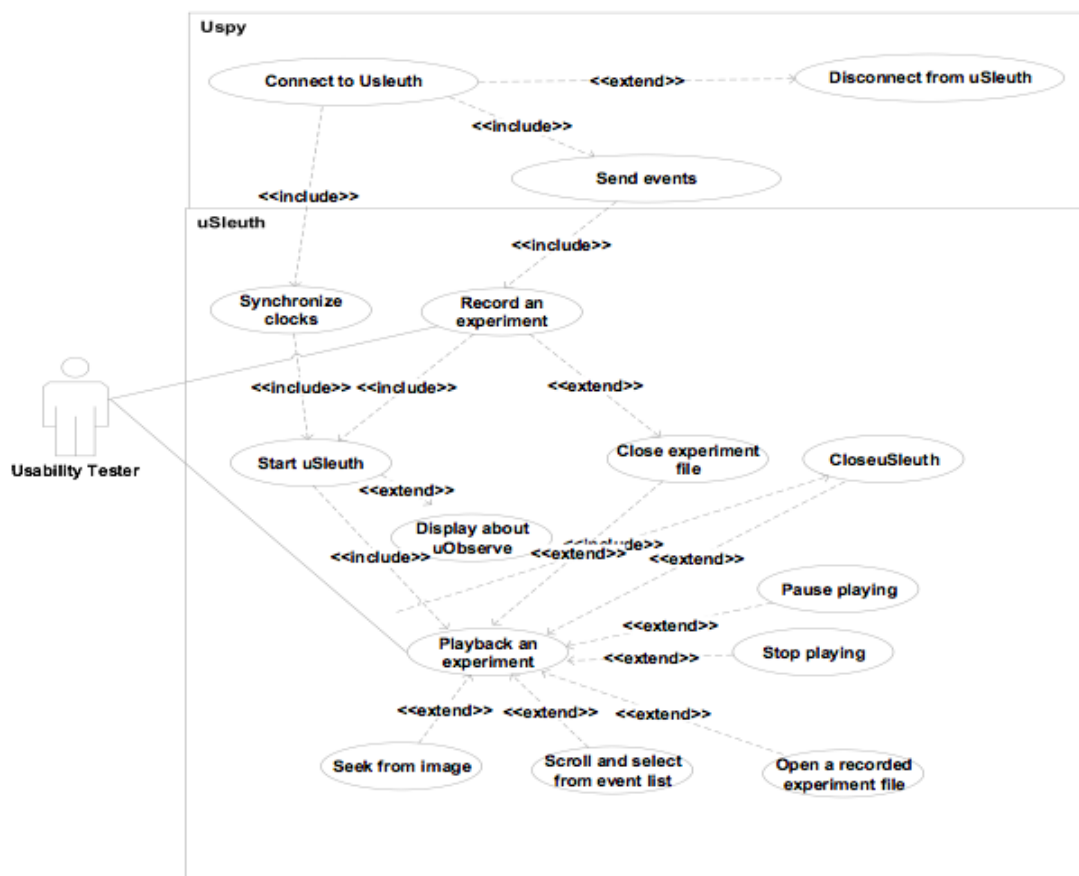


FIGURE 7.3 Diagramme de Cas d'utilisation de uObserve

(Trudel et Lavoie, 2007)

7.3 Résultat de la mesure manuelle et automatisée du projet «uObserve»

On a remarqué que les mouvements de données en « Entrée », « Sortie », « Lecture » et « écriture » ont enregistré respectivement un écart de -1 PFC (-4%), +2 PFC (5,26%), -1 PFC et -1 PFC (-33,33%). Par ailleurs, un écart de -1 PFC (-1,20%) a été enregistré entre le mesurage manuel et automatisé pour le nombre total de mouvements de données (|83 PFC - 84 PFC|). Le point de fonction COSMIC additionnel obtenu par l'outil survient suite à un doublon. Le tableau 7.4 présente un résumé des résultats de la taille fonctionnelle obtenus par les experts humains, ceux de la taille fonctionnelle obtenus de l'outil et les écarts. Par ailleurs, les résultats de mesure manuelle et automatisée sont décrits en détail à l'annexe C du document.

Mouvements de données	Mesure manuelle (PFC)	Mesure automatisée (PFC)	Écart (PFC)	Écart (%)	Précision (%)
Entré (E)	25	26	-1	-4,00%	96,00%
Sortie (S)	38	36	2	5,26%	94,74%
Lecture (L)	17	18	-1	-5,88%	94,12%
<u>écriture</u> ©	3	4	-1	-33,33%	66,67%
Total	83	84	-1	-1,20%	98,80%

TABLEAU 7.4 Résultats de mesure manuelle et automatisée du projet «uObserve».

7.3.1 Projet #5 : « C-Reg »

C-Reg est un système en ligne qui permet aux étudiants et aux professeurs d'accéder au catalogue des cours⁴. Il interagit avec le système de facturation des étudiants et le système de courriel électronique. C-Reg comporte dix-neuf (19)

4. <https://cosmic-sizing.org/wp-content/uploads/2020/04/C-Reg-Case-Study-v2.01.pdf>

fonctionnalités décrites en format de Cas d'utilisation en langage naturel. La figure 7.4 est un diagramme de Cas d'utilisation qui donne une vue globale du fonctionnement du système « C-Reg ». Elle présente la liste des fonctionnalités du système « C-Reg » qui sont mesurées.

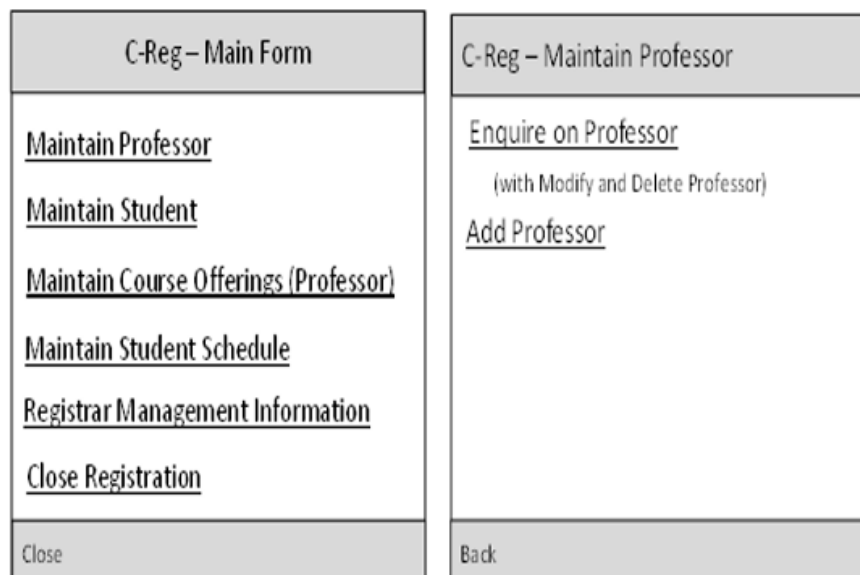


FIGURE 7.4 Liste des fonctionnalités du système « C-Reg » (Symons et al ; 2018).

7.3.2 Résultat de la mesure manuelle et automatisée du projet « C-Reg »

On a constaté que les mouvements de données en « Sortie » et en « Lecture » ont enregistré respectivement un écart de +6 CPF (12,24%) et +1 PFC (7,69%). Aucun écart n'a été enregistré pour les mouvements de donnée en « Entré » et en « écriture ». Par contre, un écart de +7 CPF (6,86%) a été enregistré entre le mesurage manuel et mesurage automatisé pour le nombre total des mouvements de données (|102 CPF-95 CPF|). Pour parvenir à identifier la source des écarts, on a examiné le document de spécification des exigences logicielles du système « C-Reg ». Après l'analyse, on a pu relever les principaux facteurs suivants causant

ces écarts :

- L’outil ne parvient pas à générer des triplets et déterminer les mouvements de données pour les scénarios des Cas d’utilisation suivants : « C-Reg requests Course Catalog to send Course Offering data » ; «C-Reg requests Course Offering data (with number of students enrolled, etc.) from the Course Catalog ». Cela vient du fait que l’outil identifie le verbe « requests » comme un nom et non comme un verbe.
- L’outil ne parvient pas à générer des triplets et déterminer les mouvements de données pour les scénarios des Cas d’utilisation suivants : « C-Reg sends The Professor’s selected Course offerings to the course Catalog » et C-Reg sends Professor’s qualifications and department to Course Catalog to retrieve eligible and committed Course Offerings.

En somme, ce projet a été difficile à mesurer, puisqu’il existe beaucoup de détails inutiles dans le document de spécification d’exigences logicielles. De plus, le format de description des Cas d’utilisation n’a pas été respecté dans son ensemble. Il existe dans ce document des scénarios des Cas d’utilisation qui sont décrits avec des verbes passifs. Tandis que les Cas d’utilisation sont décrits, selon le format standard recommandé, avec des verbes actifs. Les deux exemples de scénarios dont les verbes sont conjugués à la forme active sont respectivement : « Validated Course Offering ID’s are sent to the Course Catalog so that it can maintain the count of Students for each Course Offering ; Student’s Schedule items are marked ‘enrolled’ and made persistent on C-Reg ». Le tableau 7.5 présente un résumé des résultats de la taille fonctionnelle manuelle et automatisée. Par ailleurs, les résultats de mesurage manuel validés et publiés sur le site internet de COSMIC, ainsi que les résultats de mesurage automatisés obtenus par notre outil sont présentés en annexe.

Mouvements de données	Mesure manuelle (PFC)	Mesure automatisée (PFC)	Écart (PFC)	Écart (%)	Précision (%)
Entré (E)	30	30	0	0,00%	100,00%
Sortie (S)	49	43	+6	12,24%	87,76%
Lecture (L)	13	12	+1	7,69%	92,31%
<u>é</u> Criture ©	10	10	0	0,00%	100,00%
Total	102	95	+7	6,86%	93,14%

TABLEAU 7.5 Résultats de mesure manuelle et automatisée du projet «C-Reg».

7.4 Résultats de la taille fonctionnelle des cinq (5) projets en français

7.4.1 Projet #6 : « Système de gestion de réservation » (SGR)

Le projet #6 est une étude de cas du système de gestion des réservations d'un centre sportif (SGR) tiré du livre de « Modéliser en objets » publié par (Levesque, 2015). Ce centre sportif offre des aires de jeux pour le tennis, le squash et le racketball. Le document de spécification d'exigences fonctionnelles de ce système comporte cinq (5) Cas d'utilisation. La figure 7.5 présente le diagramme des Cas d'utilisation du système du SGR.

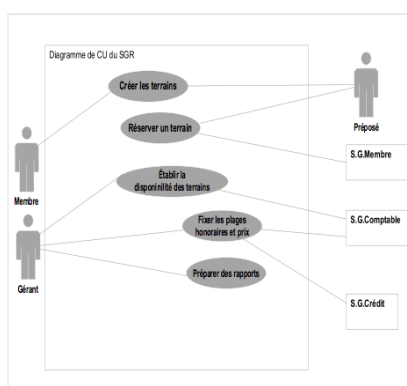


FIGURE 7.5 Cas d'utilisation du système du SGR (Levesque, 2015).

7.4.2 Résultat de mesure manuelle et automatisée du projet « SGR »

Les mouvements de données en « Entrée », « Sortie » et « Lecture » ont enregistré respectivement un écart de -1 CPF (-3,70%), +10 CPF (28,57%) et +1 CPF (4,77%). Par ailleurs, aucun écart n'a été enregistré pour les mouvements de données en « écriture ». Un écart de +10 CPF (10,10%) a été enregistré entre le mesurage manuel validé et publié par les experts et le mesurage automatisé obtenu de notre outil. La source de ces écarts vient du fait que les exigences fonctionnelles sont incomplètes. En d'autres termes, des messages d'erreur non précisés dans le document d'exigences logicielles ont été comptés par l'expert mesureur comme des points de fonction de sortie. De plus, l'outil génère deux triplets erronés pour les scénarios des Cas d'utilisation qui suivent : « Le système demande l'approbation au SGC et imprime un reçu ; Le système imprime un reçu de l'état de la demande de réservation ». Cela réside dans le fait que l'outil n'était pas capable de reconnaître l'objet « reçu ». Le tableau 7.6 résume les résultats de mesure manuelle et automatisée du projet SGR. Les résultats de mesurage manuel et automatisé sont présentés en détail à l'annexe C.

Mouvements de données	Mesure manuelle (PFC)	Mesure automatisée (PFC)	Écart (PFC)	Écart (%)	Précision (%)
Entré (E)	27	28	-1	-3,70%	96,30%
Sortie (S)	35	25	+10	28,57%	71,43%
Lecture (L)	21	20	+1	4,77%	95,23%
<u>écriture</u> ©	16	16	0	0,00%	100,00%
Total	99	89	+10	10,10%	90,90%

TABLEAU 7.6 Résultats de mesure manuelle et automatisée du projet SGR.

7.4.3 Projet #7 : « Système de gestion du commerce électronique » (SGCE)

Le projet #7 est un système de gestion du commerce électronique (SGCE) qui permet aux visiteurs et clients de commander des produits en ligne. Ce projet comporte neuf (9) Cas d'utilisation. Les descriptions de ces Cas d'utilisation sont tirées du livre de « Modéliser en objets » publié par (Levesque, 2015).

7.4.4 Résultat de mesure manuelle et automatisée

Les mouvements de données en « Lecture » ont enregistré un écart de +2 PFC (25%). Par contre, aucun écart n'a été enregistré pour les mouvements de données en « Entrée », en « Sortie » et en « écriture ». Par ailleurs, un écart de +2 PFC (5,26%) a été enregistré entre le mesurage manuel validé et publié par les experts et le mesurage automatisé. La source de ces écarts provient par le fait qu'il existe des vérifications et lectures implicites que l'expert mesureur compte pour des points de fonction. Tandis que l'outil ne parvient pas à détecter ces points de fonction implicites. Le tableau 7.7 résume les résultats de mesure manuelle et automatisée du projet SGCE.

Mouvements de données	Mesure manuelle (PFC)	Mesure automatisée (PFC)	Écart (PFC)	Écart (%)	Précision (%)
Entré (E)	9	9	0	0,00%	100%
Sortie (S)	14	14	0	0,00%	100,00%
Lecture (L)	8	6	+2	25,00%	75,00%
<u>écriture</u> ©	7	7		0,00%	100,00%
Total	38	36	+1	5,26%	94,74%

TABLEAU 7.7 Résultats de mesure manuelle et automatisée du projet SGCE.

7.4.5 Projet #8 : « Système de guichet automatique » (SGA)

Le projet #8 est une étude de cas du système de guichet automatique bancaire tiré du livre de « UML 2 par la pratique » publié par (Roques, 2018). Chaque Cas d'utilisation contient un ensemble de scénarios racontant sous forme de texte la façon dont un guichetier interagit avec le guichet automatique. Résultat de mesure automatisée et manuelle Les mouvements de données en « Entée », en « Sortie » et en « éCriture » ont enregistré respectivement un écart de +2 PFC (15,38%), -2 PFC (-18,18%), +1 PFC (33,33%). Par contre, aucun écart n'a été enregistré pour les mouvements de données en « Lecture ». Par ailleurs, un écart de +1 PFC (2,78%) a été enregistré entre le mesurage manuel et le mesurage automatisé pour le nombre total des mouvements de données ($|36 \text{ PFC} - 35 \text{ PFC}|$). La source de cet écart vient du fait qu'il y a une erreur d'orthographe à caractère sémantique dans le scénario de Cas d'utilisation suivant : « Le Porteur de carte inséré le chèque ». L'outil détecte ce scénario comme une exigence ambiguë, non compréhensible et ne compte pas « inséré » comme un mouvement de données (verbe), puisque « inséré » est un participe passé. Le tableau 7.8 résume les résultats de mesurage manuel et automatisé du système « SGA ».

Mouvements de données	Mesure manuelle (PFC)	Mesure automatisée (PFC)	Écart (PFC)	Écart (%)	Précision (%)
Entrée (E)	13	11	+2	15,38%	84,62%
Sortie (S)	11	13	-2	-18,18%	81,82%
Lecture (L)	9	9	0	0,00%	100,00%
éCriture ©	3	2	+1	33,33%	66,67%
Total	36	35	+1	2,78%	97,22%

TABLEAU 7.8 Résultats de mesurage manuel et automatisé du système « SGA ».

7.4.6 Projet #9 : « Application de contrôle d'accès à un bâtiment » (ACAB)

Le projet #9 est une application de contrôle d'accès à un bâtiment tiré du livre de « Modélisation objet avec UML » publié par (Muller et Gaertner, 2000). Les besoins de cette application sont exprimés sous forme de Cas d'utilisation dans un langage très proche des utilisateurs. Chaque Cas d'utilisation contient plusieurs scénarios décrits de manière générale du point de vue de l'acteur.

7.4.7 Résultat de mesure manuelle et automatisée

On a constaté que les mouvements de données en « Entrée », « Sortie » et en « écriture » ont enregistré respectivement un écart de -1 PFC (-2,70%), +5 PFC (11,11%) et +1 PFC (10%). Par contre, aucun écart n'a été enregistré pour les mouvements de données en « Lecture ». Un écart de +5 PFC (3,79%) a été enregistré entre le mesurage manuel et mesurage automatisé pour le nombre total des mouvements de données ($|132 \text{ PFC} - 127 \text{ PFC}|$). Pour parvenir à identifier la source des écarts, on a examiné le document de spécification. Après l'analyse, on a pu constater que l'outil n'identifie pas les points de fonction implicites pour des messages d'erreur non spécifiés dans le document d'exigences logicielles. Le tableau 7.9 résume les résultats de mesurage manuel et automatisé du projet «ACAB».

7.4.8 Projet #10

Le projet #10 est une série de Cas d'utilisation tirés du livre de (Cockburn, 2001). Le premier Cas d'utilisation décrit une personne qui achète des actions sur le Web. Le deuxième Cas d'utilisation décrit une personne qui essaie de se faire indemniser à la suite d'un accident. Quant au troisième d'utilisation, il décrit un agent de réception qui enregistre l'arrivée d'un colis. Résultat de mesure manuelle

Mouvements de données	Mesure manuelle (PFC)	Mesure automatisée (PFC)	Écart (PFC)	Écart (%)	Précision (%)
Entré (E)	37	38	-1	2,70%	97,30%
Sortie (S)	45	40	+5	11,11%	88,89%
Lecture (L)	40	40	0	0,00%	100,00%
<u>éCriture</u> ©	10	9	1	10,00%	90,00%
Total	132	127	+5	3,79%	96,21%

TABLEAU 7.9 Résultats de mesurage manuel et automatisé du projet « ACAB ».

et automatisée Les mouvements de données en « Entrée », « Sortie », « Lecture » et en « éCriture » ont enregistré respectivement un écart de + 1 PFC (9,10%), + 1 PFC (14,29%), +3 PFC (37,50%) et -1 PFC (-14,29%). Par ailleurs, un écart de + 4 PFC (12,12%) a été enregistré entre le mesurage manuel et automatisé pour le nombre total des mouvements de données (|33 PFC - 29 PFC|). La source de ces écarts vient du fait que les descriptions de ces Cas d'utilisation sont incomplètes. Cela sous-entend que l'expert mesureur identifie des points de fonction implicites dans les scénarios des Cas d'utilisation. Le tableau 7.10 présente en résumé les résultats de mesure manuelle et automatisée du projet #10. Les résultats du processus de mesure manuel et automatisé détaillés sont présentés à l'annexe.

Mouvements de données	Mesure manuelle (PFC)	Mesure automatisée (PFC)	Écart (PFC)	Écart (%)	Précision (%)
Entré (E)	11	10	+1	9,10%	90,90%
Sortie (S)	7	6	+1	14,29%	85,71%
Lecture (L)	8	5	+3	37,50%	62,50%
<u>éCriture</u> ©	7	8	-1	14,29%	85,71%
Total	33	29	+4	12,12%	87,88%

TABLEAU 7.10 Résultats de mesure manuelle et automatisée du projet #10.

7.5 Synthèse des résultats

Nous avons mesuré dix (10) projets, à raison de cinq (5) documents de spécification d'exigences logicielles écrites en anglais et de cinq (5) documents de spécification d'exigences logicielles écrites en français. Les résultats de mesure automatisée obtenus de l'outil pour les projets anglais se sont avérés plutôt cohérents avec les résultats de mesure manuelle validés et publiés par les experts avec une moyenne de 96,96%. Quant aux projets français, les résultats de mesure automatisée obtenus de l'outil se rapprochent des résultats de mesure manuelle validés par les experts avec une moyenne de 94,38%. Par contre, les résultats de mesure automatisée obtenus de l'outil pour les dix (10) documents de spécification s'élèvent à 95,71%. La précision pour les dix (10) projets varie entre 87,88% et 100%. Cet écart de 4,29% vient du fait qu'il existe des informations omises dans la rédaction des Cas d'utilisation et que l'outil n'est capable de déterminer les mouvements de données pour l'omission de ces informations. Par contre, un expert humain peut détecter ces informations omises et détermine les mouvements de données pour ces informations manquantes. De plus, l'outil ne parvient pas à déterminer les mouvements de données pour des verbes, tels que « *requests* ». Cela vient du fait que l'outil identifie le verbe « *requests* » comme un nom et non comme un verbe. D'autant plus, l'outil ne parvient pas à identifier les mouvements de données pour quelques scénarios de Cas d'utilisation du projet #5 qui étaient écrits à la voix passive et pour des scénarios de Cas d'utilisation contenant des cas possessifs tels que : « *C-Reg sends The Professor's selected Course offerings to the course Catalog* » et « *C-Reg sends Professor's qualifications and department to Course Catalog to retrieve eligible and committed Course Offerings* ». Le tableau 7.11 présente une synthèse des résultats des projets mesurés, la précision des résultats de mesurage manuel et automatisés, ainsi que les écarts.

Projet 1 - Resto Sys	119	117	98,32%
Projet 2 – ACME	33	32	96,97%
Projet 3 - Rice Cooker	24	24	100,00%
Projet 4 – uObserve	84	83	98,80%
Projet 5 - C-Reg	102	95	93,14%
Sous-total Anglais	362	351	96,96%
Projet 6 – SGR	99	89	89,90%
Projet 7- SGCE	38	39	97,37%
Projet 8-SGA	36	35	97,22%
Projet 9 – ACAB	132	127	96,21%
Projet 10	33	29	87,88%
Sous-total Français	338	319	94,38%
Total	700	670	95,71%

TABLEAU 7.11 Synthèse des résultats de mesure manuelle et automatisée.

7.6 Réponses aux sous-questions de recherche

L'objectif principal de notre recherche est de concevoir et de mettre en œuvre une nouvelle technique de rédaction d'exigences du logiciel, qui favorise l'automatisation de la taille fonctionnelle des logiciels à partir des spécifications fonctionnelles. Pour atteindre cet objectif, on a soulevé trois (3) sous questions de recherche que nous tentons de répondre dans cette section.

7.6.1 Première sous-question : difficulté d'automatisation

La première sous question de recherche a été la suivante : Pourquoi les méthodes de rédaction d'exigences du logiciel existantes ne facilitent-elles pas, au point de vue pratique, l'automatisation de la mesure de la taille fonctionnelle des logiciels ?

Les principaux outils, approches et méthodes de rédaction d'exigences du logi-

ciel existantes utilisées dans l'industrie sont notamment les Use Cases et les User Stories. Ils sont des descriptions textuelles, utilisés pour documenter les spécifications d'exigences logicielles. Cependant, la revue de littérature consultée a montré que ces méthodes et techniques de rédaction d'exigences du logiciel sont loin de permettre au point de vue pratique l'automatisation de la mesure de la taille fonctionnelle des logiciels. Cela vient du fait que les documents de spécifications, dans l'industrie, sont écrits en langage naturel et que ceux-ci ne contiennent pas de détails techniques (Wiegiers et Beatty, 2013). De plus, ces exigences sont souvent incomplètes, incohérentes et en contradiction avec d'autres exigences (Gilb et Graham, 1993)(Lévesque, 1998)(Trudel, 2012)(Sadoun, 2014). Des erreurs d'interprétation sont donc commises facilement par les analystes (Trudel, 2012). De ce fait, l'idée de parvenir, au point de vue pratique, à automatiser le processus de la mesure de la taille fonctionnelle des logiciels paraît donc difficile. Pour y remédier, nous avons proposé, dans le cadre de cette thèse, d'automatiser la taille fonctionnelle du logiciel à partir des exigences écrites sous forme de structure de triplets (sujet, prédicat, objet). Dans une telle perspective, on a présupposé que la rédaction des exigences logicielles pourrait se faire avec des prédicats de deux arguments ($f(x, y)$), où « x , f , et y » représentent respectivement les sujets, prédicats et objets. On a appliqué un ensemble de règles pour être capable d'extraire des triplets à partir des Cas d'utilisation ou Récits utilisateurs écrits en langage naturel.

7.6.2 Deuxième sous-question : avantages à automatiser la mesure

La deuxième sous question de recherche a été la suivante : Quels seraient les avantages d'automatiser le processus de mesure de la taille fonctionnelle des logiciels, à partir des spécifications par triplet ? Les exigences logicielles écrites en langage naturel sont difficiles d'être interprétables ou exploitables par des outils d'automat-

tisation (Gérançon *et al.*, 2021). De ce fait, les techniques de rédaction d'exigences logicielles, telles que les Use Case et User Stories, ne facilitent pas l'automatisation du processus de mesure de la taille fonctionnelle. Pour pallier ces difficultés, nous avons proposé l'approche par triplets comme technique de rédaction d'exigences logicielles en vue de faciliter l'automatisation du processus de la mesure de la taille fonctionnelle des logiciels à partir des spécifications fonctionnelles. Les résultats de mesure présentés précédemment ont montré que l'utilisation de l'approche par triplets favorise le processus de mesure de la taille fonctionnelle des logiciels. En d'autres mots, la rédaction des exigences logicielles sous forme de structure de triples est exploitable et interprétable par la machine et permet de concevoir des outils d'automatisation de mesure efficaces présentant des résultats de mesure de la taille fonctionnelle qui se rapprochent des résultats validés par des experts (Gérançon *et al.*, 2022). À titre d'exemple, notre outil proposé présente des résultats cohérents avec ceux des résultats de mesure validés et publiés par les experts pour les projets en anglais avec une moyenne de 96,96%, où la précision varie entre 93,14% et 100%. Tandis que dans le cadre des projets en français, l'outil fournit des résultats cohérents avec ceux des résultats de mesure trouvés avec une moyenne de 94,38%. La précision varie entre 87,88% et 100%. D'autant plus, l'approche par triplets peut permettre de résoudre les problèmes d'incohérence et d'ambiguïté d'exigences logicielles. Cette approche permet aux analystes de rédiger les exigences logicielles de façon simple et efficace avec peu d'informations. Autrement dit, elle fournit une description atomique et succincte des exigences, exprimant les besoins des utilisateurs avec peu ou sans détails superflus. L'objectif de l'approche par triplets est de s'accrocher sur la clarté des exigences logicielles. Dans une telle perspective, l'approche par triplets est une technique de rédaction d'exigences logicielles qui peut bonifier les techniques des Cas d'utilisation et des Récits utilisateurs. Nous avons développé un outil qui permet d'extraire automatiquement des triplets à partir des Cas d'utilisation ou Récits utilisateur écrits en

langage naturel. Ainsi, l'outil permet de détecter les détails inutiles ou superflus dans le document de spécification des exigences logicielles.

Un autre problème majeur au niveau du processus d'automatisation de la mesure de la taille fonctionnelle des logiciels à partir des spécifications fonctionnelles est que les outils d'automatisation n'interagissent pas avec les techniques de rédaction d'exigences logicielles existantes (Abran et Paton, 1997)(Sadoun, 2014). De ce fait, la plupart des outils d'automatisation de mesure présentent des résultats de la taille fonctionnelle différents par rapport aux résultats obtenus par les experts humains. L'approche par triplets proposée permet de résoudre ces difficultés. D'une part, elle favorise l'interaction avec les outils d'automatisation, d'autre part, elle permet à notre outil de trouver des résultats de mesure cohérents à plus de 95% par rapports aux résultats de mesure publiés par des experts humains.

En résumé, l'approche par triplets, dans le cadre d'automatisation du processus de mesure de la taille fonctionnelle des logiciels, possède de nombreux avantages. Premièrement, elle permet de développer des outils d'automatisation fournissant une présentation des résultats de mesure facilitant les analyses. Par exemple, notre outil une fois déterminé la taille fonctionnelle, identifie les mouvements de données de types Entrée (E), Sortie (S), Lecture (L) et écriture (C). Cela permet aux utilisateurs ou aux mesureurs d'analyser d'une part les données de mesure du logiciel. D'autre part, cela permettra aux analystes, concepteurs et développeur d'identifier facilement les différentes opérations du système qui seront implémentées. Car, les types de mouvements de données identifiées par l'outil correspondent aux entrants, sortants ou méthodes déclenchées par les utilisateurs ou le système. Deuxièmement, l'approche par triplets offre la possibilité aux outils d'automatisation d'interpréter les exigences logicielles écrites sous forme de structure de triplets. Cela vient du fait que la structure (sujet, prédicat, objet) est exploitable par la machine. Troisièmement, l'approche par triplets permet de développer des

outils d'automatisation efficaces et performants, c'est-à-dire d'obtenir des résultats de mesure efficaces reflétant les résultats de mesure des experts humains dans un temps très court. En effet, on a constaté pour les dix (10) projets mesurés, que notre outil d'automatisation détermine rapidement les tailles fonctionnelles et réduit considérablement l'effort de mesure entre 99,40% et 100% comme le montre le tableau ?? comparativement au processus de mesurage manuel. L'automatisation du processus de mesure de la taille fonctionnelle des logiciels à partir des exigences écrites sous forme de structure de triplets enregistre donc un gain significatif en termes de temps et d'effort pour l'obtention de la taille fonctionnelle des logiciels.

7.6.3 Troisième sous-question : approche par triplets comme méthode de rédaction

La troisième sous question de recherche a été la suivante : Dans quelle mesure la spécification des exigences par le biais de l'approche par triplets pourrait servir de méthodologie pour les analystes et concepteurs dans le domaine de développement des exigences logicielles ? Le développement des logiciels de qualité, vu les problèmes que peut causer un système logiciel défectueux dans une organisation, est une priorité en génie logiciel (Laporte et April, 2018)(Staron et Meding, 2018). La qualité du logiciel s'introduit tout au long du processus de développement en définissant des exigences logicielles de qualité (Gérançon, 2011). En effet, plusieurs chercheurs se spécialisant dans le domaine de génie de logiciel ont relaté que la source principale des erreurs du logiciel provient de la phase de spécification des exigences logicielles (Lévesque, 1998)(Ambler, 2003)(Wieggers et Beatty, 2013)(Laporte et April, 2018). Des études émanant de nombreuses sources ont montré près de 50% des erreurs du logiciel sont localisées dans la phase de spécification ou définition des exigences logicielles (Beizer, 2003)(Laporte et April, 2018)(Staron et Meding, 2018). (Laporte et April, 2018) ont soutenu que la difficulté de définition

des exigences logicielles est liée à la rédaction claire et concise d'exigences par les analystes d'affaires ou ingénieurs spécialisés en exigences. De ce fait, il est difficile pour que les architectes, concepteurs, développeurs et testeurs transforment ces exigences en spécifications utilisables par les utilisateurs (Laporte et April, 2018)(Staron et Meding, 2018). La spécification et rédaction d'exigences logicielles sous forme de structure de triplets peut aider aux analystes et aux concepteurs à éviter de définir et développer des exigences superflues et ambiguës. L'objectif de l'approche par triplets est de permettre aux analystes d'écrire ou d'exprimer les besoins des clients de manière simple et efficace avec peu d'informations. Cela signifie que la structure de triplets fournit une description atomique et succincte des exigences logicielles, exprimant les besoins des utilisateurs avec peu ou sans détails superflus. L'approche par triplets met l'emphase sur la clarté et la brièveté des exigences logicielles. Dans une telle perspective, l'approche par triplets permettra aux analystes de définir des exigences logicielles respectant les critères de qualité de bonne qualité (clairs, concis, non propices aux ambiguïtés, etc.). Ce qui permettra de réduire les erreurs et défauts logiciels injectés dans la phase de spécification d'exigences logicielles. Selon Ambler (Ambler, 2003), l'approche traditionnelle, notamment les Use Cases, augmente les risques d'échecs des projets de développement de logiciel. Cela vient du fait que les exigences détaillées tôt dans le cycle de vie d'un projet logiciel sont souvent inefficaces et ne correspondent pas très souvent aux besoins des utilisateurs (Ambler, 2003)(Laporte et April, 2018). L'approche par triplets permet de pallier ces difficultés, puisque la rédaction des exigences logicielles sous forme de structure de triplets fournit une description simple qui s'accroche principalement sur l'utilisateur (sujet), qui déclenche une opération ou un mouvement de données (prédicats) sur les composants du système (objets). Ce qui permettra aux analystes de définir des exigences transformées en spécifications directement utilisables en phases d'architecture, de conception, d'implémentation, de codage et de tests. Cela sous-entend que la mé-

thodologie par triplets peut aider aux analystes et aux concepteurs de définir et développer des exigences traçables, vérifiables et testables tout long du processus de développement de logiciels. En outre, le développement des systèmes se fait autour des objets. Ces objets nécessitent, entre autres, qu'ils soient bien identifiés, puisqu'ils seront par la suite implémentés en termes de classes ou de composants logiciels. L'approche par triplets offre une structure simple et efficace qui permet facilement d'identifier les objets, composants ou classes logicielles. En somme l'approche par triplets peut servir de méthodologie pour les analystes, architectes, concepteurs, développeurs et programmeurs dans le domaine de développement de logiciels. Elle peut permettre de réduire dès la phase d'analyse les ambiguïtés dans les spécifications du logiciel à développer, ainsi que dans toutes les phases du cycle de vie du logiciel. Puisque, les exigences logicielles seront spécifiées en un trio de concepts ou en une structure simple : sujet, prédicat, objet. La rédaction des exigences logicielles adoptant la structure de triplets facilitera aux analystes, concepteurs et développeurs d'élaborer des documents de spécification respectant les caractéristiques ou critères des exigences logicielles de qualité et de développer des systèmes logiciels répondant aux besoins des utilisateurs.

7.6.4 Question principale de recherche : Approche par triplets comme technique facilitant l'automatisation

La question de recherche principale a été formulée de la façon suivante : Dans quelle mesure une technique appropriée de rédaction d'exigences du logiciel pourrait-elle faciliter une meilleure automatisation de la mesure de la taille fonctionnelle des logiciels ? On a proposé de rédiger les exigences logicielles sous forme de structure de triplets afin de définir des exigences claires, succinctes, interprétables et manipulables par des outils d'automatisation. Ensuite, on a développé un outil permettant de générer des triplets à partir des Use Cases ou User Stories écrits

en langage naturel. Les résultats de la recherche ont montré que la génération des triplets à partir des Use Cases ou User Stories, en utilisant des règles, est exploitable par des outils d'automatisation de mesure. L'outil proposé présente des résultats de mesurage automatisés cohérents avec une moyenne de 96,96% par rapport aux résultats manuels obtenus et validés par les experts humains. Donc l'approche par triplets est une technique de rédaction d'exigences logicielle qui facilite le processus d'automatisation de la mesure de la taille fonctionnelle des logiciels.

7.7 Atteinte des objectifs de recherche

L'objectif général de notre recherche était de concevoir et de mettre en œuvre une nouvelle technique de rédaction d'exigences du logiciel, en vue de favoriser l'automatisation du processus de mesure de la taille fonctionnelle des logiciels. Cet objectif a été atteint par le développement d'un modèle de triplets pour la rédaction d'exigences du logiciel. Par la suite, nous avons développé un outil d'automatisation du processus de mesure de la taille fonctionnelle des logiciels. Cet outil permet de générer des triplets, de déterminer la taille fonctionnelle et de spécifier les différents types de mouvements de données. Notre outil d'automatisation a été développé, tel que défini par la méthode COSMIC ISO 19761 et rencontre les principales caractéristiques du cadre référentiel proposé par (Abran et Paton, 1997).

7.8 Évaluation et validation des résultats

Dans le cadre de cette thèse, nous avons mis en œuvre notre outil avec dix (10) projets. Les résultats présentés ont été comparés à ceux d'experts humains certifiés avec la méthode COSMIC. Tout d'abord, notre outil génère les triplets à partir

des exigences fonctionnelles écrites en langage naturel. Ensuite, il détermine la taille fonctionnelle en quantifiant le nombre de prédicats atomiques de chaque triplet. Les résultats de la recherche ont montré que la génération de triplets à partir des Use Cases ou User Stories peut être exploitée par des outils d'automatisation de mesure. En fait, l'outil présente des résultats automatisés cohérents avec les résultats manuels validés et publiés par des experts, avec une moyenne dépassant 95%, comme le montre le tableau . En effet, notre outil a été développé en fonction de la méthode COSMIC et du cadre référentiel proposé par (Abran et Paton, 1997). Premièrement, notre outil est appliqué aux fonctionnalités d'application d'affaires et de logiciels de type temps réel. Deuxièmement, les résultats de la thèse ont montré que les exigences fonctionnelles écrites sous forme de structure triplets sont compréhensibles ou interprétables par des machines ou par des outils d'automatisation de mesure de la taille fonctionnelle. Notre outil est capable de déterminer la taille fonctionnelle des logiciels et de spécifier les types de mouvements de données à partir du modèle de triplets proposé. Troisièmement, notre outil génère des rapports facilitant aux mesureurs ou aux utilisateurs de faire des analyses, en identifiant les mouvements de données en Entrée, Sortie, Lecture et écriture. Finalement, notre outil est capable, par sa mise en œuvre, d'exclure de par lui-même des triplets redondants.

7.9 Utilité du processus d'automatisation de la mesure de la taille fonctionnelle des logiciels

L'effort de mesurage manuel correspond au temps que les experts ont mis pour comprendre le document de spécification d'exigences fonctionnelles, appliquer la méthode COSMIC et préparer les résultats en format de chiffrier Excel. Selon une étude faite par (Trudel, 2012) dans le cadre de sa thèse, près de seize (16) mesureurs ont pris entre 60 et 90 minutes pour mesurer avec la méthode COSMIC

un document de spécifications d'exigences logicielles dont la taille fonctionnelle se totalise environ 83 PFC. En effet, le temps total de mesure pour les experts pourrait se situer en général autour de 1 à 5 minutes par point de fonction COSMIC (PFC). Il est à souligner que le temps de mesure est élastique en fonction de la clarté et de la complétude des exigences. Dans le cas où il y a des ambiguïtés dans le document de spécification d'exigences logicielles, le temps total de mesure pourrait se situer autour de 5 à 10 minutes par point de fonction COSMIC. Quant à l'effort du processus de mesure automatisée, il correspond au temps d'exécution de l'outil pour téléverser le document d'exigences, générer les triplets et déterminer les résultats de mesure de la taille fonctionnelle. En effet, l'automatisation du processus de mesure est utile dans le fait qu'elle permette de mesurer plus vite. À titre d'exemple, pour une série de projets, soit les cinq projets en français de notre expérimentation, d'environ 338 PFC, un expert humain expérimenté en COSMIC a mis environ 251 minutes, soit 4 heures et 11 minutes de temps pour mesurer leur taille fonctionnelle, c'est-à-dire d'appliquer la méthode COSMIC pour déterminer la taille fonctionnelle en inscrivant les résultats détaillés dans un chiffrier Excel. D'autres experts humains expérimentés auraient mis environ 338 minutes, à raison d'une minute par PFC. Cependant, un mesureur débutant qui prendrait hypothétiquement 5 minutes par PFC pour effectuer la même mesure, l'effort se solderait par 1 690 minutes, soit 28 heures et 10 minutes. Quant à l'outil développé, il a mis environ 114,92 secondes, soit une minute et 55 secondes, pour téléverser le document, déterminer la taille fonctionnelle et identifier les types de mouvement de données. Grosso modo, l'outil a mis, en termes d'effort environ 4 minutes et 12 secondes pour l'obtention de la taille fonctionnelle des dix (10) documents de spécification d'exigences logicielles et l'identification des mouvements de données. Par contre, un expert humain aurait mis entre 700 et 3 500 minutes, soit un intervalle de 11 heures et 40 minutes à 58 heures et 23 minutes pour l'obtention de la taille fonctionnelle de ces dix (10) documents de spécifications logicielles. En

comparant l'effort du mesurage manuel avec celui du mesurage automatisé, nous avons constaté qu'il y a une différence importante en faveur du mesurage automatisé, soit une réduction d'effort avec une moyenne de 99,40%. Cela nous permet de conclure que le mesurage automatisé garantit un effort réduit en comparaison au mesurage manuel. Cette réduction d'effort pourrait augmenter considérablement en fonction de la taille de l'application à mesurer. Il est à mentionner qu'il ne s'agisse pas, en automatisant le processus de mesure de la taille fonctionnelle des logiciels, de mettre à l'écart les experts mesureurs. Il s'agit bien au contraire d'accélérer le travail des experts. Notre outil proposé permettra d'aider les experts à aller plus vite pour un ensemble de tâches dont l'expertise est rare et que l'effort de mesurage exigé nécessite un temps difficile à prévoir quand la qualité des exigences est inconnue. Le tableau ?? présente l'effort de mesurage manuel et automatisé des dix (10) projets mesurés. On suppose que pour chaque PFC, le mesureur manuel prend dans le meilleur des cas une (1) minute. Tandis que notre outil prend en moyenne 0,34 seconde, soit 0,006 minute pour 1 PFC. La formule appliquée pour trouver l'effort manuel est la suivante : $EM = TFM * 1 \text{ minute} / \text{PFC}$ (avec EM : effort manuel ; TFM : taille fonctionnelle manuelle). Tandis que la formule appliquée pour l'obtention de l'effort automatisé est la suivante : $EA = TFM * 0,006 \text{ minute} / \text{PFC}$ (avec EA : effort automatisé ; TFM : taille fonctionnelle manuelle).

7.10 Synthèse

Dans ce chapitre, nous avons présenté une expérimentation de notre outil développé avec dix projets. L'outil génère pour chaque processus fonctionnel des triplets. Par la suite, il calcule la somme des prédicats atomiques correspondant à des mouvements de données, en vue d'obtention de la taille fonctionnelle. Notre outil fournit pour les projets anglais des résultats automatisés cohérents avec les résultats manuels validés et publiés par des experts, avec une moyenne de 96,96%,

où la précision varie entre 93,16% et 100%. Quant aux projets français, l'outil fournit des résultats cohérents avec les résultats manuels avec une moyenne de 94,38%. Par ailleurs, le taux de précision pour les projets mesurés en français a enregistré un écart de 2,58%, par rapport au taux de précision des projets mesurés en anglais. Cela vient du fait que la plupart des documents de spécification d'exigences logicielles écrites en français mesurées contiennent des informations superflues, incomplètes et incohérentes, comparativement aux documents de spécification d'exigences logicielles écrites en anglais qui ont été révisés et publiés par des experts. En effet, les résultats de cette expérimentation ont montré que le mesurage automatisé garantit un effort réduit pour les projets en anglais et en français avec une réduction de 99,40% comparativement à l'effort manuel, comme le montre le tableau 7.12. La précision varie entre 99,40% et 100%.

Projets	Taille fonctionnelle manuelle (en PFC)	Effort manuel (en minutes)	Effort automatisé (en minutes)	Effort automatisé (en secondes)
Projet#1	119	119	0,714	42,84
Projet#2	33	33	0,198	11,88
Projet#3	24	24	0,144	8,64
Projet#4	84	84	0,504	30,24
Projet#5	102	102	0,612	36,72
Sous-total Anglais	362	362	2.172	130,32
Projet#6	99	99	0.594	35,64
Projet#7	38	38	0.228	13,68
Projet#8	36	36	0.216	12,96
Projet#9	132	132	0.792	47,52
Projet#10	33	33	0.198	11,88
Sous-total Français	338	338	2.028	121,68
Total	700	700	4.2	252

TABLEAU 7.12 Effort de mesurage manuel et automatisé.

CHAPITRE VIII

CONCLUSION : CONTRIBUTION, PERSPECTIVES ET LIMITES DE RECHERCHE

8.1 Conclusion

Ce travail de recherche est inscrit dans le cadre d'automatisation du processus de mesure de la taille fonctionnelle des logiciels à partir des spécifications fonctionnelles. Nous avons proposé une approche par triplets pour spécifier, rédiger et représenter les exigences logicielles. Nous avons montré que les exigences écrites sous forme de structure de triplets favorisent l'automatisation du processus de mesure de la taille fonctionnelle des logiciels. Notre approche a été éprouvée, testée et validée par la conception d'un nouvel outil d'automatisation de la mesure de la taille fonctionnelle des logiciels, tel que défini par la méthode COSMIC ISO 19761. En effet, l'outil proposé permet de générer des triplets à partir des Use Cases ou des User Stories écrits en langage naturel. En d'autres termes, l'outil permet de construire automatiquement un triplestore par l'extraction des triplets à partir des textes (Use Cases ou User Stories). Par la suite, il détermine la taille fonctionnelle et identifie les types de mouvements de données (Entrée, Sortie, Lecture, écriture). Nous avons testé et expérimenté notre outil avec dix (10) documents de spécification d'exigences logicielles) dont les résultats de mesure manuelle sont validés et publiés par des experts certifiés avec la méthode COSMIC. Notre outil

présente des résultats automatisés cohérents avec une moyenne de 96,96% comparativement aux résultats manuels validés et publiés par les experts.

8.2 Contribution /originalité de la thèse

L'objectif de développement des systèmes logiciels est de répondre à des exigences qui représentent la valeur d'affaire du client (product owner). La réussite de tout projet de développement de logiciels repose en grande partie sur la phase de spécification des exigences dont son but est de décrire manière précise et claire les caractéristiques du système à développer (Wiegiers et Beatty, 2013)(Sadoun, 2014). Les exigences logicielles sont souvent écrites en langage naturel et contiennent généralement des informations superflues, incohérentes et des ambiguïtés sémantiques (Wiegiers et Beatty, 2013). Par conséquent, les exigences logicielles sont propices aux diverses interprétations et ne facilitent pas le processus d'automatisation de la taille fonctionnelle des logiciels. D'où la nécessité d'améliorer les techniques de rédaction d'exigences logicielles. Dans le cadre de cette thèse, nous avons premièrement proposé un modèle de triplets pour la spécification et représentation des exigences logicielles. Ce modèle est fondé sur la logique des prédicats du 1er ordre. Deuxièmement, nous avons développé un outil qui permet d'automatiser le processus de mesure de la taille fonctionnelle des logiciels à partir des exigences écrites sous forme de structure de triplets. En effet, la contribution de notre thèse est respectivement d'ordre informatique et cognitive.

8.2.1 Contribution informatique

Pour démontrer l'originalité de ce projet de recherche, nous avons relevé quelques éléments non couverts dans les travaux antérieurs : Les techniques de rédaction d'exigences du logiciel utilisées dans l'industrie, ne favorisent pas l'automatisa-

tion de la mesure de la taille fonctionnelle des logiciels. L'automatisation de la mesure de la taille fonctionnelle à partir de documents de spécifications logicielles reste jusqu'à présent dans une phase exploratoire (Levesque *et al.*, 2008)(Sadoun, 2014)(Quesada-López *et al.*, 2019). Il n'existe pas jusqu'à ce jour, d'après la littérature consultée, des outils qui automatisent complètement le processus d'obtention de la taille fonctionnelle des logiciels à partir des spécifications fonctionnelles. Plusieurs chercheurs proposent des approches, prototypes et outils. Toutefois, ces outils possèdent de nombreuses contraintes, faiblesses et limitations. Certains outils et prototypes développés nécessitent des interventions manuelles pour l'obtention des résultats du processus de mesure. D'autres chercheurs ont proposé des outils d'automatisation à partir du code source (Chamkha *et al.*, 2018)(Tarhan et Sağ, 2018)(Sahab et Trudel, 2020). Les mesures basées sur le code source s'obtiennent après la phase de développement de logiciels. Par conséquent, il est fondamental de concevoir et proposer des outils d'automatisation de mesure de la taille fonctionnelle dès le début du cycle de développement des logiciels, plus spécifiquement à partir de la phase d'analyse. Dans le cadre de cette thèse, nous avons abordé les éléments non couverts dans les travaux antérieurs. D'une part, nous avons amélioré les techniques de rédaction d'exigences du logiciel, en vue de l'automatisation de la mesure de la taille fonctionnelle des logiciels à partir des spécifications par triplets. Premièrement, nous avons développé un modèle de triplets destiné à décrire les exigences logicielles sous forme de structure de triplets (sujet, prédicat, objet) afin de faciliter le traitement automatique de telles descriptions. Les données du triplestore ou du modèle de triplets sont traduites en des formules de logique des prédicats du premier ordre. Ce qui permet de rendre automatisable le processus de mesure de la taille fonctionnelle de logiciels à partir des spécifications fonctionnelles. Deuxièmement, nous avons développé un outil qui permet de générer des triplets à partir des Use Cases ou User Stories écrits en langage naturel. Autrement dit, l'outil construit automatiquement un

triplestore par l'extraction des triplets à partir des spécifications fonctionnelles écrites en langage naturel. Par la suite, l'outil détermine, à partir du triplestore, la taille fonctionnelle des logiciels selon la méthode COSMIC. Cet outil permet d'automatiser le processus de mesure de la taille fonctionnelle des logiciels, sans nécessiter des interventions manuelles pour l'obtention de la taille fonctionnelle. Troisièmement, nous sommes parvenus à implémenter les règles de la méthode COSMIC permettant d'identifier les types de mouvements de données (Entrée, Sortie, Lecture, écriture) de chaque processus fonctionnel.

8.2.2 Contribution cognitive

Cette thèse vise à automatiser la mesure de la taille fonctionnelle des logiciels, en rendant procédurale la mesure du processus fonctionnel qui était auparavant sémantique, sans procédure automatique. La procéduralisation se fait par l'extraction des triplets à partir des Cas d'utilisation ou des Récits utilisateurs écrits en langage naturel. D'une part, la contribution cognitive de cette thèse est relative à l'usage de la logique de description et d'ontologies pour procéder à l'automatisation de la mesure de la taille fonctionnelle des logiciels. D'autre part, elle explique la place ou nature des différentes mémoires de la cognition humaine à travers le processus de conception de l'outil d'automatisation de la taille fonctionnelle.

- Usage de la logique de description et d'ontologies pour procéder à l'automatisation de la mesure de la taille fonctionnelle des logiciels

La première contribution cognitive de notre thèse sert à montrer que l'extraction des triplets sous forme de sujet, prédicat et objet, peut permettre de rendre procédural ce qui n'était auparavant qu'un contenu sémantique. L'automatisation du processus de mesure de la taille fonctionnelle des logiciels paraît difficile à partir des spécifications fonctionnelles. Par contre, la rédaction des spécifications par triplets facilite la tâche de rendre symbolique les exigences logicielles, en utilisant

la logique des prédicats du premier ordre. Ce qui permet de faciliter le traitement automatique des exigences logicielles, en vue d'automatiser le processus de mesure de la taille fonctionnelle. En termes de contribution cognitive, on présuppose que la rédaction des exigences logicielle se fait avec des prédicats dyadiques, c'est-à-dire des prédicats de deux arguments ($f(x, y)$). Le prédicat, étant une action ou un mouvement de données en COSMIC, s'est exprimé par un verbe. Quant à la variable « x », elle représente le sujet du prédicat, tandis que la variable « y » représente l'objet du prédicat. Dans une telle perspective, on a présumé que dans un système à base de règles, les règles sont basées sur l'idée que la rédaction d'exigences est l'élaboration des fonctions dyadiques. En ce sens, nous avons fait associer la fonction « f » avec les variables « x, y ». Par la suite, nous avons appliqué des règles de traitement de relation entre ($f(x, y)$). En résumé, nous avons traduit les exigences logicielles racontées sous forme de Use Case ou User Stories en informations structurées ou organisées, en construisant une ontologie via la logique de description pour pouvoir construire et générer des triplets. Ensuite, nous sommes parvenus à automatiser la mesure de la taille fonctionnelle. L'ontologie que nous avons construite est un triplestore qui permet de stocker les concepts et connaissances sur les processus fonctionnels du logiciel sous forme des triplets, ainsi que sur le processus de mesurage de COSMIC. Les exigences logicielles écrites en langage naturel se transforment en une représentation plus simple, en vue de faciliter l'automatisation. Donc, la construction de cette ontologie en utilisant la logique de description est une étape nécessaire pour l'automatisation de la taille fonctionnelle des logiciels.

- Place des différentes mémoires de la cognition humaine à travers la conception d'outil d'automatisation de la taille fonctionnelle

Le client, dans le domaine de développement de logiciels, intervient à l'entrée pour exprimer ses problèmes et besoins. Il recourt surtout à sa mémoire épisodique pour

l'expression de ses besoins, qui sont entre autres décrits sous forme de Use Case ou User Stories. Pour les rendre automatisables, nous avons traduit ces Use Cases ou User Stories en mémoire sémantique structurée en servant de la logique de description pour construire des triplets (sujet, prédicat, objet). Par la suite, on a développé l'outil d'automatisation du processus de mesure de la taille fonctionnelle des logiciels. Une fois l'outil d'automatisation est livré au client, il n'aura pas à raisonner pour mesurer la taille fonctionnelle des logiciels. Le client traitera la taille fonctionnelle comme une procédure. En d'autres termes, en livrant le produit au client, il recourt à sa mémoire sémantique, et progressivement à sa mémoire procédurale pour mettre en œuvre l'outil d'automatisation en vue d'obtention de la taille fonctionnelle des logiciels. Tout compte fait, le client a recours aux deux mémoires quand il utilise l'outil d'automatisation. Premièrement, l'outil peut être emmagasiné par le client comme étant une procédure à suivre dans sa mémoire procédurale. Deuxièmement, le client met en œuvre l'outil en utilisant son mémoire de travail, puisque la procédure est simple.

En résumé, la principale contribution cognitive de notre recherche consiste à utiliser un modèle cognitif original qui allie la mémoire épisodique, la mémoire sémantique et la mémoire procédurale dans un processus complexe visant l'automatisation du processus de mesure de la taille fonctionnelle des logiciels.

Une autre contribution cognitive de notre recherche vient du fait la spécification par triplets favorise une représentation mentale des exigences logicielles. Dans le domaine de développement de logiciels, la représentation mentale et visuelle des exigences logicielles n'est pas une tâche facile. Cela réside dans le fait qu'avec les approches de rédaction d'exigences existantes, on donne tôt trop de détails qui ne sont pas pertinents. Ce qui permet l'obtention des exigences superflues, incohérentes et ambiguës. Dans une telle perspective, il devient difficile de représenter cognitivement les vrais besoins des utilisateurs. Dans le domaine de

sciences cognitives, la représentation mentale ou cognitive est une représentation que l'on fait d'une situation ou d'un concept donné. On peut distinguer plusieurs représentations, qui sont principalement (image mentale, mémoire sémantique, concepts et catégories). En effet, les représentations mentales ont des relations avec la construction des systèmes symboliques. Dans le domaine du génie logiciel, des recherches ont été faites afin de permettre aux analystes, concepteurs et développeurs de faire une représentation mentale des concepts plus proche de la réalité du client ou des utilisateurs. Ces recherches ont conduit à l'amélioration de l'ergonomie des interfaces hommes/machines. L'extraction des triplets par l'outil pourra permettre aux analystes et concepteurs d'obtenir une meilleure représentation mentale des concepts, objets conceptuels, entités ou composants logiciels, puisque l'approche par triplets présente une structure simple (sujet, prédicat, objet), permettant de spécifier facilement les objets ou composants logiciels qui seront implémentés. Ainsi, ces objets sont représentés visuellement à partir du modèle de triplet. De plus, vu que l'extraction des triplets à partir des Use Cases permet de réduire l'ambiguïté et l'incohérence au niveau de la description des spécifications logicielles. Par conséquent, l'approche par triplets peut aider à modéliser cognitivement les besoins et concepts des logiciels et contribuer à réaliser des interfaces homme-machine conviviales reflétant les besoins des utilisateurs du système. Dans le domaine des sciences cognitives, l'interface homme-machine est, d'une façon générale, conçue comme un prolongement de la mémoire de travail de l'utilisateur (Boy, 2013). D'autres chercheurs soutiennent que les limitations liées à la mémoire humaine ont des répercussions sur l'ergonomie des interfaces homme-machine (Baddeley et Hitch, 2000). De ce fait, les chercheurs en sciences cognitives ont fait des recommandations ergonomiques en relatant que les interfaces des logiciels ne doivent pas présenter des informations superflues et ambiguës, puisqu'elles limitent l'efficacité et la capacité de travail des utilisateurs (Anderson, 2013). L'approche par triplets peut pallier ces difficultés, car les exigences

logicielles sous forme de structure de triplets sont décrites de façon succincte et claire et exprimant les besoins uniques des utilisateurs ou du client. Ce qui permettra dans les phases de conception et de codage de réaliser des interfaces hommes/machines simples, ergonomiques et faciles d'utilisation.

8.3 Perspectives et limites de recherche

Le modèle de triplets développé pour la rédaction des exigences logicielles s'inscrit dans un formalisme à base logique des prédicats du premier ordre. Notre but vise à améliorer les techniques de rédaction d'exigences logicielles en une structure simple composée d'un trio de concepts (sujet, prédicat, objet) en vue de faciliter le processus d'automatisation de la taille fonctionnelle des logiciels. Nous avons donc conçu un outil qui permet d'extraire automatiquement des triplets à partir des Use Cases ou Use Stories écrits en langage naturel. Cet outil fonctionne présentement dans deux langues telles que l'anglais et le français. L'outil pourrait s'adapter à toutes les autres langues adoptant la forme générale d'ordre sujet, prédicat, objet, telles que l'espagnol, le créole haïtien et l'italien. Par contre, la limitation de notre outil réside dans le fait qu'il ne peut pas générer des triplets pour les langues qui adoptent une règle d'ordre sujet, objet, prédicat (SOP), prédicat, sujet, objet (PSO). Selon les experts en linguistique cognitive, il existe six (6) combinaisons possibles en matière des structures des langues (Delbecque, 2006)(Hagège et Roux, 2003). Quant à (Delbecque, 2006), il soutient que les structures dominantes sont notamment, SPO, SOP et PSO.

Notre première perspective ultérieure se focalisera sur la possibilité de développer d'autres modules qui pourraient permettre à l'outil d'exploiter toutes les combinaisons dominantes de structures des langues en vue de générer des triplets à partir des Use Cases ou User Stories.

Notre deuxième perspective consistera à intégrer un module d'apprentissage automatique qui fait en sorte que le système puisse s'améliorer progressivement avec l'usage ou pendant sa mise en œuvre. Au fur et à mesure que l'outil ait une faible solution, il est ajusté par des experts mesureurs. Ces ajustements humains constituent un aspect limitatif de l'outil. On pourra, dans le cadre de l'intégration d'une composante d'apprentissage automatique, permettre à l'outil d'apprendre à résoudre des problèmes par lui-même. Il est à souligner qu'on a implémenté préliminairement cette capacité d'apprentissage, en permettant à l'outil de faire une induction par une succession de cas d'exclusion des triplets. L'outil apprend à exclure les triplets redondants exclus par les utilisateurs ou experts mesureurs. Toutefois, on pourra étendre ultérieurement la capacité d'apprentissage de l'outil, en se donnant à l'outil le moyen d'apprendre des nouvelles règles complémentaires par rapport aux premières règles qui lui sont imposées. Dans une telle perspective, l'outil ne fera pas seulement mettre en œuvre des règles. Par la mise en œuvre de ses règles, il se donnera les moyens d'apprendre des nouvelles règles complémentaires par rapport aux premières règles.

De plus, l'outil proposé n'est pas en mesure de détecter le format du document d'exigences, tel que les en-têtes, pieds de page, titres, etc. Le texte d'exigence doit être téléchargé à partir d'un fichier Word ou PDF dans notre outil, où ce fichier ne doit contenir que des textes d'exigences sans aucun format. En raison des manipulations manuelles pour créer ce fichier, il existe une possibilité d'erreurs humaines, comme par exemple un texte d'exigence n'a pas été copié ou a été copié deux fois.

Nos travaux de recherche ultérieure seront concentrés sur la possibilité de permettre à l'outil de détecter le format du document d'exigences et d'ignorer principalement les sections, sous-sections et titres qui ne sont pas pertinentes dans le cadre du processus de mesure de la taille fonctionnelle des logiciels. Il est ainsi

pertinent de souligner que les exigences fonctionnelles sont écrites avec des verbes actifs et non avec des verbes passifs. Néanmoins, il est probable de rencontrer des textes où quelques scénarios de cas d'utilisation sont écrits sous la forme passive, c'est-à-dire que le sujet et l'objet sont intervertis. Dans ce cas, l'outil détecte qu'il y a une règle de bonne rédaction d'exigences ou un format de Cas d'utilisation standard qui n'est pas respecté. Néanmoins, on pourra, dans la poursuite de nos travaux de recherche, se pencher sur ces cas afin de permettre à l'outil d'être capable de déterminer la taille fonctionnelle pour les scénarios de Cas d'utilisation écrits à la voix passive.

APPENDICE A

PUBLICATIONS DE L'AUTEUR LIÉES À CETTE THÈSE

Revue internationale avec comité de lecture :

Gérançon, B., Trudel, S., Nkambou, R. et Robert, S. (2021). Software functional sizing automation from requirements written as triplets. International Conference on Software Engineering Advances, Barcelona, Spain, 16, 23–29 (article publié) disponible : https://www.thinkmind.org/index.php?view=articlearticleid=icsea2021_14_01_8004

ICSEA 2021 : The Sixteenth International Conference on Software Engineering Advances

Software Functional Sizing Automation from Requirements Written as Triplets

Bruel Gérançon, Sylvie Trudel, Roger Kkambou, Serge Robert
Department of Computer Science
Université du Québec à Montréal (UQAM)
Montréal, Canada

e-mail: gerancon.bruel@uqam.ca, trudel.s@uqam.ca, nkambou.roger@uqam.ca, robert.serge@uqam.ca

Abstract—The domain of software functional size measurement automation, from software specification documents, has been a research topic over the last years. The literature consulted shows that attempts to automate the process of measuring the software functional size has obtained little success at the industry level. Several tools for automating the measurement of software functional size have been developed according to the Common Software Measurement International Consortium (COSMIC) method (ISO 19761) website and that of International Function Point User Group (IFPUG). However, these tools encountered many flaws, constraints, and limitations. Moreover, the methods, techniques and tools for writing software specification documents used in the industry

number of lines of code to be written for a software. Thus, the number of lines of code corresponds to the physical size of the software. Albrecht [6] proposed a method based on the number of function points, the principle being to identify and quantify user functionalities, thus giving rise to the notion of functional size. Several software measurement methods, such as COSMIC, IFPUG, NESMA, Mark II and FISMA have been proposed and approved by the International Organization for Standardization (ISO). Among the various existing methods and tools, COSMIC is a recent measurement method, developed with the aim of overcoming some limitations of the other methods. A particularity of the COSMIC method is that

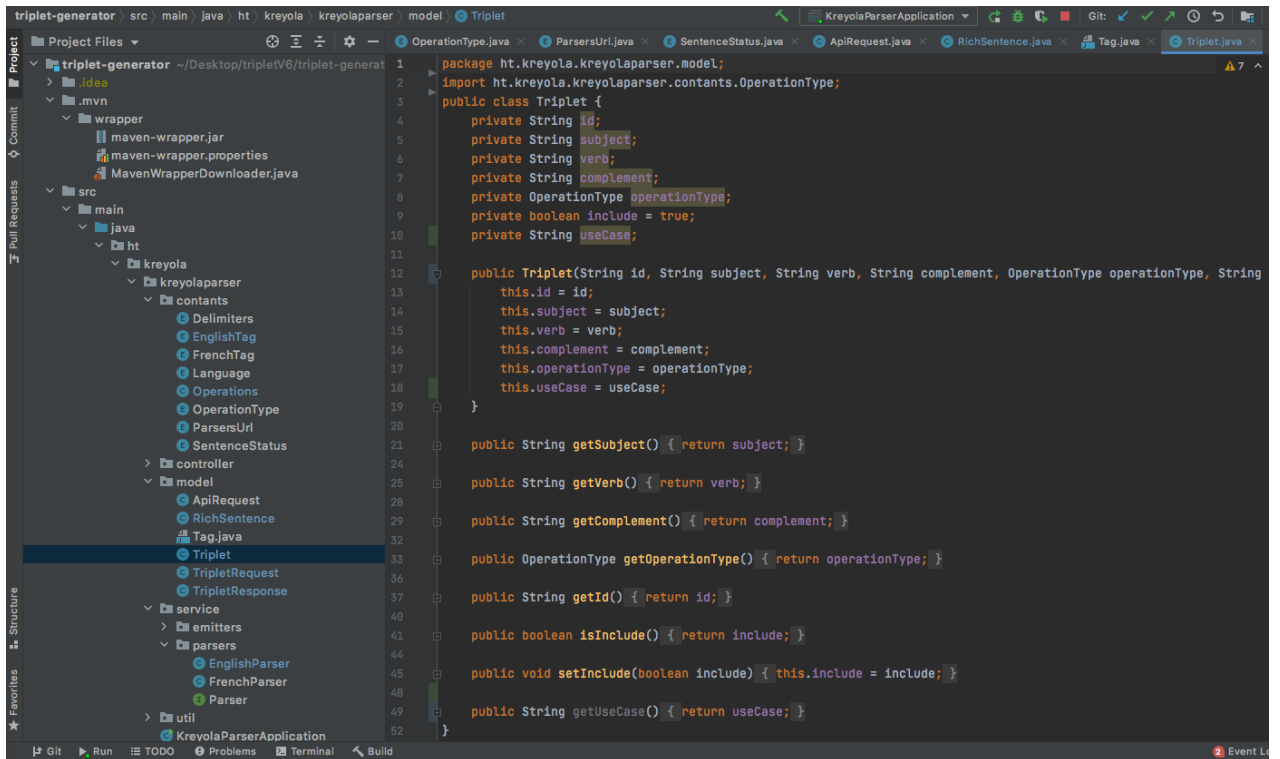
Vue de l'article publié

APPENDICE B

PROTOTYPE : QUELQUES EXEMPLES DE CLASSES D'IMPLEMENTATION

```
1 package ht.kreyola.kreyolaparser.service.emitters;
2
3
4 import edu.stanford.nlp.ling.TaggedWord;
5 import edu.stanford.nlp.ling.Word;
6 import ht.kreyola.kreyolaparser.contants.EnglishTag;
7 import ht.kreyola.kreyolaparser.model.*;
8
9
10 import java.util.*;
11
12 import static ht.kreyola.kreyolaparser.contants.EnglishTag.*;
13 import static ht.kreyola.kreyolaparser.service.emitters.Pattern.*;
14 import static ht.kreyola.kreyolaparser.util.RandomHelpers.uuid;
15 import static ht.kreyola.kreyolaparser.util.TripletHelpers.getOperationType;
16 import static java.util.Arrays.*;
17 import static java.util.Collections.*;
18 import static java.util.Objects.*;
19 import static java.util.stream.Collectors.toList;
20
21 public class EnglishEmitter implements TripletEmitter {
22
23     @private String subject(RichSentence richSentence) {
24         int numberOfTaggedWords = richSentence.getTaggedWords().size();
25         for (int index = 0; index < numberOfTaggedWords; index++) {
26             TaggedWord taggedWord = richSentence.getTaggedWords().get(index);
27             String tag = taggedWord.tag();
28             Set<String> subjectTags = new HashSet<>(asList(NNS.name(), NNS.name()));
29             if (subjectTags.contains(tag)) {
30                 // if there is another noun next to current one
31                 if (index + 1 < numberOfTaggedWords && subjectTags.contains(richSentence.getTaggedWords().get(index + 1).tag())) {
32                     // return found noun concatenated with next one
33                     return taggedWord.word() + " " + richSentence.getTaggedWords().get(index + 1).word();
34                 } else {
35                     // simply return the found tag
36                     return taggedWord.word();
37                 }
38             }
39         }
40     }
41 }
```

Exemple d'interface des classes et packages de l'outil



The screenshot shows an IDE window with the following content:

- Project Files:** A tree view on the left showing the project structure. The path is: `triplet-generator` > `src` > `main` > `java` > `ht` > `kreyola` > `kreyolaparser` > `contants` > `Triplet`.
- Code Editor:** The main window displays the implementation of the `Triplet` class in `Triplet.java`. The code is as follows:

```
1 package ht.kreyola.kreyolaparser.model;
2 import ht.kreyola.kreyolaparser.contants.OperationType;
3 public class Triplet {
4     private String id;
5     private String subject;
6     private String verb;
7     private String complement;
8     private OperationType operationType;
9     private boolean include = true;
10    private String useCase;
11
12    public Triplet(String id, String subject, String verb, String complement, OperationType operationType, String
13        this.id = id;
14        this.subject = subject;
15        this.verb = verb;
16        this.complement = complement;
17        this.operationType = operationType;
18        this.useCase = useCase;
19    }
20
21    public String getSubject() { return subject; }
22
23    public String getVerb() { return verb; }
24
25    public String getComplement() { return complement; }
26
27    public OperationType getOperationType() { return operationType; }
28
29    public String getId() { return id; }
30
31    public boolean isInclude() { return include; }
32
33    public void setInclude(boolean include) { this.include = include; }
34
35    public String getUseCase() { return useCase; }
36
37 }
```
- Bottom Bar:** The IDE's bottom bar shows icons for Git, Run, TODO, Problems, Terminal, and Build.

Implémentation de la classe de triplet

```

1  package ht.kreyola.kreyolaparser.service.parsers;
2  import ...
20 @Service
21 public class EnglishParser implements Parser {
22
23     LexicalizedParser lp ;
24     List<TripletEmitter> emitters;
25     public EnglishParser() {
26         this.lp = LexicalizedParser.loadModel(ENGLISH.value());
27         this.emitters = tripletEmitters();
28     }
29     @private List<String> toSentences(String useCase) {
30         String[] sentences = useCase.split(Delimiters.DOT.value());
31         return Arrays.stream(sentences).collect(toList());
32     }
33     @Override
34     public List<Triplet> toTriplets(final TripletRequest tripletRequest) {
35         List<RichSentence> sentences = tripletRequest.getSentences();
36
37         List<RichSentence> richSentences = sentences.stream().map(sentence -> {
38             List<CoreLabel> rawWords = Sentence.toCoreLabelList(sentence.getWords().toArray(new String[0]));
39             Tree parse = lp.apply(rawWords);
40             sentence.setTaggedWords(parse.taggedYield());
41             return sentence;
42         }).collect(toList());
43
44         return richSentences
45             .stream()
46             .flatMap(richSentence -> emitters.stream().flatMap(tripletEmitter -> tripletEmitter.emit(richSentence)))
47             .collect(toList());
48     }
49     @private List<TripletEmitter> tripletEmitters() {
50         return Collections.singletonList(
51             new EnglishEmitter()
52         );
53     }
54 }

```

Classe d'analyseur lexical des mots

```

2 package ht.kreyola.kreyolaparser.service.emitters;
3
4 import edu.stanford.nlp.ling.TaggedWord;
5 import edu.stanford.nlp.ling.Word;
6 import ht.kreyola.kreyolaparser.constants.EnglishTag;
7 import ht.kreyola.kreyolaparser.model.*;
8
9 import java.util.*;
10
11 import static ht.kreyola.kreyolaparser.constants.EnglishTag.*;
12 import static ht.kreyola.kreyolaparser.service.emitters.Pattern.*;
13 import static ht.kreyola.kreyolaparser.util.RandomHelpers.Uuid;
14 import static ht.kreyola.kreyolaparser.util.TripletHelpers.getOperationType;
15 import static java.util.Arrays.*;
16 import static java.util.Collections.*;
17 import static java.util.Objects.*;
18 import static java.util.stream.Collectors.toList;
19
20 public class EnglishEmitter implements TripletEmitter {
21
22     private String subject(RichSentence richSentence) {
23         int numberOfTaggedWords = richSentence.getTaggedWords().size();
24         for (int index = 0; index < numberOfTaggedWords; index++) {
25             TaggedWord taggedWord = richSentence.getTaggedWords().get(index);
26             String tag = taggedWord.tag();
27             Set<String> subjectTags = new HashSet<>(asList(NW.name(), NNS.name()));
28             if (subjectTags.contains(tag)) {
29                 // if there is another noun next to current one
30                 if (index + 1 < numberOfTaggedWords && subjectTags.contains(richSentence.getTaggedWords().get(index + 1).tag())) {
31                     // return found noun concatenated with next one
32                     return taggedWord.word() + " " + richSentence.getTaggedWords().get(index + 1).word();
33                 } else {
34                     // simply return the found tag
35                     return taggedWord.word();
36                 }
37             }
38         }
39         return null;

```

Générateur des triplets en anglais

```

51
52     private String verbAfterAnd(RichSentence richSentence, boolean verbIsWronglyLabel) {
53         boolean isAfterAnd = false;
54         Set<String> verbTags = new HashSet<>(asList(VB.name(), VBZ.name()));
55         int numberOfTaggedWords = richSentence.getTaggedWords().size();
56         for (int index = 0; index < numberOfTaggedWords; index++) {
57             TaggedWord taggedWord = richSentence.getTaggedWords().get(index);
58             if (taggedWord.word().toLowerCase().equals("and")) {
59                 isAfterAnd = true;
60                 continue;
61             }
62             Set<String> complementTags = new HashSet<>(asList(NW.name(), NNS.name()));
63             if (verbIsWronglyLabel && isAfterAnd && complementTags.contains(taggedWord.tag())) {
64                 // taking complement after and
65                 return taggedWord.word();
66             }
67
68             if (verbTags.contains(taggedWord.tag()) && isAfterAnd) {
69                 // return next word too if it is a verb
70                 if (index + 1 < numberOfTaggedWords && verbTags.contains(richSentence.getTaggedWords().get(index + 1).tag()))
71                     return taggedWord.word() + " " + richSentence.getTaggedWords().get(index + 1).word();
72                 // return after next word too if it is a verb : sales manager enters information and asks to save
73                 else if (index + 2 < numberOfTaggedWords && verbTags.contains(richSentence.getTaggedWords().get(index + 2).tag())) {
74                     return taggedWord.word() + " " + richSentence.getTaggedWords().get(index + 1).word() + " " + richSentence.getTaggedWords().get(index + 2).word();
75                 } else {
76                     return taggedWord.word();
77                 }
78             }
79         }
80
81         // prevent infinite loop
82         if (verbIsWronglyLabel) {
83             return null;
84         }
85         return verbAfterAnd(richSentence, verbIsWronglyLabel);
86     }
87

```

Générateur des triplets en anglais (suite)

```

81 // prevent infinite loop
82 if(verbIsWronglyLabel) {
83     return null;
84 }
85 return verbAfterAnd(richSentence, verbIsWronglyLabel: true);
86 }
87
88
89 private String getComplementAfterAnd(RichSentence rs) {
90     boolean isAfterAnd = false;
91     Set<String> complementTags = new HashSet<>(asList(NW.name(), NNS.name()));
92     int numberOfTaggedWords = rs.getTaggedWords().size();
93     for (int index = 0; index < numberOfTaggedWords; index++) {
94         TaggedWord taggedWord = rs.getTaggedWords().get(index);
95         if (taggedWord.word().toLowerCase().equals("and")) {
96             isAfterAnd = true;
97             continue;
98         }
99         // we don't want to take the word right after AND, it could be the verb, sometimes the parser wrongly label a verb as a noun
100         if (complementTags.contains(taggedWord.tag()) && isAfterAnd && !rs.getTaggedWords().get(index - 1).word().toLowerCase().equals("and")) {
101             // found a noun after and, treat it as the complement. Ex: The sales manager adds a product and asks to save the information
102             return taggedWord.word();
103         }
104     }
105
106     return null;
107 }
108
109 private String complement(RichSentence richSentence) {
110     boolean isAfterVerb = false;
111     Set<String> verbTags = new HashSet<>(asList(VB.name(), VBZ.name()));
112     Set<String> complementTags = new HashSet<>(asList(NW.name(), NNS.name()));
113     for (TaggedWord tw : richSentence.getTaggedWords()) {
114         if (verbTags.contains(tw.tag()) {

```

Générateur des triplets en anglais (suite)

```

69 // return next word too if it is a verb
70 if (index + 1 < numberOfTaggedWords && verbTags.contains(richSentence.getTaggedWords().get(index + 1).tag()))
71     return taggedWord.word() + " " + richSentence.getTaggedWords().get(index + 1).word();
72 // return after next word too if it is a verb : sales manager enters information and asks to save
73 else if (index + 2 < numberOfTaggedWords && verbTags.contains(richSentence.getTaggedWords().get(index + 2).tag())) {
74     return taggedWord.word() + " " + richSentence.getTaggedWords().get(index + 1).word() + " " + richSentence.getTaggedWords().get(index
75 } else {
76     return taggedWord.word();
77 }
78 }
79
80
81 // prevent infinite loop
82 if(verbIsWronglyLabel) {
83     return null;
84 }
85 return verbAfterAnd(richSentence, verbIsWronglyLabel: true);
86 }
87
88
89 private String getComplementAfterAnd(RichSentence rs) {
90     boolean isAfterAnd = false;
91     Set<String> complementTags = new HashSet<>(asList(NW.name(), NNS.name()));
92     int numberOfTaggedWords = rs.getTaggedWords().size();
93     for (int index = 0; index < numberOfTaggedWords; index++) {
94         TaggedWord taggedWord = rs.getTaggedWords().get(index);
95         if (taggedWord.word().toLowerCase().equals("and")) {
96             isAfterAnd = true;
97             continue;
98         }
99         // we don't want to take the word right after AND, it could be the verb, sometimes the parser wrongly label a verb as a noun
100         if (complementTags.contains(taggedWord.tag()) && isAfterAnd && !rs.getTaggedWords().get(index - 1).word().toLowerCase().equals("and")) {
101             // found a noun after and, treat it as the complement. Ex: The sales manager adds a product and asks to save the information
102             return taggedWord.word();
103         }
104     }

```

Générateur des triplets en anglais (suite)

```

142     final String complement = isNull(compAfterAnd) ? complement(rs) : compAfterAnd; // if there is a complement after and use it instead
143     final String verb = verbAfterAnd(rs, verbisWronglyLabel: false);
144
145     if (nonNull(subject) && nonNull(verb) && nonNull(complement)) {
146         Triplet triplet = new Triplet(uuid(), subject, verb, complement, getOperationType(verb), rs.getUseCase());
147         List<Triplet> tripletList = new ArrayList<>(simpleTriplets);
148         tripletList.add(triplet);
149         return tripletList;
150     } else {
151         return simpleTriplets;
152     }
153 }
154
155 List<Triplet> simpleSentence(RichSentence rs) {
156     final String subject = subject(rs);
157     final String verb = verb(rs);
158     final String complement = complement(rs);
159
160     if (nonNull(subject) && nonNull(verb) && nonNull(complement)) {
161         return singletonList(new Triplet(uuid(), subject, verb, complement, getOperationType(verb), rs.getUseCase()));
162     } else {
163         return emptyList();
164     }
165 }
166
167 private Pattern getPattern(RichSentence rs) {
168     if (rs.getContent().toLowerCase().contains("and")) {
169         return AND;
170     } else {
171         return SIMPLE;
172     }
173 }
174
175 }
176

```

Générateur des triplets en anglais

```

1 package ht.kreyola.kreyolaparser.controller;
2
3 import ...
4
5 @RestController
6 @RequestMapping("/api")
7 @CrossOrigin(origins = "+")
8 //Objet qui reçoit les requetes de l'utilisateur, les traite ou les envoie a la couche service
9 public class ParsingController {
10
11     private final Parser frenchParser;
12     private final Parser englishParser;
13     private final Set<String> excludedTriplets = new HashSet<>();
14
15     public ParsingController(Parser frenchParser, Parser englishParser) {
16         this.frenchParser = frenchParser;
17         this.englishParser = englishParser;
18     }
19
20     @PostMapping("/triplets/parser")
21     public TripletResponse getTriplets(@RequestBody ApiRequest<TripletRequest> apiRequest) {
22         final boolean useFrench = Language.FR.equals(apiRequest.getBody().getUseCaseLanguage());
23         List<Triplet> triplets = useFrench ? frenchParser.toTriplets(apiRequest.getBody()) : englishParser.toTriplets(apiRequest.getBody());
24         List<Triplet> filteredTriplets = triplets.stream().map(triplet -> {
25             String stringBuilder = triplet.getSubject() +
26                 triplet.getVerb() +
27                 triplet.getComplement();
28             triplet.setInclude(!excludedTriplets.contains(stringBuilder));
29             return triplet;
30         }).collect(Collectors.toList());
31         int inputOperationCount = (int) filteredTriplets.stream().filter(triplet -> IN.equals(triplet.getOperationType()) && triplet.isInclude()).count();
32         int outputOperationCount = (int) filteredTriplets.stream().filter(triplet -> OUT.equals(triplet.getOperationType()) && triplet.isInclude()).count();
33         int readOperationCount = (int) filteredTriplets.stream().filter(triplet -> READ.equals(triplet.getOperationType()) && triplet.isInclude()).count();
34         int writeOperationCount = (int) filteredTriplets.stream().filter(triplet -> WRITE.equals(triplet.getOperationType()) && triplet.isInclude()).count();
35         TripletResponse tripletResponse = new TripletResponse(filteredTriplets);
36

```

Implémentation de la méthode d'exclusion des triplets

```

49     tripletResponse.setFunctionalSize(filteredTriplets.size());
50     tripletResponse.setInputOperationCount(inputOperationCount);
51     tripletResponse.setOutputOperationCount(outputOperationCount);
52     tripletResponse.setReadOperationCount(readOperationCount);
53     tripletResponse.setWriteOperationCount(writeOperationCount);
54
55     return tripletResponse;
56 }
57
58 @PostMapping("/triplet/exclude")
59 public ResponseEntity<HttpStatus> excludeTriplet(@RequestBody Triplet triplet) {
60     String tripletId = triplet.getSubject() +
61         triplet.getVerb() +
62         triplet.getComplement();
63     excludedTriplets.add(tripletId);
64     return new ResponseEntity<>(HttpStatus.OK);
65 }
66
67 @PostMapping("/triplet/include")
68 public ResponseEntity<HttpStatus> includeTriplet(@RequestBody Triplet triplet) {
69     String tripletId = triplet.getSubject() +
70         triplet.getVerb() +
71         triplet.getComplement();
72     excludedTriplets.remove(tripletId);
73     return new ResponseEntity<>(HttpStatus.OK);
74 }
75
76 }
77

```

Implémentation de la méthode d'exclusion des triplets

```

2 import TripletBox from "../TripletBox";
3 import {content} from "../constants/copy";
4 // Classe qui affiche à l'utilisateur le rapport d'obtention de la taille fonctionnelle
5 class TripletsReport extends React.PureComponent {
6
7     render() {
8         const {apiResponse, userLocale, toggleTriplet, version = 'v1'} = this.props;
9
10
11         const {
12             tripletList,
13             functionalSize,
14             inputOperationCount,
15             outputOperationCount,
16             readOperationCount,
17             writeOperationCount
18         } = apiResponse;
19         const copy = content[userLocale];
20         const totalCpf = inputOperationCount + outputOperationCount + readOperationCount + writeOperationCount;
21
22         return (
23             <div className="row">
24                 {tripletList?.length > 0 && version === 'v1' && (
25                     <div style={{marginLeft: '50px'}}>
26                         <h1> {copy.TRIPLET_LIST_AND_STATS} </h1>
27                         <div>
28                             <p> {copy.FUNCTIONAL_SIZE}{ ' '} {functionalSize}</p>
29                             <p> {copy.NUMBER_OF_INPUT_OPS}{ ' '} {inputOperationCount}</p>
30                             <p> {copy.NUMBER_OF_OUTPUT_OPS}{ ' '} {outputOperationCount} </p>
31                             <p> {copy.NUMBER_OF_READ_OPS}{ ' '} {readOperationCount} </p>
32                             <p> {copy.NUMBER_OF_WRITE_OPS}{ ' '} {writeOperationCount} </p>
33                             <hr/>
34                             <h2>{copy.TRIPLET_LIST}</h2>
35                             <table style={{marginRight: '0px', width: '80%'}}>
36                                 <thead>
37                                     <tr>
38                                         <th>{copy.TRIPLET}</th>
39                                         <th>{copy.OPERATION_TYPE}</th>
40                                         <th>{copy.INCLUDE}</th>
41                                     </tr>

```

Interface d'affichage de la taille fonctionnelle automatisée

```

40     {tripletList
41       .map(triplet => (
42         <TripletBox
43           triplet={triplet}
44           onChange={toggleTriplet}
45           userLocale={userLocale}
46         />)
47       )}
48   </table>
49 </div>
50 </div>
51 )}
52 {tripletList?.length > 0 && version === 'v2' && (
53   <div style={{marginLeft: '50px'}}>
54     <h1> {copy.TRIPLET_LIST_AND_STATS}</h1>
55     <div>
56       <table style={{marginRight: '0px', width: '80%'}} className={'reports'}>
57         <tr>
58           <th style={{textAlign: 'center'}}>{copy.FUNCTIONAL_PROCESS}</th>
59           <th style={{textAlign: 'center'}}>{copy.TRIPLETS}</th>
60           <th className={'rotated-text'}>{copy.SUM_OF_ENTRIES}</th>
61           <th className={'rotated-text'}>{copy.SUM_OF_EXITS}</th>
62           <th className={'rotated-text'}>{copy.SUM_OF_READS}</th>
63           <th className={'rotated-text'}>{copy.SUM_OF_WRITES}</th>
64           <th className={'rotated-text'}>{copy.TOTAL_SUM_OF_CPF}</th>
65           <th className={'rotated-text'}>{copy.INCLUDE}</th>
66         </tr>
67         {tripletList
68           .map(triplet => (
69             <TripletBox
70               triplet={triplet}
71               onChange={toggleTriplet}
72               userLocale={userLocale}
73               version={version}
74             />)
75           )}
76       <tr style={{fontSize: '25px'}}>
77         <td>{copy.GRAND_TOTAL}</td>

```

Interface d'affichage de la taille fonctionnelle automatisée (suite)

```

78     <td></td>
79     <td className={'numbers'}>{inputOperationCount}</td>
80     <td className={'numbers'}>{outputOperationCount}</td>
81     <td className={'numbers'}>{readOperationCount}</td>
82     <td className={'numbers'}>{writeOperationCount}</td>
83     <td className={'numbers'}>{totalCpf}</td>
84   </tr>
85 </table>
86 </div>
87 </div>
88 )}
89 </div>
90 )
91 }
92 }
93 }
94 }
95 export default TripletsReport;
96

```

Interface d'affichage de la taille fonctionnelle automatisée

APPENDICE C

RÉSULTATS DÉTAILLÉS DE MESURE MANUELLE ET AUTOMATISÉE DES PROJETS EN ANGLAIS

Functional process	E	X	R	W	Sum of CFP
Login "Mobile App"	3	3			6
Add Order's Data	6	6			12
Modify Order's Data	4	4			8
Create an order	1	1		1	3
Retrieve selected table)	1	2	2		5
Save modified data	1	1		1	3
Login	1	2	1		4
Add an Employee	1	1	1	1	4
View the Employees list	1	2	1		4
View an Employee data	1	1	1		3
Modify an Employee Data	1	1		1	3
Delete an Employee	1	1		1	3
Add an Item	1	1	1	1	4
View the items list	2	2	1		5
View an Item data	1	1	1		3
Modify an item	1	1		1	3
Delete an Item	1	1		1	3
Add an item family	1	1	1	1	4
View Item families list	2	2	1		5
View Item family data	1	1	1		3
Modify an item family	1	1		1	3
Delete an item family	1	1		1	3
Add a table	1	1	1	1	4
View Tables list	2	2	1		5
View a table data	1	1	1		3
Modify table data	1	1		1	3
Delete a table	1	1		1	3
View the List of Orders	1	2	1		4
Delete an order	1	1		1	3
Total size	42	46	16	15	119

Résultats de mesurage manuel de la TF du projet #1 « Resto Sys »

Triplet generator By Bruel Ger... x +

localhost:3000

English Use word document Use case is in English Report version 2

Select file Choisir un fichier Projet1-RestoSys.docx Submit

Download

Triplets list and other statistics

Functional Process	Triplets	Entry	Exit	Read	Write	Sum CFP	Include
Login Mobile App	employee,enters,username	1	0	0	0	1	<input checked="" type="checkbox"/>
Login Mobile App	app,provides,waiter	0	1	0	0	1	<input checked="" type="checkbox"/>
Login Mobile App	app,receives,user	1	0	0	0	1	<input checked="" type="checkbox"/>
Login Mobile App	app,displays,user	0	1	0	0	1	<input checked="" type="checkbox"/>
Login Mobile App	app,receives,message	1	0	0	0	1	<input checked="" type="checkbox"/>
Login Mobile App	app,displays,message	0	1	0	0	1	<input checked="" type="checkbox"/>
Add Order's Data	waiter,enters,table	1	0	0	0	1	<input checked="" type="checkbox"/>
Add Order's Data	app,sends,table	0	1	0	0	1	<input checked="" type="checkbox"/>
Add Order's Data	app,receives,item	1	0	0	0	1	<input checked="" type="checkbox"/>
Add Order's Data	app,displays,item	0	1	0	0	1	<input checked="" type="checkbox"/>
Add Order's Data	waiter,enters,item	1	0	0	0	1	<input checked="" type="checkbox"/>
Add Order's Data	app,sends,item	0	1	0	0	1	<input checked="" type="checkbox"/>
Add Order's Data	app,receives,item	1	0	0	0	1	<input checked="" type="checkbox"/>
Add Order's Data	app,displays,list	0	1	0	0	1	<input checked="" type="checkbox"/>
Add Order's Data	waiter,enters,items	1	0	0	0	1	<input checked="" type="checkbox"/>
Add Order's Data	waiter,specifies,quantity	0	0	0	0	0	<input checked="" type="checkbox"/>
Add Order's Data	app,sends,items	0	1	0	0	1	<input checked="" type="checkbox"/>
Add Order's Data	app,displays,message	0	1	0	0	1	<input checked="" type="checkbox"/>
Modify Order's Data	waiter,enters,table	1	0	0	0	1	<input checked="" type="checkbox"/>
Modify Order's Data	app,sends,table	0	1	0	0	1	<input checked="" type="checkbox"/>
Modify Order's Data	app,receives,list	1	0	0	0	1	<input checked="" type="checkbox"/>
Modify Order's Data	app,displays,items	0	1	0	0	1	<input checked="" type="checkbox"/>

Résultats de mesurage automatisé de la TF du projet #1« Resto Sys »

Modify Order's Data	waiter,deletes,item	0	0	0	1	1	<input checked="" type="checkbox"/>
Modify Order's Data	Waiter,modifies,items	0	0	0	1	1	<input checked="" type="checkbox"/>
Modify Order's Data	waitress,modifies,items	0	0	0	1	1	<input checked="" type="checkbox"/>
Modify Order's Data	app,receives,message	1	0	0	0	1	<input checked="" type="checkbox"/>
Modify Order's Data	app,displays,message	0	1	0	0	1	<input checked="" type="checkbox"/>
Create an order	web app,receives,item	1	0	0	0	1	<input checked="" type="checkbox"/>
Create an order	web app,creates,order	0	0	0	1	1	<input checked="" type="checkbox"/>
Create an order	web app,sends,message	0	1	0	0	1	<input checked="" type="checkbox"/>
Retrieve selected table	web app,receives,table	1	0	0	0	1	<input checked="" type="checkbox"/>
Retrieve selected table	web app,retrieves,list	0	0	1	0	1	<input checked="" type="checkbox"/>
Retrieve selected table	web app,sends,list	0	1	0	0	1	<input checked="" type="checkbox"/>
Retrieve selected table	web app,retrieves,item	0	0	1	0	1	<input checked="" type="checkbox"/>
Retrieve selected table	web app,sends,list	0	1	0	0	1	<input checked="" type="checkbox"/>
Save modified data	web app,receives,list	1	0	0	0	1	<input checked="" type="checkbox"/>
Save modified data	web app,displays,message	0	1	0	0	1	<input checked="" type="checkbox"/>
Login	employee,enters,username	1	0	0	0	1	<input checked="" type="checkbox"/>
Login	employee,password,username	0	0	0	0	0	<input checked="" type="checkbox"/>
Login	web app,checks,validity	0	0	1	0	1	<input checked="" type="checkbox"/>
Login	web app,displays,user	0	1	0	0	1	<input checked="" type="checkbox"/>
Login	web app,displays,message	0	1	0	0	1	<input checked="" type="checkbox"/>
Add an Employee	admin,enters,employee	1	0	0	0	1	<input checked="" type="checkbox"/>
Add an Employee	web app,retrieves,employee	0	0	1	0	1	<input checked="" type="checkbox"/>
Add an Employee	web app,adds,employee	1	0	0	0	1	<input checked="" type="checkbox"/>
Add an Employee	web app,displays,message	0	1	0	0	1	<input checked="" type="checkbox"/>
View the Employee list	admin,selects,employee	1	0	0	0	1	<input checked="" type="checkbox"/>
View the Employee list	web app,retrieves,employee	0	0	1	0	1	<input checked="" type="checkbox"/>
View the Employee list	web app,displays,employee	0	1	0	0	1	<input checked="" type="checkbox"/>
View the Employee list	web app,displays,message	0	1	0	0	1	<input checked="" type="checkbox"/>
View an Employee data	admin,selects,user	1	0	0	0	1	<input checked="" type="checkbox"/>
View an Employee data	web app,retrieves,user	0	0	1	0	1	<input checked="" type="checkbox"/>
View an Employee data	web app,displays,waiter	0	1	0	0	1	<input checked="" type="checkbox"/>
Modifies an Employee data	admin,modifies,employee	0	0	0	1	1	<input checked="" type="checkbox"/>
Modifies an Employee data	web app,saves,change	0	0	0	1	1	<input checked="" type="checkbox"/>

Résultats de mesurage automatisé de la TF du projet #1« Resto Sys »

Modifies an Employee data	web app,displays,message	0	1	0	0	1	<input checked="" type="checkbox"/>
Delete an employee	admin,enters,employee	1	0	0	0	1	<input checked="" type="checkbox"/>
Delete an employee	web app,deletes,employee	0	0	0	1	1	<input checked="" type="checkbox"/>
Delete an employee	web app,displays,confirmation	0	1	0	0	1	<input checked="" type="checkbox"/>
Add an Item	admin,enters,data	1	0	0	0	1	<input checked="" type="checkbox"/>
Add an Item	web app,retrieves,item	0	0	1	0	1	<input checked="" type="checkbox"/>
Add an Item	web app,adds,item	1	0	0	0	1	<input checked="" type="checkbox"/>
Add an Item	web app,displays,message	0	1	0	0	1	<input checked="" type="checkbox"/>
View the item list	web app,receives,waiter	1	0	0	0	1	<input checked="" type="checkbox"/>
View the item list	admin,selects,items	1	0	0	0	1	<input checked="" type="checkbox"/>
View the item list	web app,retrieves,items	0	0	1	0	1	<input checked="" type="checkbox"/>
View the item list	web app,displays,items	0	1	0	0	1	<input checked="" type="checkbox"/>
View the item list	web app,displays,message	0	1	0	0	1	<input checked="" type="checkbox"/>
View an item data	admin,enters,item	1	0	0	0	1	<input checked="" type="checkbox"/>
View an item data	web app,retrieves,item	0	0	1	0	1	<input checked="" type="checkbox"/>
View an item data	web app,displays,item	0	1	0	0	1	<input checked="" type="checkbox"/>
Modify an item	admin,modifies,fields	0	0	0	1	1	<input checked="" type="checkbox"/>
Modify an item	admin,asks,Web	1	0	0	0	1	<input checked="" type="checkbox"/>
Modify an item	web app,saves,change	0	0	0	1	1	<input checked="" type="checkbox"/>
Modify an item	web app,displays,confirmation	0	1	0	0	1	<input checked="" type="checkbox"/>
Delete an Item	admin,selects,item	1	0	0	0	1	<input checked="" type="checkbox"/>
Delete an Item	web app,deletes,item	0	0	0	1	1	<input checked="" type="checkbox"/>
Add an Item family	admin,enters,data	1	0	0	0	1	<input checked="" type="checkbox"/>
Add an Item family	web app,retrieves,item	0	0	1	0	1	<input checked="" type="checkbox"/>
Add an Item family	web app,adds,item	1	0	0	0	1	<input checked="" type="checkbox"/>
Add an Item family	web app,displays,message	0	1	0	0	1	<input checked="" type="checkbox"/>
View Item families list	web app,receives,waiter	1	0	0	0	1	<input checked="" type="checkbox"/>
View Item families list	admin,selects,item	1	0	0	0	1	<input checked="" type="checkbox"/>
View Item families list	web app,retrieves,item	0	0	1	0	1	<input checked="" type="checkbox"/>
View Item families list	web app,displays,item	0	1	0	0	1	<input checked="" type="checkbox"/>
View Item families list	web app,displays,message	0	1	0	0	1	<input checked="" type="checkbox"/>
View Item family data	web app,retrieves,data	0	0	1	0	1	<input checked="" type="checkbox"/>
View Item family data	web app,displays,data	0	1	0	0	1	<input checked="" type="checkbox"/>

Résultats de mesurage automatisé de la TF du projet #1« Resto Sys »

Modify an Item family	admin,enters,fields	1	0	0	0	1	<input checked="" type="checkbox"/>
Modify an Item family	web app,saves,change	0	0	0	1	1	<input checked="" type="checkbox"/>
Modify an Item family	web app,displays,messages	0	1	0	0	1	<input checked="" type="checkbox"/>
Delete an Item family	admin,enters,family	1	0	0	0	1	<input checked="" type="checkbox"/>
Delete an Item family	web app,deletes,family	0	0	0	1	1	<input checked="" type="checkbox"/>
Delete an Item family	web app,displays,messages	0	1	0	0	1	<input checked="" type="checkbox"/>
Add a table	admin,enters,table	1	0	0	0	1	<input checked="" type="checkbox"/>
Add a table	system,checks,table	0	0	1	0	1	<input checked="" type="checkbox"/>
Add a table	web app,adds,table	1	0	0	0	1	<input checked="" type="checkbox"/>
Add a table	web app,displays,message	0	1	0	0	1	<input checked="" type="checkbox"/>
View Tables list	app,selects,tables	1	0	0	0	1	<input checked="" type="checkbox"/>
View Tables list	admin,selects,tables	1	0	0	0	1	<input checked="" type="checkbox"/>
View Tables list	web app,retrieves,tables	0	0	1	0	1	<input checked="" type="checkbox"/>
View Tables list	system,displays,tables	0	1	0	0	1	<input checked="" type="checkbox"/>
View Tables list	web app,displays,message	0	1	0	0	1	<input checked="" type="checkbox"/>
View a table data	admin,enters,table	1	0	0	0	1	<input checked="" type="checkbox"/>
View a table data	web app,retrieves,data	0	0	1	0	1	<input checked="" type="checkbox"/>
View a table data	web app,displays,data	0	1	0	0	1	<input checked="" type="checkbox"/>
Modify table data	admin,enters,table	1	0	0	0	1	<input checked="" type="checkbox"/>
Modify table data	web app,saves,change	0	0	0	1	1	<input checked="" type="checkbox"/>
Modify table data	web app,displays,confirmation	0	1	0	0	1	<input checked="" type="checkbox"/>
Delete a table	admin,enters,table	1	0	0	0	1	<input checked="" type="checkbox"/>
Delete a table	web app,deletes,table	0	0	0	1	1	<input checked="" type="checkbox"/>
Delete a table	web app,displays,confirmation	0	1	0	0	1	<input checked="" type="checkbox"/>
View the List of Orders	admin,selects,order	1	0	0	0	1	<input checked="" type="checkbox"/>
View the List of Orders	web app,retrieves,order	0	0	1	0	1	<input checked="" type="checkbox"/>
View the List of Orders	web app,displays,order	0	1	0	0	1	<input checked="" type="checkbox"/>
View the List of Orders	web app,displays,messages	0	1	0	0	1	<input checked="" type="checkbox"/>
Delete an order	admin,selects,order	1	0	0	0	1	<input checked="" type="checkbox"/>
Delete an order	web app,deletes,order	0	0	0	1	1	<input checked="" type="checkbox"/>
Delete an order	web app,displays,message	0	1	0	0	1	<input checked="" type="checkbox"/>
Total Of Sum CFP		42	44	16	15	117	

Résultats de mesurage automatisé de la TF du projet #1« Resto Sys »

Functional process	E	X	R	W	Sum of CFP
List customers	1	2	1		4
View customer summary	1	2	2		5
View customer details (Enquiry)	1	1	1		3
View customer details (pre-Update)	1	1	1		3
Display invoice print preview	1	3	2		6
Update customer details	1	1		1	3
Add new customer	1	1		1	3
View customer booking details	1	3	2		6
Total size	8	14	9	2	33

Résultats de mesurage manuel de la TF du projet #2« ACME Car Hire »

Triplet generator By Bruel Gera x +

localhost:3000

English Use word document Use case is in English Report version 2

Select file Choisir un fichier Projet2-ACME Submit

Download

Triplets list and other statistics

Functional Process	Triplets	Entry	Exit	Read	Write	Sum Cfg	Include
List customers	user,enters,customer	1	0	0	0	1	<input checked="" type="checkbox"/>
List customers	system,searches,customer	0	0	1	0	1	<input checked="" type="checkbox"/>
List customers	system,displays,customer	0	1	0	0	1	<input checked="" type="checkbox"/>
List customers	system,displays,message	0	1	0	0	1	<input checked="" type="checkbox"/>
View customer summary	user,selects,customer	1	0	0	0	1	<input checked="" type="checkbox"/>
View customer summary	system,searches,customer	0	0	1	0	1	<input checked="" type="checkbox"/>
View customer summary	system,searches,bookings	0	0	1	0	1	<input checked="" type="checkbox"/>
View customer summary	system,displays,customer	0	1	0	0	1	<input checked="" type="checkbox"/>
View customer summary	system,displays,booking	0	1	0	0	1	<input checked="" type="checkbox"/>
View customer details	user,clicks,button	1	0	0	0	1	<input checked="" type="checkbox"/>
View customer details	system,retrieves,detail	0	0	1	0	1	<input checked="" type="checkbox"/>
View customer details	system,displays,detail	0	1	0	0	1	<input checked="" type="checkbox"/>
Display invoice print preview	user,clicks,button	1	0	0	0	1	<input checked="" type="checkbox"/>
Display invoice print preview	user,enters,data	1	0	0	0	1	<input checked="" type="checkbox"/>
Display invoice print preview	system,displays,error	0	1	0	0	1	<input checked="" type="checkbox"/>
Display invoice print preview	user,clicks,button	1	0	0	0	1	<input checked="" type="checkbox"/>
Display invoice print preview	system,searches,customer	0	0	1	0	1	<input checked="" type="checkbox"/>
Display invoice print preview	system,retrieves,bookings	0	0	1	0	1	<input checked="" type="checkbox"/>
Display invoice print preview	system,displays,detail	0	1	0	0	1	<input checked="" type="checkbox"/>

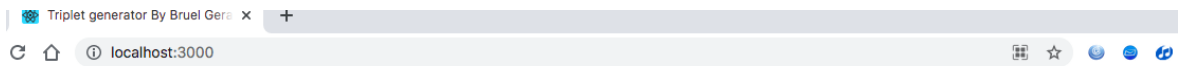
Résultats de mesurage automatisé de la TF du projet #2« ACME Car Hire »

Display invoice print preview	user,clicks,button	1	0	0	0	1	<input checked="" type="checkbox"/>
Display invoice print preview	user,enters,data	1	0	0	0	1	<input checked="" type="checkbox"/>
Display invoice print preview	system,displays,error	0	1	0	0	1	<input checked="" type="checkbox"/>
Display invoice print preview	user,clicks,button	1	0	0	0	1	<input checked="" type="checkbox"/>
Display invoice print preview	system,searches,customer	0	0	1	0	1	<input checked="" type="checkbox"/>
Display invoice print preview	system,retrieves,bookings	0	0	1	0	1	<input checked="" type="checkbox"/>
Display invoice print preview	system,displays,detail	0	1	0	0	1	<input checked="" type="checkbox"/>
Display invoice print preview	system,displays,booking	0	1	0	0	1	<input checked="" type="checkbox"/>
Display invoice print preview	system,displays,confirmation	0	1	0	0	1	<input checked="" type="checkbox"/>
Update customer details	user,enters,customer	1	0	0	0	1	<input checked="" type="checkbox"/>
Update customer details	user,presses,details	0	0	0	0	0	<input checked="" type="checkbox"/>
Update customer details	system,displays,message	0	1	0	0	1	<input checked="" type="checkbox"/>
Add new customer	user,enters,data	1	0	0	0	1	<input checked="" type="checkbox"/>
Add new customer	system,creates,customer	0	0	0	1	1	<input checked="" type="checkbox"/>
Add new customer	system,displays,message	0	1	0	0	1	<input checked="" type="checkbox"/>
View customer booking details	user,selects,view	1	0	0	0	1	<input checked="" type="checkbox"/>
View customer booking details	system,searches,bookings	0	0	1	0	1	<input checked="" type="checkbox"/>
View customer booking details	system,retrieves,bookings	0	0	1	0	1	<input checked="" type="checkbox"/>
View customer booking details	system,displays,customer	0	1	0	0	1	<input checked="" type="checkbox"/>
View customer booking details	system,displays,booking	0	1	0	0	1	<input checked="" type="checkbox"/>
View customer booking details	system,displays,message	0	1	0	0	1	<input checked="" type="checkbox"/>
Total Of Sum CFP		9	14	8	1	32	

Résultats de mesurage automatisé de la TF du projet #2« ACME Car Hire »

Functional process	E	X	R	W	Sum of CFP
Start cooking	1	2			3
Update Target temperature	2		2	1	5
Check cooker temperature	2	1	1		4
Stop Cooking	1	2			3
Control Cooker	2	1	4	2	9
Total size	8	6	7	3	24

Résultats de mesurage manuel de la TF du projet #3« Rice cooker »



Triplets list and other statistics

Functional Process	Triplets	Entry	Exit	Read	Write	Sum CFP	Include
Start cooking	button, receives, signal	1	0	0	0	1	<input checked="" type="checkbox"/>
Start cooking	heater, sends, command	0	1	0	0	1	<input checked="" type="checkbox"/>
Start cooking	lamp, sends, command	0	1	0	0	1	<input checked="" type="checkbox"/>
Update target temperature	timer, receives, signal	1	0	0	0	1	<input checked="" type="checkbox"/>
Update target temperature	timer, receives, time	1	0	0	0	1	<input checked="" type="checkbox"/>
Update target temperature	timer, gets, cooking	0	0	1	0	1	<input checked="" type="checkbox"/>
Update target temperature	timer, gets, temperature	0	0	1	0	1	<input checked="" type="checkbox"/>
Check cooker temperature	timer, receives, signal	1	0	0	0	1	<input checked="" type="checkbox"/>
Check cooker temperature	timer, gets, temperature	0	0	1	0	1	<input checked="" type="checkbox"/>
Check cooker temperature	temperature, gets, sensor	0	0	1	0	1	<input checked="" type="checkbox"/>
Check cooker temperature	heater, sends, command	0	1	0	0	1	<input checked="" type="checkbox"/>
Check cooker temperature	timer, stores, temperature	0	0	0	1	1	<input checked="" type="checkbox"/>
Stop cooking	button, receives, signal	1	0	0	0	1	<input checked="" type="checkbox"/>
Stop cooking	heater, sends, OFF	0	1	0	0	1	<input checked="" type="checkbox"/>
Stop cooking	lamp, sends, command	0	1	0	0	1	<input checked="" type="checkbox"/>
Control cooker	timer, receives, tick	1	0	0	0	1	<input checked="" type="checkbox"/>
Control cooker	timer, gets, time	0	0	1	0	1	<input checked="" type="checkbox"/>
Control cooker	timer, stores, time	0	0	0	1	1	<input checked="" type="checkbox"/>
Control cooker	timer, gets, cooking	0	0	1	0	1	<input checked="" type="checkbox"/>
Control cooker	timer, gets, temperature	0	0	1	0	1	<input checked="" type="checkbox"/>
Control cooker	timer, stores, temperature	0	0	0	1	1	<input checked="" type="checkbox"/>
Control cooker	timer, gets, target	0	0	1	0	1	<input checked="" type="checkbox"/>
Control cooker	temperature sensor, gets, sensor	0	0	1	0	1	<input checked="" type="checkbox"/>
Control cooker	heater, sends, OFF	0	1	0	0	1	<input checked="" type="checkbox"/>
Total Of Sum CFP		6	6	9	3	24	

Résultats de mesurage automatisé de la TF du projet #3« Rice cooker »

Functional process	E	X	R	W	Sum of CFP
Record an experiment	2	3	2	3	10
Start uSleuth	2	3			5
Synchronize clocks	2	3	2		7
Open a record experiment	1	4	3		8
Playback an experiment	1	4	2		7
Pause playing	1	2			3
Stop playing	1	4	2		7
Seek from image	1	3	2		6
Scroll and select from the event list	2	3	2		7
Close (eject) experiment files	2	1			3
Display about uObserve	1	1	1		3
Close uSleuth	1	2			3
Disconnected uSpy (from uSleuth)	2	1			3
Connect to uSleuth	3	2	1		6
Send events	2	1			3
Disconnected (uSpy) from uSleuth	1	1			2
Total size	25	38	17	3	83

Résultats de mesurage manuel de la TF du projet #4« uSleuth »

Triplet generator By Bruel Gera x +

localhost:3000

English Use word document Use case is in English Report version 2

Select file Choisir un fichier Projet4_uSleuth.docx Submit

Download

Triplets list and other statistics

Functional Process	Triplets	Entry	Exit	Read	Write	Sum CFP	Include
Record an experiment	UT,starts,recording	1	0	0	0	1	<input checked="" type="checkbox"/>
Record an experiment	uSleuth,displays,status	0	1	0	0	1	<input checked="" type="checkbox"/>
Record an experiment	uSleuth,creates,experiment	0	0	0	1	1	<input checked="" type="checkbox"/>
Record an experiment	uSleuth,activates,button	1	0	0	0	1	<input checked="" type="checkbox"/>
Record an experiment	uSleuth,records,audio/video	0	0	0	1	1	<input checked="" type="checkbox"/>
Record an experiment	uSleuth,discards,event	0	1	0	0	1	<input checked="" type="checkbox"/>
Record an experiment	uSleuth,captures,event	0	0	0	1	1	<input checked="" type="checkbox"/>
Record an experiment	uSleuth,adjusts,event	0	0	1	0	1	<input checked="" type="checkbox"/>
Record an experiment	uSleuth,calculates,duration	1	0	0	0	1	<input checked="" type="checkbox"/>
Record an experiment	uSleuth,displays,event	0	1	0	0	1	<input checked="" type="checkbox"/>
Record an experiment	Tester,stops,recording	0	1	0	0	1	<input checked="" type="checkbox"/>
Record an experiment	uSleuth,stops,audio/video	0	1	0	0	1	<input checked="" type="checkbox"/>
Record an experiment	uSleuth,event,log	0	0	0	0	0	<input checked="" type="checkbox"/>
Record an experiment	uSleuth,sets,audio/video	0	0	1	0	1	<input checked="" type="checkbox"/>
Record an experiment	uSleuth,discards,event	0	1	0	0	1	<input checked="" type="checkbox"/>
Start uSleuth	application,starts,application	1	0	0	0	1	<input checked="" type="checkbox"/>
Start uSleuth	uSleuth,displays,user	0	1	0	0	1	<input checked="" type="checkbox"/>
Start uSleuth	uSleuth,searches,camera	0	0	1	0	1	<input checked="" type="checkbox"/>
Start uSleuth	uSleuth,finds,camera	0	0	0	0	0	<input checked="" type="checkbox"/>
Start uSleuth	uSleuth,activates,file	1	0	0	0	1	<input checked="" type="checkbox"/>
Start uSleuth	uSleuth,displays,status	0	1	0	0	1	<input checked="" type="checkbox"/>

Résultats de mesurage automatisé de la TF du projet #4« uSleuth»

Connect to uSleuth	uSpy,displays,options	0	1	0	0	1	<input checked="" type="checkbox"/>
Connect to uSleuth	server option,selects,server	1	0	0	0	1	<input checked="" type="checkbox"/>
Connect to uSleuth	uSpy,reads,server	0	0	1	0	1	<input checked="" type="checkbox"/>
Connect to uSleuth	uSpy,sends,event	0	1	0	0	1	<input checked="" type="checkbox"/>
Connect to uSleuth	uSpy,receives,requests	1	0	0	0	1	<input checked="" type="checkbox"/>
Connect to uSleuth	time-stamped,sends,time-stamped	0	1	0	0	1	<input checked="" type="checkbox"/>
Connect to uSleuth	uSpy,receives,connection	1	0	0	0	1	<input checked="" type="checkbox"/>
Synchronize clocks	uSleuth,receives,event	1	0	0	0	1	<input checked="" type="checkbox"/>
Synchronize clocks	uSleuth,reads,network	0	0	1	0	1	<input checked="" type="checkbox"/>
Synchronize clocks	uSleuth,reads,list	0	0	1	0	1	<input checked="" type="checkbox"/>
Synchronize clocks	uSleuth,sends,synchronization	0	1	0	0	1	<input checked="" type="checkbox"/>
Synchronize clocks	uSleuth,receives,events	1	0	0	0	1	<input checked="" type="checkbox"/>
Synchronize clocks	uSleuth,calculates,time	1	0	0	0	1	<input checked="" type="checkbox"/>
Synchronize clocks	uSleuth,computes,time	0	0	1	0	1	<input checked="" type="checkbox"/>
Synchronize clocks	uSleuth,sends,connection	0	1	0	0	1	<input checked="" type="checkbox"/>
Send events	uSpy,receives,connection	1	0	0	0	1	<input checked="" type="checkbox"/>
Send events	uSpy,captures,keyboard	0	0	0	1	1	<input checked="" type="checkbox"/>
Send events	uSpy,mouse,event	0	0	0	0	0	<input checked="" type="checkbox"/>
Send events	uSpy,sends,event	0	1	0	0	1	<input checked="" type="checkbox"/>
Open a record experiment	file,selects,file	1	0	0	0	1	<input checked="" type="checkbox"/>
Open a record experiment	uSleuth,opens>window	0	0	1	0	1	<input checked="" type="checkbox"/>
Open a record experiment	directories,browses,directories	0	0	1	0	1	<input checked="" type="checkbox"/>
Open a record experiment	experiment selection,confirms,experiment	0	1	0	0	1	<input checked="" type="checkbox"/>
Open a record experiment	uSleuth,closes,file	0	1	0	0	1	<input checked="" type="checkbox"/>
Open a record experiment	uSleuth,opens,audio/video	0	0	1	0	1	<input checked="" type="checkbox"/>
Open a record experiment	uSleuth,displays,audio/video	0	1	0	0	1	<input checked="" type="checkbox"/>
Playback an experiment	option,selects,option	1	0	0	0	1	<input checked="" type="checkbox"/>
Playback an experiment	uSleuth,replaces,option	1	0	0	0	1	<input checked="" type="checkbox"/>
Playback an experiment	uSleuth,displays,status	0	1	0	0	1	<input checked="" type="checkbox"/>
Playback an experiment	uSleuth,starts,audio/video	1	0	0	0	1	<input checked="" type="checkbox"/>
Playback an experiment	uSleuth,highlights,event	0	0	1	0	1	<input checked="" type="checkbox"/>
Playback an experiment	uSleuth,replaces,option	1	0	0	0	1	<input checked="" type="checkbox"/>
Pause playing	Pause,selects,Pause	1	0	0	0	1	<input checked="" type="checkbox"/>
Pause playing	uSleuth,replaces,Pause	1	0	0	0	1	<input checked="" type="checkbox"/>

Résultats de mesurage automatisé de la TF du projet #4« uSleuth»

Stop playing	current,highlights,event	0	0	1	0	1	<input checked="" type="checkbox"/>
Stop playing	uSleuth,replaces,option	1	0	0	0	1	<input checked="" type="checkbox"/>
Seek from image	frames,browses,frames	0	0	1	0	1	<input checked="" type="checkbox"/>
Seek from image	uSleuth,sets,event	0	0	1	0	1	<input checked="" type="checkbox"/>
Seek from image	uSleuth,refreshes,event	0	1	0	0	1	<input checked="" type="checkbox"/>
Scroll and select from the event list	list,browses,list	0	0	1	0	1	<input checked="" type="checkbox"/>
Scroll and select from the event list	uSleuth,displays,portion	0	1	0	0	1	<input checked="" type="checkbox"/>
Scroll and select from the event list	event,selects,event	1	0	0	0	1	<input checked="" type="checkbox"/>
Scroll and select from the event list	uSleuth,highlights,event	0	0	1	0	1	<input checked="" type="checkbox"/>
Scroll and select from the event list	uSleuth,sets,audio/video	0	0	1	0	1	<input checked="" type="checkbox"/>
Scroll and select from the event list	uSleuth,refreshes,frame	0	1	0	0	1	<input checked="" type="checkbox"/>
Close (eject) experiment files	option,selects,option	1	0	0	0	1	<input checked="" type="checkbox"/>
Close (eject) experiment files	uSleuth,closes,audio/video	0	1	0	0	1	<input checked="" type="checkbox"/>
Close (eject) experiment files	uSleuth,empties,list	0	0	1	0	1	<input checked="" type="checkbox"/>
Close (eject) experiment files	uSleuth,refreshes,display	0	1	0	0	1	<input checked="" type="checkbox"/>
Close (eject) experiment files	uSleuth,displays,camera	0	1	0	0	1	<input checked="" type="checkbox"/>
Close (eject) experiment files	uSleuth,displays,video	0	1	0	0	1	<input checked="" type="checkbox"/>
Display About uObserve	uSleuth,displays,data	0	1	0	0	1	<input checked="" type="checkbox"/>
Display About uObserve	uSleuth,closes>window	0	1	0	0	1	<input checked="" type="checkbox"/>
Display About uObserve	uSleuth,refreshes,user	0	1	0	0	1	<input checked="" type="checkbox"/>
Disconnect uSpy from uSleuth	server,selects,server	1	0	0	0	1	<input checked="" type="checkbox"/>
Disconnect uSpy from uSleuth	uSpy,stops,mouse	0	1	0	0	1	<input checked="" type="checkbox"/>
Disconnect uSpy from uSleuth	uSpy,sends,event	0	1	0	0	1	<input checked="" type="checkbox"/>
Disconnect uSpy from uSleuth	uSleuth,stops,activities	0	1	0	0	1	<input checked="" type="checkbox"/>
Disconnect uSpy from uSleuth	uSleuth,displays,uSpy	0	1	0	0	1	<input checked="" type="checkbox"/>
Disconnect uSpy from uSleuth	uSleuth,clears,event	1	0	0	0	1	<input checked="" type="checkbox"/>
Disconnect uSpy from uSleuth	uSleuth,deactivates,button	1	0	0	0	1	<input checked="" type="checkbox"/>
Close uSleuth	uSleuth,displays,Closing	0	1	0	0	1	<input checked="" type="checkbox"/>
Close uSleuth	uSleuth,closes,interface	0	1	0	0	1	<input checked="" type="checkbox"/>
Close uSleuth	uSleuth,sends,request	0	1	0	0	1	<input checked="" type="checkbox"/>
Close uSleuth	uSleuth,stops,audio/video	0	1	0	0	1	<input checked="" type="checkbox"/>
Close uSleuth	uSleuth,event,log	0	0	0	0	0	<input checked="" type="checkbox"/>
Total Of Sum CFP		26	36	18	4	84	

Résultats de mesurage automatisé de la TF du projet #4« uSleuth »

Functional process	E	X	R	W	Sum of CFP
Add a Professor	1	1	1	1	4
Enquire on a Professor	1	2	1		4
Modify a Professor	1	1		1	3
Delete a Professor	2	2		1	5
Enquire on Course Offerings (Professor)	2	4	1		7
Create Course Offering commitments	3	3			6
Modify Course Offering commitments	3	4			7
Delete Course Offering commitments	1	3			4
Add a Student's details	1	2		1	4
Enquire on Course Offerings (Student)	1	2	1		4
Modify a Student's detail	1	1		1	3
Delete a Student's details	1	1	1	1	4
Enquire on Course Offerings (Student)	2	4			6
Create a Student schedule	2	2	1	1	6
Modify a Student schedule	2	3	2	1	8
Delete a Student schedule	1	3	1	1	6
Monitor Course Offering Enrolment progress	2	4			6
Monitor Student Schedule Enrolment progress	1	2	2		5
Close registration	2	5	2	1	10
Total size	30	49	13	10	102

Résultats de mesurage manuel de la TF du projet #5« C-Reg»

Triplet generator By Bruel Gera x +

localhost:3000

English Use word document Use case is in English Report version 2

Select file Choisir un fichier Project5-C-Reg.docx Submit

Download

Triplets list and other statistics

Functional Process	Triplets	Entry	Exit	Read	Write	Sum CFP	Include
Add a Professor	registrar,enters,detail	1	0	0	0	1	<input checked="" type="checkbox"/>
Add a Professor	C-Reg,checks,detail	0	0	1	0	1	<input checked="" type="checkbox"/>
Add a Professor	C-Reg,creates,professor	0	0	0	1	1	<input checked="" type="checkbox"/>
Add a Professor	registrar,displays,message	0	1	0	0	1	<input checked="" type="checkbox"/>
Enquire on a Professor	registrar,enters,professor	1	0	0	0	1	<input checked="" type="checkbox"/>
Enquire on a Professor	C-Reg,retrieves,professor	0	0	1	0	1	<input checked="" type="checkbox"/>
Enquire on a Professor	C-Reg,displays,professor	0	1	0	0	1	<input checked="" type="checkbox"/>
Enquire on a Professor	registrar,displays,error	0	1	0	0	1	<input checked="" type="checkbox"/>
Modify a Professor	registrar,modifies,detail	0	0	0	1	1	<input checked="" type="checkbox"/>
Modify a Professor	C-Reg,validates,detail	1	0	0	0	1	<input checked="" type="checkbox"/>
Modify a Professor	registrar,displays,error	0	1	0	0	1	<input checked="" type="checkbox"/>
Delete a professor	C-Reg,asks,course	1	0	0	0	1	<input checked="" type="checkbox"/>
Delete a professor	catalog,replies,"yes	1	0	0	0	1	<input checked="" type="checkbox"/>
Delete a professor	C-Reg,deletes,professor	0	0	0	1	1	<input checked="" type="checkbox"/>
Delete a professor	registrar,displays,error	0	1	0	0	1	<input checked="" type="checkbox"/>
Enquire on Course Offerings (Professor)	professor,enters,ID	1	0	0	0	1	<input checked="" type="checkbox"/>
Enquire on Course Offerings (Professor)	C-Reg,retrieves,professor	0	0	1	0	1	<input checked="" type="checkbox"/>
Enquire on Course Offerings (Professor)	C-Reg,sends,professor	0	1	0	0	1	<input checked="" type="checkbox"/>
Enquire on Course Offerings (Professor)	C-Reg,sends,department	0	1	0	0	1	<input checked="" type="checkbox"/>

Résultats de mesurage automatisé de la TF du projet #5« C-Reg »

Enquire on Course Offerings (Professor)	catalog,returns,course	0	1	0	0	1	<input checked="" type="checkbox"/>
Enquire on Course Offerings (Professor)	C-Reg,displays,professor	0	1	0	0	1	<input checked="" type="checkbox"/>
Enquire on Course Offerings (Professor)	C-Reg,displays,course	0	1	0	0	1	<input checked="" type="checkbox"/>
Enquire on Course Offerings (Professor)	registrar,displays,error	0	1	0	0	1	<input checked="" type="checkbox"/>
Create Course Offering commitments	professor,selects,course	1	0	0	0	1	<input checked="" type="checkbox"/>
Create Course Offering commitments	C-Reg,sends,course	0	1	0	0	1	<input checked="" type="checkbox"/>
Create Course Offering commitments	C-Reg,receives,count	1	0	0	0	1	<input checked="" type="checkbox"/>
Create Course Offering commitments	C-Reg,receives,ID	1	0	0	0	1	<input checked="" type="checkbox"/>
Create Course Offering commitments	C-Reg,displays,message	0	1	0	0	1	<input checked="" type="checkbox"/>
Modify Course Offering commitments	professor,modifies,course	0	0	0	1	1	<input checked="" type="checkbox"/>
Modify Course Offering commitments	C-Reg,sends,course	0	1	0	0	1	<input checked="" type="checkbox"/>
Modify Course Offering commitments	C-Reg,receives,count	1	0	0	0	1	<input checked="" type="checkbox"/>
Modify Course Offering commitments	C-Reg,receives,ID	1	0	0	0	1	<input checked="" type="checkbox"/>
Modify Course Offering commitments	e-mail,sends,e-mail	0	1	0	0	1	<input checked="" type="checkbox"/>
Modify Course Offering commitments	C-Reg,displays,error	0	1	0	0	1	<input checked="" type="checkbox"/>
Delete Course Offering commitments	professor,enters,presses	1	0	0	0	1	<input checked="" type="checkbox"/>
Delete Course Offering commitments	professor,presses,delete	0	0	0	0	0	<input checked="" type="checkbox"/>
Delete Course Offering commitments	C-Reg,sends,ID	0	1	0	0	1	<input checked="" type="checkbox"/>
Delete Course Offering commitments	e-mail,sends,e-mail	0	1	0	0	1	<input checked="" type="checkbox"/>
Delete Course Offering commitments	C-Reg,displays,error	0	1	0	0	1	<input checked="" type="checkbox"/>
Add a Student's data	registrar,enters,student	1	0	0	0	1	<input checked="" type="checkbox"/>
Add a Student's data	C-Reg,creates,student	0	0	0	1	1	<input checked="" type="checkbox"/>
Add a Student's data	C-Reg,obtains,student	0	1	0	0	1	<input checked="" type="checkbox"/>
Add a Student's data	C-Reg,displays,error	0	1	0	0	1	<input checked="" type="checkbox"/>
Enquire on a Student's details	registrar,enters,student	1	0	0	0	1	<input checked="" type="checkbox"/>
Enquire on a Student's details	C-Reg,obtains,student	0	1	0	0	1	<input checked="" type="checkbox"/>
Enquire on a Student's details	C-Reg,displays,student	0	1	0	0	1	<input checked="" type="checkbox"/>
Enquire on a Student's details	C-Reg,displays,error	0	1	0	0	1	<input checked="" type="checkbox"/>
Modify a Student details	registrar,modifies,student	0	0	0	1	1	<input checked="" type="checkbox"/>
Modify a Student details	registrar,displays,error	0	1	0	0	1	<input checked="" type="checkbox"/>
Delete a Student details	registrar,enters,command	1	0	0	0	1	<input checked="" type="checkbox"/>
Delete a Student details	C-Reg,checks,student	0	0	1	0	1	<input checked="" type="checkbox"/>
Delete a Student details	C-Reg,deletes,student	0	0	0	1	1	<input checked="" type="checkbox"/>
Delete a Student details	C-Reg,displays,error	0	1	0	0	1	<input checked="" type="checkbox"/>
Enquire on Course Offerings (Student)	student,enters,ID	1	0	0	0	1	<input checked="" type="checkbox"/>

Résultats de mesurage automatisé de la TF du projet #4« C-Reg »

Enquire on Course Offerings (Student)	C-Reg,asks,catalog	1	0	0	0	1	<input checked="" type="checkbox"/>
Enquire on Course Offerings (Student)	C-Reg,receives,course	1	0	0	0	1	<input checked="" type="checkbox"/>
Enquire on Course Offerings (Student)	C-Reg,displays,student	0	1	0	0	1	<input checked="" type="checkbox"/>
Enquire on Course Offerings (Student)	registrar,displays,error	0	1	0	0	1	<input checked="" type="checkbox"/>
Create a Student Schedule	student,selects,schedule	1	0	0	0	1	<input checked="" type="checkbox"/>
Create a Student Schedule	student,selects,course	1	0	0	0	1	<input checked="" type="checkbox"/>
Create a Student Schedule	C-Reg,checks,prerequisites	0	0	1	0	1	<input checked="" type="checkbox"/>
Create a Student Schedule	C-Reg,validates,course	1	0	0	0	1	<input checked="" type="checkbox"/>
Create a Student Schedule	C-Reg,saves,student	0	0	0	1	1	<input checked="" type="checkbox"/>
Create a Student Schedule	C-Reg,displays,error	0	1	0	0	1	<input checked="" type="checkbox"/>
Modify a Student Schedule	student,selects,course	1	0	0	0	1	<input checked="" type="checkbox"/>
Modify a Student Schedule	C-Reg,retrieves,student	0	0	1	0	1	<input checked="" type="checkbox"/>
Modify a Student Schedule	C-Reg,adds,item	1	0	0	0	1	<input checked="" type="checkbox"/>
Modify a Student Schedule	student,modifies,course	0	0	0	1	1	<input checked="" type="checkbox"/>
Modify a Student Schedule	C-Reg,checks,prerequisites	0	0	1	0	1	<input checked="" type="checkbox"/>
Modify a Student Schedule	C-Reg,marks,schedule	1	0	0	0	1	<input checked="" type="checkbox"/>
Modify a Student Schedule	C-Reg,sends,course	0	1	0	0	1	<input checked="" type="checkbox"/>
Modify a Student Schedule	C-Reg,displays,error	0	1	0	0	1	<input checked="" type="checkbox"/>
Delete a Student Schedule	student,selects,schedule	1	0	0	0	1	<input checked="" type="checkbox"/>
Delete a Student Schedule	C-Reg,retrieves,schedule	0	0	1	0	1	<input checked="" type="checkbox"/>
Delete a Student Schedule	C-Reg,displays,schedule	0	1	0	0	1	<input checked="" type="checkbox"/>
Delete a Student Schedule	C-Reg,prompts,student	0	1	0	0	1	<input checked="" type="checkbox"/>
Delete a Student Schedule	student,confirms,deletion	0	1	0	0	1	<input checked="" type="checkbox"/>
Delete a Student Schedule	C-Reg,deletes,schedule	0	0	0	1	1	<input checked="" type="checkbox"/>
Delete a Student Schedule	C-Reg,sends,course	0	1	0	0	1	<input checked="" type="checkbox"/>
Delete a Student Schedule	C-Reg,displays,error	0	1	0	0	1	<input checked="" type="checkbox"/>
Monitor Course Offering Enrolment progress	registrar,selects,course	1	0	0	0	1	<input checked="" type="checkbox"/>
Monitor Course Offering Enrolment progress	C-Reg,receives,course	1	0	0	0	1	<input checked="" type="checkbox"/>
Monitor Course Offering Enrolment progress	C-Reg,outputs,department	0	1	0	0	1	<input checked="" type="checkbox"/>
Monitor Course Offering Enrolment progress	C-Reg,outputs,number	0	1	0	0	1	<input checked="" type="checkbox"/>
Monitor Student Schedule Enrolment progress	registrar,selects,student	1	0	0	0	1	<input checked="" type="checkbox"/>
Monitor Student Schedule Enrolment progress	C-Reg,retrieves,student	0	0	1	0	1	<input checked="" type="checkbox"/>
Monitor Student Schedule Enrolment progress	C-Reg,retrieves,student	0	0	1	0	1	<input checked="" type="checkbox"/>
Monitor Student Schedule Enrolment progress	C-Reg,displays,names	0	1	0	0	1	<input checked="" type="checkbox"/>

Résultats de mesurage automatisé de la TF du projet #5« C-Reg »

Monitor Student Schedule Enrolment progress	C-Reg,displays,schedule	0	1	0	0	1	<input checked="" type="checkbox"/>
Close Registration	registrar,selects,registration"	1	0	0	0	1	<input checked="" type="checkbox"/>
Close Registration	C-Reg,receives,course	1	0	0	0	1	<input checked="" type="checkbox"/>
Close Registration	C-Reg,checks,student	0	0	1	0	1	<input checked="" type="checkbox"/>
Close Registration	C-Reg,sends,statuses	0	1	0	0	1	<input checked="" type="checkbox"/>
Close Registration	C-Reg,sends,schedule	0	1	0	0	1	<input checked="" type="checkbox"/>
Close Registration	C-Reg,retrieves,student	0	0	1	0	1	<input checked="" type="checkbox"/>
Close Registration	C-Reg,sends,info	0	1	0	0	1	<input checked="" type="checkbox"/>
Total Of Sum CFP		30	43	12	10	95	

Résultats de mesurage automatisé de la TF du projet #5« C-Reg »

Processus fonctionnel	E	S	L	C	Somme de PFC
Réserver un terrain au comptoir	3	2	2	1	8
Se connecter	1	2	2	0	5
Réserver un terrain par internet	4	3	1	2	10
Créer les terrains	1	2	2	1	6
Établir les horaires et disponibilités des terrain	3	4	1	4	12
Fixer le prix de location de terrain	2	6	4	1	13
Payer une réservation au comptoir	3	6	3	2	14
Payer une réservation par internet	3	3	1	1	8
Réserver un terrain en permanence	5	5	3	3	16
Sous-louer une plage horaire en réservation permanente	2	2	2	1	7
Taille totale	27	35	21	16	99

Résultats de mesurage manuel de la TF du projet #6« SGR »

Triplet generator By Bruel Gera x +

localhost:3000

French Utiliser un document word Cas d'utilisation est en Français Rapport version 2

Veuillez sélectionner un fichier Choisir un fichier Projet6-SGR-Sport.docx Soumettre

Telecharger

Liste des triplets generée et autres statistiques

Processus Fonctionnel	Triplets	Entrée	Sortie	Lecture	Ecriture	Somme Cfp	Inclure
Réserver un terrain au comptoir	membre,réserve,terrain	1	0	0	0	1	<input checked="" type="checkbox"/>
Réserver un terrain au comptoir	préposé,inscrit,numéro membre	1	0	0	0	1	<input checked="" type="checkbox"/>
Réserver un terrain au comptoir	système,valide,carte membre	0	0	1	0	1	<input checked="" type="checkbox"/>
Réserver un terrain au comptoir	système,vérifie,disponibilité terrains	0	0	1	0	1	<input checked="" type="checkbox"/>
Réserver un terrain au comptoir	système,affiche,disponibilité terrains	0	1	0	0	1	<input checked="" type="checkbox"/>
Réserver un terrain au comptoir	préposé,indique,plage	1	0	0	0	1	<input checked="" type="checkbox"/>
Réserver un terrain au comptoir	système,enregistre,réservation	0	0	0	1	1	<input checked="" type="checkbox"/>
Se connecter	membre,connecte,système	1	0	0	0	1	<input checked="" type="checkbox"/>
Se connecter	système,vérifie,code	0	0	1	0	1	<input checked="" type="checkbox"/>
Se connecter	système,valide,code	0	0	1	0	1	<input checked="" type="checkbox"/>
Se connecter	système,vérifie,transactions	0	0	1	0	1	<input checked="" type="checkbox"/>
Se connecter	système,affiche,transactions	0	1	0	0	1	<input checked="" type="checkbox"/>
Réserver un terrain par internet	membre,fait,réservation	1	0	0	0	1	<input checked="" type="checkbox"/>
Réserver un terrain par internet	système,choisit,terrain	1	0	0	0	1	<input checked="" type="checkbox"/>
Réserver un terrain par internet	membre,indique,terrain	1	0	0	0	1	<input checked="" type="checkbox"/>
Réserver un terrain par internet	système,vérifie,disponibilités	0	0	1	0	1	<input checked="" type="checkbox"/>
Réserver un terrain par internet	système,affiche,disponibilités	0	1	0	0	1	<input checked="" type="checkbox"/>
Réserver un terrain par internet	membre,confirme,choix réservation	0	1	0	0	1	<input checked="" type="checkbox"/>
Réserver un terrain par internet	membre,fournit,informations paiement	1	0	0	0	1	<input checked="" type="checkbox"/>
Réserver un terrain par internet	système,enregistre,paiement	0	0	0	1	1	<input checked="" type="checkbox"/>
Réserver un terrain par internet	svystème,enregistre,réservation	0	0	0	1	1	<input checked="" type="checkbox"/>

Résultats de mesurage automatisé de la TF du projet #6« SGR »

Réserver un terrain par internet	système,affiche,confirmation commande	0	1	0	0	1	<input checked="" type="checkbox"/>
Créer les terrains	gérant,créé,terrain	0	0	0	1	1	<input checked="" type="checkbox"/>
Créer les terrains	système,complète,informations	0	0	0	1	1	<input checked="" type="checkbox"/>
Créer les terrains	gérant,fournit,informations	1	0	0	0	1	<input checked="" type="checkbox"/>
Créer les terrains	système,vérifie,données	0	0	1	0	1	<input checked="" type="checkbox"/>
Créer les terrains	système,enregistre,terrain	0	0	0	1	1	<input checked="" type="checkbox"/>
Créer les terrains	système,affiche,message confirmation	0	1	0	0	1	<input checked="" type="checkbox"/>
Établir les horaires et disponibilités des terrains	gérant,demande,liste terrains	1	0	0	0	1	<input checked="" type="checkbox"/>
Établir les horaires et disponibilités des terrains	système,complète,date début	0	0	0	1	1	<input checked="" type="checkbox"/>
Établir les horaires et disponibilités des terrains	gérant,fournit,informations	1	0	0	0	1	<input checked="" type="checkbox"/>
Établir les horaires et disponibilités des terrains	système,vérifie,données	0	0	1	0	1	<input checked="" type="checkbox"/>
Établir les horaires et disponibilités des terrains	système,créé,plages	0	0	0	1	1	<input checked="" type="checkbox"/>
Établir les horaires et disponibilités des terrains	gérant,indique,dates	1	0	0	0	1	<input checked="" type="checkbox"/>
Établir les horaires et disponibilités des terrains	système,valide,données	0	0	1	0	1	<input checked="" type="checkbox"/>
Établir les horaires et disponibilités des terrains	système,affiche,plages	0	1	0	0	1	<input checked="" type="checkbox"/>
Établir les horaires et disponibilités des terrains	système,affiche,dates	0	1	0	0	1	<input checked="" type="checkbox"/>
Établir les horaires et disponibilités des terrains	système,finalise,liste disponibilité	1	0	0	0	1	<input checked="" type="checkbox"/>
Fixer le prix de location des terrains	gérant,fixe,prix location	1	0	0	0	1	<input checked="" type="checkbox"/>
Fixer le prix de location des terrains	système,vérifie,type terrain	0	0	1	0	1	<input checked="" type="checkbox"/>
Fixer le prix de location des terrains	système,affiche,grille	0	1	0	0	1	<input checked="" type="checkbox"/>
Fixer le prix de location des terrains	gérant,fournit,informations	1	0	0	0	1	<input checked="" type="checkbox"/>
Fixer le prix de location des terrains	système,valide,données	0	0	1	0	1	<input checked="" type="checkbox"/>
Fixer le prix de location des terrains	système,affiche,grille	0	1	0	0	1	<input checked="" type="checkbox"/>
Fixer le prix de location des terrains	système,demande,confirmation données	1	0	0	0	1	<input checked="" type="checkbox"/>
Fixer le prix de location des terrains	gérant,confirme,données	0	1	0	0	1	<input checked="" type="checkbox"/>
Fixer le prix de location des terrains	système,vérifie,données	0	0	1	0	1	<input checked="" type="checkbox"/>
Fixer le prix de location des terrains	système,enregistre,prix	0	0	0	1	1	<input checked="" type="checkbox"/>
Fixer le prix de location des terrains	système,retourne,message confirmation	0	1	0	0	1	<input checked="" type="checkbox"/>
Payer une réservation par internet	membre,accède,système	1	0	0	0	1	<input checked="" type="checkbox"/>
Payer une réservation par internet	système,valide,code	0	0	1	0	1	<input checked="" type="checkbox"/>
Payer une réservation par internet	système,affiche,profil	0	1	0	0	1	<input checked="" type="checkbox"/>
Payer une réservation par internet	système,vérifie,transactions	0	0	1	0	1	<input checked="" type="checkbox"/>
Payer une réservation par internet	système,affiche,transactions	0	1	0	0	1	<input checked="" type="checkbox"/>
Payer une réservation par internet	membre,choisit,Mes réservations	1	0	0	0	1	<input checked="" type="checkbox"/>

Résultats de mesurage automatisé de la TF du projet #6« SGR »

Payer une réservation par internet	membre,choisit,carte crédit	1	0	0	0	1	<input checked="" type="checkbox"/>
Payer une réservation par internet	système,enregistre,paiement	0	0	0	1	1	<input checked="" type="checkbox"/>
Payer une réservation par internet	système,confirme,paiement	0	1	0	0	1	<input checked="" type="checkbox"/>
Payer une réservation au comptoir	membre,paie,terrain	1	0	0	0	1	<input checked="" type="checkbox"/>
Payer une réservation au comptoir	préposé,inscrit,numéro membre	1	0	0	0	1	<input checked="" type="checkbox"/>
Payer une réservation au comptoir	système,valide,carte membre	0	0	1	0	1	<input checked="" type="checkbox"/>
Payer une réservation au comptoir	système,vérifie,état réservations	0	0	1	0	1	<input checked="" type="checkbox"/>
Payer une réservation au comptoir	système,affiche,état réservations	0	1	0	0	1	<input checked="" type="checkbox"/>
Payer une réservation au comptoir	préposé,indique,réservations	1	0	0	0	1	<input checked="" type="checkbox"/>
Payer une réservation au comptoir	préposé,inscrit,montant	1	0	0	0	1	<input checked="" type="checkbox"/>
Payer une réservation au comptoir	système,enregistre,paiement	0	0	0	1	1	<input checked="" type="checkbox"/>
Payer une réservation au comptoir	système,enregistre,réservation	0	0	0	1	1	<input checked="" type="checkbox"/>
Payer une réservation au comptoir	système,confirme,réservation	0	1	0	0	1	<input checked="" type="checkbox"/>
Réserver un terrain en permanence	préposé,inscrit,numéro membre	1	0	0	0	1	<input checked="" type="checkbox"/>
Réserver un terrain en permanence	système,valide,carte membre	0	0	1	0	1	<input checked="" type="checkbox"/>
Réserver un terrain en permanence	système,vérifie,état réservation	0	0	1	0	1	<input checked="" type="checkbox"/>
Réserver un terrain en permanence	système,affiche,état réservation	0	1	0	0	1	<input checked="" type="checkbox"/>
Réserver un terrain en permanence	préposé,confirme,choix	0	1	0	0	1	<input checked="" type="checkbox"/>
Réserver un terrain en permanence	système,enregistre,demande	0	0	0	1	1	<input checked="" type="checkbox"/>
Réserver un terrain en permanence	préposé,inscrit,informations paiement	1	0	0	0	1	<input checked="" type="checkbox"/>
Réserver un terrain en permanence	système,enregistre,paiement	0	0	0	1	1	<input checked="" type="checkbox"/>
Réserver un terrain en permanence	système,enregistre,réservation	0	0	0	1	1	<input checked="" type="checkbox"/>
Réserver un terrain en permanence	système,confirme,paiement	0	1	0	0	1	<input checked="" type="checkbox"/>
Réserver un terrain en permanence	système,affiche,	0	1	0	0	1	<input checked="" type="checkbox"/>
Réserver un terrain en permanence	système,imprime,	0	1	0	0	1	<input checked="" type="checkbox"/>
Sous louer une plage horaire en réservation permanence	membre,sous-loue,terrain	1	0	0	0	1	<input checked="" type="checkbox"/>
Sous louer une plage horaire en réservation permanence	préposé,inscrit,numéro membre	1	0	0	0	1	<input checked="" type="checkbox"/>
Sous louer une plage horaire en réservation permanence	système,valide,carte membre	0	0	1	0	1	<input checked="" type="checkbox"/>
Sous louer une plage horaire en réservation permanence	système,vérifie,état réservations	0	0	1	0	1	<input checked="" type="checkbox"/>
Sous louer une plage horaire en réservation permanence	système,affiche,état réservations membre	0	1	0	0	1	<input checked="" type="checkbox"/>
Sous louer une plage horaire en réservation permanence	préposé,demande,mise disponibilité terrain	1	0	0	0	1	<input checked="" type="checkbox"/>
Sous louer une plage horaire en réservation permanence	système,enregistre,plage	0	0	0	1	1	<input checked="" type="checkbox"/>
Sous louer une plage horaire en réservation permanence	système,confirme,sous-location	0	1	0	0	1	<input checked="" type="checkbox"/>
Somme CFP Totale		28	25	20	16	89	

Résultats de mesurage automatisé de la TF du projet #6« SGR »

Processus fonctionnel	E	S	L	C	Somme de PFC
Acheter un produit	5	7	3	4	19
Rechercher produit	1	1	1	0	3
Ouvrir un compte	1	1	1	1	4
Ajouter un article au panier	1	3	2	1	7
Modifier un panier	1	2	1	1	5
Taille totale	9	14	8	7	38

Résultats de mesurage manuel de la TF du projet #7 « SGCE »

French Utiliser un document word Cas d'utilisation est en Français Rapport version 2

Veillez selectionner un fichier Choisir un fichier Projet7_SGCE.docx

Soumettre

Telecharger

Liste des triplets generée et autres statistiques

Processus Fonctionnel	Triplets	Entrée	Sortie	Lecture	Ecriture	Somme CPP	Inclure
Acheter un produit	visiteur,accède,site	1	0	0	0	1	<input checked="" type="checkbox"/>
Acheter un produit	visiteur,saisit,article	1	0	0	0	1	<input checked="" type="checkbox"/>
Acheter un produit	système,vérifie,articles	0	0	1	0	1	<input checked="" type="checkbox"/>
Acheter un produit	système,affiche,articles	0	1	0	0	1	<input checked="" type="checkbox"/>
Acheter un produit	visiteur,démarre,recherche	1	0	0	0	1	<input checked="" type="checkbox"/>
Acheter un produit	système,vérifie,disponibilité articles	0	0	1	0	1	<input checked="" type="checkbox"/>
Acheter un produit	visiteur,ajoute,article	0	0	0	1	1	<input checked="" type="checkbox"/>
Acheter un produit	système,affiche,panier	0	1	0	0	1	<input checked="" type="checkbox"/>
Acheter un produit	système,retourne,formulaire	0	0	0	0	0	<input type="checkbox"/>
Acheter un produit	visiteur,remplit,formulaire	1	0	0	0	1	<input checked="" type="checkbox"/>
Acheter un produit	système,enregistre,compte	0	0	0	1	1	<input checked="" type="checkbox"/>
Acheter un produit	système,affiche,identité visiteur	0	1	0	0	1	<input checked="" type="checkbox"/>
Acheter un produit	système,passe,commande	0	0	0	1	1	<input checked="" type="checkbox"/>
Acheter un produit	visiteur,confirme,commande	0	1	0	0	1	<input checked="" type="checkbox"/>
Acheter un produit	système,demande,adresse livraison	0	1	0	0	1	<input checked="" type="checkbox"/>
Acheter un produit	visiteur,saisit,adresse livraison	1	0	0	0	1	<input checked="" type="checkbox"/>
Acheter un produit	système,affiche,modes livraison	0	1	0	0	1	<input checked="" type="checkbox"/>
Acheter un produit	système,demande,mode livraison	0	1	0	0	1	<input checked="" type="checkbox"/>
Acheter un produit	visiteur,sélectionne,mode livraison	1	0	0	0	1	<input checked="" type="checkbox"/>
Acheter un produit	système,affiche,page inscription	0	0	0	0	0	<input type="checkbox"/>

Résultats de mesurage automatisé de la TF du projet #7« SGCE »

Acheter un produit	visiteur,remplit,informations carte crédit	1	0	0	0	1	<input checked="" type="checkbox"/>
Acheter un produit	visiteur,valide,paielement	0	0	1	0	1	<input checked="" type="checkbox"/>
Acheter un produit	système,recherche,autorisation	0	0	1	0	1	<input checked="" type="checkbox"/>
Acheter un produit	système,enregistre,paielement	0	0	0	1	1	<input checked="" type="checkbox"/>
Acheter un produit	système,confirme,paielement	0	1	0	0	1	<input checked="" type="checkbox"/>
Rechercher un produit	visiteur,démarre,page recherche	1	0	0	0	1	<input checked="" type="checkbox"/>
Rechercher un produit	système,affiche,page recherche	0	0	0	0	0	<input type="checkbox"/>
Rechercher un produit	visiteur,l'article,	0	0	0	0	0	<input type="checkbox"/>
Rechercher un produit	système,vérifie,articles	0	0	1	0	1	<input checked="" type="checkbox"/>
Rechercher un produit	système,affiche,articles	0	1	0	0	1	<input checked="" type="checkbox"/>
Ouvrir un compte	visiteur,accède,site	1	0	0	0	1	<input checked="" type="checkbox"/>
Ouvrir un compte	visiteur,créé,compte	0	0	0	1	1	<input checked="" type="checkbox"/>
Ouvrir un compte	système,fiche,	0	0	0	0	0	<input checked="" type="checkbox"/>
Ouvrir un compte	visiteur,fiche,	0	0	0	0	0	<input type="checkbox"/>
Ouvrir un compte	système,enregistre,compte	0	0	0	1	1	<input checked="" type="checkbox"/>
Ajouter un article au panier	visiteur,demande,d'ajouter produits	0	1	0	0	1	<input checked="" type="checkbox"/>
Ajouter un article au panier	système,affiche,catalogue articles	0	1	0	0	1	<input checked="" type="checkbox"/>
Ajouter un article au panier	visiteur,ajoute,articles panier	0	0	0	1	1	<input checked="" type="checkbox"/>
Ajouter un article au panier	système,affiche,articles	0	1	0	0	1	<input checked="" type="checkbox"/>
Modifier un article au panier	visiteur,modifie,panier	0	0	0	0	0	<input type="checkbox"/>
Modifier un article au panier	système,affiche,catalogue articles	0	1	0	0	1	<input checked="" type="checkbox"/>
Modifier un article au panier	visiteur,modifie,panier	0	0	0	0	0	<input type="checkbox"/>
Modifier un article au panier	système,vérifie,articles	0	0	1	0	1	<input checked="" type="checkbox"/>
Modifier un article au panier	système,affiche,articles	0	1	0	0	1	<input checked="" type="checkbox"/>
Somme CFP Totale		9	14	6	7	36	

Résultats de mesurage automatisé de la TF du projet #7« SGCE »

Processus fonctionnel	E	S	L	C	Somme des PFC
Retirer de l'argent	6	5	2	2	15
Déposer de l'argent	4	3	3	1	11
Consulter solde	3	3	4		10
Taille totale	14	17	9	2	36

Résultats de mesurage manuel de la TF du projet #8« SGA »

French Utiliser un document word Cas d'utilisation est en Français Rapport version 2

Veillez selectionner un fichier Choisir un fichier Projet10-SAB

Soumettre

Telecharger

Liste des triplets generée et autres statistiques

Processus Fonctionnel	Triplets	Entrée	Sortie	Lecture	Ecriture	Somme Crp	Inclure
Retirer de l'argent	Porteur,introduit,carte	1	0	0	0	1	<input checked="" type="checkbox"/>
Retirer de l'argent	GAB,vérifie,carte	0	0	1	0	1	<input checked="" type="checkbox"/>
Retirer de l'argent	GAB,demande,code	0	1	0	0	1	<input checked="" type="checkbox"/>
Retirer de l'argent	Porteur,saisit,code	1	0	0	0	1	<input checked="" type="checkbox"/>
Retirer de l'argent	GAB,compare,code	0	0	1	0	1	<input checked="" type="checkbox"/>
Retirer de l'argent	GAB,sollicite,autorisation	1	0	0	0	1	<input checked="" type="checkbox"/>
Retirer de l'argent	système,donne,accord	0	1	0	0	1	<input checked="" type="checkbox"/>
Retirer de l'argent	système,indique,crédit	1	0	0	0	1	<input checked="" type="checkbox"/>
Retirer de l'argent	GAB,demande,montant retrait	0	1	0	0	1	<input checked="" type="checkbox"/>
Retirer de l'argent	Porteur,saisit,montant retrait	1	0	0	0	1	<input checked="" type="checkbox"/>
Retirer de l'argent	GAB,contrôle,montant crédit	0	0	1	0	1	<input checked="" type="checkbox"/>
Retirer de l'argent	GAB,enregistre,retrait	0	0	0	1	1	<input checked="" type="checkbox"/>
Retirer de l'argent	GAB,veut,ticket	0	0	0	0	0	<input type="checkbox"/>
Retirer de l'argent	Porteur,sélectionne,ticket	1	0	0	0	1	<input checked="" type="checkbox"/>
Retirer de l'argent	GAB,rend,carte	0	1	0	0	1	<input checked="" type="checkbox"/>
Retirer de l'argent	Porteur,repren,carte	0	0	0	0	0	<input type="checkbox"/>
Retirer de l'argent	GAB,délivre,billets	0	1	0	0	1	<input checked="" type="checkbox"/>
Retirer de l'argent	GAB,délivre,ticket	0	1	0	0	1	<input checked="" type="checkbox"/>
Retirer de l'argent	porteur,prend,billets	0	0	0	0	0	<input checked="" type="checkbox"/>
Retirer de l'argent	porteur,prend,ticket	0	0	0	0	0	<input type="checkbox"/>

Résultats de mesurage automatisé de la TF du projet #8« SGA »

Déposer de l'argent	Porteur,introduit,carte	1	0	0	0	1	<input checked="" type="checkbox"/>
Déposer de l'argent	système,vérifie,carte	0	0	1	0	1	<input checked="" type="checkbox"/>
Déposer de l'argent	système,demande,code	0	1	0	0	1	<input checked="" type="checkbox"/>
Déposer de l'argent	Porteur,saisit,code	1	0	0	0	1	<input checked="" type="checkbox"/>
Déposer de l'argent	système,vérifie,code	0	0	1	0	1	<input checked="" type="checkbox"/>
Déposer de l'argent	système,demande,chèque	0	1	0	0	1	<input checked="" type="checkbox"/>
Déposer de l'argent	Porteur,inséré,chèque	0	0	0	0	0	<input checked="" type="checkbox"/>
Déposer de l'argent	système,valide,montant	0	0	1	0	1	<input checked="" type="checkbox"/>
Déposer de l'argent	système,affiche,montant	0	1	0	0	1	<input checked="" type="checkbox"/>
Déposer de l'argent	système,enregistre,dépôt	0	0	0	1	1	<input checked="" type="checkbox"/>
Déposer de l'argent	Porteur,sélectionne,	1	0	0	0	1	<input checked="" type="checkbox"/>
Déposer de l'argent	système,retourne,ticket	0	1	0	0	1	<input checked="" type="checkbox"/>
Consulter solde	Porteur,introduit,carte	1	0	0	0	1	<input checked="" type="checkbox"/>
Consulter solde	système,vérifie,carte	0	0	1	0	1	<input checked="" type="checkbox"/>
Consulter solde	système,demande,code	0	1	0	0	1	<input checked="" type="checkbox"/>
Consulter solde	Porteur,saisit,code	1	0	0	0	1	<input checked="" type="checkbox"/>
Consulter solde	système,vérifie,code	0	0	1	0	1	<input checked="" type="checkbox"/>
Consulter solde	système,vérifie,liste opérations	0	0	1	0	1	<input checked="" type="checkbox"/>
Consulter solde	système,affiche,liste opérations	0	1	0	0	1	<input checked="" type="checkbox"/>
Consulter solde	système,affiche,solde compte	0	1	0	0	1	<input checked="" type="checkbox"/>
Somme CFP Totale		11	13	9	2	35	

Résultats de mesurage automatisé de la TF du projet #8« SGA »

Processus fonctionnel	E	S	L	C	Somme de PFC
Identification	1	2	1	0	4
Modification des portes	2	3	2	1	8
Modification d'une personne	2	3	2	1	8
Modification des groupes de personnes	2	3	2	1	8
Modification des groupes de portes	2	3	2	1	8
Recherche d'une personne en fonction d'un badge	1	1	1		3
Recherche des portes franchissables par une personne donnée	3	3	3	0	9
Recherche des groupes qui contiennent une personne donnée	2	2	2	0	6
Recherche des personnes qui appartiennent à un groupe donné	2	2	2	0	6
Modification des accès d'une groupe de personnes à un groupe de portes	4	7	7	1	19
Modification d'une semaine type	4	4	4	3	15
Recherche des droits d'accès d'une personne pour une porte donnée	4	4	4		12
Rapport des événements	2	1	1		4
Purge des événements	1			1	2
Rapport des alarmes	1	1	1		3
Ouverture manuelle des portes	2	4	4	1	11
Incendie	1	1	1		3
Autorisation de passage	1	1	1		3
Taille totale	37	45	40	10	132

Résultats de mesurage manuel de la TF du projet #9« ACAB »

Triplet generator By Bruel Gera x +

localhost:3000

French Utiliser un document word Cas d'utilisation est en Français Rapport version 2

Veillez sélectionner un fichier Choisir un fichier Projet9-ACAB.docx Soumettre

Telecharger

Liste des triplets generée et autres statistiques

Processus Fonctionnel	Triplets	Entrée	Sortie	Lecture	Ecriture	Somme Crp	Inclure
Identification	superviseur,connecte,système	1	0	0	0	1	<input checked="" type="checkbox"/>
Identification	système,vérifie,identité superviseur	0	0	1	0	1	<input checked="" type="checkbox"/>
Identification	système,autorise,connexion	0	1	0	0	1	<input checked="" type="checkbox"/>
Modification des portes	superviseur,demande,liste portes	1	0	0	0	1	<input checked="" type="checkbox"/>
Modification des portes	système,vérifie,liste portes	0	0	1	0	1	<input checked="" type="checkbox"/>
Modification des portes	système,affiche,liste portes	0	1	0	0	1	<input checked="" type="checkbox"/>
Modification des portes	superviseur,choisit,porte	1	0	0	0	1	<input checked="" type="checkbox"/>
Modification des portes	système,vérifie,porte	0	0	1	0	1	<input checked="" type="checkbox"/>
Modification des portes	système,affiche,informations	0	1	0	0	1	<input checked="" type="checkbox"/>
Modification des portes	superviseur,modifie,informations	1	0	0	0	1	<input checked="" type="checkbox"/>
Modification des portes	système,vérifie,informations	0	0	1	0	1	<input checked="" type="checkbox"/>
Modification des portes	système,enregistre,informations	0	0	0	1	1	<input checked="" type="checkbox"/>
Modification d'une personne	superviseur,demande,liste personnes	1	0	0	0	1	<input checked="" type="checkbox"/>
Modification d'une personne	système,vérifie,personnes	0	0	1	0	1	<input checked="" type="checkbox"/>
Modification d'une personne	système,affiche,personnes	0	1	0	0	1	<input checked="" type="checkbox"/>
Modification d'une personne	superviseur,choisit,personne	1	0	0	0	1	<input checked="" type="checkbox"/>
Modification d'une personne	système,vérifie,personne	0	0	1	0	1	<input checked="" type="checkbox"/>
Modification d'une personne	système,affiche,état	0	1	0	0	1	<input checked="" type="checkbox"/>
Modification d'une personne	système,affiche,confidentialité	0	1	0	0	1	<input checked="" type="checkbox"/>
Modification d'une personne	superviseur,modifie,informations	1	0	0	0	1	<input checked="" type="checkbox"/>
Modification d'une personne	système,vérifie,informations	0	0	1	0	1	<input checked="" type="checkbox"/>

Résultats de mesurage automatisé de la TF du projet #9« ACAB »

Modification des portes	système,enregistre,informations	0	0	0	1	1	<input checked="" type="checkbox"/>
Modification d'une personne	superviseur,demande,liste personnes	1	0	0	0	1	<input checked="" type="checkbox"/>
Modification d'une personne	système,vérifie,personnes	0	0	1	0	1	<input checked="" type="checkbox"/>
Modification d'une personne	système,affiche,personnes	0	1	0	0	1	<input checked="" type="checkbox"/>
Modification d'une personne	superviseur,choisit,personne	1	0	0	0	1	<input checked="" type="checkbox"/>
Modification d'une personne	système,vérifie,personne	0	0	1	0	1	<input checked="" type="checkbox"/>
Modification d'une personne	système,affiche,état	0	1	0	0	1	<input checked="" type="checkbox"/>
Modification d'une personne	système,affiche,confidentialité	0	1	0	0	1	<input checked="" type="checkbox"/>
Modification d'une personne	superviseur,modifie,informations	1	0	0	0	1	<input checked="" type="checkbox"/>
Modification d'une personne	système,vérifie,informations	0	0	1	0	1	<input checked="" type="checkbox"/>
Modification d'une personne	système,enregistre,informations	0	0	0	1	1	<input checked="" type="checkbox"/>
Modification des groupes de personnes	superviseur,demande,liste groupes	1	0	0	0	1	<input checked="" type="checkbox"/>
Modification des groupes de personnes	système,vérifie,groupes	0	0	1	0	1	<input checked="" type="checkbox"/>
Modification des groupes de personnes	système,affiche,groupes personnes	0	1	0	0	1	<input checked="" type="checkbox"/>
Modification des groupes de personnes	système,affiche,informations	0	1	0	0	1	<input checked="" type="checkbox"/>
Modification des groupes de personnes	superviseur,modifie,informations	1	0	0	0	1	<input checked="" type="checkbox"/>
Modification des groupes de personnes	système,vérifie,informations	0	0	1	0	1	<input checked="" type="checkbox"/>
Modification des groupes de personnes	système,enregistre,informations	0	0	0	1	1	<input checked="" type="checkbox"/>
Modification des groupes de portes	superviseur,demande,liste groupes portes	1	0	0	0	1	<input checked="" type="checkbox"/>
Modification des groupes de portes	système,vérifie,groupes portes	0	0	1	0	1	<input checked="" type="checkbox"/>
Modification des groupes de portes	système,affiche,groupes portes	0	1	0	0	1	<input checked="" type="checkbox"/>
Modification des groupes de portes	superviseur,choisit,groupe portes	1	0	0	0	1	<input checked="" type="checkbox"/>
Modification des groupes de portes	système,vérifie,groupe portes	0	0	1	0	1	<input checked="" type="checkbox"/>
Modification des groupes de portes	système,affiche,informations	0	1	0	0	1	<input checked="" type="checkbox"/>
Modification des groupes de portes	superviseur,modifie,informations	1	0	0	0	1	<input checked="" type="checkbox"/>
Modification des groupes de portes	système,vérifie,informations	0	0	1	0	1	<input checked="" type="checkbox"/>
Modification des groupes de portes	système,enregistre,informations	0	0	0	1	1	<input checked="" type="checkbox"/>
Recherche d'une personne	superviseur,donne,numéro badge	0	1	0	0	1	<input checked="" type="checkbox"/>
Recherche d'une personne	système,retrouve,personne	0	0	1	0	1	<input checked="" type="checkbox"/>
Recherche d'une personne	système,affiche,informations	0	1	0	0	1	<input checked="" type="checkbox"/>
Recherche des portes franchissables par une personne donnée	superviseur,demande,liste personnes	1	0	0	0	1	<input checked="" type="checkbox"/>
Recherche des portes franchissables par une personne donnée	système,vérifie,personnes	0	0	1	0	1	<input checked="" type="checkbox"/>
Recherche des portes franchissables par une personne donnée							

Résultats de mesurage automatisé de la TF du projet #9« ACAB »

Recherche des portes franchissables par une personne donnée	système,affiche,personnes	0	1	0	0	1	<input checked="" type="checkbox"/>
Recherche des portes franchissables par une personne donnée	superviseur,choisit,personne	1	0	0	0	1	<input checked="" type="checkbox"/>
Recherche des portes franchissables par une personne donnée	système,vérifie,personne	0	0	1	0	1	<input checked="" type="checkbox"/>
Recherche des portes franchissables par une personne donnée	système,affiche,liste portes	0	1	0	0	1	<input checked="" type="checkbox"/>
Recherche des portes franchissables par une personne donnée	superviseur,choisit,porte	1	0	0	0	1	<input checked="" type="checkbox"/>
Recherche des portes franchissables par une personne donnée	système,vérifie,porte	0	0	1	0	1	<input checked="" type="checkbox"/>
Recherche des portes franchissables par une personne donnée	système,affiche,état	0	1	0	0	1	<input checked="" type="checkbox"/>
Recherche des portes franchissables par une personne donnée	système,vérifie,accès	0	0	1	0	1	<input checked="" type="checkbox"/>
Recherche des portes franchissables par une personne donnée	système,affiche,accès	0	1	0	0	1	<input checked="" type="checkbox"/>
Recherche des groupes qui contiennent une personne donnée	superviseur,demande,liste personnes	1	0	0	0	1	<input checked="" type="checkbox"/>
Recherche des groupes qui contiennent une personne donnée	système,vérifie,personnes	0	0	1	0	1	<input checked="" type="checkbox"/>
Recherche des groupes qui contiennent une personne donnée	système,affiche,personnes	0	1	0	0	1	<input checked="" type="checkbox"/>
Recherche des groupes qui contiennent une personne donnée	superviseur,choisit,personne	1	0	0	0	1	<input checked="" type="checkbox"/>
Recherche des groupes qui contiennent une personne donnée	système,vérifie,liste groupes	0	0	1	0	1	<input checked="" type="checkbox"/>
Recherche des groupes qui contiennent une personne donnée	système,affiche,liste groupes	0	1	0	0	1	<input checked="" type="checkbox"/>
Recherche des personnes qui appartiennent à un groupe donné	superviseur,demande,liste groupes personnes	1	0	0	0	1	<input checked="" type="checkbox"/>
Recherche des personnes qui appartiennent à un groupe donné	système,vérifie,liste groupes personnes	0	0	1	0	1	<input checked="" type="checkbox"/>
Recherche des personnes qui appartiennent à un groupe donné	système,affiche,groupes personnes	0	1	0	0	1	<input checked="" type="checkbox"/>
Recherche des personnes qui appartiennent à un groupe donné	superviseur,choisit,groupe	1	0	0	0	1	<input checked="" type="checkbox"/>

Résultats de mesurage automatisé de la TF du projet #9« ACAB »

Recherche des personnes qui appartiennent à un groupe donné	système,vérifie,liste personnes	0	0	1	0	1	<input checked="" type="checkbox"/>
Recherche des personnes qui appartiennent à un groupe donné	système,affiche,liste personnes	0	1	0	0	1	<input checked="" type="checkbox"/>
Recherche des accès d'un groupe de personnes à un groupe de porte	superviseur,demande,liste groupes personnes	1	0	0	0	1	<input checked="" type="checkbox"/>
Recherche des accès d'un groupe de personnes à un groupe de porte	système,vérifie,groupes	0	0	1	0	1	<input checked="" type="checkbox"/>
Recherche des accès d'un groupe de personnes à un groupe de porte	système,affiche,groupes	0	1	0	0	1	<input checked="" type="checkbox"/>
Recherche des accès d'un groupe de personnes à un groupe de porte	superviseur,choisit,groupe	1	0	0	0	1	<input checked="" type="checkbox"/>
Recherche des accès d'un groupe de personnes à un groupe de porte	système,vérifie,nom groupe	0	0	1	0	1	<input checked="" type="checkbox"/>
Recherche des accès d'un groupe de personnes à un groupe de porte	système,affiche,nom groupe	0	1	0	0	1	<input checked="" type="checkbox"/>
Recherche des accès d'un groupe de personnes à un groupe de porte	système,affiche,liste personnes	0	1	0	0	1	<input checked="" type="checkbox"/>
Recherche des accès d'un groupe de personnes à un groupe de porte	système,vérifie,accès	0	0	1	0	1	<input checked="" type="checkbox"/>
Recherche des accès d'un groupe de personnes à un groupe de porte	système,affiche,liste accès	0	1	0	0	1	<input checked="" type="checkbox"/>
Recherche des accès d'un groupe de personnes à un groupe de porte	superviseur,choisit,accès	1	0	0	0	1	<input checked="" type="checkbox"/>
Recherche des accès d'un groupe de personnes à un groupe de porte	système,vérifie,plages	0	0	1	0	1	<input checked="" type="checkbox"/>
Recherche des accès d'un groupe de personnes à un groupe de porte	système,affiche,plages	0	1	0	0	1	<input checked="" type="checkbox"/>
Recherche des accès d'un groupe de personnes à un groupe de porte	superviseur,modifie,informations	1	0	0	0	1	<input checked="" type="checkbox"/>
Recherche des accès d'un groupe de personnes à un groupe de porte	système,vérifie,informations	0	0	1	0	1	<input checked="" type="checkbox"/>
Recherche des accès d'un groupe de personnes à un groupe de porte	système,enregistre,informations	0	0	0	1	1	<input checked="" type="checkbox"/>
Modification d'une semaine type	superviseur,demande,liste semaines	1	0	0	0	1	<input checked="" type="checkbox"/>
Modification d'une semaine type	système,vérifie,semaines	0	0	1	0	1	<input checked="" type="checkbox"/>
Modification d'une semaine type	système,affiche,semaines	0	1	0	0	1	<input checked="" type="checkbox"/>
Modification d'une semaine type	superviseur,choisit,semaine type	1	0	0	0	1	<input checked="" type="checkbox"/>

Résultats de mesurage automatisé de la TF du projet #9« ACAB »

Modification d'une semaine type	système,vérifie,nom semaine	0	0	1	0	1	✓
Modification d'une semaine type	système,affiche,	0	1	0	0	1	✓
Modification d'une semaine type	système,vérifie,accès	0	0	1	0	1	✓
Modification d'une semaine type	système,affiche,accès	0	1	0	0	1	✓
Modification d'une semaine type	superviseur,modifie,informations	1	0	0	0	1	✓
Modification d'une semaine type	système,vérifie,informations	0	0	1	0	1	✓
Modification d'une semaine type	système,enregistre,informations	0	0	0	1	1	✓
Recherche des droits d'accès d'une personne pour une porte donnée	superviseur,demande,liste personnes	1	0	0	0	1	✓
Recherche des droits d'accès d'une personne pour une porte donnée	système,demande,liste portes	1	0	0	0	1	✓
Recherche des droits d'accès d'une personne pour une porte donnée	système,vérifie,personnes	0	0	1	0	1	✓
Recherche des droits d'accès d'une personne pour une porte donnée	système,affiche,personnes	0	1	0	0	1	✓
Recherche des droits d'accès d'une personne pour une porte donnée	système,vérifie,portes	0	0	1	0	1	✓
Recherche des droits d'accès d'une personne pour une porte donnée	système,affiche,portes	0	1	0	0	1	✓
Recherche des droits d'accès d'une personne pour une porte donnée	superviseur,choisit,personne	1	0	0	0	1	✓
Recherche des droits d'accès d'une personne pour une porte donnée	système,vérifie,personne	0	0	1	0	1	✓
Recherche des droits d'accès d'une personne pour une porte donnée	système,vérifie,porte	0	0	1	0	1	✓
Recherche des droits d'accès d'une personne pour une porte donnée	système,affiche,calendrier	0	1	0	0	1	✓
Recherche des droits d'accès d'une personne pour une porte donnée	système,vérifie,plages	0	0	1	0	1	✓
Recherche des droits d'accès d'une personne pour une porte donnée	système,affiche,plages	0	1	0	0	1	✓
Rapport des événements	gardien,spécifie,dates débuts	1	0	0	0	1	✓
Rapport des événements	système,vérifie,événements	0	0	1	0	1	✓
Rapport des événements	système,affiche,événements	0	1	0	0	1	✓
Purger des événements	gardien,spécifie,dates débuts	1	0	0	0	1	✓
Purger des événements	système,détruit,événements	0	0	0	1	1	✓
Rapport des alarmes	gardien,spécifie,délai	1	0	0	0	1	✓

Résultats de mesurage automatisé de la TF du projet #9« ACAB »

Rapport des alarmes	système,affiche,alarmes	0	1	0	0	1	<input checked="" type="checkbox"/>
Ouverture manuelle des portes	gardien,demande,liste portes	1	0	0	0	1	<input checked="" type="checkbox"/>
Ouverture manuelle des portes	système,vérifie,liste portes	0	0	1	0	1	<input checked="" type="checkbox"/>
Ouverture manuelle des portes	système,affiche,liste portes	0	1	0	0	1	<input checked="" type="checkbox"/>
Ouverture manuelle des portes	gardien,choisit,porte	1	0	0	0	1	<input checked="" type="checkbox"/>
Ouverture manuelle des portes	système,affiche,état	0	1	0	0	1	<input checked="" type="checkbox"/>
Ouverture manuelle des portes	système,affiche,valeur confidentialité	0	1	0	0	1	<input checked="" type="checkbox"/>
Ouverture manuelle des portes	système,affiche,événements	0	1	0	0	1	<input checked="" type="checkbox"/>
Ouverture manuelle des portes	gardien,force,ouverture porte	1	0	0	0	1	<input checked="" type="checkbox"/>
Ouverture manuelle des portes	système,enregistre,événement	0	0	0	1	1	<input checked="" type="checkbox"/>
Incendie	gardien,déclenche,incendie	1	0	0	0	1	<input checked="" type="checkbox"/>
Incendie	système,ouvre,portes	0	0	1	0	1	<input checked="" type="checkbox"/>
Autorisation passage	personne,présente,badge	1	0	0	0	1	<input checked="" type="checkbox"/>
Autorisation passage	système,détermine,accès	0	0	1	0	1	<input checked="" type="checkbox"/>
Autorisation passage	système,commande,ouverture porte	0	1	0	0	1	<input checked="" type="checkbox"/>
Autorisation passage	gardien,force,ouverture porte	1	0	0	0	1	<input checked="" type="checkbox"/>
Autorisation passage	système,détermine,accès	0	0	1	0	1	<input checked="" type="checkbox"/>
Autorisation passage	système,commande,ouverture porte	0	1	0	0	1	<input checked="" type="checkbox"/>
Autorisation passage	système,enregistre,événement	0	0	0	1	1	<input checked="" type="checkbox"/>
Somme CFP Totale		38	40	40	9	127	

Résultats de mesurage automatisé de la TF du projet #9« ACAB »

Processus fonctionnel	E	S	L	C	Somme des PFC
Acheter des actions sur le Web	3	4	3	2	12
Se faire indemniser à la suite d'un accident de voiture	4	1	3	4	12
Enregistrer l'arrivée d'un colis	4	2	2	1	9
Taille totale	11	7	8	7	33

Résultats de mesurage manuel de la TF du projet #10

French ▼ Utiliser un document word ▼ Cas d'utilisation est en Français ▼ Rapport version 2 ▼

Veillez sélectionner un fichier

Choisir un fichier

Projet-Cockburn.docx

Soumettre

Telecharger

Liste des triplets generée et autres statistiques

Processus Fonctionnel	Triplets	Entrée	Sortie	Lecture	Ecriture	Somme CFP	Inclure
Acheter des actions sur le Web	L'acheteur,sélectionne,fonction	1	0	0	0	1	<input checked="" type="checkbox"/>
Acheter des actions sur le Web	CPF,obtient,nom site	0	1	0	0	1	<input checked="" type="checkbox"/>
Acheter des actions sur le Web	CPF,ouvre,connexion site	0	1	0	0	1	<input checked="" type="checkbox"/>
Acheter des actions sur le Web	acheteur,parcourt,site	1	0	0	0	1	<input checked="" type="checkbox"/>
Acheter des actions sur le Web	acheteur,passe,ordres achat	0	0	0	1	1	<input checked="" type="checkbox"/>
Acheter des actions sur le Web	CPF,intercepte,réponses site	0	1	0	0	1	<input checked="" type="checkbox"/>
Acheter des actions sur le Web	CPF,actualise,portefeuille l'acheteur	0	0	0	1	1	<input checked="" type="checkbox"/>
Acheter des actions sur le Web	CPF,montre,état portefeuille	0	1	0	0	1	<input checked="" type="checkbox"/>
Acheter des actions sur le Web	CPF,enregistre,délai	0	0	0	1	1	<input checked="" type="checkbox"/>
Acheter des actions sur le Web	système,reçoit,suggestion	0	0	1	0	1	<input checked="" type="checkbox"/>
Acheter des actions sur le Web	CPF,lance,	1	0	0	0	1	<input checked="" type="checkbox"/>
Acheter des actions sur le Web	CPF,actualise,achat	0	0	0	1	1	<input checked="" type="checkbox"/>
Se faire indemniser suite à un accident	demandeur,soumet,demande	1	0	0	0	1	<input checked="" type="checkbox"/>
Se faire indemniser suite à un accident	demandeur,soumet,données	1	0	0	0	1	<input checked="" type="checkbox"/>
Se faire indemniser suite à un accident	compagnie,vérifie,police	0	0	1	0	1	<input checked="" type="checkbox"/>
Se faire indemniser suite à un accident	agent,examine,dossier	0	0	1	0	1	<input checked="" type="checkbox"/>
Se faire indemniser suite à un accident	Compagnie,demande,informations	1	0	0	0	1	<input checked="" type="checkbox"/>
Se faire indemniser suite à un accident	Demandeur,fournit,informations	1	0	0	0	1	<input checked="" type="checkbox"/>
Se faire indemniser suite à un accident	Compagnie,indemnise,Demandeur	0	0	0	1	1	<input checked="" type="checkbox"/>
Se faire indemniser suite à un accident	compagnie,clôt,dossier	0	1	0	0	1	<input checked="" type="checkbox"/>

Se faire indemniser suite à un accident	compagnie,clôt,dossier	0	1	0	0	1	<input checked="" type="checkbox"/>
Se faire indemniser suite à un accident	Compagnie,rejette,demande	0	1	0	0	1	<input checked="" type="checkbox"/>
Se faire indemniser suite à un accident	compagnie,enregistre,démarches	0	0	0	1	1	<input checked="" type="checkbox"/>
Se faire indemniser suite à un accident	compagnie,fixe,règlement	0	0	0	1	1	<input checked="" type="checkbox"/>
Enregistrer l'arrivée d'un colis	L'agent,reçoit,colis	0	0	0	0	0	<input type="checkbox"/>
Enregistrer l'arrivée d'un colis	L'agent,ouvre,colis	0	0	0	0	0	<input type="checkbox"/>
Enregistrer l'arrivée d'un colis	L'agent,valide,l'identifiant colis	0	0	1	0	1	<input checked="" type="checkbox"/>
Enregistrer l'arrivée d'un colis	L'agent,signe,bon livraison papier	0	0	0	0	0	<input checked="" type="checkbox"/>
Enregistrer l'arrivée d'un colis	Agent,enregistre,colis	0	0	0	1	1	<input checked="" type="checkbox"/>
Enregistrer l'arrivée d'un colis	système,stocke,nombre sacs	1	0	0	0	1	<input checked="" type="checkbox"/>
Enregistrer l'arrivée d'un colis	système,stocke,valeur	1	0	0	0	1	<input checked="" type="checkbox"/>
Enregistrer l'arrivée d'un colis	système,vérifie,compagnie transport	0	0	1	0	1	<input checked="" type="checkbox"/>
Enregistrer l'arrivée d'un colis	système,stocke,compagnie	1	0	0	0	1	<input checked="" type="checkbox"/>
Somme CFP Totale		10	6	5	8	29	

Résultats de mesurage automatisé de la TF du projet #10

RÉFÉRENCES

- Abran, A. (2010). *Software metrics and software metrology*. John Wiley & Sons.
- Abran, A. (2015). *Software project estimation : The fundamentals for providing high quality information to decision makers*. John Wiley & Sons.
- Abran, A., Desharnais, J.-M. et Aziz, F. (2016). 3.5 measurement convertibility from function points to cosmic ffp. *Cosmic Function Points : Theory and Advanced Practices*, p. 214.
- Abran, A., Fagg, P. et Lesterhuis, A. (2019). The cosmic functional size measurement method version 5.0. [Disponible en ligne : <https://cosmic-sizing.org/>; consulté le 7-novembre-2021].
- Abran, A., Lesterhuis, A., Symons, C. et Trudel, S. (2018). Rice cooker case study v2.0.1 : Realtime software size measurement. [Disponible en ligne : <https://cosmic-sizing.org/wp-content/uploads/2018/08/Rice-Cooker-2.0.1-GoogleDocs.pdf>; consulté en octobre-2021].
- Abran, A. et Paton, K. (1997). Automation of function points counting : Feasibility and accuracy ?
- Abualkishik, A. Z. (2019). A review study on the accuracy of conversion between ifpug fpa and cosmic function points. *Smart Technologies and Innovation for a Sustainable Future*.
- Abualkishik, A. Z. et Lavazza, L. (2018). Ifpug function points to cosmic function points convertibility : A fine-grained statistical approach. *Information and Software Technology*, 97, 179–191.
- Albrecht, A. J. (1979). Measuring application development productivity. Dans *Proc. Joint Share, Guide, and IBM Application Development Symposium, 1979*.
- Albrecht, A. J. et Gaffney, J. E. (1983). Software function, source lines of code, and development effort prediction : a software science validation. *IEEE transactions on software engineering*, (6), 639–648.

- Ambler, S. W. (2003). Examining the “big requirements up front (bruf) approach”. *Scott &ℓ*.
- Anderson, J. R. (2013). *The architecture of cognition*. Psychology Press.
- Andersson, A. (2017). *Measurement Technology for Process Automation*. CRC Press.
- Aussenac-Gilles, N., Despres, S. et Szulman, S. (2008). The terminae method and platform for ontology engineering from texts.
- Azzouz, S. et Abran, A. (2004). A proposed measurement role in the rational unified process and its implementation with iso 19761 : Cosmic-ffp.
- Baader, F., Calvanese, D., McGuinness, D., Patel-Schneider, P., Nardi, D. et al. (2003). *The description logic handbook : Theory, implementation and applications*. Cambridge university press.
- Baader, F., Ganter, B., Sertkaya, B. et Sattler, U. (2007). Completing description logic knowledge bases using formal concept analysis. Dans *IJCAI*, volume 7, 230–235.
- Baader, F., Lutz, C., Milicic, M., Sattler, U. et Wolter, F. (2005). Integrating description logics and action formalisms : First results. Dans *AAAI*, volume 5, 572–577.
- Baddeley, A. D. et Hitch, G. J. (2000). Development of working memory : Should the pascual-leone and the baddeley and hitch models be merged ? *Journal of experimental child psychology*, 77(2), 128–137.
- Barkallah, S. (2012). *Conception d’un profil UML spécifique à la méthode COSMIC-ISO 19761 pour supporter la mesure de la taille fonctionnelle des logiciels*. (Thèse de doctorat). École de technologie supérieure.
- Barone, L., Monteleone, M. et Silberztein, M. (2017). *Automatic Processing of Natural-Language Electronic Texts with NooJ : 10th International Conference, NooJ 2016, České Budějovice, Czech Republic, June 9-11, 2016, Revised Selected Papers*, volume 667. Springer.
- Beck, K. et West, D. (2014). User stories in agile software development.
- Beizer, B. (2003). *Software testing techniques*. Dreamtech Press.
- Bennaceur, A., Tun, T. T., Yu, Y. et Nuseibeh, B. (2019). Requirements engineering. In *Handbook of Software Engineering* 51–92. Springer.

Berardi, D., Calvanese, D. et De Giacomo, G. (2005). Reasoning on uml class diagrams. *Artificial intelligence*, 168(1-2), 70–118.

Berners-Lee, T., Hendler, J. et Lassila, O. (2001). The semantic web. *Scientific american*, 284(5), 34–43.

Bévo, V., Lévesque, G. et Abran, A. (1999). Application de la methode ffp a partir d'une specification selon la notation uml : Compte rendu des premiers essais d'application et questions. Dans *9th International Workshop Software Measurement, Lac Supérieur, Canada*.

Bevo, W. E. V. (2005). *Analyse et formalisation ontologique des procédures de mesure associées aux méthodes de mesure de la taille fonctionnelle des logiciels : De nouvelles perspectives pour la mesure*. (Thèse de doctorat).

Black, S. E. et Wigg, J. (1999). X-ray : A multi-language industrial strength tool. Dans *9th International Workshop on Software Measurement*, 39–50.

Boehm, B. *et al.* (1981). Software engineering economics. *New York, 197*.

Boehm, B., Abts, C., Brown, A., Chulani, S., Clark, B., Horowitz, E., Madachy, R., Reifer, J. et Steece, B. (2000). Software cost estimation with cocomo ii. prentice hall ptr. *Upper Saddle River, NJ*.

Boy, G. A. (2013). Cognitive engineering. In *Orchestrating human-centered design* 35–57. Springer.

Breitman, K., Casanova, M. A. et Truszkowski, W. (2007). *Semantic web : concepts, technologies and applications*. Springer Science & Business Media.

Chamkha, N., Sellami, A. et Abran, A. (2018). Automated cosmic measurement of java swing applications throughout their development life cycle. Dans *IWSM-Mensura*, 20–33.

Charton, E., Torres-Moreno, J.-M. et SanJuan, E. (2008). Réécriture statistique de phrases basée sur des modèles de langage. *les actes de JADT*, 1–11.

Cimiano, P. et Völker, J. (2005). A framework for ontology learning and data-driven change discovery. Dans *Proceedings of the 10th International Conference on Applications of Natural Language to Information Systems (NLDB)*, 227–238. Springer.

Clark, B. K. (1997). *The effects of software process maturity on software development effort*. University of Southern California.

Cleland-Huang, J., Gotel, O., Zisman, A. *et al.* (2012). *Software and systems*

traceability, volume 2. Springer.

Cockburn, A. (2001). *Writing effective use cases*. Pearson Education India.

Cohen, N. J. et Squire, L. R. (1980). Preserved learning and retention of pattern-analyzing skill in amnesia : Dissociation of knowing how and knowing that. *Science*, 210(4466), 207–210.

Coman, I. D., Sillitti, A. et Succi, G. (2009). A case-study on using an automated in-process software engineering measurement and analysis system in an industrial environment. Dans *2009 IEEE 31st International Conference on Software Engineering*, 89–99. IEEE.

Commeynes, C., Abran, A. et Djouab, R. (2016). Effort estimation with story points and cosmic function points -. *Software Measurement News, Las Vegas, USA*, 21, 25–36.

Delbecq, N. (2006). Linguistique cognitive. comprendre comment fonctionne le langage. préface de Jean-Rémi Lapaire. *Bruxelles : Champs linguistiques de Boeck*.

Desgranges, B. et Eustache, F. (2011). Les conceptions de la mémoire déclarative d'endel Tulving et leurs conséquences actuelles. *Revue de neuropsychologie*, 3(2), 94–103.

Diab, H., Frappier, M. et St-Denis, R. (2002). A formal definition of function points for automated measurement of b specifications. Dans *International Conference on Formal Engineering Methods*, 483–494. Springer.

Diab, H., Koukane, F., Frappier, M. et St-Denis, R. (2005). μ crose : automated measurement of cosmic-ffp for rational rose realtime. *Information and Software Technology*, 47(3), 151–166.

Domingos, P. (2012). A few useful things to know about machine learning. *Communications of the ACM*, 55(10), 78–87.

Dumke, R. et Abran, A. (2016). *COSMIC Function Points : Theory and Advanced Practices*. CRC Press.

El Ghazi, H. (2009). *Une approche multi-vues pour la gestion de la traçabilité des exigences : (MV-TMM multi view traceability management method)*. (Thèse de doctorat). Paris 1.

Fenton, N. et Bieman, J. (2019). *Software metrics : a rigorous and practical approach*. CRC press.

Fortuna, B., Grobelnik, M. et Mladenic, D. (2006). Semi-automatic

- data-driven ontology construction system. Dans *Proceedings of the 9th International multi-conference Information Society IS-2006, Ljubljana, Slovenia*, 223–226.
- Fotso, S. J. T. (2019). *Vers une approche formelle d'ingénierie des exigences outillée et éprouvée*. (Thèse de doctorat). Université Paris Est Créteil Val de Marne (Paris 12) ; Université de Paris Est.
- Galorath, D. D. et Evans, M. W. (2006). *Software sizing, estimation, and risk management : when performance is measured performance improves*. Auerbach Publications.
- Garmus, D. et Herron, D. (1996). *Measuring the software process : a practical guide to functional measurements*. Prentice-Hall, Inc.
- Gencel, C. et Bideau, C. (2012). Exploring the convertibility between ifpug and cosmic function points : preliminary findings. Dans *2012 Joint Conference of the 22nd International Workshop on Software Measurement and the 2012 Seventh International Conference on Software Process and Product Measurement*, 170–177. IEEE.
- Gilb, T. et Graham, D. (1993). *Software inspections*. Addison-Wesley Reading, Massachusetts.
- Goldfarb, W. D. (1979). Logic in the twenties : the nature of the quantifier¹. *The Journal of Symbolic Logic*, 44(3), 351–368.
- Goodfellow, I., Bengio, Y. et Courville, A. (2016). *Deep learning*. MIT press.
- Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge acquisition*, 5(2), 199–220.
- Gérançon, B. (2011). *Évaluation des processus de développement de deux très petits organismes d'Haiti en technologie de l'information à l'aide du profil basique de la norme ISO/IEC 29110*. Université du Québec à Montréal.
- Gérançon, B., Nkambou, R., Trudel, S. et Robert, S. (2022). A cognitive model for supporting software functional sizing automation from requirements written as triplets. *The 20th International Conference on Software Engineering Research Practice, Las Vegas, USA, 20*, 30–37.
- Gérançon, B., Trudel, S., Nkambou, R. et Robert, S. (2021). Software functional sizing automation from requirements written as triplets. *International Conference on Software Engineering Advances, Barcelona, Spain, 16*, 23–29.

- Hagège, C. et Roux, C. (2003). Entre syntaxe et sémantique : Normalisation de la sortie de l'analyse syntaxique en vue de l'amélioration de l'extraction d'information à partir de textes. Dans *Actes de la 10ème conférence sur le Traitement Automatique des Langues Naturelles. Articles longs*, 145–154.
- Happel, H.-J., Maalej, W. et Seedorf, S. (2010). Applications of ontologies in collaborative software development. In *Collaborative software engineering* 109–129. Springer.
- Happel, H.-J. et Seedorf, S. (2006). Applications of ontologies in software engineering. Dans *Proc. of Workshop on Semantic Web Enabled Software Engineering"(SWESE) on the ISWC*, 5–9. Citeseer.
- Hevner, A. R., March, S. T., Park, J. et Ram, S. (2004). Design science in information systems research. *MIS quarterly*, 75–105.
- Huijgens, H., Van Deursen, A., Minku, L. L. et Lokan, C. (2017). Effort and cost in software engineering : A comparison of two industrial data sets. Dans *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering*, 51–60.
- Jacobson, I. (2003). Use cases and aspects-working seamlessly together. *Journal of Object Technology*, 2(4), 7–28.
- Jones, C. (2007). *Estimating software costs : Bringing realism to estimating*. McGraw-Hill Education.
- Karim, S., Liawatimena, S., Trisetyarso, A., Abbas, B. S., Suparta, W. *et al.* (2017). Automating functional and structural software size measurement based on xml structure of uml sequence diagram. Dans *2017 IEEE International Conference on Cybernetics and Computational Intelligence (CyberneticsCom)*, 24–28. IEEE.
- Kitchenham, B. A. et Stell, J. G. (1997). The danger of using axioms in software metrics. *IEE Proceedings-Software Engineering*, 144(5), 279–285.
- Laird, L. M. et Brennan, M. C. (2006). *Software measurement and estimation : a practical approach*, volume 2. John Wiley & Sons.
- Lamma, E., Mello, P. et Riguzzi, F. (2004). A system for measuring function points from an er-dfd specification. *The Computer Journal*, 47(3), 358–372.
- Laporte, C. Y. et April, A. (2018). *Software quality assurance*. John Wiley & Sons.
- LeCun, Y., Bengio, Y. et Hinton, G. (2015). Deep learning. *nature*,

521(7553), 436–444.

Lehmann, J. et Voelker, J. (2014). An introduction to ontology learning. *Perspectives on Ontology Learning*, 7–14.

Lévesque, G. (1998). *L'analyse de système orientée-objet et génie logiciel : concept, méthodes et applications*. Éditions de la Chenelière.

Levesque, G., Bevo, V. et Cao, D. T. (2008). Estimating software size with uml models. Dans *Proceedings of the 2008 C3S2E Conference*, 81–87.

Mäder, P., Olivetto, R. et Marcus, A. (2017). Empirical studies in software and systems traceability.

Marín, B., Pastor, O. et Giachetti, G. (2008). Automating the measurement of functional size of conceptual models in an mda environment. Dans *International Conference on Product Focused Software Process Improvement*, 215–229. Springer.

Minsky, M. (2019). *A framework for representing knowledge*. de Gruyter.

Mitchell, T. M. (1997). Does machine learning really work? *AI magazine*, 18(3), 11–11.

Mondary, T., Després, S., Nazarenko, A. et Szulman, S. (2008). Construction d'ontologies à partir de textes : la phase de conceptualisation. Dans *19èmes Journées Francophones d'Ingénierie des Connaissances (IC 2008)*, 87–98.

Muller, P.-A. et Gaertner, N. (2000). *Modélisation objet avec UML*, volume 514. Eyrolles Paris.

Nuseibeh, B. et Easterbrook, S. (2000). Requirements engineering : a roadmap. Dans *Proceedings of the Conference on the Future of Software Engineering*, 35–46.

Quesada-López, C., Martínez, A., Jenkins, M., Salas, L. C. et Gómez, J. C. (2019). Automated functional size measurement : A multiple case study in the industry. Dans *International Conference on Product-Focused Software Process Improvement*, 263–279. Springer.

Roques, P. (2018). *UML 2.5 par la pratique : études de cas et exercices corrigés*. Editions Eyrolles.

Sadoun, D. (2014). *Des spécifications en langage naturel aux spécifications formelles via une ontologie comme modèle pivot*. (Thèse de doctorat). Université Paris Sud-Paris XI.

- Sahab, A. et Trudel, S. (2020). Cosmic functional size automation of java web applications using the spring mvc framework. Dans *IWSM-Mensura*.
- Selami, A., Haoues, M. et Ben-Abdallah, H. (2019). Sizing natural language/user stories/uml use cases for web and mobile applications using cosmic fsm. [Disponible en ligne :<https://cosmic-sizing.org/publications/restosys-case-v1-2/>; consulté en octobre 2021].
- Sellami, A., Hakim, H., Abran, A. et Ben-Abdallah, H. (2015). A measurement method for sizing the structure of uml sequence diagrams. *Information and Software Technology*, 59, 222–232.
- Singh, J. (2017). *Functional software size measurement methodology with effort estimation and performance indication*. John Wiley & Sons.
- Soubra, H. (2011). *Automatisation de la mesure fonctionnelle COSMIC-ISO 19761 des logiciels temps-réel embarqués, en se basant sur leurs spécifications fonctionnelles*. (Thèse de doctorat). École de technologie supérieure.
- Squire, L. R. et Knowlton, B. J. (2000). The medial temporal lobe, the hippocampus, and the memory systems of the brain. *The new cognitive neurosciences*, 2, 756–776.
- St-Pierre, D., Maya, M., Abran, A., Desharnais, J.-M. et Bourque, P. (1997). Full function points : Counting practices manual. *Software Engineering Management Research Laboratory and Software Engineering Laboratory in Applied Metrics*.
- Staron, M. et Meding, W. (2018). Software development measurement programs. *Springer*. https://doi.org/10.1007/978-3-319-91836-5_10, 3281333.
- Taconnat, L. (2012). Fonctionnement et dysfonctionnement de la mémoire humaine. *Le Journal des psychologues*, (4), 18–23.
- Tarhan, A. et Sağ, M. A. (2018). Cosmic solver : A tool for functional sizing of java business applications. *Balkan Journal of Electrical and Computer Engineering*, 6(1), 1–8.
- Tremblay, M.-P. (2017). *Le rôle de la mémoire sémantique dans la reconnaissance des émotions*. (Thèse de doctorat).
- Trudel, S. (2012). *Using the COSMIC functional size measurement method (ISO 19761) as a software requirements improvement mechanism*. (Thèse de doctorat). École de technologie supérieure.

- Trudel, S. et Boisvert, M. (2011). *Choisir l'agilité : Du développement logiciel à la gouvernance*. Dunod.
- Trudel, S. et Lavoie, J. (2007). uobserve software specification. *Montreal, Canada : École de Technologie Supérieure*.
- Tulving, E. (1972). 12. episodic and semantic memory. *Organization of memory/Eds E. Tulving, W. Donaldson, NY : Academic Press*, 381–403.
- Uemura, T., Kusumoto, S. et Inoue, K. (1999). Function point measurement tool for uml design specification. Dans *Proceedings Sixth International Software Metrics Symposium (Cat. No. PR00403)*, 62–69. IEEE.
- Uschold, M. et Jasper, R. (1999). A framework for understanding and classifying ontology applications. Dans *Proceedings of the IJCAI-99 Workshop on Ontologies and Problem-Solving Methods (KRR5), Stockholm, Sweden*, volume 2.
- Wentzlaff, I. (2017). Establishing a requirements baseline by functional size measurement patterns. Dans *REFSQ Workshops*.
- Wieggers, K. et Beatty, J. (2013). *Software requirements*. Pearson Education.
- Wynne, M., Hellesoy, A. et Tooke, S. (2017). *The cucumber book : behaviour-driven development for testers and developers*. Pragmatic Bookshelf.
- Xunmei, G., Guoxin, S. et Hong, Z. (2006). The comparison between fpa and cosmic-ffp. Dans *Proceedings of Software Measurement European Forum (SMEF) Conference*, 113–114. Citeseer.
- Zouaq, A. et Nkambou, R. (2008). Building domain ontologies from text for educational purposes. *IEEE Transactions on learning technologies*, 1(1), 49–62.
- Zouaq, A. et Nkambou, R. (2009). Evaluating the generation of domain ontologies in the knowledge puzzle project. *IEEE Transactions on Knowledge and Data Engineering*, 21(11), 1559–1572.