Towards an Enhancement Effort Estimation Approach using Machine Learning Techniques

Zaineb Sakhrawi

▶ **To cite this version:**

Zaineb Sakhrawi. Towards an Enhancement Effort Estimation Approach using Machine Learning Techniques. Computer Science [cs]. université de sfax, 2022. English. NNT : . tel-04430712

HAL Id: tel-04430712
https://hal.science/tel-04430712

Submitted on 1 Feb 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

TUNISIAN REPUBLIC
Ministry of Higher Education
and Scientific Research

UNIVERSITY OF SFAX

Faculty of Economics and Management of Sfax

DOCTORAL THESIS

For obtaining the title of Doctor in

Computer science

Towards an Enhancement Effort Estimation Approach using Machine Learning Techniques

Presented and publicly supported by:

Zaineb SAKHRAWI

Defended on 24 September 2022 in front of the jury composed by:

Mrs. Ikram AMOUS	**Full Professor - ENET'Com-Sfax**	**Chair**
Mrs. Nadia BOUASSIDA	**Full Professor - ISIM-Sfax**	**Supervisor**
Mr. Ali IDRI	**Full Professor – ENSIAS- Mohammed V-souissi, Rabat**	**Reviewer**
Mr. Hatem HADJ KACEM	**Associate Professor- FSEG-Sfax**	**Reviewer**
Mr. Mounir MARRAKCHI	**Associate Professor- FS-Sfax**	**Examiner**
Mrs. Asma SELLAMI	**Assistant Professor - ISIM-Sfax**	**Co-supervisor**

Academic year: 2021-2022

Dedicace

All praise is to ALLAH the most passionate and the merciful. May his Prophet be blessed with peace and blessings. I am grateful to ALLAH for endowing me with the ability to perform this thesis. I dedicate this work to my mother and father, whose prayers and love never left me while I worked on this thesis away from home. When I encountered difficulties, their prayers and encouragement made my work easier. It would not have been possible for me to pursue higher education in Sfax without the kind support and encouragement of my family. Special thanks to my two adorable sisters « Awatef and Afef » for their assistance and love. Thank you for being sisters who know how to make life more worthwhile. Thanks to my four kind brothers « Abdel Sattar, Hassen, Abdel Raouf, and Abdel Latif » for their moral and financial support and encouragement.

Zaineb SAKHRAWI

Acknowledgement

I want to thank everyone who helped make this thesis a reality. First and foremost, I would like to express my gratitude to my two supervisors, Dr. Nadia Bouassida and Dr. Asma Sellami, for providing me with the opportunity to complete this doctoral research. Thank you for all of your suggestions and discussions that assisted me in completing this work.

Thank you Dr. Nadia Bouassida, for your moral support in overcoming the Ph.D. challenges, and I am grateful for your dedication to assisting me in making the most of my Ph.D. I would also like to thank Dr. Asma Sellami for her advice, friendship, understanding, and financial support as well as for teaching me how to be a successful researcher.

Aside from my supervisors, I'd like to thank everyone on the jury who read and commented on my thesis. I would like to express my gratitude to the members of the thesis examining committee, Mrs. Ikram AMOUS, Mr. Hatem HADJ KACEM, Mr. Ali IDRI and Mr. Mounir MARRAKCHI for the time they spent on reviewing my thesis.

I also want to thank all of the MIRACL team researchers for their assistance and support. Finally, I'd like to express my gratitude to all of my friends, whose company and unwavering support made this journey exciting and enjoyable.

Zaineb SAKHRAWI

Résumé

Vers une approche d'estimation de l'effort des changements dans les exigences logicielles en utilisant les techniques d'apprentissage automatique
Zaineb SAKHRAWI

L'estimation a souvent été considéré comme l'un des défis les plus importants dans la plupart des organisations logicielles. Plusieurs projets s'achèvent en retard, en dehors des budgets, avec moins de fonctionnalités que prévues et sans aucune indication sur leur degré de qualité. Des considérations comme celles de l'utilisation des estimations inexactes influent fortement sur le succès des projets logiciels. En effet, des estimations inexactes suscitent des attentes irréalistes et contribuent à l'insatisfaction du client. Des estimations précises conviennent à des prises de décisions appropriées au moment opportun. D'autre part, des demandes d'améliorations pour ajouter de nouvelles exigences, changer des exigences existantes ou même améliorer l'usage du produit logiciel constitue une source d'erreurs dans ces estimations. Par conséquent, ils peuvent augmenter le coût de développement ou de maintenance de logiciels, perturber le calendrier du projet et même influencer la qualité du produit final. Plusieurs approches avec divers modèles d'estimation sont proposées pour fournir une estimation plus précise de l'effort des projets logiciels. Il existe trois grandes catégories de ces modèles tels que le jugement d'expert, les modèles algorithmiques (exemple, COCOMO II) et les modèles non algorithmiques (tels que les techniques d'apprentissage automatique). Plusieurs chercheurs s'entendent sur l'efficacité de l'utilisation des techniques d'apprentissage automatique comparativement aux autres techniques d'estimations.

Face aux problèmes ci-dessus énumérés, nous avons proposé les contributions suivantes:

— La première contribution consiste à mener une étude d'arrimage systématique de la littérature (SMS) sur l'estimation de l'effort requis pour compléter une amélioration dans les projets logiciels en se basant sur « A Systematic Mapping Study –

SMS in Software Engineering [1] ». La revue a été réalisée en examinant les articles pertinents durant la période de 1995 à 2020 pour déterminer les principaux facteurs utilisés dans l'évaluation des améliorations et l'estimation de l'effort correspondant à l'aide des techniques de ML. L'approche par SMS a pu sélectionner 30 études pertinentes. 19 extraits de revues et 11 actes de conférence via quatre moteurs de recherche (Google Scholar, IEEExplore, ACM Digital library, and ScienceDirect). Cette revue supporte les chercheurs à identifier et à structurer les méthodes utilisées dans le domaine d'estimation de l'effort dans les projets d'améliorations de logiciels. Les résultats de l'étude SMS ont montré qu'il y a très peu d'investigation dans l'estimation de l'effort requis pour implémenter une amélioration dans les projets d'améliorations de logiciels. La plupart des approches proposées utilisent les techniques d'apprentissage automatique.

— La deuxième contribution est la proposition d'une nouvelle approche de prédiction de l'effort d'une amélioration dans les exigences logicielles (SEEE). Cette approche porte sur deux volets. Le premier volet consiste à proposer un nouveau modèle basé sur l'ontologie pour la classification des demandes d'améliorations en des changements fonctionnels et des changements techniques. Cette étude a été conduite en se basant sur des résultats expérimentaux menés sur des projets réels provenant de l'industrie du logiciel et sur le référentiel PROMISE. La classification permet aux gestionnaires et aux intervenants d'être sélectifs dans l'utilisation des méthodes MTF (Mesure de taille fonctionnelle de logiciels). Ainsi, nous avons construit notre ensemble de données en associant chaque demande d'amélioration fonctionnelle avec son effort correspondant en utilisant le jugement d'expert. Le deuxième volet porte sur la prédiction de l'effort d'améliorations des exigences logicielles en utilisant l'ensemble de données construit dans le premier volet. Nous avons choisi quatre méthodes d'apprentissage automatiques pour faire la prédiction: Ada Boost Regression (ABR), Gradient Boosting Regression (GBR), Linear support Vecteur Regression (Linear SVR) et Random Forest Regression (RFR). Les résultats ont montré que le degré de précision de SEEE est meilleur en utilisant l'ontologie avec l'algorithme RFR.

— La troisième contribution consiste à examiner l'utilisation des méthodes MTF du logiciel de IFPUG et COSMIC pour vérifier l'impact de l'utilisation de la taille fonctionnelle d'une amélioration logicielle sur la précision de l'estimation de l'effort requis pour compléter cette amélioration. Cette contribution a abouti à l'efficacité de la deuxième génération de MTF (COSMIC) comparativement à la première génération (IFPUG) pour mesurer la taille des améliorations et son utilisation pour prédire l'effort d'amélioration des exigences et celui de produit logiciel résultant.

— La quatrième contribution consiste à utiliser l'algorithme de sélection des caractéristiques corrélées (CFS) pour sélectionner les attributs (features) les plus pertinents en utilisant le référentiel ISBSG (International Software Benchmarking Standards Group). L'application du CFS a montré qu'il y a une forte corrélation entre la taille et l'effort d'amélioration logicielle. Nous avons utilisé l'algorithme M5P pour prédire l'effort (SEEE). La performance de cet algorithme (M5P) a été comparée par rapport à trois techniques de régression d'apprentissage automatique: Gradient Boosting Regressor (GBRegr), Linear support Vector Regression (LinearSVR) et Random Forest Regression (RFR). Les résultats ont montré que le degré de précision de SEEE est meilleur en utilisant l'algorithme CFS avec l'algorithme M5P.

— La cinquième contribution consiste à proposer une nouvelle approche qui a étudié l'utilisation de la technique « Stacking Ensemble » pour accroitre le degré de précision de SEEE. Le modèle Stacking Ensemble que nous avons construit combine trois modèles de régression: GBRegr, LinearSVR et RFR. Comparativement à l'approche basée sur l'utilisation d'un seul modèle d'apprentissage (M5P), le modèle Stacking Ensemble a donné des résultats plus précis.

— La sixième contribution consiste à développer une application web "ERWebApp" pour obtenir rapidement le SEEE. L'application Web développée est destinée à générer tout d'abord la taille fonctionnelle d'une amélioration, puis à estimer l'effort correspondant à cette amélioration à l'aide du modèle « Stacking Ensemble ».

Mots-clés: Étude d'arrimage systématique (SMS), amélioration ou Changement fonctionnel, Prédiction, Estimation, Effort, Apprentissage Automatique, Agile

Abstract

Towards a Software Enhancement Effort Estimation Approach Using Machine Learning Techniques

Zaineb SAKHRAWI

Estimating has often been seen as one of the biggest challenges in most software organizations. Several projects are ending late, out of budget, with less functionality than expected, and without any indication of their levels of quality. Considerations such as the use of inaccurate estimates strongly influence the success of software projects. This is because inaccurate estimates raise unrealistic expectations and contribute to customer dissatisfaction. Accurate estimates are suitable for making appropriate decisions at the right time. On the other hand, enhancement requests to add new requirements, improve existing requirements or change the usage of software products are a source of errors in these estimates. Therefore, they can increase the cost of software development or Enhancement (maintenance) projects, disrupt the project schedule, and even influence the quality of the final product. Many approaches with various estimation models are proposed to provide a more accurate effort estimation of software development and enhancement projects. There are three main categories of these models such as expert judgment, algorithmic models (*e.g.,* COCOMO II), and non-algorithmic models (such as Machine Learning techniques). Several researchers agree on the effectiveness of the use of ML techniques compared to other estimation techniques.

To resolve those problems listed above, we proposed the following contributions:

— The first contribution consists in conducting a review on estimating the effort required to complete an enhancement in software projects based on "A Systematic Mapping Study – SMS in Software Engineering [1]". The SMS was carried out by surveying relevant papers from 1995 to 2020 to determine the main factors used in

evaluating ER and estimating the corresponding effort using ML techniques. The SMS selects 30 relevant studies. 19 published journals and 11 conference proceedings via four search engines (Google Scholar, IEEExplore, ACM Digital library, and ScienceDirect). This review supports researchers in identifying and structuring methods used in the field of effort estimation in software development and enhancement projects. The results of the SMS showed that there is a very little investigation on estimating the effort required to implement an enhancement in software enhancement projects. Most of the proposed approaches used ML techniques.

— The second contribution consists in proposing a new approach for estimating the effort required to implement an enhancement in software requirements. This approach has two phases. The first phase consists in proposing an Ontology-based Model Classification (OMC) for classifying customer ER as either Functional Change or Technical Change. This study was conducted based on experimental results carried out on real projects from the software industry and on the PROMISE repository. The classification allows managers and stakeholders to be selective in the use of the FSM (Functional size measurement) method. Thus, we built a data set by associating each Enhancement Request (ER) with its corresponding effort using Expert judgment. The second phase deals with the prediction of Software enhancement effort (SEEE) using the dataset built in the first part. Four machine learning methods were selected to make the prediction: Ada Boost Regressor (ABR), Gradient Boosting Regressor (GBR), Linear support Vector Regression (Linear SVR), and Random Forest Regression (RFR). Results showed that the level of accuracy of the SEEE is improved when using the ontology with the RFR algorithm.

— The third contribution consists in investigating the impact of an enhancement functional size through the use of IFPUG and COSMIC FSM methods on the accuracy of the SEEE. This contribution resulted in the effectiveness of the second generation COSMIC FSM method compared to the first generation IFPUG for sizing an enhancement and its use to make an enhancement estimation, and that of the resulting software product.

— The fourth contribution consists in using the Correlated Feature Selection (CFS) algorithm to select the most relevant features using the ISBSG (International Software Benchmarking Standards Group) repository. The application of CFS has shown that there is a strong correlation between size and software enhancement effort. The M5P algorithm was used to provide the SEEE. The performance of this algorithm was compared against three ML regression techniques: Gradient Boosting Regressor (GBRegr), Linear support Vector Regression (LinearSVR), and Random

Forest Regression (RFR). Results showed that the accuracy of SEEE was improved when using the CFS algorithm with the M5P algorithm.

— The fifth contribution consists in proposing a new approach that investigates the use of the “Stacking Ensemble” model to increase the level of accuracy of SEEE. Our constructed Stacking Ensemble model combines three regression models: GBRegr, LinearSVR, and RFR. Compared to the approach based on using a single learning model (M5P), the Stacking Ensemble model gives more accurate results.

— The sixth contribution consists in developing a Web application named "ERWebApp" to quickly make SEEE. The developed Web application is intended to first generate the functional size of an enhancement, then estimate the effort corresponding to this enhancement using the “Stacking Ensemble” model.

Keywords: Systematic Mapping Study (SMS), Enhancement Request (ER), Ontology, Software Enhancement Effort Estimation (SEEE), Machine Learning (ML) techniques, Agile (Scrum)

Glossary

CFP: COSMIC Function Point.

CFS: Correlation based Feature Selection.

COSMIC: Common Software Measurement International Consortium.

COCOMO: COnstructive COst MOdel.

CR: Change Request.

ER: Enhancement Request.

E: Entry.

FC: Functional Change.

TC: Technical Change.

FP: Functional Process.

FPA: Function Point Analysis.

FS: Functional Size.

FS(FC): Functional Size of a Functional Change.

FSM: Functional Size Measurement.

FUR: Functional User Requirements.

IFPUG: International Function Point Users Group.

ISBSG: International Software Benchmarking Standard Group.

ISO: International Organization for Standardization.

KSLOC: Kilo-Source Lines of Code.

LOC: Lines of Code.

ML: Machine Learning.

NFR: Non-Functional Requirements.

PRC: Project Requirements and Constraints.

R: Read.

SE: Software Engineering.

SF: Scale Factors.

SDLC: Software Development Life Cycle.

SMLC: Software Maintenance Life-Cycle.

SLOC: Source Lines of Code.

SEEE: Software Enhancement Effort Estimation.

SR: System Requirement.

SVR: Support Vector Regression.

SP: Software Process.

UML: Unified Modeling Language.

UC: Use Case.

US: User Story.

W: Write.

X: eXit.

Contents

List of Figures

List of Tables

INTRODUCTION

"I feel very strongly that any introduction to a new subject written by a competent person confers a real benefit on the whole science."

Ernest William Hobson

Context

Managing and planning software projects are regarded as one of the challenging problems in software engineering. A software project may include three types of development: new development, enhancement, and redevelopment [2]. Unlike software development which is requirements driven [3], software maintenance is event-driven and it is categorized as correction or enhancement [4]. Although most of the software projects are maintained [5], Software maintenance has not received the same degree of attention that the other phases have [6]. Among the major key issues of software maintenance is the cost estimation [6]. More accurate estimation of software maintenance effort or cost contributes to better management and control for software maintenance [7]. The category of Software maintenance used in our study is enhancement. The enhancement is defined as a change to an existing software product to meet a new requirement [4]. Software maintenance is problematic if the software product is not developed for maintenance [8]. Software estimation (also known as software prediction) is one of the key activities of software development and maintenance project planning. According to a study on Software development effort estimation [9], effort prediction is needed. However, only a few studies investigate the importance of enhancement effort [5]. Accurate estimates reduced the costs of software enhancement projects. However, inaccurate estimates have a significant impact on the

success of software projects by creating unrealistic expectations and contributing to customer dissatisfaction. Even in industries that use modern software development methods such as agile methodology, software estimation is still a critical process. Regarding the listed issues, this thesis looked at how to obtain an accurate estimate of software enhancement effort when using ML techniques in both traditional and agile software projects. It focused also on identifying the impact of the Functional Size (FS) of an Enhancement Request (ER) generated from different standardized Functional Size Measurement (FSM) methods on the effort estimation performance. It proposes an approach for estimating software enhancement effort that is based on the use of the COSMIC FSM method as a primary independent variable and Machine Learning (ML) techniques.

In this introduction, we will discuss the different research problems in Software Enhancement Effort Estimation (SEEE). Then, we list the thesis objectives and contributions. Afterward, in comparison to previous studies, we present the originality of our work. Following that, we provide an overview of the research process methodology. Finally, we outline the thesis's structure.

Research Problem Statement

Customers may now share, post, and tweet their satisfaction or dissatisfaction with any services, and as a result, they have the power to directly influence the software organization's vision. As Bill Gate said, "Your most unhappy customers are your greatest source of learning"[1]. Indeed, the ability to meet all customer requests should never be used to determine the success of a software project. It can only be proven by its ability to meet the customer's requests over time [10]. Because enhancement requests are frequent software project planning should be reviewed regularly. Thus, customers often expressed their feedback or ER in natural language. These requests are most often ambiguous and not well defined. In order to provide an accurate SEEE, an improved ER description is needed. It is important to note that the accuracy of the SEEE model varies according to the nature of the project and the type of software to be developed or maintained. When compared to software development, the cost of software maintenance typically necessitates more time and resources. Many models have been developed to help with estimation [11]. Indeed, for some organizations, approximate estimates are sufficient to determine the project's required value. On the other hand, other organizations believe that accurate

1. https://www.oxfordreference.com/view/10.1093/acref/9780191826719.001.0001/q-oro-ed4-00012282

estimates are the foundation of good project management. Many approaches with various estimation models are proposed to provide more accurate software effort estimation. This mandate requires immediate attention to several Research Questions (RQ):

— RQ1: How should ER be identified and classified?

— RQ2: What is the most suitable FSM method to use for measuring the FS of an enhancement?

— RQ3: What attributes (or features) influence estimate accuracy?

— RQ4: How can effort estimates be improved?

— RQ5: How assessing the performance or the accuracy of a model for estimating software enhancement effort?

— RQ6: How important is the accurate estimate for improving both traditional and Agile software project management?

Objectives and Contributions

In short, to meet the listed RQ, the objectives of this thesis are:

— Identify and classify ER using a semantic model.

— Investigate the effectiveness of using ISO Functional Size Measurement Methods such as IFPUG ISO 20926 (first generation) and COSMIC ISO 19761 (second generation) as independent variables.

— Verify the impact of using FSM methods on the accuracy of SEEE in both traditional and modern (agile) enhancement projects.

— Identify the appropriate ML techniques for generating accurate SEEE.

— Provide automatically SEEE through a web application to help estimators quickly and efficiently make SEEE.

To achieve these objectives, we have targeted the following contributions:

1. Conduct a Systematic Mapping Study (SMS) to identify the pertinent area in SEEE.
2. Develop an ontology-based model to identify and classify customer ER as either Functional Change (FC) or Technical Change (TC).
3. Use the correlation-based feature selection (CFS) to select the relevant variables to our estimate (SEEE).

4. Construct SEEE model using ML techniques separately: M5P algorithm, Ada Boost Regression (ABReg), Gradient Boosting Regression (GBReg), LinearSupport Vector Regression (LinearSVR), and Random Forest Regression (RFR).
5. Build a stacking ensemble model to predict the total enhancement effort in person-hours for a software enhancement project. The constructed Stacking ensemble model combines three different ML techniques (GBRegr, LinearSVR, and RFR). Prediction results using the stacking ensemble model will be compared to the model using only the M5P algorithm.
6. Develop a web application named "ERWebApp" to quickly and efficiently make SEEE. The developed ERWebApp is intended to first generate the enhancement size in CFP units, and then estimate its corresponding effort using the stacking ensemble model.

Originality

Regarding the software maintenance effort prediction approaches proposed in the literature, a small number of researchers have considered the quality of the estimation inputs. This thesis highlights the influence of dataset quality on estimate accuracy. Choosing the appropriate model input with a good understanding of enhancement requests is necessary for an accurate prediction of the effort required for their implementation. Besides, classifying enhancement requests will provide an understanding of both the project level (effect on the project) and the requirements level (effect on other requirements). This thesis aims to improve estimation models inputs by:

— Exploring the use of Ontology for semantically identifying and classifying ER. Ontologies have long been used in the life sciences to formally represent and reason about domain knowledge. It has recently become more popular as a source of background knowledge in similarity-based analysis and ML techniques.

— Identifying the relationship between the independent variables (e.g, Enhancement FS) and the dependent variable (the enhancement effort) using the Feature Selection methods, namely the CFS algorithm.

When improving the accuracy of software effort estimation models, research papers investigated the use of ML techniques. Compared to the existing estimation models that are based on algorithmic and non-algorithmic, ML techniques have been successfully used

as effective models for an accurate estimation of software effort. In the field of SM, we discovered that only single models were used to predict maintenance effort. In this context, this thesis aims to improve the estimation model output by constructing a stacking ensemble model. The built stacking ensemble model combines three different ML techniques (GBRegr, LinearSVR, and RFR). Predictions made with the stacking ensemble model will be compared to those made with different single algorithms separately.

Since agile methodology encourages changes, several estimation techniques have been proposed. In practice, story point-based estimations are used. Even in the scrum context, some ER are trivial while others pose serious threats to the software project's success. Consequently, software organizations need rigorous and structured approaches to evaluate ER. In this context, this thesis aims to improve the estimation model by investigating:

— The effectiveness of COSMIC methods for describing ER and sizing User Story (US) in CFP units within the Scrum context.

— The use of FS (generated by COSMIC) of ER (expressed in the form of US) as a primary independent variable for predicting enhancement effort for a scrum software project.

Consequently, this thesis proposes a new Enhancement effort estimation approach that can be applied in traditional and agile projects as well.

Research Design Methodology

Figure 1 presents our research process, where:

— **Phase 1:** Reviews relevant papers in the field of SEEE using the SMS Methodology. Phase 1 is detailed in Chapter 2. The results are published in [12].

— **Phase 2:** First, proposes an Ontology-based Classification Model (OCM) that distinguishes two types of ER associated with software enhancement projects which are referred to as functional Change (FC) and technical Change (TC). The results are reported in [13]. Second, it uses the proposed OCM and the Expert Judgment estimation approach in order to construct a semantic dataset. This dataset is used as input for constructing a ML-based SEEE model. The results are published in [14]. Phase 2 is detailed in Chapter 3.

— **Phase 3:** Investigates the applicability of the CFS algorithm for making a comparison between the first and second FSM generations. IFPUG and COSMIC methods

are used for measuring the size of functional changes. Therefore the measurement results are used as input for predicting SEE. The findings are published in [15]. Phase 3 is detailed in Chapter 4.

— **Phase 4:** Investigates the applicability of the COSMIC FSM method for measuring the size of functional changes within the scrum context and determining their impact on enhancement effort prediction. The findings are published in[16]. Phase 4 is detailed in Chapter 4.

— **Phase 5:** Improves the input (International Software Benchmarking Standards Group (ISBSG) dataset) of the SEEE where a set of selected ML techniques are trained on a dataset with relevant features using the CFS algorithm. Then, the use of the stacking ensemble model for estimating the enhancement effort of software projects is evaluated. The stacking ensemble model is used to improve prediction accuracy over individual models. As using this model manually is time-consuming, we developed an ERWebApp to quickly and efficiently make SEEE. The results of this phase are published in [17] and [18]. Phase 5 is detailed in Chapter 5.

Thesis Structure

This thesis is divided into five chapters, where:

— **Chapter 1** introduces the general concepts of effort estimation for software maintenance projects and software size measurement.

— **Chapter 2** presents an SMS on the use of ML techniques for SEEE. The SMS was carried out by surveying relevant papers from 1995 to 2020. We followed well-known procedures.

— **Chapter 3** provides an OCM for classifying ER as FC or TC. The proposed approach is applied to real-world projects from the software industry (six software development project datasets including functional requirements requests and the PROMISE repository including ER). We constructed our dataset by associating each ER classified as FC with its corresponding effort using Expert Judgement. The constructed dataset is used as input for various ML techniques for SEEE.

— **Chapter 4** Investigates the use of the first and second FSM generations (*i.e.,* IFPUG and COSMIC FSM methods for respectively sizing ER). And also the use of the CFS algorithm. Firstly, the obtained FS will be used as input for estimating

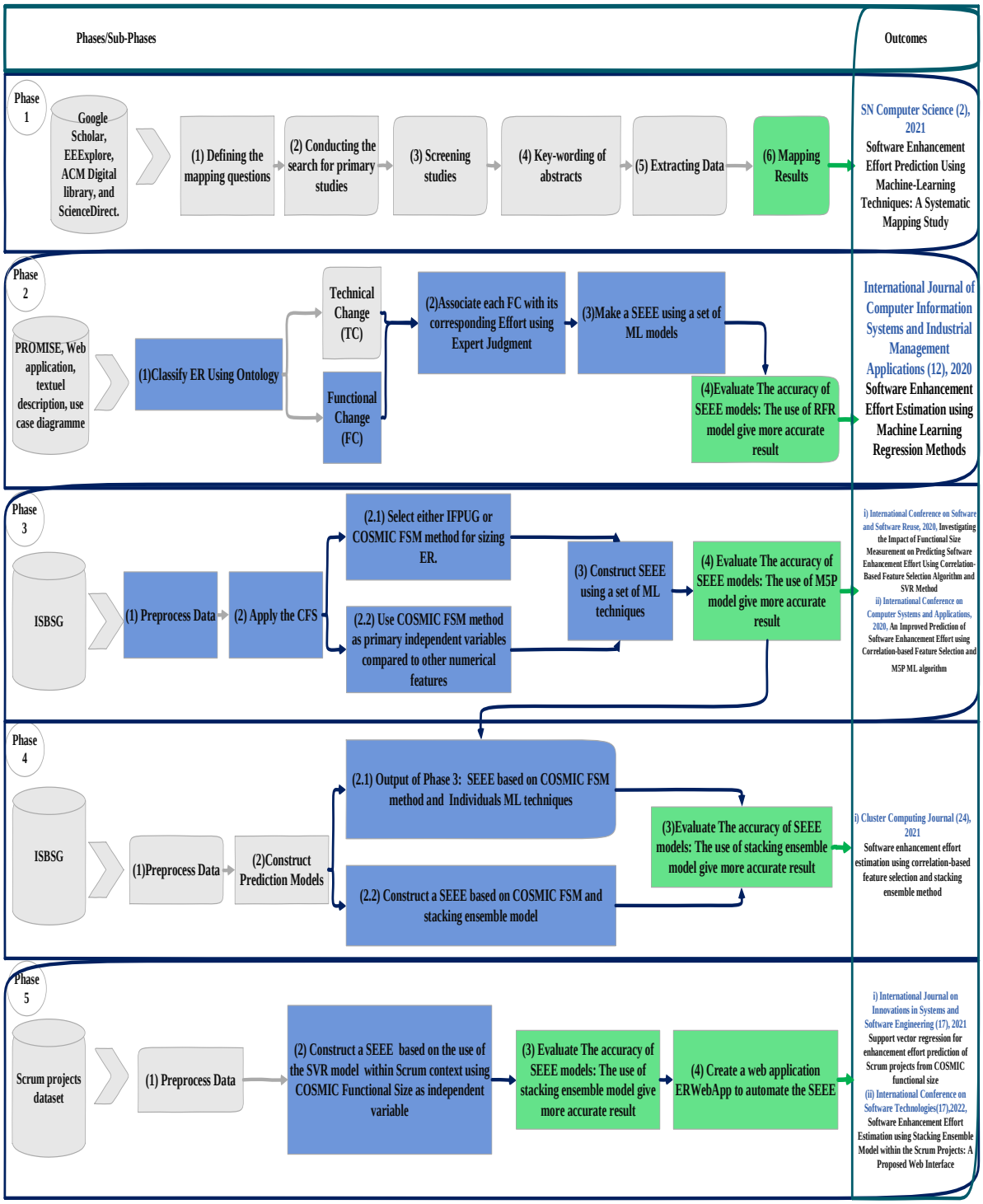

Figure 1 – Research Work Process

software effort in traditional enhancement projects. Following that, a comparison of IFPUG with COSMIC is conducted to provide which one is more effective. Second, in this chapter, we also investigate the use of the COSMIC FSM method as an independent variable for software estimation using the SVR enhancement estimation model within the Scrum context.

— **Chapter 5** Investigates the use of the stacking ensemble model for estimating the enhancement effort (EME) of software projects. The goal is to improve the accuracy of the estimation model over the individual models' ones as described in Chapter 4. We compare two ML-based approaches for predicting SEE: the M5P (as an individual model) and the stacking ensemble model as an ensemble method combining different regression models (GBRegr, LinearSVR, and RFR) on the ISBSG dataset. Afterward, we develop the ERWebApp. The proposed ERWebApp is designed to first generate the enhancement FS and then estimate its corresponding effort.

— Finally, we summarize our work, highlight the restrictions and outline some of its prospective extensions.

Table 1.1 summarizes the research problem tackled in this thesis.

Table 1 – Summary of the research problem

Motivation	Enable stakeholders in a software company (project management, development team, scrum master, etc.) to be selective (1) in their use of the appropriate measurement method (COSMIC or IFPUG) regarding the customers' ER and (2) using the suitable effort estimation model that improves the accuracy of the results (or estimates). This is useful for the software company to not only evaluate but also make appropriate decisions regarding the effort required to implement an ER.
Object/Inputs	ER classified through an ontology model, FS measured through COSMIC FSM method, ML techniques
Purpose	Explore the applicability of (1) the COSMIC FSM method for sizing ER (2) identify the impact of using enhancement size on the estimated effort that is required to implement this ER in both traditional and agile software projects (3) use of ML techniques to provide more accurate estimated effort (in the traditional and agile project)
Perspective	The software company roles include customer, project manager, analysis and design team, implementation and validation team, etc.
Domain	Requirements enhancement management, requirements engineering, size measurement, effort estimation and ML techniques
Scope	PROMISE dataset, ISBSG dataset, web application and Scrum enhancement projects dataset

Chapter 1

Background: The nature of Software Maintenance, Measurement and Estimating

Contents

In Short

For most software organizations, it is a challenge to produce high-quality software product that meets user requirements and customer expectations within a specified time, scope, and cost. Indeed, software organizations' survival depends not only on rapid development but also on the adaptability to the enhancement requests. This thesis addresses the problem of enhancement requirements management including enhancement identification, presentation, measurement, and effort estimation. We start this chapter by describing three basic concepts which are: software maintenance, software measurement, and software project estimation.

1.1 Introduction

In software maintenance activity, responding to the user ER is a critical task. The more ER are clear, precise, and well-defined, the better software developers/ maintainers will understand the functionality to be enhanced. Oppositely, unclear, imprecise, and inaccurate ER forces software maintainers to ask for more clarifications. Consequently, this may slow down the software maintenance progress. Identifying ER is integrated into the requirements management activity. According to various research studies, the main cause of project failure is the inability to manage changes to requirements or ER [19]. As a result, managing requirements is important for understanding the impact of ER on the enhancement effort and the project progress.

Requirements management is one of the four main software requirements process activities [20], which are requirements elicitation, requirements analysis, requirements acceptance, and requirements management. The requirements elicitation activity concerns the identification of the users' needs and customers' expectations. During the requirements analysis phase, more details about the users' needs and the customer's expectations are provided. Both activities (requirements elicitation/analysis) require negotiation with the customer and users. The requirements acceptance activity concerns the requirements verification. By the end of this phase, the requirements baseline is provided. Thereafter, when an ER is proposed, a requirement management activity is required to analyze the enhancement impact on the requirements baseline. Hence, some modifications can be made. ER can be expensive in terms of the common predictable variables; that is, effort, time, and size. The effort represents the number of hours needed to develop or maintain a software product in terms of person-months [21]. Time also named software project schedule, duration, etc [2]. Afterward, the software size is expressed in terms of lines of code or Function points [22]. To make changes and deliver the enhanced software product with respect to the estimated deadlines and budget, it is required to establish an effective evaluation of every single ER. To address this issue, in this chapter, we will investigate the importance of using formal representation tools (such as ontology) to provide a clear understanding of the enhancement purpose and assess its impact on the software system as well as the development and maintenance progress.

In the same context, we propose to improve the ER description (*i.e.,* a detailed description of the ER that facilitates their sizing). Measuring the Functional Size (FSM) of a given requirement or an enhancement is one of the most important factors influ-

encing the effort estimation associated with this requirement implementation. Thus, the more accurate the measurement results, the fewer errors are detected, and better decisions are made. Inadequate measurements, for example, can lead to overestimating or underestimating.

Several FSM methods have been proposed since they can be used to estimate development/maintenance (enhancement) efforts. There is only one second-generation FSM method, the COSMIC, and four first-generation FSM methods, including the IFPUG. Measurement methods that have been approved by the international community are required to provide accurate measurements and avoid errors. Several researchers observed that the size variables are closely related to the required effort [23]. In particular, functional sizing can be used to improve organizational performance, estimate the effort/cost of new development, estimate the effort for enhancement, and control software development, among other things [24, 25]. Nevertheless, the most difficult challenge for estimators and other stakeholders is selecting how functional size is to be derived. In other words, how determining the efficiency of an FSM method and its use in the SEEE model to generate accurate estimates.

With a clear description of an ER, an appropriate software measurement, and an accurate SEEE model, a high level of quality of software product can be achieved (*i.e.,* satisfying customers' needs within a specified time, effort, and cost). This chapter provides a quick overview of software maintenance activity, the Software Measurement methods the most used, and the most common software effort estimation approaches.

1.2 Software Maintenance

Face a highly competitive market, software organizations must have the ability to maintain software to meet the customer's requests. Software maintenance is regarded as an essential component of software development, but it has not received the same attention as other Software Development Life Cycle (SDLC) fields [5].

1.2.1 Definition

Software maintenance is defined as « the totality of activities required to provide cost-effective support to a software product » (ISO/IEC 14764) [4]. Software maintenance activities are identified [11] as: « process implementation; problem and change analysis;

change implementation; maintenance review/acceptance; migration; and retirement. » Software maintenance is the field of Software Engineering which have been ignored over the last period. It has not received the same degree of attention that the other phases have [6]. Nevertheless, it is the most crucial field in software life [26]. In most software organizations, software development is far more important than software maintenance. Software maintenance begins when software development begins and continues until the software system enters the retirement process, as shown in Figure 1.1. Software maintenance is the general process of changing a system after it has been delivered. Typically, the term "Software maintenance" refers to custom software in which separate development teams are involved before and after delivery. Changes to the software may be simple (*e.g.,* correcting coding errors) complex (*e.g.,* correcting design errors), or significant (*e.g.,* correcting specification errors) or accommodating new requirements. Changes are made by altering existing system components and, if necessary, adding new ones. The ISO 14764 standard defines enhancement maintenance as "a change to an existing software product to meet a new requirement." According to ISO 14764, software maintenance is categorized into four types: corrective, adaptive, preventive, and perfective [6]. ISO 14764 classifies adaptive and perfective maintenance as enhancements and corrective and preventive maintenance as the correction. Adaptive maintenance is the modification of a software product performed after delivery to keep a software product usable in a changed or changing environment[4]. Perfective maintenance is defined as "modifying a software product after delivery to detect and correct latent faults in the software product before they manifest as failures"[4]. The same document defines corrective maintenance

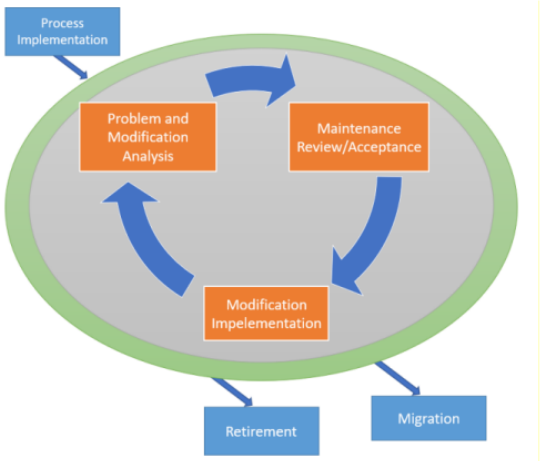

Figure 1.1 – Software Maintenance Process [26]

as "reactive modification of a software product performed after delivery to correct discovered problems." Preventive maintenance is defined as "modifying a software product after delivery to detect and correct latent faults in the software product before they become operational faults."

1.2.2 Software Maintenance Challenge

Many issues that are related to software maintenance can be traced to the requested changes [11]. It is difficult to predict how the system will respond if the software code is changed for maintenance purposes, which can lead to overestimation. ER are welcome when they provide a means for software improvement [19]. But, in some cases, they are large in scope and create some challenges for maintainers. software maintenance is difficult if the software product was not designed for it. One problem encountered in the software development progress (including the maintenance phase) is how to provide an accurate estimate? Hence, distinguishing FC from TC is a necessity. We believe that the SEEE must take into account the (i) type of ER (*i.e.,* either a FC or a TC. Another reason could be that the organization is not concerned with perfection, but rather with getting the system up and running as soon as possible [26]. Once the ER is identified, the software project planner/manager must estimate: (i) how long will take the maintenance phase and (ii) the required effort for each ER. This information is needed for the schedule planning and resources (hardware and software) distribution. In the next sections, we describe the estimations approaches the most used in practice. However, despite their popularity, they have been widely criticized. For instance, Pressman considers that estimation models used many variables (human, technical, environment, etc.) which can affect the estimation results [27]. Whereas, the project managers prefer accurate estimates in terms of effort, time, and cost. In addition, inaccurate measures of the software size lead to false effort estimation and consequently late delivery. For this reason, instead of focusing only on how to determine the effort estimation, we will also focus on the ER descriptions and their corresponding sizing.

1.3 Software Measurement

Measuring is no longer restricted to a single area. It is critical in science, engineering, and even everyday life. Measurement is used in all aspects of human activity, including

social, medical, industrial, and academic activities. Measurement has been popular in engineering fields since the 18Th century. It has been investigated in the Software Engineering (SE) literature to discover and mitigate various issues in software businesses, such as missed time-to-market deadlines, negative cost trends, omissions in client needs, and so on. As noted in [28], measuring software provide managers with valuable information for improving communication, tracking specific project objectives, identifying and correcting problems early, making crucial trade-off decisions, etc. Furthermore, practitioners must understand the significance of software measurements. This requirement is reflected in the IEEE definition of SE: "The application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of software; that is, the application of engineering to software" [6]. In addition, software measurement enables project, product, and process management. DeMarco [29] depicted the following: "You cannot control what you cannot measure". As a result, measuring has emerged as a critical phase in the field of SE.

Two key reasons for measuring software size were summarized by Symons [30]:

— Measurement is fundamental to improving project management of software projects. Software size helps managers to regulate both new software creation and the maintenance or upgrade of current software. Furthermore, the size of the program allows managers to track performance metrics (project productivity, project speed, etc.).

— Measurement is used to estimate the time and effort needed to complete a software project. In this situation, measuring the program size early in the SLC is critical (requirements phase).

In the SE, software measurement is a crucial process. The interpretation and analysis of measurement results help managers evaluate the software project's progress and quality. As a result, a great variety of measures, such as software size measurement methods have been proposed over the last forty years.

1.3.1 Software Size

Software, like any other concept, can be measured in terms of its size. Even though size measures do not convey external factors such as "coding difficulty," they are extremely valuable. FSM methods measure the software size by quantifying the Functional User Requirements (FUR). User requirements are classified into three categories: Functional

User Requirements (FUR), Non-Functional Requirements (NFR) and Project Requirements and Constraints (PRC) [31]. Where:

— FUR express "what the software is expected to do in terms of tasks and services." [31].

— NFR includes "any requirement for a hardware/software system or for a software product, including how it should be designed and maintained, as well as how it should perform in operation" [31].

— PRC describes "how a software system project should be managed and resourced, as well as the constraints that affect its performance" [31].

Abran differentiated between two types of software size: technical (*e.g.,* length measures) and functional (*e.g.,* functionality measures) [32]:

— The developer's perspective is used to determine the technical size. It is based on the number of elements such as the number of LOC, modules, components, etc.

— The user's view is used to determine the functional size. It is measured in terms of software functionality, regardless of any technological constraints or implementation decisions.

1.3.2 Source Lines of Code (SLOC)

The amount of source instructions in the program to be produced is counted to determine the software size in terms of SLOC. The comments and header lines are not included in the count [33]. Managers break down the problem into modules to estimate the number of Lines of Code (LOC) at the start of the SLC. Where each module can be subdivided into a set of sub-modules, and so on, until the sizes of the various entities (modules, submodules, etc.) can be approximated [33].

LOC counting like SLOC (Source LOC) and TLOC (Total LOC) is straightforward and easy to count. Calculating LOC is quite popular since it is provided most simply when compared to the available sizing. Furthermore, measurement results expressed in terms of LOC can be used as an input to the vast majority of estimating models and approaches. Despite their widespread use, the use of LOC can cause several difficulties. Counting the number of LOCs in software is akin to counting the bricks in a structure, although a building's area is typically represented in terms of the number and size of rooms [33]. Furthermore, LOC is related to the programming language used.

1.3.3 Functional Size Measurement Methods

FSM methods have been developed to overcome the limitations of LOC measures. Many standards have been developed over the years by various organizations. The sections that follow provide a brief history as well as a detailed comparison of the most commonly used FSM methods. FSM methods quantify the Functional User Requirements to determine the size of the software. FURs express *"what the software shall do in terms of tasks and services"* [31]. User needs are represented through a series of artifacts, releases, or documentation across the SLC.

Several FSM methods have been proposed because they are useful when estimating development/enhancement efforts. However, the most difficult challenge for managers and other stakeholders in determining the effectiveness of an FSM method and selecting an accurate SEEE model.

1.3.3.1 IFPUG FSM Method

IFPUG FSM methods were founded to foster and promote the evolution of the Function Point method [24]. Since then, the method has been renamed IFPUG Function Point Analysis (or simply IFPUG) and has been standardized as ISO/IEC 20926:2009. The IFPUG sizing method is used to assess the functionality impacted by software development, and maintenance, regardless of the technology used for implementation.

The size of an application is determined by IFPUG based on its Functional User Requirements (or by other software artifacts that can be abstracted in terms of FURs). Each FUR is functionally decomposed into Base Functional Components (BFC), and each BFC is classified into one of five Data or Transactional BFC Types to identify the software's set of features [24].

1.3.3.2 COSMIC FSM Method

The COSMIC FSM method was proposed in 1999 to correct some of the structural deficiencies of the first-generation FSM methods and overcome a number of their limitations. It is widely used all over the world for a variety of purposes, the most common ones are:

— measure and compare projects performance with similar characteristics as the software is being measured.

— estimates the project effort/cost in terms of hours.

— drive decisions on the development project activities.

The COSMIC method was designed to be independent of any implementation decisions embedded in the operational artifacts of the software to be measured. Each data movement is measured as 1 COSMIC Function Point (CFP). The COSMIC measurement process [25] includes three phases: Measurement strategy phase, Mapping phase, and Measurement phase.

— The **Measurement Strategy Phase**: Before starting the measurement, it is required to identify a set of parameters to ensure a correct interpretation of the measurement results. These main parameters are measurement purpose, scope, level of decomposition, functional users, and level of granularity. For instance, the Purpose identifies why the measurement is needed and what the measurement results will be used for. For example, measurement purpose can be *"to estimate the effort of implementing a change to manage the project scope creep"* [34]. The Scope determines the set of functionality to be measured. For example, the measurement scope can be *"all the changes required for a new release of a piece of existing software"* [34].

— The **Mapping Phase**: At this phase, the FUR are mapped to the COSMIC "Generic Software Model" components.

— The **Measurement Phase**: At this phase, one CFP is attributed for each identified data movement. The Functional Size of a Functional Process (noted by FS(FP)) is given by Equation 1.1:

$$FS(FP_i) = \Sigma size(E_i) + \Sigma size(X_i) + \Sigma size(R_i) + \Sigma size(W_i) \tag{1.1}$$

Where:

— $FS(FP_i)$: the functional size of the functional process FP_i.

— $\Sigma size\ (E_i)$: the functional size of entries in FP_i.

— $\Sigma size\ (X_i)$: the functional size of exits in FP_i.

— $\Sigma size\ (R_i)$: the functional size of reads in FP_i.

— $\Sigma size\ (W_i)$: the functional size of writes in FP_i.

The functional size of software is obtained by performing an arithmetic addition of the functional sizes of its functional processes, as given by Equation 1.2:

$$FS(SW) = \sum_{i=1}^{n} FS(FP_i) \tag{1.2}$$

Where:

— FS(SW) is the functional size of the software.

— FS(FPi) is the functional size of the functional process FPi.

— n is the number of the functional processes identified in the software.

1.3.4 Sizing software enhancement using COSMIC FSM Method

Compared to other FSM methods, COSMIC is designed to objectively measure the functional size of a change to software as well as the size of the software that is added, modified, or deleted [34]. Results showed that the COSMIC method has been successfully used in the software industry. It contributes to exploring ways of increasing productivity, monitoring software progress, and performance specifications, while other measures have been tried and were found to be lacking [34]. COSMIC FSM is used by many organizations for agile projects due to its objectivity and ability to be used at all levels of aggregation [34]. Before determining the effort required to implement the US, the functional size of the US can be measured using the COSMIC method [34].

An ER that affects the FUR is called an FC. An FC may propose the addition of new functionality or the deletion/modification of existing functionality. Measuring the functional size of an FC (noted by FS(FC)) is needed to estimate the required cost/effort to handle the change. COSMIC can be used to measure the size of an FC to software in terms of CFP units and the size of the software that is added, changed, or deleted as well [30]. COSMIC defines a FE as *"any combination of additions of new data movements or modifications or deletions of existing data movements"* [34]. The FS(FC) is given by the aggregation of the sizes of all the added, deleted, and modified data movements (see Equation 1.3). The FS(FC) has at least a value of 1 CFP with no upper limits.

$$FS(FC) = \Sigma FS(added data movements) + \Sigma FS(deleted data movements) + \Sigma FS(modified data movements) \tag{1.3}$$

Where:

— $FS(FC)$ is the functional size of the functional change.

— $\Sigma FS(addeddatamovements)$ is the functional size of the added data movements.

— $\Sigma FS(deleteddatamovements)$ is the functional size of the deleted data movements.

— $\Sigma FS(modifieddatamovements)$ is the functional size of the modified data movements.

The software's functional size after changing its functionality is measured as given by Equation 1.4. Taking into account the original size, **plus** the functional size of all the added data movements, **minus** the functional size of all the removed data movements [34]. Modifying data movements has no influence on the software functional size measured after the FC since the modified data movements exist before and after the FC.

$$FS(SW)_{final} = FS(SW)_{initial} - \Sigma FS(deleteddatamovements) + \Sigma FS(addeddatamovements) \tag{1.4}$$

Where:

— $FS(SW)_{final}$ is the functional size of the software after the FE.

— $FS(SW)_{initial}$ is the functional size of the software before the FE.

— $\Sigma FS(addeddatamovements)$ is the functional size of the added data movements.

— $\Sigma FS(deleteddatamovements)$ is the functional size of the deleted data movements.

To illustrate how this works, let's examine the example of a software with three functional processes (FP1, FP2 and FP3), where: FS(FP1) = 10 CFP, FS(FP2) = 11 CFP, FS(FP3) = 9 CFP and
$FS(SW)_{initial}$ = 10 CFP + 11 CFP + 9 CFP = 30 CFP.

We assume that a FE request proposes to:

— add one new functional process (FP4), where the FS(FP4) has value of 6 CFP;

— add one data movement to FP2;

— modify three data movements in FP3; and

— delete two data movements in FP1.

Hence, the total size of the FE is the sum of: 6 + 1 + 3 + 2 = 12 CFP. The software size after the change is equal to: $FS(SW)_{final}$ = $FS(SW)_{initial}$ + 6 + 1 - 2 = 35 CFP. The sizes of the functional processes after the FE are as follow: FS(FP1) = 10 - 2 = 8 CFP, FS(FP2) = 11 + 1 = 12 CFP, FS(FP3) = 9 CFP and finally FS(FP4) = 6 CFP.

1.4 Software Project Estimation Models

Based on the functional size of the software, estimators can provide estimates in terms of effort, duration, and cost required to implement the software. In practice, after identifying the user requirements, it is possible to generate the software functional size using an FSM method. However, during the software development process, software size may change when ER occurs. The impact on the software development progress may put off the delivery of the final product. For this reason, researchers proposed to estimate the effort required to handle an ER (or Functional change) to (i) evaluate if the available resources (human and material) will be enough to implement the change or the original budget needs to be maintained, and (ii) determine if the risks associated to the change is increased or reduced.

Like the traditional project, agile projects also use an estimation-based approach to predict the software maintenance effort in person-hours or to determine the actual working hours required to complete development/maintenance tasks. As depicted in Figure 1.2, there are three popular approaches or models: algorithmic, non-algorithmic, and ML techniques that have been widely used to predict maintenance effort [35].

Effort Estimation	
1- Algorithmic Model	COCOMO-II
	SLIM ...
2- Non-Algorithmic Model	Expert judgment (traditional project)
	Planning poker (agile project)
3- Machine Learning models	Supervised learning Algorithms
	Unsupervised learning Algorithms
	Reinforcement Learning Algorithms

Figure 1.2 – Classification of software effort estimation models

1.4.1 Algorithmic model

Algorithmic models are used for statistical and mathematical formulations [36]. There are several different algorithmic techniques in the software engineering literature. Many

are known as regression analysis techniques. For example, COCOMO-II (COnstructive COst MOdel), Putnam Software LIfe cycle Management (SLIM), SEER-SEM, and True Planning. The primary input to these models is software size. Typically, the size is measured in terms of function points, source lines of code (SLOC), or use case points (UCP).

1.4.2 Non-algorithmic model

Non-algorithmic models are based on analytical assessments and interpretations for estimating software effort [36]. These models analyze historical data from previously completed projects. The expert judgment is also referred to as the expert opinion-based process [31].

Several studies showed that the expert judgment approach is one of the most common estimation techniques used in software project estimation [36]. Because of its simplicity and flexibility, software development teams prefer to use this technique over formal estimation models [36]. However, there is no evidence in the consulted literature that the results produced by this approach are completely accurate.

1.4.3 ML techniques for Regression problem

ML techniques are alternatives to algorithmic models [37]. ML is a subset of artificial intelligence that focuses on the creation of models that can be trained on some data and then used to predict new data in the future. An ML approach can be divided into three parts: the Data Preprocessing phase, Prediction models phase, and Decision-making phase.

1.4.3.1 Data Preprocessing phase

The data preprocessing phase in the knowledge discovery process is guided through the data transformation activities from raw data to training and test data [38]. This complex and multidisciplinary phase involves concepts and structured knowledge in distinct and specific ways in the literature and specialized tools, necessitating the services of data scientists with appropriate expertise. Datasets including customers' ERs must be managed consistently to reduce the risk exposure and guarantee the software development progress. The requirements process spans the whole software maintenance life cycle. Change management and maintaining requirements in a state that accurately mirrors the

software to be built, or that has already been built, are critical to the success of the software engineering process [6]. Requirements Change Management (RCM) is concerned with making rational decisions about whether or not to implement a requested change. It is also concerned with justifying the decision for determining which information (such as documents and other requirements) will be impacted by the proposed change. Even in the best of circumstances, change management is difficult to execute, and it becomes even more difficult when executed globally due to the nature of distributed development projects and the diversity of stakeholders [20]. In this thesis, we will make focus on semantically describing customer ER to facilitate the RCM process by proposing an Ontology-based model. Ontology can be considered a useful data preprocessing technique in ML are [39]. Recently, there has been a lot of interest in the design of ontologies for data mining, resulting in a plethora of ontologies for various purposes. Instead of focusing on the use of ontology, we will focus on another popular data preprocessing technique named the CFS algorithm. According to Rashwan, ontology is a conceptual modeling tool that describes information systems at the semantic and knowledge levels [40]. The goal is to capture knowledge in related fields, identify commonly recognized terms in this field, describe the semantics of concepts through conceptual relationships, and provide a common understanding of field knowledge. Recently, there has been a lot of interest in the design of ontologies for data mining, resulting in a plethora of ontologies for various purposes. Ontologies [40] is a well-established tool for modeling knowledge in various domains, and as such, they can play an important role in modeling the various steps of a data mining application and supporting application design.

On the other hand, the goal of feature selection is to find the best feature in the data set [41]. Data can be classified using ML techniques into a set of class features and targets. ML techniques can classify data into a set of class features and targets. ML or pattern recognition applications have variable or feature domains containing tens to hundreds of variables or features. Several techniques have been developed to address the issue of removing irrelevant and excessive variables. Feature selection (variable elimination) aids in data comprehension, reduces computing requirements, reduces dimensional curse effects, and improves performance. Filters, wrappers, and hybrid algorithms are the three types of feature selection algorithms [41].

— Filter methods select features based on the characteristics of the dataset without the use of any learning techniques. Following that, this subset of features is fed into

a classification/prediction algorithm as input.

— Wrapper methods choose feature subsets based on the performance of a given learning technique, as measured by a performance metric.

— Embedded or hybrid methods combine filter and wrapper techniques to perform the selection and model building steps simultaneously. Dependency measures are one of the measures used in feature selection. There have been numerous proposals for dependency-based algorithms.

In this thesis, we will use the main measure called CFS. CFS is selected since it evaluates all the possible combinations [42]. It can update the subset of selected features during the evaluation process. In contrast to greedy forward selection and greedy backward elimination, they do not update the subset of features during the evaluation process [41]. CFS uses correlation to evaluate a feature subset derived from the Pearson correlation coefficient [42]. This method is a multivariate feature Filter, which means it evaluates various feature subsets and selects the best one. Hall proposed the concept of CFS which evaluates feature subsets using a heuristic evaluation function [42]. This thesis is based on the hypothesis "A good feature subset contains features highly correlated with the class, yet uncorrelated with each other" [42]. The choice of feature selection methods differs among various application areas [42, 41].

1.4.3.2 Prediction models phase

There are four types of ML algorithms: supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning. No method is thought to be superior to the others, and their strengths and weaknesses are frequently complementary. As a result, in our context, we used various experiment learning methods to determine which one was best suited to our situation. Thus, we present in this section a number of ML techniques to be used in this thesis. Each technique will be explained in detail next.

A. Supervised Learning Algorithm

Supervised learning is a labeling learning technique [38]. Supervised learning algorithms are given a labeled training dataset (*i.e.,* inputs and known outputs) to build the system model representing the learned relationship between the input and output. When a new input is fed into the system after training, the trained model can be used to obtain the expected output [38]. Regression analysis falls under supervised ML, especially where the

constructed model tries to predict a value for an input based on previous information. The selected supervised learning algorithms used in this thesis are detailed as follows:

(a) Random Forest Regression (RFR)

Breiman created the RFR at first [43]. RFR is an improved classification and regression tree method that has gained popularity due to its robustness and flexibility inappropriately modeling the input-output functional relationship. A random forest is made up of numerous decision trees. To reduce overfitting and improve accuracy, the random forest method constructs each decision tree by randomly selecting a subset of the feature space. The random forest method is used to classify a new data sample in three steps: (a) Assign a data sample to each tree in the forest. (b) Each tree provides a classification result, which serves as its "vote." (c) The data sample will be classified into the class with the highest number of votes.

(b) Linear Support Vector Regression (LinearSVR)

Vapnik first introduced the SVR in 1995 [38]. To use the Support Vector algorithm for regression estimation, an analog of the margin is constructed in the space of the target values using Vapnik's insensitive loss function [44]. To generalize the Support Vector algorithm to regression estimation, an analog of the margin is constructed in the space of the target values using Vapnik's insensitive loss function [44]. Variables in the SVR model structure belong to continuous space. The kernel function chosen is determined by the training dataset. If the dataset is linearly separable, the linear kernel function performs well. If the dataset cannot be separated linearly, polynomial and RBF kernel functions are commonly used [44].

(c) Ada Boost Regression (ABReg)

Freund Schapire were the first researchers who proposed the AdaBoost algorithm [45]. This algorithm solved many of the problems encountered by previous boosting algorithms [46]. The idea behind AdaBoost is to build a strong model by combining multiple weak classifiers into a single strong classifier. A weak classifier is a classifier that performs poorly but outperforms random guessing [47].

(d) Gradient Boosting Regressor (GBReg)

GBR is an ML technique that is widely used today. The advantage of using GBR is that it can solve almost all objective functions. It is also effective in many cases, and allows for flexibility in the selection of loss functions [9].

(e) M5P algorithm

M5P is a powerful implementation of Quinlan's M5 algorithm for inducing both Models Trees and Regression Trees [48]. It is an extended work based on the M5 algorithm [48]. M5 algorithm was originally developed by Quinlan and used in data mining which combines the decision tree and multilinear regression. Decision trees are used to classify input and output. The M5 tree development has three steps: tree construction, tree pruning, and smoothing.

M5P is a modified version of the M5 tree algorithm. It is designed to handle enumerated attributes and missing values. M5P is used in categorical and continuous variables and missing values. It is based on surrogate splitting to deal with missing values. After the splitting is done the missing values are converted by the average values of the attributes of the training example.

(f) Ensemble learning

Ensemble learning techniques are types of ML techniques in which different base models are combined to create a single best-fit predictive model. Ensemble learning has been shown to outperform ensemble members [49]. It is successfully used in both supervised and unsupervised learning tasks (regression, classification, and distance learning) (density estimation). The three types of ensemble methods we employed in our research are as follows [50]:

1. BAGGing is an abbreviation for Bootstrap AGGregating. BAGGing gets its name from the fact that it combines Bootstrapping and Aggregation into a single ensemble model. A given sample of data is used to generate multiple bootstrapped subsamples. A Decision Tree is built on each of the bootstrapped subsamples. An algorithm is used to aggregate the Decision Trees to form the most efficient predictor after each subsample Decision Tree is formed.

2. Boosting: The boosting ensemble also combines different types of classifiers. Boosting is a sequential ensemble method in which each model or classifier is run based on attributes that will be used by the following model. The boosting method distinguishes a stronger learner model from a weaker learner model by averaging their weights. In other words, a stronger trained model is reliant on a number of weakly trained models. A weak learner, also known as a weak trained model, has a low correlation with true classification. In contrast, the next weak learner is slightly more correlated with true classification. When such disparate weak learners are combined, a strong learner is produced. It is highly correlated with the true classification.
3. Stacking: This method also employs a meta-classifier or meta-model to combine multiple classifications or regression techniques. The combined model is trained to use the outputs of the lower-level models, which are trained using the complete training dataset. Unlike boosting, each lower-level model is subjected to parallel training. The training dataset for the next model is the prediction from the lower-level models, generating a stack in which the top layer of the model is more trained than the bottom layer of the model. The top layer model has high prediction accuracy and is built based on lower-level models. The stack grows until the best prediction is carried out with the least amount of error. The combined models or meta-prediction models are based on the predictions of the various weak models or lower layer models. It aims to create a model with fewer biases.

(B) Unsupervised Learning Algorithm

Unlike supervised learning, an unsupervised learning algorithm uses a set of unlabeled inputs (*i.e.,* without labeled) [38]. An unsupervised learning algorithm seeks to discover patterns, structures, or knowledge in unlabeled data by grouping sample data into different groups based on their similarity. Unsupervised learning techniques are widely used in data clustering and aggregation [38].

(C) Reinforcement Learning Algorithm

Reinforcement Learning is another well-known learning technique [38]. An agent, a State-space S, and an Action space A are all involved in Reinforcement Learning. The agent is a learning entity that interacts with its surroundings to determine the best course of action

to maximize its long-term reward. The long-term reward is a discounted cumulative reward that includes both immediate and future rewards.

1.4.3.3 Decision-making phase

For SEEE, there are various evaluation metrics used to evaluate and compare the accuracy of the estimation model. The choice of the appropriate performance metric and evaluation is consistently situated according to the problem type which can be a regression, classification, or clustering [51].
The evaluation metrics used in this thesis can be summarized as follows: (1) AE: absolute error (2) Pred (x): prediction level (3) MMER: mean magnitude of error relative to the estimate (4) MMRE: mean magnitude of relative error (5) RMSE: root mean squared error.

1.5 Estimating in the Context of Traditional and Agile Software Projects

Since the year 2000, Agile Methodologies (Scrum) have grown in popularity. The software maintenance phase of the software lifecycle is the most expensive and time-consuming, and it makes extensive use of Agile methodologies [52]. It promotes flexibility and adaptability to make software upgrades over time.

1.5.1 Switching from Waterfall to Agile

Since 1970, all software companies used the traditional waterfall model for software development. If a particular phase is not completed and approved, moving to the next phase or any other is not possible. Due to its shortcomings, the software development using the Waterfall model follows a linear, sequential design approach in which the project is delivered throughout a set of phases. These should be completed one after another. Even making changes is difficult. The maintenance cost associated with the use of this approach was increasing. To address these issues, the Agile methodology in which all the phases overlap and the requirements are gathered iteratively and incrementally, was introduced. This minimizes the shortcomings of the waterfall model and hence improves the software development process cost-efficiently. With agile, maintaining software becomes quite easy which enhances the quality as well as reduces the cost. Agile is based

on its four factors which include: Cost, Schedule, Requirements, and Quality [52].

1.5.2 The different Agile Approaches

Many agile approaches implement the values and principles of Agile manifest differently. Most of these approaches are used in developing and maintaining the software at a lower cost. The common Agile approaches are Scrum, XP, Kanban [52].

1.5.2.1 eXtreme Programming

eXtreme Programming is one of the most widely adopted agile methodologies which was created by Kent Beck [53]. It primarily focuses on the development phase rather than the managerial aspect of software projects [53]. A release plan is developed initially. Users write user stories to describe what they want and are part of the developer team. This ensures that all the requirements are being added in accordance with and presence of a user. The team breaks the tasks into iterations and at the end of it.

1.5.2.2 Scrum

Ken Schwaber and Mike Beedle popularized Scrum in the 1990s [54]. Scrum is also one of the most popular Agile approaches and is incremental and iterative. Scrum is based not only on development but also on management [55]. Scrum divides work into sprints, which are cycles of work. Requirements are prioritized and referred to as user stories during each sprint.

1.5.2.3 Kanban

Kanban is described by Anderson as "an evolutionary change strategy that uses a kanban (small k) pull system, visualization, and other tools to accelerate the adoption of Lean principles [56]. It is an iterative and evolutionary process". The main goal of the Kanban system is to maximize unit productivity by reducing the process idle time. When used correctly, the Kanban system is a very cost-effective process [56].

Although there are a lot of similarities between Agile approaches, Scrum is used the most in practice [55].

1.5.3 Differences between Traditional and Agile approaches

The main advantage of agile over the traditional (waterfall) model is that it is based on the concept of iterations. Following each iteration, users will be able to obtain a working version of their project. Based on this, even after the design phase has begun, the user can easily add or change the requirement. However, in the waterfall model, all requirements must be submitted at the start of the project. With the involvement of the user/customer in the agile process, the software product becomes most easy to enhance and stay within the allocated budget [57].

Table 1.1 – Agile vs. Traditional software development

Waterfall model	**Agile development process**
Linear, sequential design approach	Iterative and incremental
Fixed scope	Flexible scope
Late warning of risk	Early warning of risk
Low uncertainty	High uncertainty
Lack of customer involvement	High customer interaction
Suitable for large project	Not suitable for large project
Make changes	Embrace change
Late and unpredictable delivery	Early and predictable delivery

1.5.4 Estimating in the context of Scrum

Estimating is the process of predicting or approximating the effort required to complete a particular US, including analysis, development, testing, and maintenance effort. It is done at a high level and detailed level during release planning and iterative planning. Several estimation techniques are proposed. For instance, expert judgment-based techniques and planning poker are commonly used in agile effort estimation studies [55].

(A) Planning Poker

Planning Poker is a widely used estimation technique. With the PP technique, the agile team used values that can be a Fibonacci sequence or any other choice progression. The values represent the effort required to develop or maintain the particular US. Planning Poker operates based on team consensus [58]. Typically, the outcomes would be acceptable, with fewer risks and errors [58]. Participants in this process will be given special decks of Planning Poker cards. Basically, the numbers in the sequence reflect that uncertainty grows proportionally with the story size. A high estimate means that the

story is not well understood and should be decomposed into smaller stories (if possible). The Fibonacci sequence is often utilized by software teams [59], even though there is no consensus on these values.

(B) Expert Judgment

Expert is the most experienced person working in the software industry regarding the estimation of various projects [60]. They have extensive knowledge, which has a significant impact on development time and cost, as well as the deviation of actual costs from estimates. As a result, experts' advice is frequently sought when analyzing project costs and timelines. Expert's judgment method is influenced by a variety of subjective factors such as bias, the influence of the work environment, the type of projects handled by the concerned experts, and human errors [61]. It is a practical and efficient method for estimating small and medium-sized projects [61].

1.6 Conclusion

In this chapter, we established a background on software maintenance (enhancement), software measurement, and effort estimation in software projects within both traditional and agile contexts. We presented an overview of FSM and a comparison between the FSM methods supported by ISO standards. We showed that COSMIC ISO 19761 has been widely used to measure the functional change size of any type of software. Finally, we presented the techniques used when estimating effort in the context of traditional and Scrum projects. We presented the most popular effort models used in software development and enhancement projects, in particular algorithmic and non-algorithmic, and ML techniques. In the next chapter, we will present an SMS on the use of ML techniques for SEEE.

Chapter 2

Systematic Mapping Study: Software Enhancement Effort Estimation using Machine Learning Techniques

Contents

In Short

In the software industry, estimating is crucial for the success of software project planning and management. Several approaches used ML techniques to anticipate the software project effort to improve the accuracy of estimates. This second chapter is about SEEE with the use of ML techniques. Its goal is to present an SMS on the use of ML techniques for SEEE. The SMS was carried out by reviewing pertinent papers from 1995 to 2020. We followed well-known procedures. We found 30 relevant studies using four search engines: 19 from journals and 11 from conference proceedings. Based on the results obtained in this SMS, estimators should be aware that SEEE using ML techniques as part of a non-algorithmic model has demonstrated high predictive accuracy compared to algorithmic models. The use of ML techniques, in general, provides reasonable precision when using the enhancement functional size as an independent variable.

2.1 Introduction

The software industry has been progressing over recent years. The cost of software maintenance is greater than that of software development [5]. Similar to software development, software maintenance is also important. Among the cost drivers, human effort is the most important. In software engineering, the expression "Effort estimation" is similar to "Cost estimation" [62]. As a result, the majority of the proposed effort estimation is expressed in terms of Person-Month [26]. Software effort estimation, also known as software effort prediction, is regarded as the most important domain of both software development and software maintenance projects [37]. In software industries, the development phase has traditionally been valued more than the software maintenance phase [6]. Indeed, according to a recent survey on the software maintenance process [63], software maintenance is the longest and, in most cases, the most expensive phase of the software maintenance life cycle. As a result, predicting software maintenance effort is an important task that, if done correctly, can reduce the costs of software development and maintenance projects. Consequently, accurate estimates of software maintenance effort have a positive impact on project planning and management [64]. By reducing uncertainty and increasing customer confidence.

Even with the use of modern software development approaches such as agile, software industries have to revise and refine effort estimation when changes occur. It may be necessary to make an accurate estimate based on relevant and standardized information. This is a useful tool for project managers who want to improve their industries. Of course, when the same definition and measurement standard are used as inputs to the prediction process. However, predicting will be ineffective, or will damage future business opportunities, if not done correctly (underestimate, overestimate). Many approaches with various estimate models are presented to give more accurate software estimation. Expert judgment, algorithmic models (such as COCOMO II), and non-algorithmic models are the three types of models (such as ML techniques [37]). Many researchers claim that applying ML techniques (as a non-algorithmic model) can increase estimation accuracy [65].

Software development becomes more complex as the software project grows and evolves. Some ER is required to improve software quality characteristics that are likely to be transformed into functional requirements (*e.g.,* perfective maintenance). Other requests are used to implement new requirements (*e.g.,* Adaptive maintenance). It is

clear when managing the software maintenance process, particularly after the software has been delivered, the estimated cost of which exceeds 70 percent of the total costs of the software development process [10].

Consequently, Enhancement is a type of software maintenance that may necessitate major project re-planning and improved implementation. Special attention is devoted to evaluating ER (*i.e.,* functional changes affecting user requirements) and predicting their impact on the estimated effort. As a result, user ER appears not only during the software maintenance phase but also throughout the software development project. For these reasons, estimating the effort required to implement ER or functional changes should be regarded as the key activity in managing a software project. Although there are several methods for improving the accuracy of SEEE, the choice of suitable SEEE is not only limited to the ability of software developers/maintainers to handle ER but also to its evaluation. When assessing the performance of a prediction model, the choice of a suitable one is based on the quality of its inputs, data sets, and, most importantly, the use of international standards [32]. As a result, when managing a software project, choosing the appropriate prediction model when the user's requirements are subject to change has become a significant challenge.

This chapter reports on the SMS of relevant research papers (journals and conference proceedings papers) investigating the topic of SEEE using ML techniques. Our SMS investigates SEEE to identify research gaps, recommends future research avenues, obtains a detailed overview of the proposed SEEE models, and identifies the ML technique most commonly used for predicting enhancement maintenance effort. Between 1995 and 2020, we looked for the most popular digital database libraries in computer science. We examined 30 related studies published in 19 journals and 11 conference proceedings. Our primary goal is to examine the SEEE state, as well as its limitations and challenges. In this chapter, we used the Petersen research method [66] [1]. SMS is beneficial to both software engineering researchers and practitioners.

The remainder of this chapter is organized in the following manner. Section 2.2 describes the SMS's research methodology. Section 2.3 displays the SMS results. Section 2.4 presents and discusses the SMS main findings. Section 2.5 concludes with limitations and research gaps.

2.2 SMS Methodology for SEEE

A SMS identifies the nature and extent of empirical study data that is accessible to answer a specific mapping research question in a systematic and objective manner [1].

According to [1], SMS is divided into six steps : (1) Defining the mapping questions, (2) Finding primary studies, (3) Screening studies, (4) Abstract key-wording, (5) Data extraction, and (6) Mapping Results.

2.2.1 Defining the mapping questions

The mapping questions (MQ) addressed in this SMS, as well as their associated objectives, are listed in table 2.1.

Table 2.1 – Mapping questions and objectives

ID	Question Details	Objective
MQ1	What are the trends in software maintenance prediction research from 1995 to 2020?	To ascertain the temporal trends in SEEE publications.
MQ2	How can software enhancement (*i.e.,* functional changes) effort prediction be improved?	To explain how to account for changes in SEEE.
MQ3	How is enhancement effort predicted and assessed?	To describe the methods for measuring and evaluating enhancement effort prediction that have been proposed.
MQ4	What type of ML problems addressed in SEEE fall into?	To investigate the different types of ML problems: regression, classification, and clustering.
MQ4.1	What are the proposed methods for predicting software enhancement effort?	To describe the proposed methods used for predicting software enhancement effort.
MQ4.2	What are the SEEE datasets used to build prediction models?	To identify various datasets commonly used in the SEEE domain.
MQ4.3	What are the independent variables used to improve SEEE model performance?	To assess the significance of using the FSM method for evaluate the accuracy of SEEE.
MQ4.4	Which single models are most frequently used for SEEE?	To identify the most commonly used single models for SEEE.

2.2.2 Conducting the search for primary studies

We followed Kitchenham's guideline for conducting SMS studies in order to have a clear description of our search strategy. Following the steps discussed, the research was carried out in four digital libraries:

— Google Scholar,

— IEEExplore,

— ACM Digital library, and

— ScienceDirect.

Table 2.2 contains relevant journal and conference proceedings in the SEEE field for identifying search sources. The search was limited to articles published between 1995 and 2020. A review of various ML techniques for predicting SEEE revealed that estimation accuracy can be achieved [67] [68]. However, we chose some studies that did not employ ML techniques because they answered some of our MQs. The following is a list of search strings: (Mapping OR literature) AND (context OR area OR field) AND (approach OR method OR technique) AND (information OR requirement) AND (maintain OR enhance OR modify OR change) (effort, cost, size, or time) AND (estimate, predict) AND (software, system, application, or product) AND (project OR activity).

2.2.3 Screening studies

The following inclusion and exclusion criteria were used to select the most relevant studies.

Studies that fulfill the following criteria are selected for inclusion:

— Studies including the keywords directly related to the Software enhancement maintenance field (adaptive/perfective maintenance, software change), effort prediction/estimation.

— Studies that contain the exact keyword "effort prediction" or synonyms.

— For the period 1995-2020, studies written in English and including the most recent publication are included.

Excluded studies are those that do not meet the inclusion criteria:

— Studies without the keywords “software enhancement effort prediction”

— Studies without the exact keyword “effort prediction”, or its synonym.

Table 2.2 – Selected journals and conference proceedings

Source	**Number of studies**	**Published by**	**Impact factor/ Rank**
Journal			
Applied Soft Computing	1	Elsevier	4.873
Empirical Software Engineering	1	IEEE	4.457
IEEE Transactions on software engineering	5	IEEE	3.92
Information and Software Technology (IST)	6	Elsevier	2.76
Journal of Software Maintenance and Evolution: Research and Practice	2	Wiley	1.320
Journal of Quality in Maintenance Engineering	1	Emerald Group Publishing Ltd.	1.46
Journal of systems and software	1	Elsevier	1.352
International Journal of Software Engineering and Knowledge Engineering	1	World Scientific Publishing Co. Pte Ltd	0.644
International Arab Journal of Information Technology	1	Zarqa University	0.410
Conference			
International Conference on Software Maintenance (ICSM)	5	IEEE	A
Asia-Pacific Software Engineering Conference (APSEC)	2	IEEE	B
IEEE International Conference on Software Maintenance and Evolution	2	IEEE	A

— Studies without the combination of “software enhancement” and “effort prediction”, or its synonym

2.2.4 Key wording of abstracts

The following titles were chosen after screening the Paper's title (30) [69, 70, 51, 67, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 68, 87, 88, 89, 5, 90, 91, 92, 93] software quality effort prediction, software maintenance effort prediction, software maintenance effort prediction using ML techniques, and software change effort prediction.

The following keywords appear in the paper's keywords and abstract (25): software maintenance, software enhancement, change request, software enhancement maintenance effort prediction, ML techniques, evaluation metrics.

The number of studies chosen after screening Full Papers (21) taking into account the models and approaches proposed in the area of software maintenance effort prediction; the evaluation metrics used to perform the proposed models

Number of studies selected after screening Papers' Inclusion and Exclusion Criteria (14) taking into account the discussed inclusion and exclusion criteria.

Number of studies passed our quality assessment (14) The quality assessment follows the defined quality checklist as proposed by Kitchenham [66]. The primary goal of the quality assessment is to evaluate and select studies that answer our MQs, as well as to support more in-depth analysis of inclusion and exclusion criteria. The quality assessment questions scoring procedure

The scoring procedure for the quality assessment questions of our chosen primary studies (see Table 2.6) is as follows [51]:

— 1 represents Yes.

— 1/2 represents Partly.

— 0 represents No.

The scores rank the papers into three categories [51]:

$$excellent(6.5 \leq score \leq 8)$$

$$good(2.5 \leq score \leq 6)$$

$$fair(1 \leq score \leq 2)$$

2.2.5 Data Extraction

When using the SMS, data extraction is an important step to take [94]. As a result, in order to answer the mapping questions listed in Table 2.1, the data extraction was designed to collect all relevant data from each chosen primary study, which would then be used to answer our research questions. As shown in Table 2.3, the listed mapping questions are associated with five major fields: software maintenance challenges, Software enhancement (changes request), SEEE models, Software enhancement evaluation metric, and Software enhancement single (individual) models.

Table 2.3 – Data Extraction Properties with their mapping questions

Research Questions Field	**Research question**
software maintenance challenges	MQ1,MQ2,MQ4
Software enhancement (changes request)	MQ2
SEEE models	MQ4, MQ4.1, MQ4.2, MQ4.3
Software enhancement evaluation metric	MQ3
Software enhancement single (individual) models	MQ4, MQ4.4

2.3 Mapping results

This section discusses the mapping questions to be addressed.

MQ1: What are the trends in software maintenance prediction research from 1995 to 2020? This mapping study will be conducted between 1995 and 2020 (see table 2.4). The distribution of research over the years is shown in Figure 2.1. Although a large number of predictive models have been proposed for development projects, few predictive models have been proposed for software enhancement maintenance projects. Since most of the research is published in trusted impact factor journals and leading software engineering conferences, the importance of this mapping research has increased.

From 2005 to 2010, highly ranked publications were recorded. Six studies are retrieved from the IEEE Explore Digital Library during this time period, and six studies are retrieved from Science Direct. High-quality papers are published in the IEEExplore Digital Library and Science Direct. This emphasizes the challenge of SEEE using ML techniques to researchers. 65 percent of the 30 primary studies were published in journals,

Table 2.4 – The distribution of years for SEEE area

Year/DataBase	IEEE Xplore Digital Library	Google Scholar	ACM Digital library	Science Direct
1995-2000	3	3	1	1
2000-2005	3	7	5	2
2005-2010	6	11	6	6
2010-2015	5	6	2	1
2015-2020	7	11	3	4

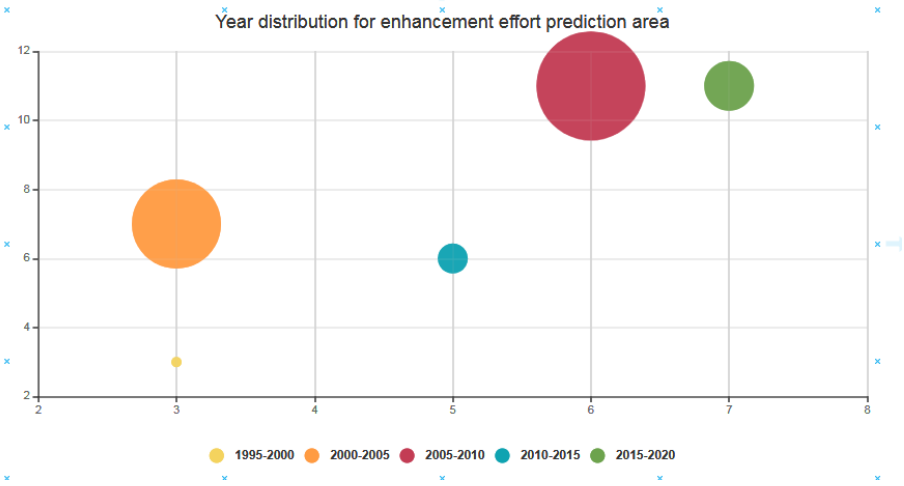

Figure 2.1 – Distribution of years for SEEE area

and 35 percent were presented at conferences (see Figure 2.2).

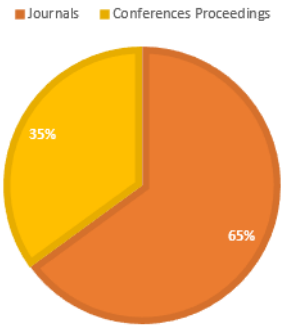

Figure 2.2 – The distribution of publication type

MQ2: How can SEEE be improved ? Enhancement is related to operational (error correction) or technical purposes (goal modification) requested by the user after project completion. It is critical to classify the Requirement ER. As a result, ER can be further classified based on their origin (internal or external), type (add, modify, or delete), etc.

When estimating the effort of a software enhancement project, identifying only the types of changes requested is effective. In other words, in order to provide an accurate effort estimation in software projects, the characteristics of each type of requirements ER must be identified [95]. Therefore, effort estimates must be performed before maintaining software products. Furthermore, it is critical to meet and satisfy the changing needs of the customer. In the same context, Basri et al. [19] assume that the effort required in the maintenance phase (after the software product is delivered) is less than the effort required in the development phase. As a result, software managers must make sound decisions when dealing with changes during software development or maintenance. The prediction of the change effort produced by the changes is one type of information that assist decision-making. Predicting the effort of implementing an ER is not an easy task for both the manager and the development team when the software is being developed. An accurate effort prediction can prevent software projects from going over or under budget. Overestimation can lead to financial failure for the organization, while underestimation can lead to exhausted software quality and, as a result, software project failure. Estimates may be inaccurate because there is not enough information on the project at the start, and requirements are almost always poorly described. The 2018 study by PMI's Pulse of the Profession puts the average number of the primary causes of those failures. The main causes of software project failures are changes in requirements (such as organizational priorities, project objectives, and so on), inaccuracy in requirements gathering, and inaccurate estimates (in terms of costs or time) [20].

The amount of information required to develop a detailed cost/effort estimate is frequently missing in the early stages of the SLC. A poorly defined requirement is a big obstacle to project success. Furthermore, requirements are represented in many formats. Software requirements, for example, might be expressed by a textual description of a UML use case or the US within the scrum process. Table 2.5 lists the quality criteria for requirements. To assist in identifying business opportunities and facilitating system design, the quality requirement should be correct, feasible, necessary, unambiguous, complete, consistent, testable, clear, and traceable. A detailed description of requirements throughout the SLC helps to predict their size more precisely. Following that, it contributes to a more precise estimation of the change effort, resulting in better-informed decisions. This is in contrast to many other prediction models, which do not take quality requirements into account.

Table 2.5 – Characteristics of a "good" Requirement

Criteria [96]	**Explanation**
Correct	Each requirement must accurately describe the functionality to be delivered.
Feasible	Implement each requirement within the known capabilities and limitations of the system and its environment.
Necessary	Each requirement should document something the customers really need.
Unambiguous	Write each requirement in simple straightforward language of the user domain.
Complete	No requirements or necessary information should be missing.
Consistent	Consistent requirements do not conflict with other software requirements or with higher level requirements.
Testable	Testers should be able to verify whether the requirement is implemented correctly.
Clear	concise, simple, precise
Traceable	Linking each requirement to its source is essential, to achieve it, each requirement must be written in a structured and precise way

In [97], effort was predicted based only on software functional size. In [98] model, the quality requirements attributes have increased software effort by 30 percent.

MQ3: How is enhancement effort predicted and assessed? There are four types of software maintenance classified as correction maintenance including corrective maintenance and preventive maintenance, and enhancement maintenance including adaptive maintenance and perfective maintenance. The main study selected for this SMS (see Table 2.6) includes two types of software enhancement effort prediction. For each type of maintenance, we provide more detailed information on the definitions and methods used to predict the software effort.

Table 2.6 – Software Maintenance type used for effort prediction

Study	**Type of Maintenance**	**Description**
S1 [67]	Enhancement	Applying ML technique: a radial Basis Function Neural Network (RBFNN), Support Linear Regression (SLR), Multiplier Linear Perceptron (MLP), and Gradient Regression Neural Network (GRNN).

S2 [68]	Adaptive, Corrective, preventive	Contributing the employment of use-cases for predicting the effort software maintenance
S3 [99]	Corrective, Enhancive, Reductive	Evaluating the software maintenance effort during the three maintenance types, particularly the effort spent by the programmer on maintenance activities.
S4 [89]	Adaptive; Corrective; perfective	Representing software maintenance effort using some determined software measures
S5 [92]	Adaptive	Performing the prediction of adaptive maintenance based on the prediction of lines of codes to be changed.
S6 [100]	Corrective, Perfective	Proposing a maintenance effort model that is based on the function point measure and new maintenance productivity factors.
S7 [71]	Enhancement, corrective	Proposing a maintenance prediction effort method based on an improved version of analogy with Virtual Neighbor.
S8 [93]	Adaptive, Corrective, Perfective, preventive	Applying Function-point based regression models for the maintenance effort prediction.
S9 [72]	Adaptive	Presenting a model and metrics for the adaptive maintenance effort estimation/prediction.
S10 [73]	Adaptive, Corrective, Perfective, preventive	Implementing, evaluating and improving software maintenance effort prediction model based on expert judgment approach.
S11 [74]	Adaptive, Corrective, Perfective, preventive	Proposing six models based on eight different indicators of evolution activity, their predictive power is examined and compared to that of two baseline models.
S12 [90]	Enhancement	Applying stochastic gradient boosting (SGB) algorithm to predict software maintenance effort. make a comparison between the prediction performance of the SGB algorithm and statistical regression, neural network, SVR , decision trees, and association rules.

S13 [5]	Enhancement	Proposing an enhancement prediction model based on the use of two types of SVR (-SVR and -SVR).
S14 [91]	Enhancement	Proposing a prediction software enhancement maintenance model based on the use of ANN (Artificial Neural Network): Multi-Layered Feed Forward Neural Network and trained with Back Propagation training.

The two current models algorithmic and non-algorithmic models are widely used for estimating enhancement effort. The algorithmic model predicts maintenance effort using mathematical formulas. Non-algorithmic models predict using past project experiences.

Table 2.6 shows that ML techniques (or non-algorithm models) are most commonly used for enhancement effort prediction. Various ML techniques including RBFNN, neural network (NN), the rule engine (RE), multi-regression, a multilayer feed-forward neural network (MFFNN) with back-propagation algorithm and Bayesian regularization training, stochastic gradient boosting (SGB) model, Virtual Neighbor (VN), and SVR, are used in the selected primary studies (SVR). It is worth noting that COCOMO II is one of the most widely used algorithmic models in the industry for estimating effort [91] [21].

From table 2.6, the main questions to answer are: Is there any relationship between the techniques used? Is it possible to use a combination of several techniques? To be clear about the first question, the choice of ML Techniques is dependent on the situation surrounding the project taking for example the first study [67] the author used case-based reasoning and decision trees because they are intuitive and easy to understand. The neural networks algorithm can learn non-linear functions. There is evidence here that the situation, or more specifically the nature of the historical dataset to be used for such a learning problem. It has a significant impact on selecting the ML technique that best meets the accurate prediction. Regarding the second question, it should be noted that no works have proposed the use of Ensemble models which explore the combination of two or more ML techniques to predict effective enhancement effort. Many researchers assessed the performance of their proposed model using various error metrics such as MRE, Pred, etc. The metrics used to evaluate the accuracy of SEEE are listed in Table 2.7. Table 2.7 also reveals that the majority of the selected primary studies were used for performance evaluation of the Magnitude of Relative Error (MRE). The mean absolute

Table 2.7 – Criteria used for evaluating SEEE

Study	**Prediction accuracy criterion**
S1 [67]	Absolute Residuals Friedman statistical
S2[68], S3 [99], S7 [71], S8 [93], S9 [72], S10 [73], S11 [74], S13 [5], S14 [91]	Mean Relative Error (MRE)
S4 [89]	Correlation coefficient
S6 [100]	Coefficient of determination (r2)
S5 [92]	Standard error of the estimate, MMRE - Mean Magnitude of Relative Error, MdMRE - Median Magnitude of Relative Error, PRED(25)
S12 [90], S13 [5]	Absolute Residuals (AR), Mean of Absolute Residuals(MAR)
S4 [89], S8 [93]	Pred(25)
S10 [73]	Pred(50)
S11 [74]	MdMRE - Median Magnitude of Relative Error, Pred(10) and Pred(50)
S14 [91]	Mean-Square-Error (MSE)

error (MAE) is the average of the absolute value differences between the actual and predicted effort. N denotes the total number of projects. It is calculated as follows: MRE equals (actual effort - estimated effort / actual effort). The distribution of evaluation metrics used by selected primary studies is depicted in Figure 2.3. The MRE is the most widely used evaluation metric for predicting enhancement effort (48%) followed by Absolute Residuals and PRED (25%) (12 percent).

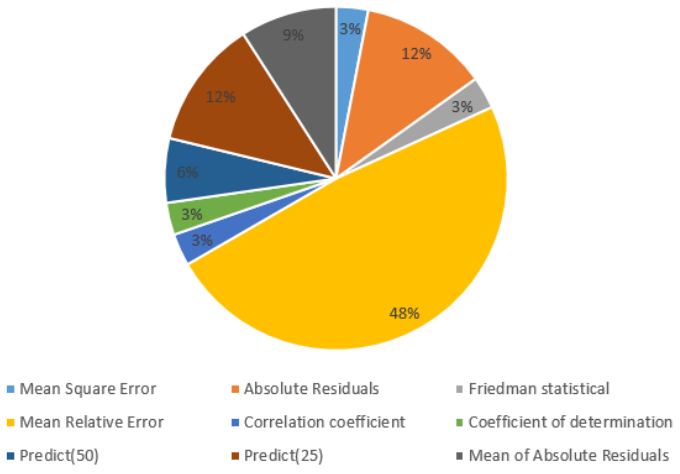

Figure 2.3 – The distribution of performance metrics used for evaluating SEEE

MQ4: What type of ML problems addressed in SEEE fall into? Several performance metrics are used to evaluate and compare the performance or the accuracy of the SEEE model. The appropriate performance evaluation metric is consistently selected based on the problem type which can be a regression, classification, or clustering problem [51]. In terms of what is used to provide a more accurate SEEE model, we find that regression problems outnumber those of ML (see Table 2.6). The regression models used in the selected studies were statistical regressions [89, 91, 73], General Regression [67], Support Linear Regression models [67], SVR [5], and decision trees stochastic gradient boosting [90]. The purpose of regression models is to construct a function *f(x)* that adequately maps a set of independent variables (X1, X2,..., Xn) into a dependent variable Y [37].

MQ4.1: What are the proposed models for SEEE? Many researchers proposed developing SEEE models in order to obtain more accurate estimates. The findings of a literature review for which various ML techniques are investigated in terms of their ability to generate accurate prediction models are presented. Between 1995 and 2020, 14 studies proposing SEEE models were published. Table 2.8 displays the techniques and data sets used for SEEE.

Table 2.8 – ML techniques and data sets used for SEEE

Study	**Used Techniques**	**Source**
S1 [67]	Radial Basis Function Neural Network, SLR, MLP and GRNN	ISBSG Release 11
S2 [68]	Linear Regression Analysis	Commercial software Process Management tool.
S3 [99]	Three LOC metrics: LOC added, modified, and deleted.	24 projects from 23 graduate students and one senior majoring in computer science.
S4 [89]	Linear Regression	200 software projects maintained in the NASA Goddard Space Flight Center
S5 [92]	Multiple Regression, Simple Regression	32 projects and an industrial project developed in Lexington, Kentucky.
S6 [100]	Regression analysis with Functional Points	26 maintained software projects

S7 [71]	Analogy with Virtual Neighbor compared to The normalized dimension value (NDE) method, the closest neighbor (CN) method, and the original COCOMO81 method	24 projects from a Hong Kong branch office of an international financial institution
S8 [93]	Function-point based regression models	145 maintenance and development projects
S9 [72]	Multilinear regression analysis	145 software projects of a single outsourcing company.
S10 [73]	Regression analysis, neural networks and pattern recognition	109 tasks from a Norwegian organization
S11 [74]	Univariate Regression, Multivariate Regression	121 observations from 1981 to 1998
S12 [90]	Statistical regression (SR), NN, SVR, Decision Tree (DT) and Stochastic Gradient Boosting (SGB)	Five SIS data sets selected from the ISBSG Release 11.
S13 [5]	SVR, Decision Tree (DT)	Software enhancement projects were obtained from ISBSG Release 11
S14 [91]	Artificial Neural Network	COCOMO data-set

Table 2.8 shows that the majority of the primary studies used regression methods such as Statistical Regression (SR), SVR, and Stochastic Gradient Boosting (SGB) for SEEE. Table 2.8 also demonstrates that the Artificial Neural Network (ANN) was used in a large number of the primary studies.

MQ4.2: What are the SEEE datasets used to build prediction models? Table 2.6 highlights the variety of datasets used for SEEE. A dataset is a "collection of connected sets of data that may be utilized to run ML-based models, and it is regarded as the foundation for developing prediction models" [101]. When constructing a learning model, the dataset is divided into two parts: a training set for model input and a testing set for evaluating the built model [101].

Table 2.8 depicts a summary of the various types of datasets used in the selected primary studies. Primary study datasets are classified into two types.

— Public dataset: The dataset can be found as an appendix or table in a published paper or in a publicly accessible repository [51], such as the ISBSG Release 11 in S1[67], S13 [5], and S12 [90]. The ISBSG maintains a Development and Enhancement Repository [2] (also known as the "ISBSG dataset"). In S14 [91] and S7 [71], we additionally acknowledged the use of the COCOMO II dataset, as well as the NASA Goddard Space Flight Center dataset in S4 [89].

— Private dataset: The dataset is not publicly available and was obtained from a private software system, such as in S8 [93], S10 [73], S6 [100], S9 [72] and S5 [92].

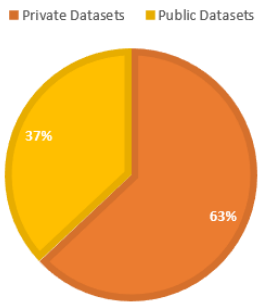

Figure 2.4 – Percentage of studies using each type of datasets

Figure 2.4 shows the percentage of studies using the different types of datasets. It is also a difference in the number of studies using public versus private datasets. The majority of the selected primary studies were conducted using private datasets, as opposed to public datasets. This explains the lack of studies addressing SEEE with public datasets. As a result, we observed a lack of comparative studies among researchers' effort prediction results in the field of software maintenance. This is since each dataset's features are unique.

As stated in MQ3, one of the important factors influencing one choice over another is the dataset used for learning. In other words, the selection of the datasets' features is critical for more accurate prediction. Some research studies focused on feature selection [102] in order to select the relevant features and provide the best configuration. Other research studies, on the other hand, focused on the use of independent variables (that we detailed in the next question). As a result, several factors, such as the size of the dataset, can have an impact on the prediction results [103]. Therefore, many research studies used a single dataset [67]. However, we observed the use of more than one dataset in other research studies [92].

MQ4.3: What are the independent variables used to improve performance of SEEE model? The data features derived from the ISBSG dataset are used for three distinct purposes in SEEE [23]:

— Filtering features [23]: The filtering phase is carried out in order to identify the most relevant set of projects.

— Dependent features [23]: The dependent variable in effort prediction models is usually 'Effort'. The dependent variable representing the output is the one that must be predicted.

— Independent features [23]: A large number of the 118 features in the ISBSG dataset are most likely considered effort factors. As a result, the criteria for selecting appropriate independent variables are not standardized. It all depends on the area of study. In most cases, the independent variable displays the most significant data values [23].

Table 2.9 lists the independent variables used in the primary studies to predict maintenance effort. Table 2.9 shows that there is a broader range of using functional size (in

Table 2.9 – Independent variables used for SEEE

Study	**Independent Variables**
S1 [67], S6 [100], S8 [93], S13 [5], S12 [90]	Function Points Size
S2 [68]	Size Parameter (use cases)
S4 [89]	Lags time (Lag time), the number of LOC changed (LOC change), and the number of modules changed (Module change)
S5 [92]	Percentage of operators changed and the number of lines of codes changed edited, added or deleted (DLOC)
S9 [72]	Size and complexity
S10 [73]	Size of the changed application
S11 [74]	SubsysInclCreations, SubsysChanged, SubsysHandled, ModulesCreated, ModulesHandled, ModulesChanged, TotalHandlings, Modiflandlings
S14 [91]	KLOC

terms of function points) as independent variables. The authors [23] assume that the size features are solely related to the amount of effort required. Sizing is considered one of

the most accepted methods which have a greater impact on predicting software project effort [94]. The Size metric and effort estimations are the most intriguing metrics which, if correct, have a positive impact on software project planning and management [64].

Nowadays, FSM methods, including the first generation (*e.g.,* IFPUG) and second-generation (*i.e.,* COSMIC) are widely used in the software industry to the size software product. The obtained functional size is identified as an independent variable in the prediction models. The IFPUG software size is frequently used as an independent variable for estimating software effort. Furthermore, the COSMIC functional size is used to appropriately estimate software size as well as the size of an ER (add, modify, delete)[15]. Industry results assume that the COSMIC sizing method is successfully used in the software industry estimating process.

MQ4.4: Which single models are most frequently used for SEEE? When investigating the primary selected studies, Table 2.6 reveals that all studies used only single prediction models (*i.e.,* individual models). The most commonly used single models for predicting maintenance effort are SVR (including linear and multi regression) and ANN (see Figure 2.5). The evaluation metrics used to compare and evaluate the prediction

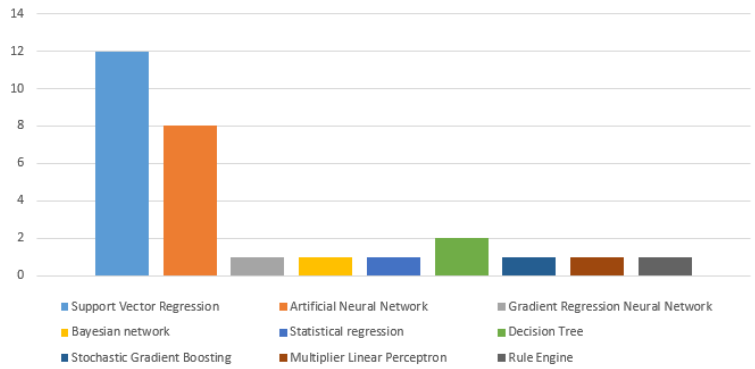

Figure 2.5 – The distribution of singles models used for enhancement effort prediction

accuracy of the single prediction models employed in each study are used to determine which model is the best (see table 2.7). Table 2.10 enumerates the set of single prediction models used in each investigation, along with the performance of the MRE value when using the ISBSG dataset, and the best model in each study. Figure 2.5 indicates that Radial Basis Function Neural Network (RBFNN), Stochastic Gradient Boosting (SGB), and PK-SVR are close to meeting the MAR criteria to build an accurate SEEE model when using ISBSG dataset.

Table 2.10 – Performance of MRE value for selected primary studies using ISBSG dataset

Study	Used Techniques	The best accuracy prediction model	MRE
S1 [67]	Radial Basis Function Neural Network (RBFNN), SLR, MLP and GRNN	Radial Basis Function Neural Network (RBFNN)	0.46
S12 [90]	Statistical regression (SR), NN, SVR, DT and Stochastic Gradient Boosting (SGB)	Stochastic Gradient Boosting (SGB)	0.36
S13 [5]	SLR, MLP, DT, ARU, -SVR and -SVR	PK-SVR	0.49

2.4 Implication for research and practice

Understanding and defining the product to be predicted is one of the first steps in any estimate. However, predicting is difficult because a software system is intangible, invisible, and intractable [20]. Understanding and predicting a product or process that cannot be seen or touched is inherently more difficult. Furthermore, software grows and changes as it is written. As a result, the stakeholders and manager must make good decisions when managing requirements changes during software development or maintenance.

Because underestimation is the more prevalent trend, it is critical to consider factors (*i.e.,* project size, team maturity, etc.) that may cause project delays. Identifying and taking into account these factors can help to reduce overestimation. Some researchers argue that because change is frequent, it is difficult to provide an accurate estimate and predict the future (it is a waste of time). However, the effort required to implement a change must be determined. This is to avoid failures and make appropriate decisions. There are numerous software cost/effort estimation/prediction models/techniques in the literature, including algorithmic models and non-algorithmic models. Recent research on software maintenance effort prediction has focused on comparing the accuracy of ML techniques as non-algorithmic for effort prediction.

A review of various ML techniques (such as regression learning models) used in predicting effort revealed that estimate accuracy can be achieved. No model is necessarily superior to another. Strengths and weaknesses are frequently complementary to one another. Which prediction models should be used for a specific project based on the project's environment? As a result, having a good understanding of requirements changes is re-

quired for predicting the effort required to rework them. However, no standard method for classifying requirements changes exists that can provide an understanding of both the project level (effect on the project progress) and the requirements level (effect on other requirements).

Size in software engineering datasets can refer to "the physical size of the program, measured in lines of code (LOC); or the functional size of the problem, measured in Function Points."[23]. FSM is especially useful when development/maintenance effort must be predicted. Despite the fact that COSMIC FSM method [2] is the most widely used standard of the FSM method, we discovered that none of the primary studies datasets used for SEEE used it, only the ISBSG dataset. The COSMIC FSM method [34] can also be used to approximate software size at the start of the software life-cycle. The COSMIC FSM method has been used successfully to size data manipulation-rich software as well as some scientific/engineering software [34]. Recently, research papers looked at the use of an ensemble of learning machines to improve software effort estimation [104]. An ensemble of learning machines is defined as "a group of learners who have been trained to perform the same task and have been combined to improve predictive performance." [104]. According to our mapping study findings, only single models have been used to predict maintenance effort. According to the findings of our mapping study, only single models have been used to predict maintenance effort. Even when considering ensemble, Leandro et al. [104] conclude that there is very unlikely to be a universally best model.

The estimation process of a software project is divided into three major parts in the software industry: estimation inputs, estimation model, and estimation output [32]. Frequently, the first section, known as estimation inputs, includes all three types of software requirements: functional and non-functional requirements, as well as project constraints. However, there is no standardized method for ensuring the quality of each of these parts. No agreement has been reached on the quality of each part of the estimation process. The software industry is frequently plagued by a number of poor estimation practices [32]. The majority of researchers are focused on developing new prediction models. A minority of researchers, however, have considered the quality of estimation input [32].

Finally, it is important to note that the inability to assess the quality of prediction models influences the selection of the best model. Indeed, employing the prediction process will help in the elimination of common research errors (*i.e.,* focusing on the

improvement of prediction models, instead of focusing on improving the input to these prediction models). As a result, using common terminology and measurement standards in the same context is essential for improving estimate quality (the output). Then, how estimates are gathered and used can have an impact on their usefulness for planning and, as a result, the success of a software project. The main goal of this chapter is to investigate and bring to light the major shortcomings of SEEE models involved in literature reviews. Results showed a lack of studies dealing with the SEEE using ML techniques. Based on the use of SMS, the findings of the mapping research questions were as follows:

— MQ1: The time frame for this mapping study has been set between 1995 and 2020. Although many prediction models have been proposed for development projects, few have been proposed for software maintenance.

— MQ2: It is critical for Software Managers to (1) make effective decisions when managing changes during software maintenance and (2) focus on software system aspects that are likely to change.

— MQ3: The majority of selected primary studies used regression methods such as Statistical Regression (SR), SVR, and Stochastic Gradient Boosting (SGB) to predict software maintenance (enhancement) effort. Whereas the majority of the primary studies used the Magnitude of Relative Error (MRE) to evaluate performance.

— MQ4: When it comes to the use of ML techniques for SEEE models, regression problems are more common than other ML problems.

— MQ4.1: Enhancement (including adaptive and perfective maintenance) is not well considered for effort prediction. ML techniques are the most frequently non-algorithm method used for enhancement effort prediction.

— MQ4.2: For SEEE, two types of datasets were used: public datasets and private datasets. The majority of the primary studies chosen were conducted using private datasets rather than public datasets.

— MQ4.3: There is a wide range of using software functional size (in terms of CFP) as independent variables.

— MQ4.4: When using the ISBSG dataset, the results show that SVR is the most commonly used model, and Stochastic Gradient Boosting (SGB) is closer to meeting the MAR criteria (=0.36) for building an accurate SEEE model.

2.5 Conclusion

As this chapter was designed as an SMS, our analysis was limited to a broad overview of the software maintenance (enhancement) effort prediction research field. When proposing a new ML prediction technique in the field of software maintenance, this mapping review takes an unbiased approach to decision-making. This chapter investigated some of the most important issues that should be addressed in the context of SEEE. The first review used 30 studies from 1995 to 2020. Then we chose 14 studies to be examined. This study enables researchers and practitioners to determine what needs to be done in the field of SEEE. In the next chapter, we discuss the effectiveness of using both ontology and ML techniques to improve the accuracy of SEEE models.

Chapter 3

Ontology-based Classification of Enhancements with their corresponding Effort Estimation

Contents

In Short

This chapter outlines how ontology knowledge representation including semantic similarity measures can improve the accuracy of ML techniques. First, an ontology-based Model (OMC) is designed to specify, present, and classify enhancement requests. Next, each enhancement request classified as Functional Change will be associated with its corresponding enhancement effort using Expert Judgment. Finally, the constructed ontology model is populated with a historical dataset to predict the required SEEE to complete an effort for software enhancement using ML techniques.

3.1 Introduction

Software enhancement is regarded as one of the critical activities in the software maintenance life cycle. It is defined as changes made to an existing application in which new functionality is added or existing functionality is modified or deleted. This would include adding a module to an existing application, regardless of whether any of the existing functionality is changed or removed [105]. Given that effort estimation is one of the primary activities of software project planning, it is necessary to define the components of an estimation process. The quality of an estimation process's outcome is determined by the quality of its inputs (such as product requirements, software development process, and project constraints) [32]. Requirements are the foundation of any software project. Identifying complete and clear requirements throughout the SLC is a difficult task. As a result, adjustments are required. ER can occur during the development of software or even after it has been delivered. To effectively evaluate enhancements and the effort required to complete these enhancements, the use of both an appropriate measurement method and an accurate estimation model is required. However, choosing an appropriate measurement method will depend on the type of ER.

ER are most commonly expressed in natural language, accounting for up to 90 percent of all specifications [40]. ER expressed in natural language are difficult to analyze and may result in confusion, inefficient distinction of requirement types, ambiguity, etc. According to [20], each proposed ER has to be analyzed to determine whether it is "in-scope" or "out-of-scope". "In-scope" ER fall within the scope of the project so that they can be implemented with little or no disruption to the planned activities. They involve minor adjustments to an existing requirement. They usually have a minor effort within the project process. While "out-of-scope" ER falls outside the scope of the project and must be accompanied by an adjustment to project planning with significant effort impact [6].

As mentioned in chapter 1, requirements for software system projects are divided into three categories [6]: functional user requirements (FUR), non-functional requirements (NFR), and project requirements and constraints (PRC). In this chapter, we propose to classify ER into two categories they affect. Using ontology, ER that affect Functional User Requirements are classified as FC. ER that affects NFR or PRC is classified as TC. The classification of ER allows stakeholders to be selective in the use of the appropriate measurement method. As a result, they can evaluate the impact of ER on the effort required for their implementations. This is useful when they need to improve their

understanding of management decisions.

In this chapter, we first propose an OMC which will be used for classifying ER into FC or TC. Therefore, every effort is made based on this categorized ER using Expert Judgement (which serves as input to the SEEE model). A detailed description of the enhancement request will be presented in this chapter. Next, We estimate the software enhancement effort based on the use of four ML techniques. We evaluate how well FC is correlated to the SEEE. We also present which of the selected ML techniques provides more accuracy for estimating.

The remainder of this chapter is organized as follows: section 3.2 gives a detailed description of our research work process. Section 3.3 presents the gathering data phase. Section 3.4 presents the Ontology based-Semantic Classification model. Section 3.5 presents the experiments and results and addresses the threats to validity. Section 3.6 presents the discussion and the limitations of our contribution. Finally, section 3.7 provides our conclusions.

3.2 Research Work Process Overview

In order to ensure that results are generally valid, the empirical evaluation of the proposed models must cover a wide range of conditions. These conditions include various parameter settings and datasets of varying size, skewness, and noisiness, as well as various preprocessing approaches. Our research work process (see Figure 3.1) includes the following three steps:

1. Gathering Data
2. Ontology Semantic Classification
3. Constructing Prediction Models and Evaluation

3.3 Gathering Data

Many scientific disciplines make extensive use of public experiment repositories to facilitate the sharing of experiment data. On the other hand, unambiguous description languages which are based on a careful examination of the concepts are created to be used within a domain and its relationships. This is formally represented by ontologies, which are machine-manipulable domain models that clearly describe each concept (class).

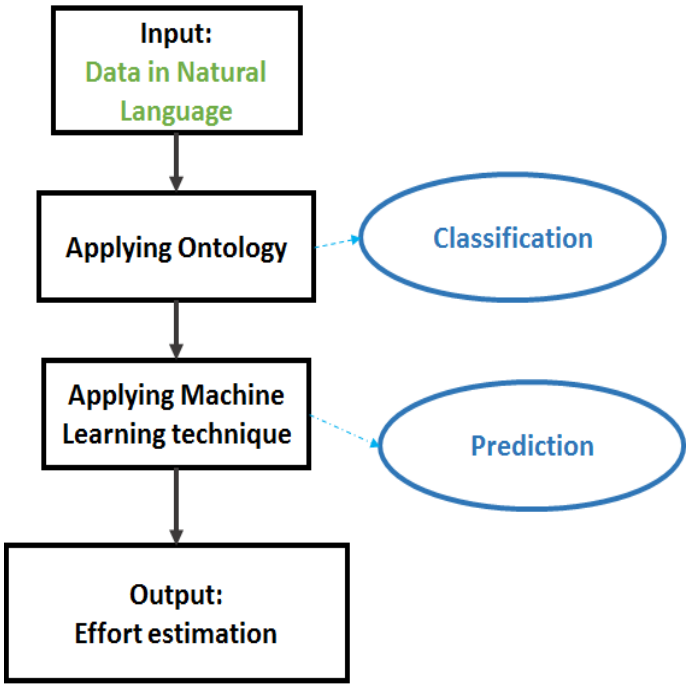

Figure 3.1 – Research Work Process Overview

Accordingly, in this step, we collect a set of data from two types of databases. The first database contains requirements for software projects including ER described in natural language and collected from use case diagrams, class diagrams, and project tutorials from previously developed real projects in the software industry. The second database contains ER collected from customers' reviews in PROMISE[1]. These sources provide the system contextual requirements including system purpose, system scope, and system overview. Our research study takes into account the ER as an input, (1) identifies its types (add, modify or delete), (2) measures the actual effort corresponding to each ER classified as FC derived from experts' opinions, uses the outputs of (1) and (2) as an input to construct an SEEE based ML techniques.

3.4 Ontology based-Semantic Classification

As shown in Figure 3.2, this phase proposed an OCM where we focused on the impact of semantic classification for improving the performance of ML experiment results.

An ontology consists of the following elements: (i) a set of concepts; (ii) a set of relations describing concept hierarchy or taxonomy; (iii) a set of relations linking concepts non-taxonomic; and (iv) a set of axioms, usually expressed in a formal language—for a

1. http://promise.site.uottawa.ca/SERepository/datasets-page.html

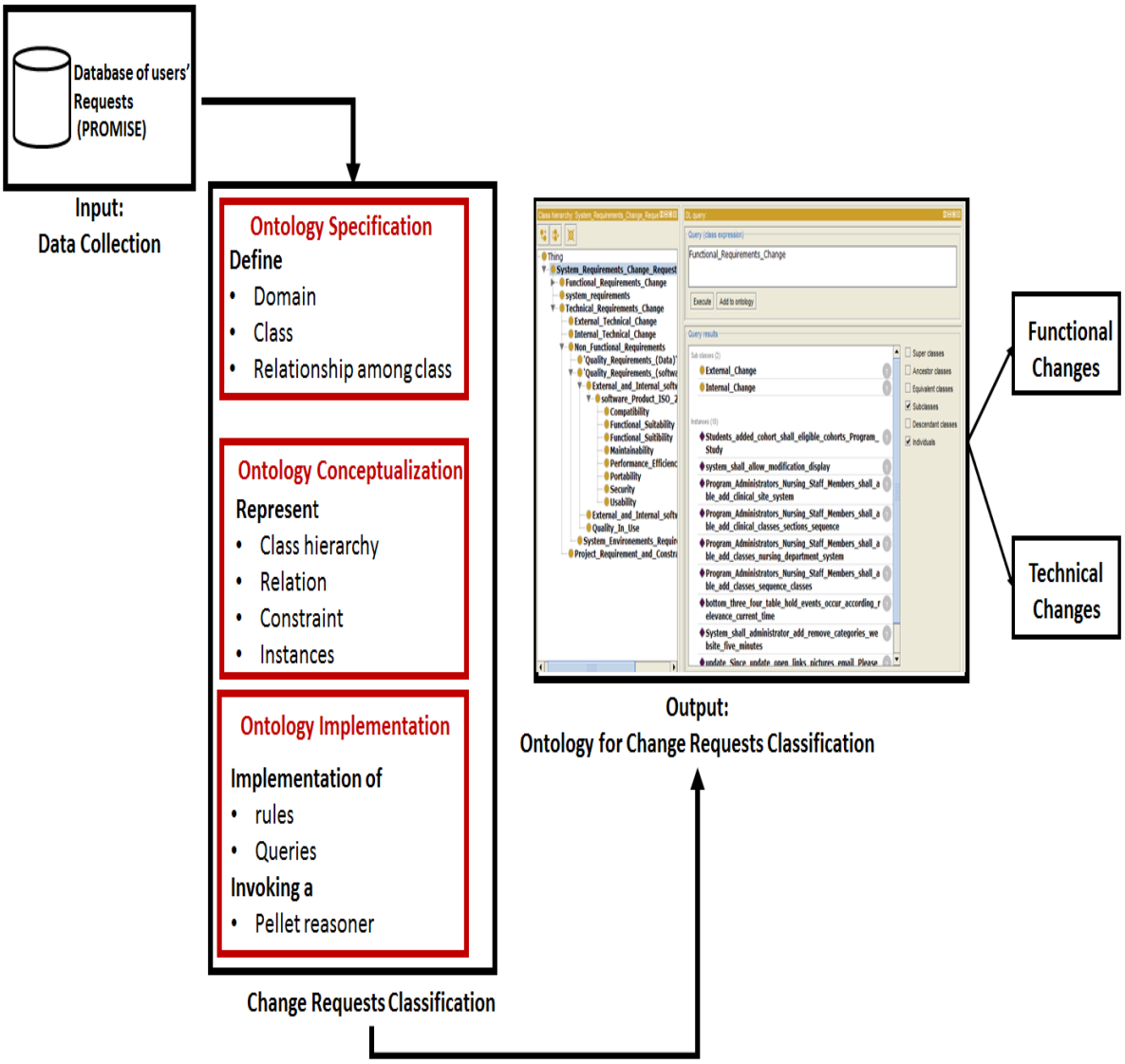

Figure 3.2 – Semantic Classification with Ontology

formal definition of ontology [40]. The main goal of our proposed ontology is to semantic classify the gathered ER into FC or TC. TC is further classified into one of the eight ISO 25010 quality characteristics and Project Requirements and Constraints.

The Pellet Reasoner and a set of DL queries are implemented to accomplish data cleaning based on the knowledge expressed in the ontology domain. The Pellet Reasoner uses ontology constraints to consistently detect data inconsistencies. The reasoners' inconsistencies are then detected and corrected by a SPARQL query. Therefore, queries are used to explore the data and build constraints for detecting and correcting inconsisten-

cies. Thus, we carry out the ontology development through three main steps: Ontology Specification, Ontology Conceptualization, and Ontology Implementation.

3.4.1 Ontology Specification

This step usually includes analyzing concepts to determine whether they are classes or entities, as well as the relationship between classes. This is to identify their categories (as either FC or TC). The classification of ER is intended to identify the type of enhancement to be made, the actions to be taken and the appropriate service to be implemented. Table 3.1 lists the main classes of our proposed ontology. Table 3.2 depicts the interrelationships between classes (domain/range).

Table 3.1 – Ontology class specifications

Class	**Description**
System ERs	Customers ER from PROMISE repository.
FC	Customers ERs are changes that affect FUR
TC	Customers ERs are changes that affect either NFR or PRC
System ER Domain	Enhancement communicators (internal or external). Examples of some of the internal change communicators are business analysts, development team, designers, testers, etc. Examples of some of the external communicators are users, customers, managers, Product owner, etc.
Internal TC	TC derived from Internal stakeholders
External TC	TC derived from External stakeholders
Internal FC	FC derived from Internal stakeholders
External FC	FC derived from External stakeholders
Internal FC Modification	Internal FC to be modified
Internal TC Modification	Internal FC to be modified
Delete in Internal FC	Internal FC to be deleted
Delete in Internal TC	Internal TC to be deleted
Add in Internal TC	Internal TC to be created
Add in Internal FC	Internal FC to be created
External FC Modification	External FC to be modified
External TC Modification	External TC to be modified
Delete in External FC	External FC to be created
Delete in External TC	External TC to be created

Add in External TC	External TC to be created
Add in External FC	External FC to be created

Table 3.2 – Ontology Inter-relationship description

Inter-relationship among classes	Domain	Range
is composed of	System ER	FC, TC
Decomposed in	TC	External TC, Internal TC.
Decomposed in	FC	External FC, Internal FC
Decomposed in	External FC	Modify in External FC, Delete in External FC, Add in External FC
Decomposed in	Internal FC	Modify in Internal FC, Delete in Internal FC, Add in Internal FC
Decomposed in	Internal TC	Modify in Internal TC, Delete in Internal TC, Add in Internal TC
Decomposed in	External TC	Modify in External TC, Delete in External TC, Add in External TC
Is Equivalent to	TC	Non-FC, Projects ER and constraint

3.4.2 Ontology Conceptualization

Using the Protégé 4.3 ontology editor, we create an ontology-based on the enumerated concepts in table 3.1 and table 3.2. The conceptual model entails a set of domain concepts and their relationships. Of course, concepts such as class, attributes, objects property, data property, and their relationships must be defined. Figure 3.3 depicts the various classes of the proposed ontology model:

— The main class is named “System Requirements Enhancement Request".

— The classes named “Functional Enhancement Request" and “Technical Enhancement Request" are subclasses of the class “System Requirements Enhancement Request".

— The classes named “Non-Functional Enhancement Request" and "Project Enhancement Requirement and Constraints" are subclasses of the class “Technical Enhancement Request".

— The classes named “External Functional Enhancement Request" and "Internal Functional Enhancement Request" are subclasses of the class “Functional Enhancement Request".

— The classes named “External Technical Enhancement Request" and "Internal Technical Enhancement Request" are sub-classes of the class “Technical Enhancement Request".

— The classes named “Modify in External Technical Enhancement Request", "Delete in External Technical Enhancement Request" and "Add in External Technical Enhancement Request" are sub-classes of the class “External Technical Enhancement Request".

— The classes named “Modify in Internal Technical Enhancement Request", "Delete in Internal Technical Enhancement Request" and "Add in Internal Technical Enhancement Request" are sub-classes of the class “Internal Technical Enhancement Request".

— The classes named “Modify in Internal Functional Enhancement Request", "Delete in Internal Functional Enhancement Request" and "Add in Internal Functional Enhancement Request" are sub-classes of the class “Internal Functional Enhancement Request".

— The classes named “Modify in External Functional Enhancement Request", "Delete in External Functional Enhancement Request" and "Add in External Functional Enhancement Request" are sub-classes of the class “External Functional Enhancement Request".

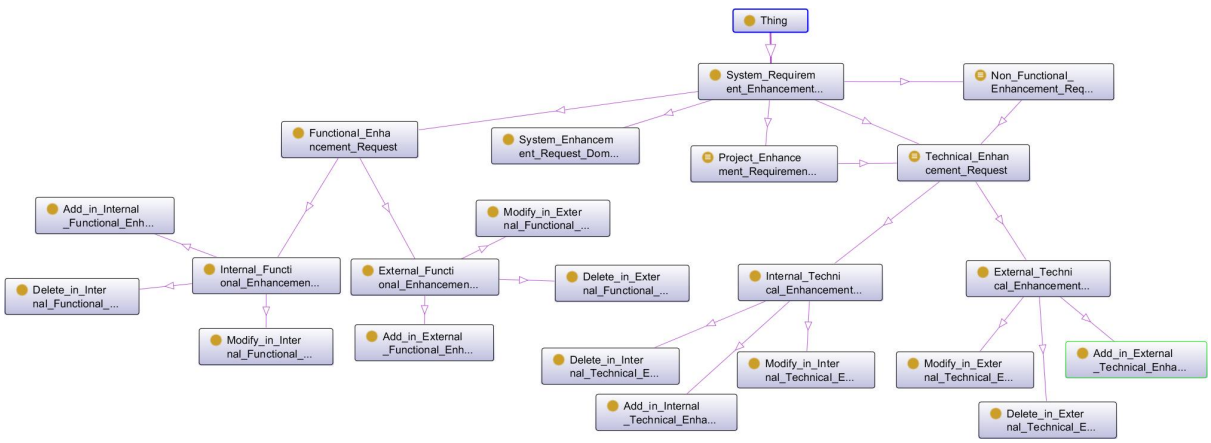

Figure 3.3 – Ontology-based Semantic Classification Model of ER

3.4.2.1 Populating Ontology with FC

This step includes populating the ontology using the PROMISE repository (an example of Ontology's class and its corresponding customer's review (*i.e.,* Customer ER) is listed in appendix 5.7). Ontology population is accomplished by creating an instance of each class and providing links based on the inter-relationships between classes (domain/range). We chose one software requirements specification document with a total of 832 non-functional requirements (NFR) and 93 functional requirements (FUR).

Manipulation of instances is a critical step in our ontology model. There are many approaches used by ontology management systems: OWL schema[2] and object-oriented development[3]. We used OWL schema and Jena to populate automatically our ontology with ER (*i.e.,* users' reviews) derived from the PROMISE repository. These requests are conducted on implemented software and previous software development projects. The collected data (ER) is used to complete the task of the Ontology population.

3.4.2.2 Populating Ontology with Enhancement effort derived from Expert judgment approach

In this step, we employ Expert Judgment to identify each FC of an ER with its corresponding effort. When a company does not have any historical data in its database [106], the Expert Judgment will be useful. Here, the estimates have been updated and calibrated based on the organization's past expert experience. In our study, as shown in Table 3.3, we asked seven estimators with at least three years of experience. As

Years of software Experience	Numbers of estimators
3 years	2
4 years	3
10 years	2

Table 3.3 – Expert Judgement Experience

shown in Table 3.4, each ER is associated with its corresponding estimated effort based on Expert judgments. The outcomes are based on information provided by estimators such as the product's previous history (*i.e.,* previous changes), the functional size of the new function, similar previous implementations, the amount of new code, the deadline pressure, and the product's expected lifetime.

2. https://www.w3.org/OWL/
3. https://www.w3.org/TR/sw-oosd-primer/

Project	**ER**	**ER Type**	**Estimated effort**
Mobile game application	Personalize Information Add notification For localization	Modify Add	3h/per 7h/per
SOCOMENIN	Personalize Information Personalize Information for invoice	Modify Add	3h/per 3h/per

Table 3.4 – Example: Enhancement Effort Estimation based on Expert Judgement

3.4.3 Ontology Implementation

In our contribution, we proposed a set of Semantic Web Rules Language (SWRL)[4] and Description Logic Query (DL query)[5] based on the definition of the ISO 25010 software quality characteristics and the description of the users' reviews within PROMISE repository. The first step is to choose a set of terms that are relevant to the domain, which can be done manually or automatically. It is associated with the recognition of subject, object, and relationships. The linguistic motivation for this identification is that the meaning of common terms is hidden in their relationships with other terms. These terms are organized into classes using the Protégé 4 editor. And then, converted into a set of rules. These concepts can be used to quickly identify the required Instances. Table 3.5 lists our proposed classes and their corresponding key concepts (*i.e.,* users' reviews from PROMISE repository). The following steps are required for our proposed ontology solution: (A) Implementation of the rules, (B) Queries about the knowledge using DL query, (C) Invoke a pellet reasoner that builds a knowledge-based ontology domain, and (D) Ontology output and solution discussion.

4. https://www.w3.org/Submission/SWRL/
5. https://github.com/stardog-union/DLQuery

Table 3.5 – Categorizing the Customer's ER

Classes	Key concepts
Functional Enhancement	must contain, play, view, select, manage, operate
Technical Enhancement	maintain, produce, corporate, load, upload, synchronize, appearance, transaction
External-Enhancement	Cannot, please, doesn't, none, problems, no access, bugs, stopped working
Internal-Enhancement	Product must, product shall, administrators must, system must, application parameters, change
Create	Add, build, design, generate, organize, set up, produce,
Delete	Delete, black out, destroy, exclude, cut out, eliminate, cancel.
Modify	adapt, revise, modify, correct, rework, repair

A. Implementation of rules

Following the conceptualization step, we propose a set of rules for our Ontology. With the explanation of the first rule, Table 3.6 includes four columns: the name of class (class), the attribute (Data property), the instances (individuals) and the result (output). One line including the proposed rules. This table is applicable to the following rules (R1, R2, and R3).

— R1: SystemRequirementsChangeRequest(?F), Change Value(?F, ?V), contains(?V, "stopped played ") — External Change(?F)

Table 3.6 – Rule 1

class	Data property	individuals	output
System Requirements Change Request	Change_Value	stopped played	External Enhancement

R1 is used to determine the source of ER. An ER may come from multiple stakeholders, each with its own set of priorities. The reviews of users are classified as either external ER or internal ER. External ER is related to the users' perspectives. For example, "I loved the app, but since I installed iOS7 and updated it, it no longer works; please fix it". External ER help to identify and define internal ER (from the developer's perspective).

It is critical to distinguish internal ER from external ER to better prioritize ER and determine the role of stakeholders.

— R2: SystemRequirementsChangeRequest(?F), Change Value(?F,?V), contains(?V, “events”), contains(?V, “update”)- Functional Change(?F)

R2 is used to identify FE that affect functional requirements (FUR).

— R3: SystemRequirementsChangeRequest(?F), Change Value(?F,?V), contains(?V, "resources"), contains(?V, "update")- Technical Change(?F)

R3 is used to identify TC that affect quality requirements including both (NFR and PRC).

B. Queries about the knowledge using DL query

The reasoning features of the proposed rules specified in the DL Query were used to verify and validate the ontology. For searching a classified ontology, DL includes a powerful and simple feature. The query language (class expression) supported by the plugin is built on the Manchester OWL syntax, a user-friendly OWL-DL syntax. Because of its expressiveness and power, we selected OWL to represent our ontology-based approach. Figure 3.4 illustrates a query for FC. As a result, a list of inferred individuals related to

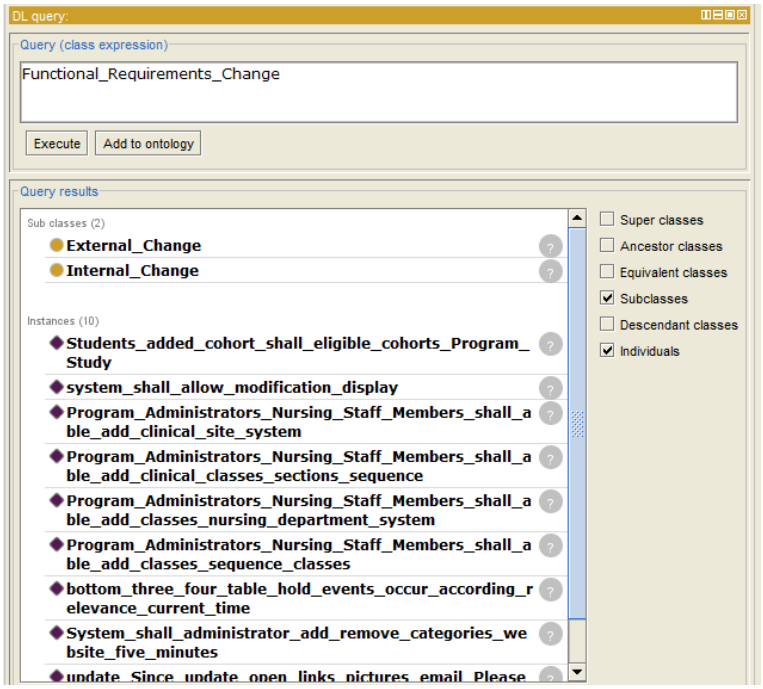

Figure 3.4 – Functional Change DL Rule result

FC is considered (as explained earlier in rule R2).

C. Invoking a pellet reasoner that builds a knowledge-based ontology domain

To illustrate the use of our ontology, we used the DL Query tab in conjunction with the reasoner pellet to retrieve all of the corresponding class instances. The reason why ontology is used is for reasoning. In Protégé, there are two types of models: asserted and inferred (Figure 3.5). Test results are displayed in the form of inferred individuals.

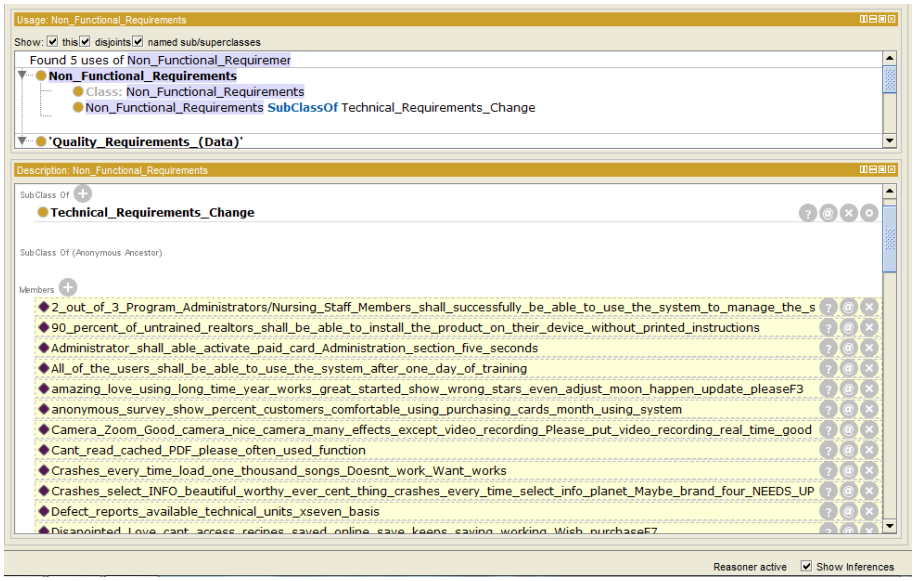

Figure 3.5 – Ontology with Reasoner

The ontology reasoner may discover significant connections and implications among the various components (concepts, relations, and properties) used to construct our ontology.

D. Ontology output and solution discussion

For a more appropriate response to an ER, we identify three types of requirements ER, which are categorized by FC and TC: Addition (), Deletion (), and Modification ().

— If the ER is an "Addition of a new requirement," it will produce more information.

— If the ER is a "Deletion of an existing requirement," deletion output will be provided.

— If the ER is for a "Modification of an Existing Requirement," the sources of the modification types must be identified (Refine or Replace).

As an example of rule R1, "the system must be able to display student information," and ER means "the system must be able to display student information: full name and grade level."

To make ontology information available as background knowledge for ML techniques, it is required to use a statistic and semantic-based approach to process textual data.

One of the most widely used techniques to process textual data is TF-IDF [107]. A common phrase (or non-unique phrase) that appears frequently in a document may not be important if it also appears in many other documents. To consider the uniqueness of key phrases, the Term Frequency and Inverse Document Frequency (TF-IDF) approach is recommended [107]. As its name implies, TF-IDF vectorizes/scores a word by multiplying the word's Term Frequency (TF) with the Inverse Document Frequency (IDF), where:

— Term Frequency (TF) is the number of times a term or word appears in a document in comparison to the total number of words in the document.

— Inverse Document Frequency: IDF of a term reflects the proportion of documents in the corpus that contain the term.

3.5 Constructing Prediction Models and Evaluation

In this section, the chosen ML techniques are trained and tested for a variety of experiments. Our prediction model was created using the Google Colaboratory Python programming language. Google Colaboratory, also known as Google Colab, is an open-source service that Google offers to anyone with a Gmail account. Google Colab[6] provides GPU for research to the people who do not have enough resources or cannot afford one.

Six software development projects were used to test our proposed SEEE model. In this section, two types of experiments are carried out. In the first set of experiments, the dataset is randomly divided into two subsets, a training set, and a test set. The second set of experiments is carried out using the widely used tenfold cross-validation method.

3.5.1 Simple split

For the first set of experiments, we used the classic approach that is to do a simple 70%-30%. We frequently divide our data into two sets: training and validation/test. The training set is used to train the model, and the validation/test set is used to validate data that it has never seen before. The results of our built SEEE models are compared to a widely used set of evaluation metrics such as mean square error (MSE), root mean square error (RMSE), and mean absolute error (MAE) as demonstrated in Table 3.7. All error metrics indicate quite values. It is evident from the results (see Figure 3.6) that the RFR method delivers the best performance when compared with the other three MLRM.

6. https://colab.research.google.com/notebooks/welcome.ipynb

Method/parameters	**MAE**	**MSE**	**RMSE**
ABReg	0.450	0.263	0.513
GBReg	0.108	0.070	0.265
RFR	0.040	0.045	0.215
LinearSVR	0.100	0.479	0.190

Table 3.7 – Prediction analysis using MAE, MSE and RMSE

It shows evidence of its powerful predictive capacity. In addition, the GBReg presents good results. However, the bad results are presented by the ABReg method.

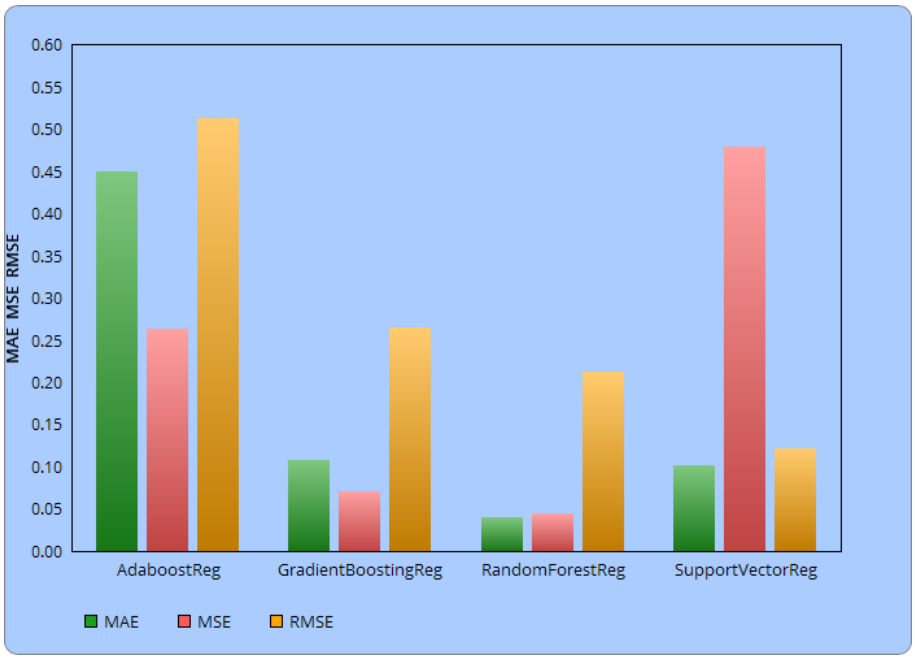

Figure 3.6 – Prediction analysis using MAE, MSE and RMSE

3.5.2 Cross validation

In cross-validation, we do more than one split. We used 10-fold cross-validation in our model. It has been used in a variety of experiments to assess the performance of four MLRM. We can obtain more metrics and draw important conclusions about our algorithm and data by employing Cross-Validation. One of the most obvious reasons for performing cross-validation is that it allows us to make better use of our data and provides us with a lot more information about the performance of our algorithms. Table 3.8 illustrates the results of using these two metrics (accuracy/prediction) for Cross-validation method.

Techniques	ABReg	GBReg	RFR	LinearSVR
Accuracy$_{If}$	0.8203498	0.8634263	0.8836048	0.7168095
Cross-Predicted(K-Fold)	0.8066543	0.8496022	0.8175633	0.8011495

Table 3.8 – 10-Fold Cross Validation accuracy

3.6 Discussion and Comparison

The use of Ontology with ML techniques improves the accuracy of the SEEE. The advantage of using Ontology is its ability to categorize ER from simple classification to semantic classification. As stated in section 3.4, the data quality (models input) was improved using an ontology-based classification approach. Furthermore, this approach necessitates that the request for enhancements is clearly and completely defined.

From Figure 3.6, we can confirm the validity of ML techniques as an alternative to the traditional estimation approaches (such as Expert Judgement). This estimation will help experts make decisions on whether to accept, defer or deny an ER.

All the participants in the software project recognize the importance of developing an accurate SEEE since it plays a key role in the success of the software project planning and management. The main idea of our research is to present an effective model for SEEE. We focused on the importance of semantic classification, and therefore we investigate their impacts for a good prediction.

The proposed SEEE model is quite effective and demonstrates the minimum MAE of 0.040 using a real dataset project. After learning, the ML techniques were able to produce reasonably accurate predictions. This study and experiment were done to evaluate four ML methods ABR, GBR, LinearSVR, and RFR. These methods are used to predict effort for an ER that occurs in the software development project. The RFR is established to be the more effective algorithm when compared with the other three methods.

We used two methods for evaluation. The first method used a simple split. The second method used 10-fold cross-validation. In addition, we used the R2 score for cross-validation. Based on the obtained results, we noted small values of MSE, MRE, and RMSE when applying a simple method alone. It demonstrates the effectiveness of the used methods. Good accuracy of 90% is obtained when the 10-fold cross-validation with an R2 score in the best scenario is used. To identify the effective determinants of the

SEEE, we calculate the importance of each feature. Furthermore, a model that uses Actual effort and ER features delivers superior performance as compared to a model that uses all proposed features. In addition, a model that incorporates a combination of 10-fold cross-validation and R2 score demonstrates better performance when compared with a model that uses a simple split (train/test). As a result, we can conclude that the RFReg and GBReg techniques improve estimate accuracy.

3.7 Conclusion

In this chapter, we investigated the problem of providing accurate SEEE. We designed an OCM for classifying ER as FC or TC. We populate the OCM with real-world projects from the software industry where we associate each ER with its corresponding effort using Expert Judgement. The output of the OCM (dataset) is used as input to make SEEE using ML techniques. Four ML techniques (ABR, GBR, LinearSVR, and RFR.) are used for estimation. The RFR gives a more accurate SEEE compared to the others selected ML techniques. In the next chapter, we discuss the effectiveness of using the COSMIC FSM method as a primary independent variable for improving the accuracy of SEEE.

Chapter 4

Towards the use of COSMIC FSM method for improving SEEE within the context of classical and Agile projects

Contents

In Short

One of the most important factors influencing effort estimation is the software size. Several FSM methods have been proposed to be used as input for estimating development/maintenance effort. There is only one second-generation FSM method, the COSMIC, and four first-generation FSM methods, including the IFPUG. There are two main contributions to be investigated in this chapter: (1) investigate the effectiveness of the first and second FSM generations for sizing functional changes and examining their impact on predicting software enhancement effort in traditional and agile projects, (2) the applicability of COSMIC sizing as an independent variable in scrum projects. The use of the CFS algorithm in conjunction with the Support Vector Regression (SVR) model.

4.1 Introduction

The majority of the well-known estimation models, techniques, and tools use the software size as an input for their estimations [2]. As we mentioned in Chapter 1, the software size can be expressed in SLOC or function points units. Effort or cost estimations using functional size measures are gaining more and more attention since the software functional size can be generated at any phase of the SLC compared to the SLOC. Moreover, the obtained measurement results using FSM methods are independent of any technical criteria. These two advantages motivated the researchers to use the FSM methods for more accurate effort estimation.

Despite a large number of proposals interested in finding accurate estimates, there is no clear evidence in determining which model is the best for estimating enhancement effort (the factors to be considered when choosing one model over others). As a result, it is critical to identify and assess the inputs to estimation models. Customer dissatisfaction and project failure are the results of inaccurate estimates. On the other hand, accurate estimates reduce uncertainty and facilitate more effective software project management [6].

In this context, software size is widely recognized as a major cost driver for the effort and cost required for software projects. Researchers believe that the size variables are closely related to the required effort [23]. It is important to note that functional size is the only standardized way to measure the software size [106]. As sensitivity to changes in functional size has a greater impact on project effort [94], knowing the functional size of the software to be developed/redeveloped or maintained is useful. Software size can be used for many purposes such as: improve organizational performance, estimate the effort of new development, estimate the enhancement effort, and control software development, and so on [24, 25]. Several studies used the IFPUG and COSMIC sizing to predict the effort in software development project. However, only a few research studies investigated the efficiency and the impact of using FSM on predicting the effort is software enhancement project. On the other hand, predicting effort for managing scrum projects is performed differently compared to the traditional ones [55]. There are many prediction techniques such as Expert opinion, Planning Poker, and a few more [55]. A Survey of five studies conducted on Basic Estimation techniques showed that the most popular effort prediction technique used in Scrum projects is the Planning Poker (PP) technique [55]. The basis of the PP technique is practitioners' opinions that are expressed in terms

of Story Points. In practice, it is used for predicting the effort required to complete software requirements or User Stories. In scrum, enhancements that are categorized as Functional User Requirements are represented in the form of US at a high level of detail. The US is a brief description of the user's request [108]. Besides the need of PP, several international standards provide well-documented methods for measuring or approximating the US functional size, such as the COSMIC FSM method. Indeed, there is a growing body of work on the use of the CFP [25] for prediction and performance measurement of software development projects, which can be adapted for predicting agile software enhancement effort as well.

In this chapter, we make two contributions. The first contribution is for making comparison between the most popular FSM methods (IFPUG and COSMIC) when they are used as independent variables in predicting SEEE in the context traditional project. The second contribution is to investigate the application of the best SEEE model (the results of the first contribution) in the Agile (scrum) enhancement project). For both contributions, we used the CFS algorithm, SVR model and ISBSG dataset.

The rest of this chapter is organized as follows: Section 4.2 describes the first contribution. Section 4.3 describes the second contribution. Section 4.4 provides the results and discussion. Section 4.5 presents the conclusion.

4.2 On the use of FSM methods for more accurate Prediction in the traditional software Enhancement projects

In this section, we carry out two experiments setting up two regression-based models: one using the IFPUG and the other using the COSMIC. Then, we compare their prediction accuracy to determine whether the COSMIC method provides more accurate results than the IFPUG for the SEEE. We use a training dataset to predict the total effort for the software enhancement projects in man-hours. Our research methodology is depicted in Figure 4.1.

4.2.1 Data Preprocessing

The ISBSG Release 12 dataset was used to train and test the prediction model [23]. The ISBSG dataset is widely used for estimating software projects [23]. It maintains a repository of finished software projects, including new, improved, and redeveloped ver-

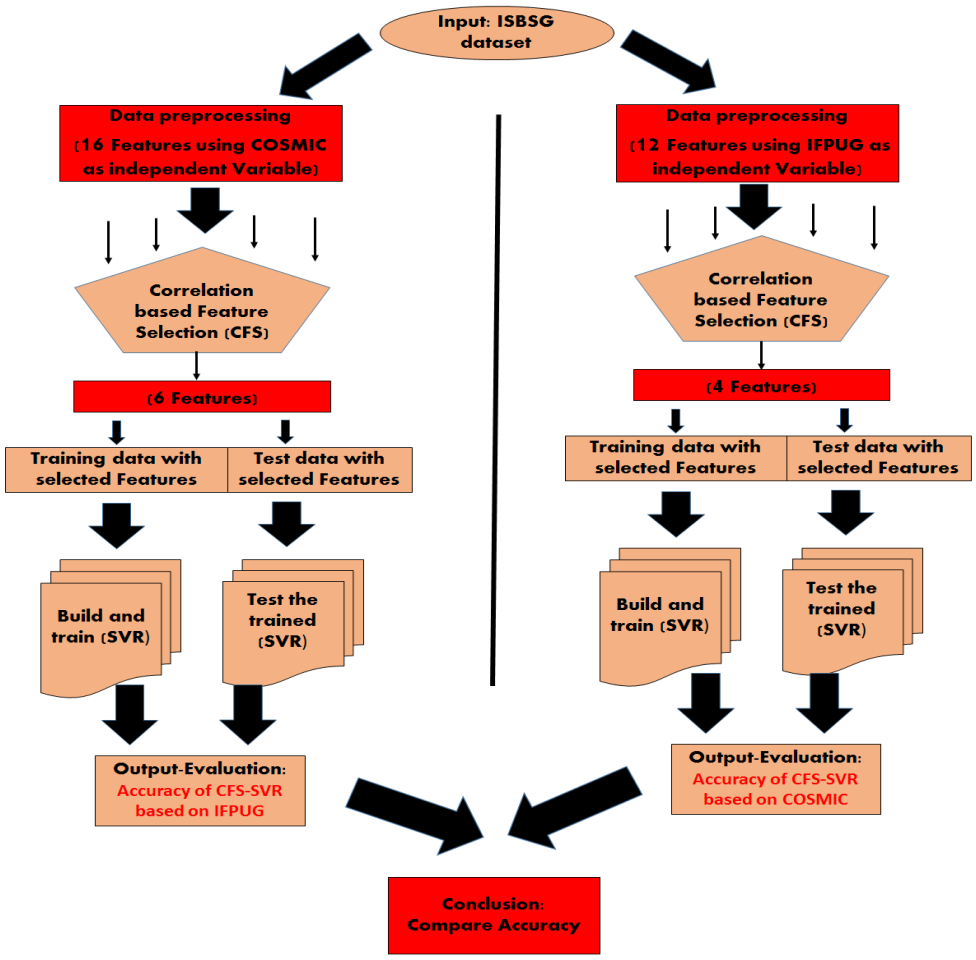

Figure 4.1 – Research method design

sions. The ISBSG dataset contains a variety of information about finished software projects, including methods, tools, and methodologies, as well as process and product data, that may be used for benchmarking, monitoring, quality control, and performance management [23]. The ISBSG dataset is the largest available for effort estimating research, and it has already been used in several publications. It has been thoroughly examined for its suitability in the construction of effort estimation models, including the effects of outliers and missing data [109]. We selected the data regarding « enhancement » as the « development type » where « count approaches » were IFPUG and COSMIC methods. In addition, we consider only data with soundness and high level of integrity (*i.e.*, records having « Data Quality Rating » of « A » or « B »). Table 4.1 shows the data fields, the values assigned to them in this study, and the number of projects. Following the preprocessing phase, we chose 17 attributes/features for COSMIC data and 13 attributes/features for IFPUG data.

Table 4.1 – First Data of software enhancement projects from the ISBSG dataset

ISBSG data field	**Selected Values for COSMIC _dataset**	**Selected Values for IFPUG _dataset**	**Discarded Values**	**Projects for COSMIC _dataset**	**Projects for IFPUG _dataset**
Data quality rating	A, B	A,B	C,D	4000	4000
Count Approach	COSMIC	IFPUG	NESMA, FISMA, etc.	449	3104
development Type	Enhancement	Enhancement	New development and Redevelopment	302	1084

4.2.2 Using the CFS algorithm

Following the selection of a project with high-quality data (after the preprocessing phase), we propose using the CFS algorithm to select the features that are relevant for predicting effort for a software enhancement project. That is, after building the CFS algorithm, we determine which features appear in the optimal set of features globally and consistently. The Pearson's Correlation Coefficient algorithm is one of the most commonly used algorithms, which is used to filter the data in this step [110]. The Pearson correlation coefficient is a single number that expresses the strength and direction of a linear relationship between two continuous variables. The range of possible values is -1 to +1, with 0 indicating no correlation, 1 indicating total positive correlation, and - 1 indicating whole negative correlation [110]. We will use the Pearson correlation heat map in our example. Each attribute is sorted according to the p correlation score (See Equation 4.2).

$$p = \frac{cov(Xi,Y)}{\sqrt{var(Xi)var(Y)}} \tag{4.1}$$

Where var(Xi) and cov(Xi,Y) represent the variance of feature Xi and the covariance between a feature Xi and the target class Y, respectively.

4.2.2.1 Computation of Score P for the selected features from COSMIC_dataset using Pearson's correlation coefficient

Following the preprocessing phase, we chose 17 attributes, 16 of which are independent variables and one of which is a dependent variable (NormalizedWorkEffort). Pearson's

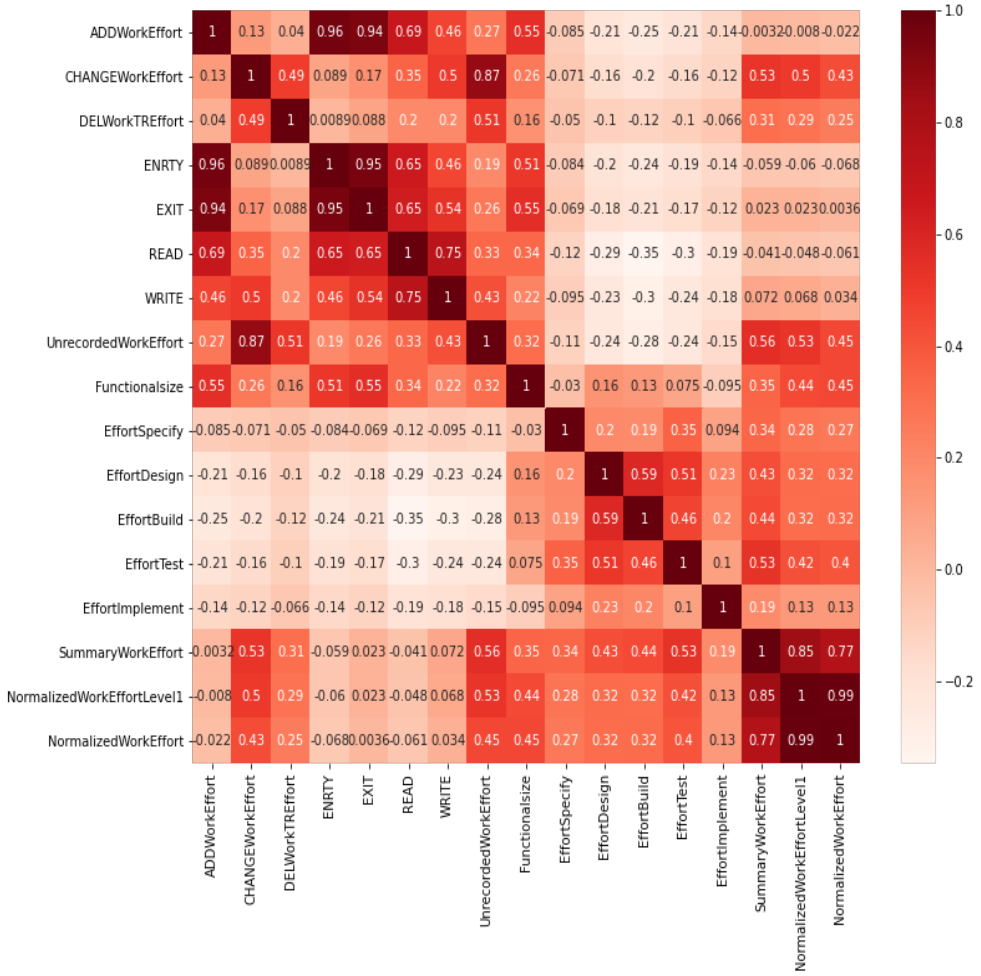

Figure 4.2 – Pearson's correlation heat map for COSMIC_dataset

correlation coefficient is used to generate a list of attributes that are sorted according to their degree of correlation to the module class (*i.e.,* NormalizedWorkEffort). In our case, only features with a correlation greater than 0.4 (taking absolute value into account) are chosen for the output variable. The use of CFS algorithm selects 37.5 percent (6 out of 16 features) (see Figure 4.2).

COSMIC sizing is an effective FSM method for determining the functional size of an enhancement (*i.e.,* functional change size) that has been identified within the enhancement projects. Table 4.2 illustrates the selected features and their corresponding correlation

Table 4.2 – Selected Feature correlation when using COSMIC_dataset

Selected Features	**Value - Round(Correlation target)**
CHANGEWorkEffort	0.4
UnrecordedWorkEffort	0.5
FunctionalSize	0.5
EffortTest	0.4
SummaryWorkEffort	0.8
NormalizedWorkEffortLevel1	1

coefficients (score greater than 0.4) between functional change size (FunctionalSize) and enhancement effort, estimated using COSMIC_dataset (NormalizedWorkEffort). When compared to other features, the correlation coefficient value of 0.5 indicates an adequate correlation of functional change size with enhancement effort (such as CHANGEWorkEffort and UnrecordedWorkEffort). As a result, the size of the functional change is chosen as the key independent variable.

4.2.2.2 Computation of Score P for the selected features from IFPUG_dataset using Pearson's correlation coefficient

Following the preprocessing phase, we chose a total of 13 features, 12 of which are independent variables and one of which is a dependent variable (NormalizedWorkEffort). Only features with a correlation greater than 0.4 (taking into account absolute value) are chosen for the output variable. The CFS algorithm selects 33.3 percent of the IFPUG dataset (4 out of the 12 features) (see Fig. 4.3). The correlation coefficient between

Table 4.3 – Selected Feature correlation when using IFPUG_dataset

Selected Features	**Value -Round(Correlation target)**
EffortBuilt	0.8
EffortTest	0.9
SummaryWorkEffort	1
NormalizedWorkEffortLevel1	1

Functional change size and enhancement effort when using data from the ISBSG dataset is 0.1 (see Fig. 4.3). When compared to other features, this number suggests a relatively poor correlation between functional change size and enhancement effort (such as

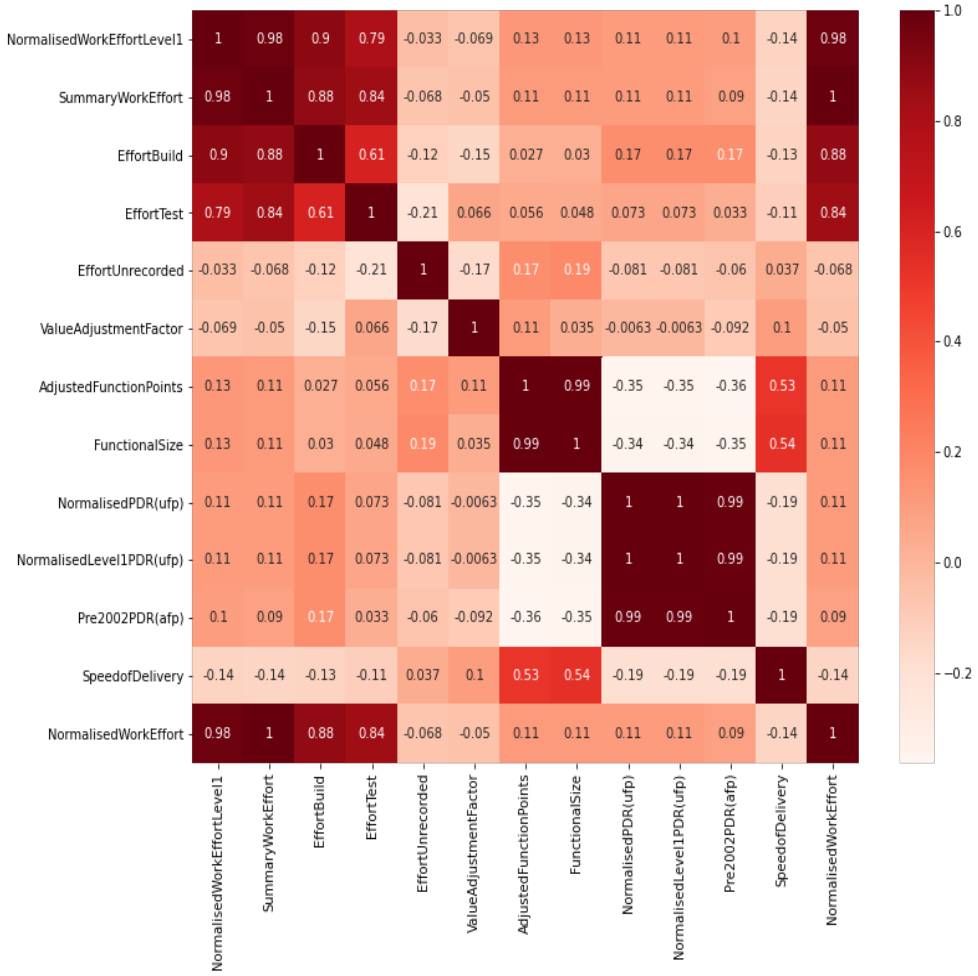

Figure 4.3 – Pearson's correlation heat map for IFPUG_dataset

EffortBuilt and EffortTest). The findings suggest that using a functional change size derived from IFPUG sizing as the key independent variable may not give an appropriate estimate of enhancement effort. On the other hand, a close examination of the Pearson's correlation algorithm's feature list (see Table 4.3) reveals that the EffortTest, Effort-Built, SummaryWorkEffort, and NormalizedWorkEffortLevel1 were the most important features in predicting effort.

4.2.3 Constructing SEEE Models

This section investigated the use of the SVR method through a series of experiments. The traditional method for conducting experiments is to divide the sample into two parts: 70 percent and 30 percent. The ISBSG dataset is divided into two parts: training and validation/test. The « test_size » argument specifies the size of the split. In our model,

30 percent of the dataset is allocated to the test set, while 70 percent is allocated to the training set. The training set is used to train the model, and the validation/test set is used to validate it on new data. Following that, the CFS algorithm and the validation test prediction of the SVR method were carried out using Google Colab. Table 4.4 lists the predefined range of parameters values of the SVR method.

Table 4.4 – Parameters values for Grid Search

ML Technique	**Parameters**
SVR	Kernel=Linear; Complexity=1, 2; epsilon=0.2; Deviation=0.001, 0.0001

4.2.4 Empirical Results

This section assesses the prediction performance of the SVR used in this study, which includes two types of experiments using the CFS algorithm. We used a variety of evaluation metrics [111] to assess the accuracy of the SEEE models, including root mean square error (RMSE) and mean absolute error (MAE). We also used the Standardized Accuracy (SA) metric based on the MAE as described in [111].

4.2.4.1 Performance Assessment when using COSMIC sizing

The CFS algorithm selects features for SVR to be trained and tested. When the Functional Change Size is utilized as the independent variable, using CFS algorithms with the SVR approach can lead to a more accurate SEEE (see Table 4.5). Using the SVR approach, error measurements (such as MAE and RMSE) indicate quite values (MAE=0.0382; RMSE=0.1082).

Table 4.5 – Prediction analysis using MAE, RMSE and SA for COSMIC_dataset

Method/ parameters	**MAE**	**RMSE**	**SA(%)**
SVR	0.0382	0.1082	98%

4.2.4.2 Performance Assessment when using IFPUG sizing

The CFS algorithm selects features for SVR training and testing. The selected features (*i.e.,* features selected by CFS and the IFPUG feature) combined with the SVR method provide an accurate SEEE (see Table 4.6). Error metrics (such as MAE and RMSE) show relatively low values (MAE=0.0734; RMSE=0.1950).

Table 4.6 – Prediction analysis using MAE, RMSE and SA for IFPUG_dataset

Method/ parameters	**MAE**	**RMSE**	**SA(%)**
SVR	0.0734	0.1950	98%

4.2.5 Discussion and Comparison

Using the ISBSG dataset, we notice that the prediction accuracy increases when the COSMIC method is selected as « count approaches » for « enhancement projects » in comparison with the IFPUG method (see Table 4.5 and 4.6). The key reason for selecting software functional size as an independent variable in our study is that it is highly correlated to software project effort. The sensitivity to the functional change size has a stronger impact on Software project effort [112]. The relevance of each feature is identified using the CFS algorithm in order to discover the effective determinants for SEEE. When software is being maintained, the SVR model is used to predict the effort to implement an enhancement. When compared to the IFPUG FPA method, the COSMIC FSM method is deemed to be the most effective. It is obvious from the results, which show a minimal MAE of 0.0382 and a 98 percent accuracy rate for accurate predictions.

Furthermore, the CFS algorithm results demonstrate the significance of the Functional Change Size feature when employing the COSMIC FSM method. As expected, the CFS algorithm in our case has significantly contributed not only to reducing the number of features required to achieve prediction performance but also to improving such performance. To summarize the findings of this study, software organizations interested in planning and managing software enhancement projects should select the appropriate sizing method based on their objectives and capabilities. As a result, a good measurement program is an investment for project success because it allows for an accurate evaluation of an enhancement and the effort required to complete this enhancement.

4.3 On the use of COSMIC method for more accurate SEEE in Scrum

sThe research process adopted in this contribution is depicted in Fig. 4.4. Our work is divided into four steps: data collection, applying the CFS algorithm, creation of prediction models, and evaluation (empirical results). Each step is described next.

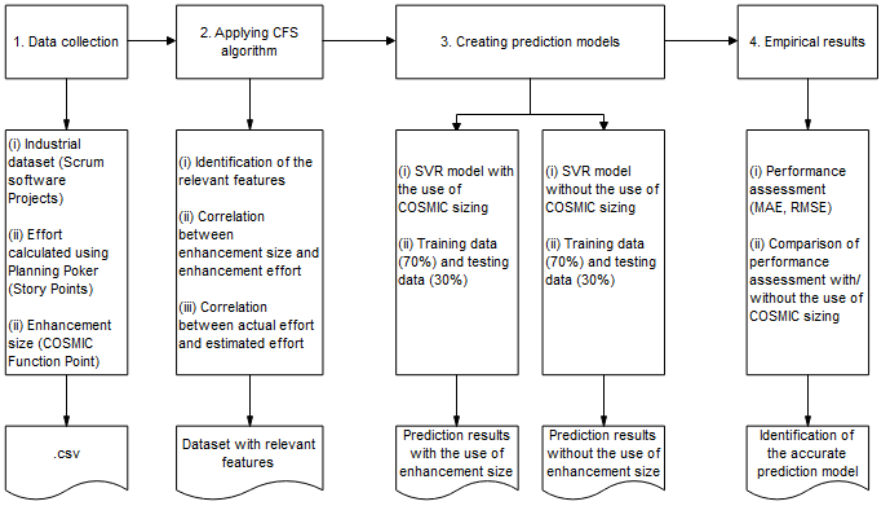

Figure 4.4 – Research work process

4.3.1 Data collection

The data collection in our study is based on real projects developed using the Scrum framework and derived from industry. The dataset is called “Example Application Functional Specification"[1] founded in 2007. It was published by Tony Heap, a Business Analyst on the HMRC API Platform team, who has 25 years of experience in IT[2] (indexed by Google in June 2013). The selected dataset specifies an example of software maintenance Enhancement where the functional specifications can be considered as an enhancement.

4.3.1.1 Effort generated from the application of Planning Poker technique

An enhancement can be requested by one of the stakeholders (*e.g.,* the product owner, the development team, etc.) and described in the form of “US". The US can be quantified and estimated using Planning Poker (expressed in Story Points). It is also sized using

1. https://fr.scribd.com/
2. http://www.its-all-design.com/

COSMIC FSM method [113].

Several factors (Technical/Non-technical factors) can affect the prediction results such as the size of the database (technical factor) [11]. For this reason, several research studies used a single dataset while other research studies used more than one dataset [92]. Some data set attributes still missing in certain data sets [114]. Some other research studies focused on the use of independent variables [102].

Accordingly, the choice of features in the dataset is important for more accurate predictions [41]. The dataset used in our study includes 93 US that is executed by the same team on the same application within a set of eight iterations. The effort devoted to implementing the US is expressed in terms of Story Points. It consists of four worksheets: summary, details, actors, and stats.

1. The “Summary" focuses on the kinds of information used in the Stories such as the status of each US. This means each US status can range from “being elaborated" right through to “gone live". Since our work focus on enhancement, to exclude trivial projects, the following filters are applied: Actual Development Time (man_days), full life cycle effort for a project is greater than 80 man-hours, and "Stats" other than "implement" were excluded.
2. The “Details" provide the detailed information for each US. In the selected dataset the US involves determining: Who will do the US or find it valuable $< Actor >$, What it can be used for $< Goal >$, and Why it is valuable or important <value or expected benefit>. In scrum, the description of a US is most often described as follows:

   As an <Actor>
   I want to <Goal>
   so that <value or expected benefit>
3. The “Actors" specifies all actors (Human users or external systems).
4. The “Stats" show the development progress for individual increments in the form of the “burndown chart". The release burndown chart is presented daily to the Scrum Master to monitor progress toward completion of the increments [115]. When the US is completed, the development team may use the Planning Poker technique to generate the actual effort expressed in terms of story points. Thereafter, the estimated effort will be compared against the actual effort to determine how well the project is progressing.

4.3.1.2 Enhancement Size generated form the application of COSMIC method

The enhancement functional size of the same set of 93 US were measured using the COSMIC FSM method. It was carried out by examining the four worksheets documentation of each US. As described in chapter 1, the COSMIC FSM process includes three phases: the measurement strategy, the mapping of concepts, and the measurement of the identified concepts (see Figure 4.5).

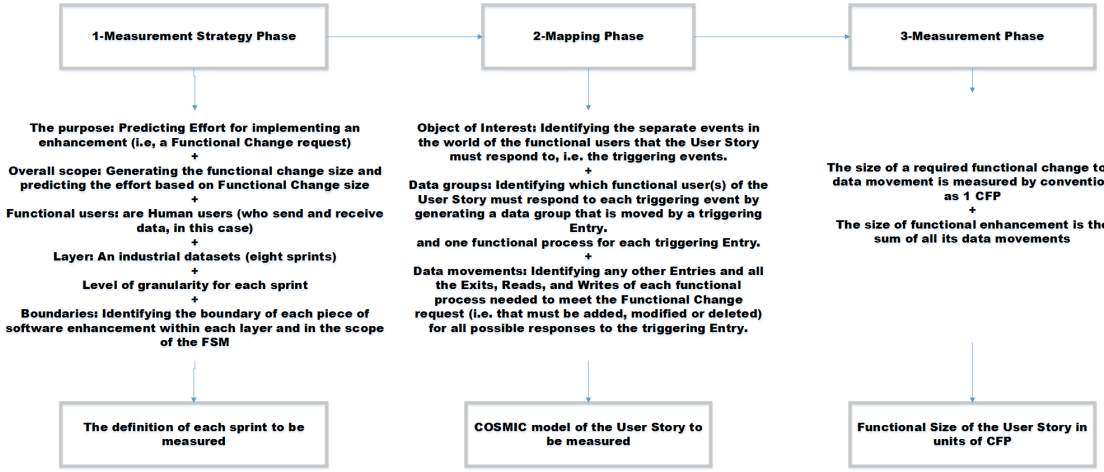

Figure 4.5 – The COSMIC FSM method

4.3.1.2.1 Measurement strategy phase The main parameters that must be identified in this phase are detailed as follows:

— The purpose: Predicting Effort for implementing an enhancement (*i.e.*, FC).

— Overall scope: Generating the functional size of an ER and predicting its corresponding effort.

— Functional users: are human users in this case.

— Layer: An industrial dataset (eight sprints).

— Level of granularity: one level of granularity

4.3.1.2.2 Mapping Phase: US to COSMIC Functional Process (FP) The US exhibits a high-level requirement description. There is no general standard for the US representation [113]. The level of granularity for sizing requirements (or enhancements) in the form of a US must be that of the COSMIC FP [108]. For that reason, the mapping of a US to a COSMIC FP necessitates the identification of the following concepts (see Table 4.7) [116].

Table 4.7 – Mapping of US in sprint 1 with COSMIC Functional Change

US Id	US description			COSMIC Functional Change description	
	As an . . . (actor)	I can . . . (Goal)	Story Points effort	Change Type	FC description
1	Organization User	Add a custom evidence type to an assessment criterion (because the standard evidence types are not appropriate for me)	4	ADD( )	Add Custom Evidence Type
2	XYZ ABC User	Create bulk emails to be sent to users	5	ADD( )	Create Bulk Email
4	Organisation User	Add an improvement action against a specific attainment criterion	5	ADD( )	Add Improvement Action
5	Assessment Completer	Add and amend general comments for my requirement response	3	ADD( )	Add Widget Comments
6	General User	View an organisation's Assessment Report with various enhancements	4	MODIFY( )	View Improved Assessment Report
7	Organisation User	Create, amend and delete users within my organisation user (using the simplified interface)	3	MODIFY( )	Maintain Users (Simplified Interface)
8	XYZ ABC User	Delete a pending bulk email	1	DELETE( )	Delete Bulk Email
9	General User	View Improved Assessment Report - HTML Version	3	ADD( )	Create an HTML version of the assessment report

— <Actor>represents the user of the US referred to as the functional user in COSMIC.

— <Goal>represents the ER of the US referred to as an enhancement or a FP in COSMIC.

Functional Changes are mainly classified into three types: add (new requirements to be created), delete (existing requirements to be deleted), and modify (existing requirements

to be modified) [13]. Several stakeholders find it useful to evaluate rapidly a change and improve their understanding and managing decisions [13].

— Each US is typically written using the following template (As an actor...I can). The effort required to implement a US (representing a ER) is estimated using PP technique (story points).

— For each US, the corresponding COSMIC FP is presented and classified by type (add, delete, modify).

4.3.1.2.3 Measurement phase According to [117], the data movements can be identified based on some common word cases (such as create, select, delete, add, share, display, etc.). Using the identified data movement types that are repeatedly executed by the user will facilitate the measurement process.

The table in Appendix 5.7 shows that there is a perfect size-effort relationship. When sizing increases, the level of total effort increases. This information can be explained by the ability of the COSMIC FSM method on detailing the Functional change process (from process to sub-process) as presented in Table 4.8, which exposes more iterations and therefore more data movements.
Once the dataset for estimating is created, the question is how well the Functional Size of the change is correlated to the SEEE in the scrum context?

4.3.1.3 Applying CFS algorithm

The Pearson's Correlation Coefficient algorithm, the most commonly used algorithm, is used for filtering in this step. Pearson's correlation coefficient is a single number that indicates the strength and direction of a linear relationship between two continuous variables. The values can range from -1 to +1, where -1 represents a total negative correlation, 0 represents no correlation, and 1 represents a total positive correlation. We will create a Pearson correlation heat map in this case. Each feature is ranked according to the correlation score obtained (See Equation 4.2) [110]:).

$$p = \frac{cov(Xi,Y)}{\sqrt{var(Xi)var(Y)}} \tag{4.2}$$

where, cov(Xi, Y) represent the covariance between a feature Xi and the target class Y, and var(Xi), var (Y) represents the variance of feature Xi, feature Y, respectively. The

Table 4.8 – Sizing the "Add Custom Evidence Type" enhancement in CFP units

FP Description	Functional User	Description of Sub-FP	Data Group	Object of interest	Data movement type	CFP
Add Custom Evidence Type	Organisation User	Request to view a specific widget	Custom Evidence type	Custom Evidence type	Entry	1
		Add a "custom" evidence type against a specific attainment criterion	Custom Evidence Type	Custom Evidence type	_	_
	Organisation User	Add a custom evidence type and give it a name (free form text)	Custom Evidence Type	Custom Evidence type	Entry	1
		Verify changes	Custom Evidence Type	Custom Evidence type	Read	1
		Save change	Custom Evidence Type	Custom Evidence Type	Write	1
		The custom evidence type is saved	Custom Evidence Type	Custom Evidence type	eXit	1
		The custom evidence type is saved	Custom Evidence Type	Custom Evidence type	eXit	1
					Total	6

expression of statistical results with effect sizes (such as Pearson correlation) and confidence intervals provides a more comprehensive method of statistical result interpretation in terms of not only statistical significance but also the size of treatment effects [118].

4.3.1.3.1 Identifying relevant features After the preprocessing phase, we selected a total of five attributes where four are independent variables and one is the dependent variable. The use of Pearson's correlation coefficient provides a list of features that are sorted based on their degree of correlation to the module class (*i.e.,* the actual value in Story Points).

Table 4.9 illustrates the definition of the features used to facilitate the interpretation of

the Pearson correlation heat map. Following the advice of Field [119], we selected only

Table 4.9 – Description of Selected Features

Selected Features	**Description**
Man-days(Today)	Effort-cumulative
StoryPoints(Actualtoday)	Effort-actual-today
FunctionalSize(COSMIC FP)	Functional Size using COSMIC
StoryPoints(Actual)	Effort-actual-cumulative calculated based on the use of Story Point
StoryPoints(Estimated)	Effort-estimated-cumulative calculated based on the use of Story Point

the correlated features having a coefficient larger than 0.4 with the output variable. Of course, when taking into account absolute value. The use of the CFS algorithm selects 80% (4 out of 5 features) (see Figure 4.6).

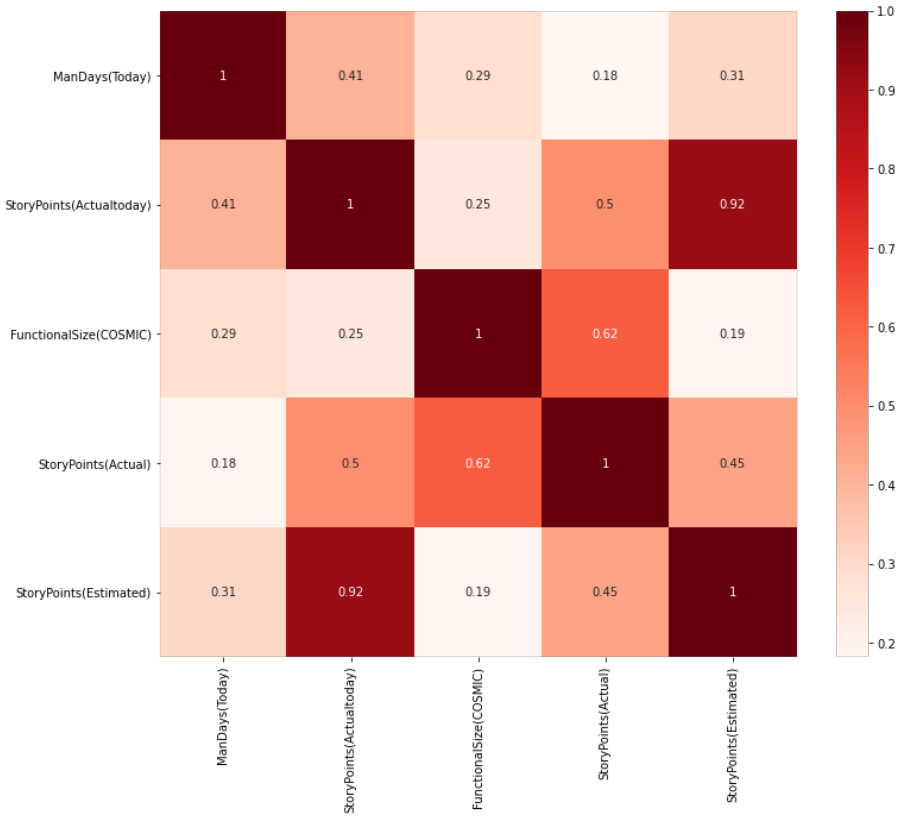

Figure 4.6 – Pearson's correlation heat map

Table 4.10 shows the selected features with their corresponding correlation coefficient value, computed using a real dataset from industry within the context of the Scrum software project. The selected features are "StoryPoints(Actualtoday), Functional-

Size(COSMIC), StoryPoints(Estimated), and enhancement effort “StoryPoints(Actual)".

Table 4.10 – Selected Features

Selected Features Correlated with StoryPoints(Actual)	Correlation target
StoryPoints(Actualtoday)	0.495189
FunctionalSize(COSMIC)	0.618766
StoryPoints(Estimated)	0.445416
StoryPoints(Actual)	1.000000

4.3.1.3.2 Determining the correlation between the COSMIC Functional Size of an enhancement and its corresponding effort Determining the correlation between the COSMIC Functional Size of an enhancement and its corresponding effort answer the question of what is the key factor that impacts the enhancement effort prediction? From the results, the correlation coefficient (score P) has a value of 0.6 which indicates an acceptable correlation of the enhancement Functional Size with its corresponding enhancement effort expressed in terms of Story Points (see Figure 4.6). The use of Functional size (COSMIC) can significantly improve the prediction of an enhancement effort when compared to other features (*e.g.,* StoryPoints(Estimated)). Therefore the enhancement functional Size is chosen as the primary independent variable. It has been observed that COSMIC sizing is an efficient standardized method for measuring not only software size but also the functional size of an enhancement that may occur during the scrum enhancement project. The resulted enhancement sizes are objective and well correlated to the actual effort.

4.3.1.3.3 Determining the correlation between the SEEE and the Actual effort With the determination of the correlation coefficient, the deviation between the predicted enhancement effort and the actual enhancement effort in terms of Story points can be identified. As shown in table 4.10, the correlation (Score P) has a value of 0.4, which indicates a weak correlation of the predicted enhancement effort with the actual enhancement effort. It has been observed that the use of Story Points is only practicable when measuring the actual effort of USs. However, Story Points cannot help in predicting the total software effort early in a project where usually a cost/benefit analysis is needed

before committing to the project. Thus, it is recommended for software organizations to adopt the COSMIC FSM method for sizing enhancement and providing accurate effort of the enhancement. The main reason for using COSMIC is its objectivity and it can be used at any phase of the SLC, and at all levels of aggregation, [117].

4.3.2 Creating Prediction Models

This section conducted a series of empirical analyses to investigate the use of the SVR ML techniques. For empirical analysis, we have used the hold-out method. The main reason for using the hold-out method (70%-30% split) compared to Leave-one-out cross-validation is because it is good to use for a very large dataset. Nevertheless, the Leave-one-out cross-validation, which is a special case of k-fold cross-validation is used when a data set or for a class value is small [120]. The data split ratio for training and testing is one of the most important factors to consider when using the retention method. This is a particularly difficult decision to make, and even a minor error in selecting the check size can result in over or under adjustment [120]. The retention method, on the other hand, works well when we have a very large data set or when starting to build an initial model in a data science project. We split data into training (70%) and test sets (30%). The training set is used to train the model, and the test set is used to validate it on data it has never seen before. Thereafter, to carry out the empirical analysis, the CFS algorithm and validation test prediction of the SVR model were performed using the Google Colab python programming. Table 4.11 lists the predefined range of parameter values of the SVR model. An important factor affecting SVR performance is how to correctly select model parameters which play an important role in obtaining good performance:

— Kernel=linear: It must be one of ‘linear’, ‘poly’, ‘rbf’, ‘sigmoid’, ‘precomputed’ or a callable. If none is given, ‘rbf’ will be used. The most suitable one to use in our model is the linear kernel.

— Complexity (C): The larger C is, the better performance SVR has. However, if C is too large, then the goal is only to minimize the empirical risk, without considering the complexity of the model in the optimization formula. In our situation, we achieve good performance when using 1,2.

— epsilon=0.2: A larger epsilon value results in fewer SVs selected and in more “flat” (less complex) regression estimates. The selected value in our situation gives an

accurate estimation.

— Deviation=0.001, 0.0001; The smaller deviation value is the best performance of SVR.

Table 4.11 – Parameters values for Grid Search

ML Technique	**Parameters**
SVR	Kernel=Linear; Complexity=1, 2; epsilon=0.2; Deviation=0.001, 0.0001

4.3.3 Empirical Analysis Results

This section evaluates the prediction performance of the SVR model used in this study. We conduct two types of empirical analysis: with and without the use of the enhancement Size. The appropriate performance evaluation metric is always chosen based on the problem type, which can be regression, classification, or clustering [120]. In our situation which is a regression problem, we used two types of evaluation metrics. The root mean square error (RMSE) and the mean absolute error (MAE) to evaluate the accuracy of our prediction models.

4.3.3.1 Performance Assessment without the Enhancement size feature

Table 4.12 shows that the Error metrics (such as MAE and RMSE) provide quite better results using the SVR model (MAE=0.6174; RMSE=0.6763). Using the SVR

Table 4.12 – MAE and RMSE without sizing the “Functional change" feature

Algorithm/ parameters	**MAE**	**RMSE**
SVR	0.6174	0.6763

model, we observed that the SEEE is more accurate when the effort is estimated using story points (see Figure 4.7).

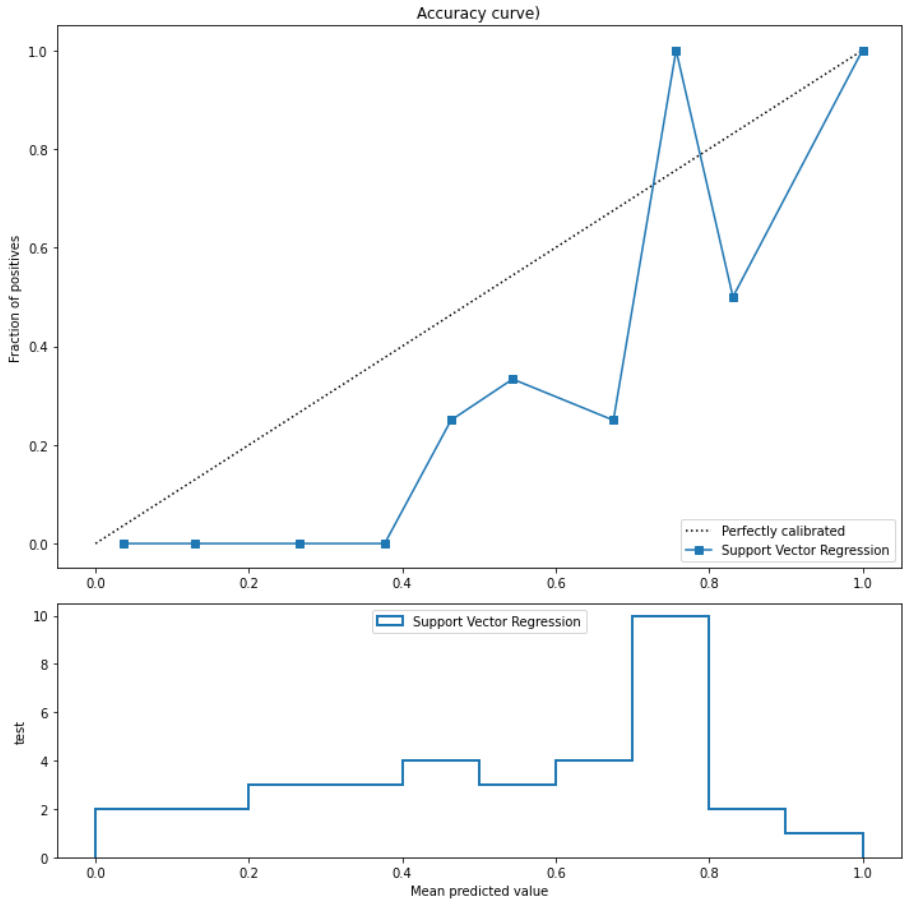

Figure 4.7 – Performance Assessment without using the Enhancement size feature

4.3.3.2 Performance Assessment with the use of the Enhancement size feature

Performance Assessment when using COSMIC sizing and SVR is trained and tested using features selected by the CFS algorithm. Using the CFS algorithm with the SVR model can lead to a more accurate SEEE when the enhancement Size is used as the independent variable (see Figure 4.8). Table 4.13 illustrates that error metrics (such as MAE and RMSE) are quite better results using the SVR model (MAE=0.2402; RMSE= 0.6469).

Table 4.13 – Prediction analysis using MAE, RMSE with the Enhancement size feature

Model/ parameters	**MAE**	**RMSE**
SVR	0.2402	0.6469

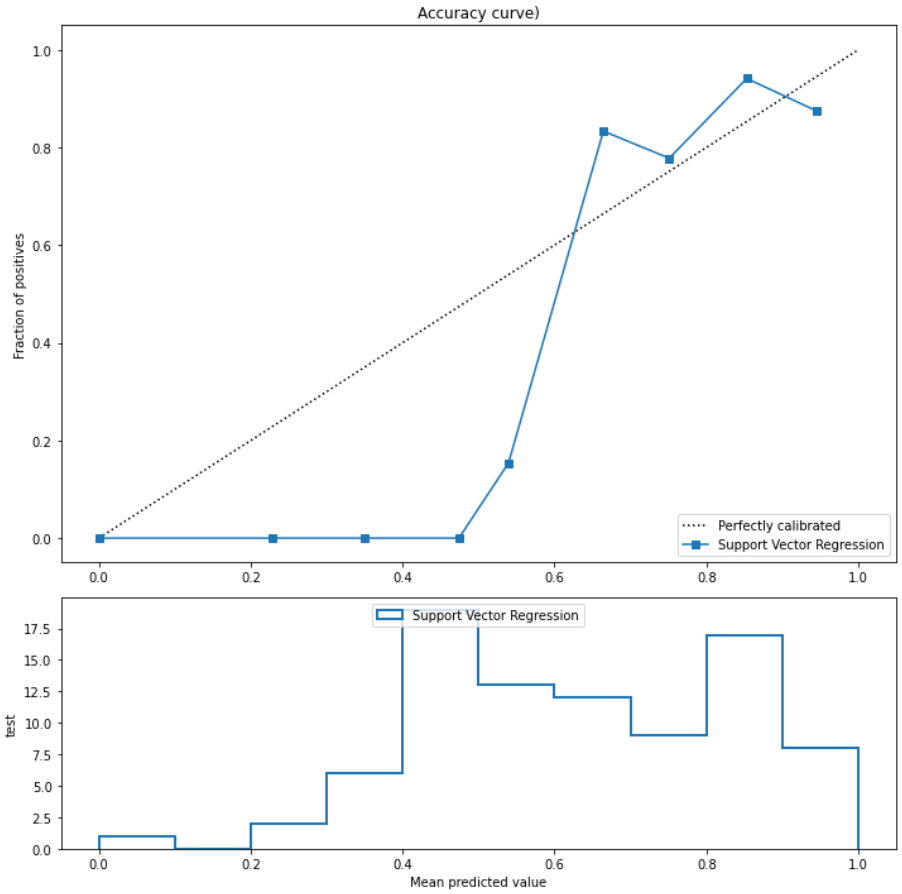

Figure 4.8 – Performance Assessment with the Enhancement size feature

4.4 Discussion and Comparison

In this chapter, we demonstrated the effectiveness of the COSMIC FSM method not only for sizing the functional changes but also when it is used as an independent variable to estimate the effort required to complete an enhancement in both traditional and agile software enhancement projects contexts. The SVR model is used to predict the effort of a new enhancement when it occurs throughout the scrum process. The results showed a minimum MAE of 0.2402 and a Standardized Accuracy (SA) metric of 98% which indicates an accurate prediction. The main reason behind selecting COSMIC sizing is its accurate measurement results compared to the IFPUG method.

Besides, the CFS algorithm is also applied to identify the significant drivers of effort in the enhancement project with Scrum. The correlation between the enhancement size measured using the COSMIC FSM method and the actual effort expressed in terms of Story Points is provided. Consequently, the COSMIC functional size of an enhancement is evaluated to be an effective significant feature, since it increases the accuracy of the

SEEE. This is demonstrated by the results, which have a Score P of 0.6. Using a burn-

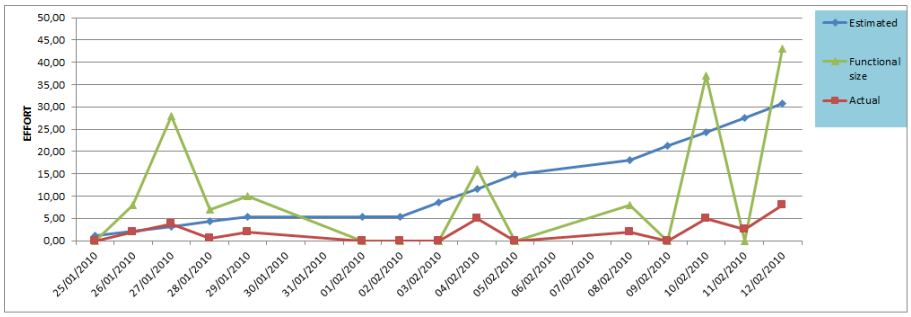

Figure 4.9 – Positive correlation curve

down chart which is the popular method of reporting progress when using Agile Project Management [57]. We illustrate in Figure 4.9 that the feature "actual effort" increases when the feature "Functional size" increases. The feature "actual effort" decreases as the feature "Functional size" decreases. It means that the two discussed features are positively correlated. A positive correlation exists as is illustrated by the CFS algorithm. The feature "Functional change size" will impact the feature "actual effort". A positive correlation is a relationship between two variables (or features) in which both variables move in tandem—that is, in the same direction[3]. It is also noted that the curve presenting the effort required to implement an enhancement having a size derived from the use of the COSMIC FSM method is higher than the curve presenting the enhancement effort derived from the use of the PP technique. This explains that sizing a US (representing an enhancement) expressed in terms of COSMIC Functional Points gives more details than sizing the US expressed in terms of Story Points. Consequently, the COSMIC FSM method is evaluated to be the most effective when compared to the use of the PP technique. It is evident from the results with the minimum MAE of 0.06. To summarize, the benefits of this study are oriented to the software organizations focusing on planning and managing software enhancement projects in the scrum context. Predicting the effort required to implement an enhancement (described in the form of US) is important to help in managing software enhancement projects successfully. It is preferred to be sized using a standardized method such as COSMIC FSM. Accurate estimates reduce uncertainty and support software project management more effectively. Therefore, it is necessary to use COSMIC enhancement size as an independent variable for the SEEE within the scrum project.

3. https://www.investopedia.com/terms/p/positive-correlation.asp

4.5 Conclusion

The findings of our empirical study investigated in this chapter were as follows:

— The correlation score between COSMIC enhancement size and enhancement effort has a value of 0.6 which denotes a good correlation as compared to the estimated Story Points. Therefore, choosing the COSMIC enhancement size as the primary independent variable provides a more accurate SEEE.

— The estimated effort that results from the SVR model provides good accuracy. Consequently, COSMIC can be considered as an effective measurement method for sizing an enhancement within a Scrum project and thereafter predicting its corresponding effort.

— It was found that the SVR enhancement effort-based prediction model is more accurate with small MAE and RMSE values results and with quite good performance when the enhancement Functional size is used as the independent variable.

In the next chapter, we discuss the effectiveness of using the stacking ensemble model for improving the accuracy of the SEEE.

Chapter 5

Software Enhancement Effort Estimation using Stacking Ensemble method

Contents

In Short

In software project management, estimating software enhancement efforts has become a difficult task. Recent studies have concentrated on identifying the best ML algorithms for software project estimation. The majority of the research papers looked at the use of ensemble learning to improve the accuracy of software project estimates. Intending to increase the estimation accuracy over individual models, this chapter investigates the use of the stacking ensemble method for SEEE. We make a comparison between two ML-based approaches for SEEE: The M5P (as an individual model) and the stacking as an ensemble method combining different regression models (GBRegr, LinearSVR, and RFR) using the ISBSG dataset. The CFS algorithm is used to achieve efficient data reduction. The selected two experiments models were trained and tested on the dataset with relevant features leading to the improvement of the SEEE accuracy.

5.1 Introduction

The benefits of using SEEE models are numerous: Estimation models, for example, can help in making decisions about when to restructure or re-engineer a software component to make it more maintainable, or in understanding the underlying causes of the difficulty in correcting specific types of errors [121]. ML techniques are widely used in this field to improve the accuracy of SEEE. ML techniques are best suited for dealing with high-dimensional problem modeling [121]. However, there is no agreement among researchers on the technique that can achieve better estimation [122]. Case-based reasoning, neural networks (NN), decision trees (DT), Bayesian networks, support vector machines (SVM), genetic algorithms, genetic programming, and association rules are all examples of statistical regressions or ML-based models that have been proposed for SEEE (ARU) [122]. More recently, research publications investigated the use of ensemble learning for improving the accuracy of the software effort estimation [50]. Consequently, various ensemble methods, such as those presented in chapter 1 are considered [50].

The main motivation for this research study comes from the fact that existing single techniques for SEEE have several limitations, while other innovative approaches for estimating, such as the ensemble method, have yet to be adopted in the industry [123]. In this chapter, we have set up two SEEE models. Regarding the first model, the SEEE is carried out by applying four individuals ML techniques (M5P, GBRerg, LinearSVR, and RFR). Constructing the first model, we have also made the focus on the selection of optimal feature set in the ISBSG dataset using the CFS algorithm investigating the impact of the COSMIC FSM method on improving SEEE performance. Three different ML-based models (GBRegr, LinearSVR, and RFR) are combined in the second SEEE model named Stacking ensemble model.

Following that, we make a comparison between the two SEEE based on ML approaches. We highlight the impact of using individuals and the stacking ensemble model for improving the accuracy of SEEE. The four chosen ML techniques were trained and tested using industrial projects from the ISBSG Release 12 dataset [2]. Staking estimation results will be compared to those obtained using an individual algorithm (M5P).

The rest of this chapter is organized as follows: In Section 5.2, we present an overview description of our research methodology process consisting for achieving better SEEE. In section 5.3, the results of using individuals ML techniques are evaluated and discussed. In section 5.4, the results of using the stacking ensemble model are evaluated and discussed.

In Section 5.5, we intend to discuss the experimental results. Section 5.6 proposes SEEE automation through the development of a web application named "ERWebApp". Finally, in Section 5.7, we conclude the chapter.

5.2 Research Process

Our research process set up two models to predict SEEE (see Figure 5.1). The first model is constructed using four selected regression ML techniques (M5P, LinearSVR, GBRegr, and RFR) separately. The second model constructs a stacking ensemble method (that combines LinearSVR, GBRegr, and RFR). For this second model, the meta-model provided via the “final_estimator” argument (LinearSVR) is trained to combine the estimation of the chosen regression ML techniques provided via the "estimators" argument (GBReg, RFR). Finally, we make a comparison of the estimation accuracy of the two mentioned models.

5.2.1 Data Collection

The dataset used to train and test the SEEE model is from ISBSG Release 12. The description of the used dataset is already detailed in chapter 4 (section 4.2.1) with the same filter considering only "enhancement" as the "development type", the "count approach" was the COSMIC FSM method, and data with a high level of integrity and soundness (*i.e.,* Records with a "Data Quality Rating" of "A" or "B").

5.2.2 Relevant Features Extraction based on the CFS algorithm

We applied the CFS algorithm to determine which features globally and consistently appear in the optimal set of features. A correlation matrix is extracted using Pearson correlation. Matrix results are presented in Figure 5.2. The use of CFS algorithm selects 37.5% (6 out of 16) of features (see Table 5.1). The CFS algorithm is used not only to select features, but also to evaluate the impact of the enhancement size feature on the accuracy of the SEEE.

As described in Chapter 4, it has been observed that COSMIC sizing is an effective method for measuring not only software size but also the functional size of the ER that will occur throughout the maintenance life cycle [124]. Figure 5.2 shows that the correlation coefficients between enhancement functional size and enhancement effort have a value of

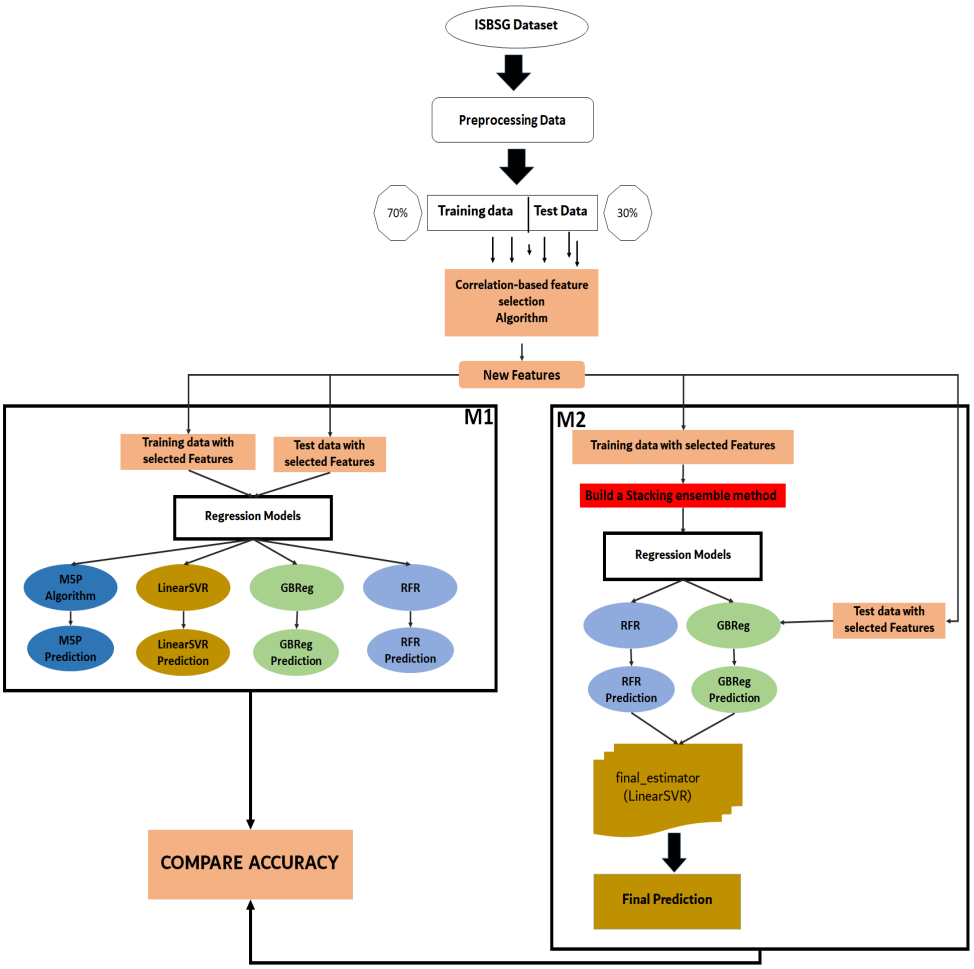

Figure 5.1 – Research method design

Table 5.1 – Selected Feature correlation

Features Selection Methods	**Selected Features with value (Round(Correlation target))**
Pearson correlation	CHANGEWorkEffort=0.4; UnrecordedWorkEffort=0.5; Functionalsize=0.5; EffortTest=0.4; SummaryWorkEffort=0.8; NormalizedWorkEffortLevel1=1

0.5 which is adequate. This investigation indicates a suitable correlation when put next to other features (such as CHANGEWorkEffort and UnrecordedWorkEffort). Change functional Size was therefore chosen as the primary independent variable.

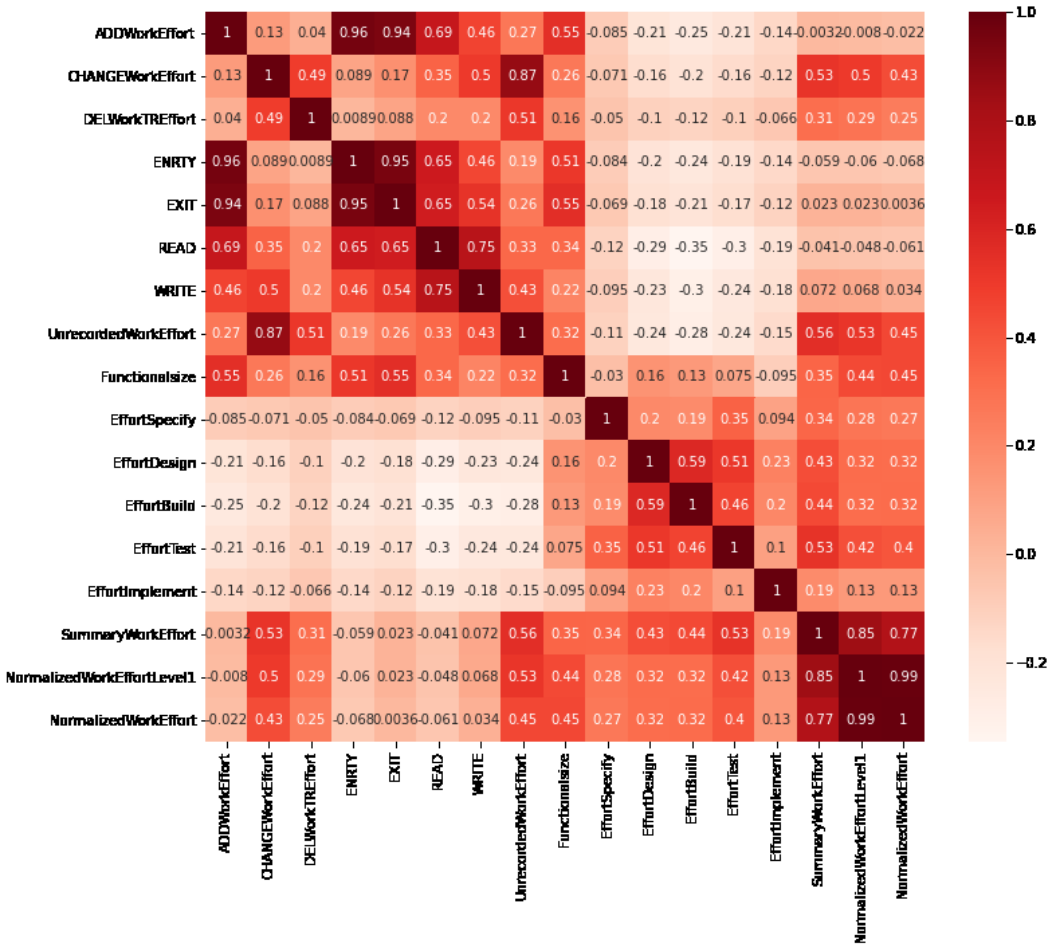

Figure 5.2 – Pearson correlation heat map

5.3 Constructing Individuals Estimation Models

This section carried out a series of experiments to investigate the performance of the chosen ML techniques (M5P, GBRegr, LinearSVR, and RFR). We divided the data into two sets: training and validation/test (70 percent-30 percent split respectively). Following that, various tools were used to carry out the experiments.

The new M5P model (tree-based model) was created with the help of Weka software[1]. It is widely used in education, research, and industrial applications. It includes a plethora of built-in tools for common ML tasks. The Google Colab Python programming was used to perform 10-fold cross-validation and validation test prediction of the GBRegr, SVR, and RFR models for the feature selection methods. Table 5.2 lists the selected ML techniques with their corresponding predefined range of parameters values. The selected ML techniques are trained and tested using (1) dataset without filter and (2) features

1. https://www.cs.waikato.ac.nz/ml/weka/

Table 5.2 – Parameters values for Grid Search

ML Techniques	Parameters
M5P	Instances=5
GBRegr	random_state=0; min_samples_split=2
LinearSVR	Kernel=Linear; Complexity={1,2}; epsilon={0.2}; Deviation={0.001, 0.0001}
RFR	random_state=0; min_samples_leaf={1,2,3}; Max_depth={2,4,6}; min_samples_split={2}

selected during the preprocessing phase (CFS).

5.3.1 Performance Assessment without using CFS algorithm

The prediction errors (MAE and RMSE) are calculated for each prediction. The predicted values are compared against the actual target values. It is evident from the results that M5P method gives better performance when compared to other three ML techniques (with MAE=0.4035, RMSE=0.4002). We also used cross-validation method

Table 5.3 – Prediction analysis using MAE and RMSE

Method/ parameters	MAE	RMSE
M5P	0.4035	0.4002
GBRerg	0.6635	0.7501
LinearSVR	0.6331	0.7267
RFR	0.5445	0.5646

(10-fold cross-validation) [120]. Cross-validation methodology is used to compare models by dividing data into two segments: one used to learn or train a model and the other used for testing to validate the model [120]. The accuracy of prediction models can be increased as the model predicts new data that were not used in its prediction. Cross-validation strives to measure the generalization power of a model: how well it will predict new data [120]. In this study, we used 10-fold cross-validation. Since it is the most used for experiments and to analyze the performance of ML techniques [120]. Table 5.4 illustrates the results of using these metrics.

Table 5.4 – Prediction analysis using 10-folds Cross Validation methods

Method/ parameters	**MAE**	**RMSE**	**SA(%)**
M5P	0.0612	0.3381	97.25%
GBRegr	0.1846	0.3634	50.55%
SVR	0.2407	0.3379	54.30%
RFR	0.1738	0.3300	56.42%

5.3.2 Performance Assessment using CFS algorithm

Table 5.5 shows that error metrics (such as MAEs and RMSEs) reveal relatively low values when using M5P (MAE=0.0571; RMSE=0.2514). The results show that the M5P method outperforms the other three ML techniques, with a SA of 99 percent (See Figure 5.3). Using the CFS algorithm with the selected ML techniques leads to more accurate SEEE when the enhancement Size is used as the independent variable (see Table 5.5).

Table 5.5 – Prediction analysis using MAE, RMSE and SA

Method/ parameters	**MAE**	**RMSE**	**SA(%)**
M5P	0.0571	0.2514	99.36%
GBRegr	0.2625	0.3447	85.43%
LinearSVR	0.1110	0.3020	89.69%
RFR	0.1665	0.3187	87.54%

5.4 Constructing Estimation stacking ensemble model

Our stacking ensemble technique is based on the idea that "when weak models are correctly aggregated, the strength of the union leads to higher performance and more accurate SEEE." To construct the staking model it is important to (1) choose which models to use as "estimators" and which models to use as "meta-models," and (2) then generate estimates by the feeding estimator predictions into the selected meta-model.

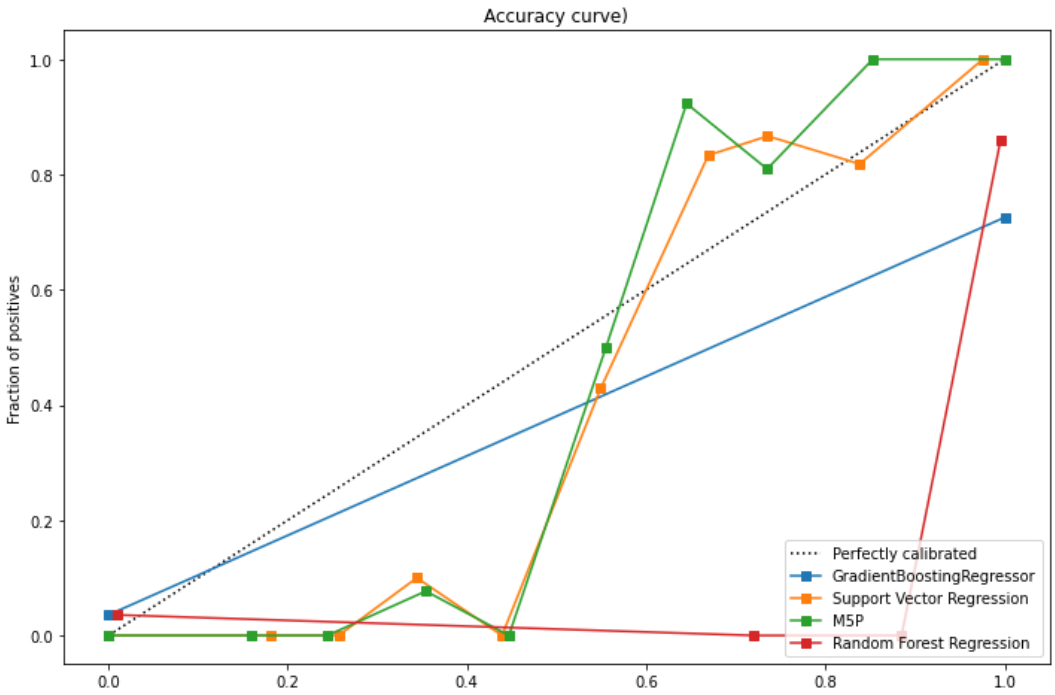

Figure 5.3 – ML techniques accuracy

5.4.1 Selecting estimators and meta-model

The main parameters of the stacking ensemble regression model are defined in scikit-learn[2] as follows: StackingRegressor(estimators, final_estimator=None, *) explained in Table 5.6. Thus, we try to identify which technique from the three ML techniques can

Table 5.6 – Stacking ensemble regression model parameters'

Parameters	Description
estimators	Base estimators which will be stacked together.
final_estimator	An estimator which will be used to combine the base estimators

be used as "final_estimator" and which ones should be used as "estimators". In this case, we selected the r2_score evaluation metric[3] to evaluate the overall performance of the selected prediction model to provide an adequate combination. Table 5.7 illustrates the r2_score results where the best possible score stands at 1.0. Figure 5.4 shows the ML "estimators" and the average of their predictions.

2. https://scikit-learn.org/.../sklearn.ensemble.StackingRegressor.html
3. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.r2_score.html

Table 5.7 – Prediction analysis using R2 Score

Method/ parameters	**R2 Score**
GBRegr	0.981
LinearSVR	0.956
RFR	0.980

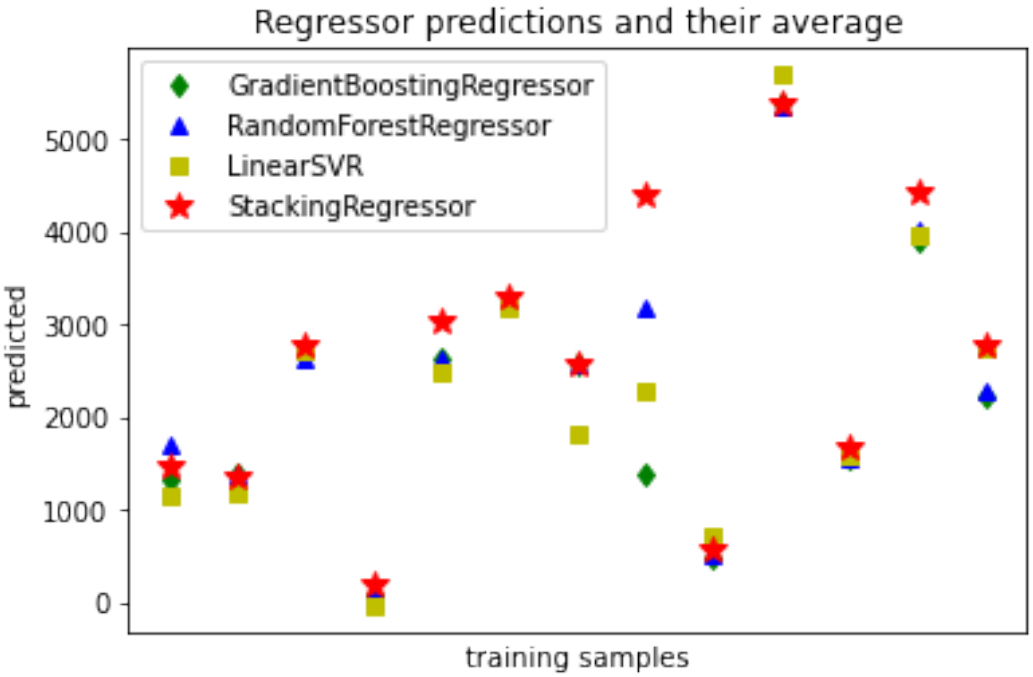

Figure 5.4 – ML "estimators" and the average of their predictions

5.4.2 Constructing the SEEE model

Each ML regression technique is trained on the ISBSG dataset with relevant features filtered using the CFS algorithm allocated for training (see section 5.2.2). The outputs of "estimators" are therefore fed into the "final_estimator" which combines each regression estimator model with a weight and delivers the final estimation. Regarding Table 5.7, LineanrSVR is selected to be used as the final_estimator. Table 5.8 shows the stacking ensemble method parameter that defines the best combination.

When the enhancement functional Size is utilized as the independent variable (see Table 5.9), the CFS algorithm combined with the constructed stacking ensemble approach yields an appropriate SEEE. When the findings are compared to the other three ML techniques, it is clear that the stacking ensemble method outperforms them all. The r2 score now stands at 0.987 (See Figure 5.7).

Table 5.8 – Parameters values for Grid Search

ML Techniques	**Parameters**
Stacking model	estimators=[(GBRegr,RFR)], final estimator= LinearSVR()
GBRegr	alpha=0.9, ccp_alpha=0.0, criterion='friedman_mse', init=None, learning_rate=0.1, loss='ls', max_depth=3
LinearSVR	C=1.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.1, gamma='scale', max_iter=-1, shrinking=True, tol=0.001, verbose=False
RFR	max_depth=3, max_features=1, max_leaf_nodes=None, min_samples_leaf=25, min_samples_split=2, min_weight_fraction_leaf= n_estimators=25

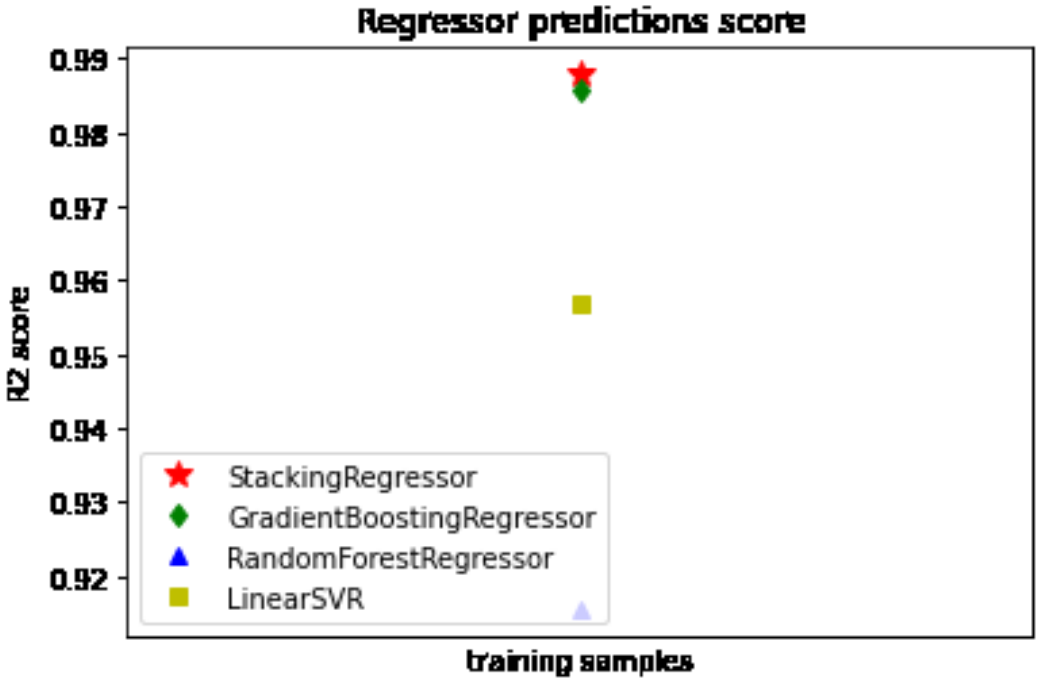

Figure 5.5 – Regressor estimation score

5.5 Discussion and Comparison

The first experimental model was conducted to evaluate the accuracy of four ML techniques (M5P, GBRerg, LinearSVR, and RFR) separately. The selected ML techniques are used to provide the SEEE when software is being maintained. M5P is the most effective of the ML algorithms tested. The results back this up, with a minimum MAE of

0.0612. The results achieved when using a simple method demonstrate M5P's efficiency. It has small MAEs and RMSEs values. A good accuracy (SA) of 99% is obtained when using the 10-fold cross-validation.

The relevance of each feature is computed using the CFS algorithm to discover the effective determinants for SEEE. Furthermore, when compared to the model that used all of the selected features, the model utilizing the CFS method performs better (17 features). As a result, adopting the M5P ML technique enhances the SEEE accuracy.

In the second experiment model, we studied the possibility of adopting a stacking ensemble method by merging the weak ML techniques to secure the above results (GBR-erg, LinearSVR, and RFR). The M5P algorithm is used to compare the experimental outcomes (see Table 5.9). The results demonstrate the usefulness of the stacking ensemble method (see Figure 5.6 and Figure 5.7). This is supported by the results with the minimum MAE of 0.0383, RMSE of 0.1973, and a good r2_score of 0.987.

Table 5.9 – Prediction analysis using MAE, RMSE and r2_score

Method/ parameters	**MAE**	**RMSE**	**r2_score**
M5P	0.0612	0.2514	0.985
Stacking Regressor	0.0383	0.1973	0.987

In this chapter, we have used the CFS algorithm for selecting the attributes from one of the well-known historical software project datasets (the ISBSG dataset that contains many attributes). Since we restricted the study to numerical attributes only 17 features have been selected which constitutes 17% of all the attributes in the ISBSG dataset after the phase of prepossessing data. Six features have been selected after using the CFS algorithm that constitutes 6% of all the attributes in the ISBSG dataset. This is why the findings of this work may differ from other studies that use other types of data. Indeed, conducting more experiments with other kinds of datasets that present quality characteristics are required. Although the experiments were performed using CFS, it is still compulsory to test other FS algorithms with different ML techniques.

The SEEE in our study is provided based on the independent variable (*i.e.,* the enhancement size). Even the results about the performance accuracy of the selected ML techniques provide good accuracy with 99%, the correlation coefficients computed between enhancement functional size and enhancement effort is still a moderate value, this is because enhancement functional size is identified at a high level of abstraction

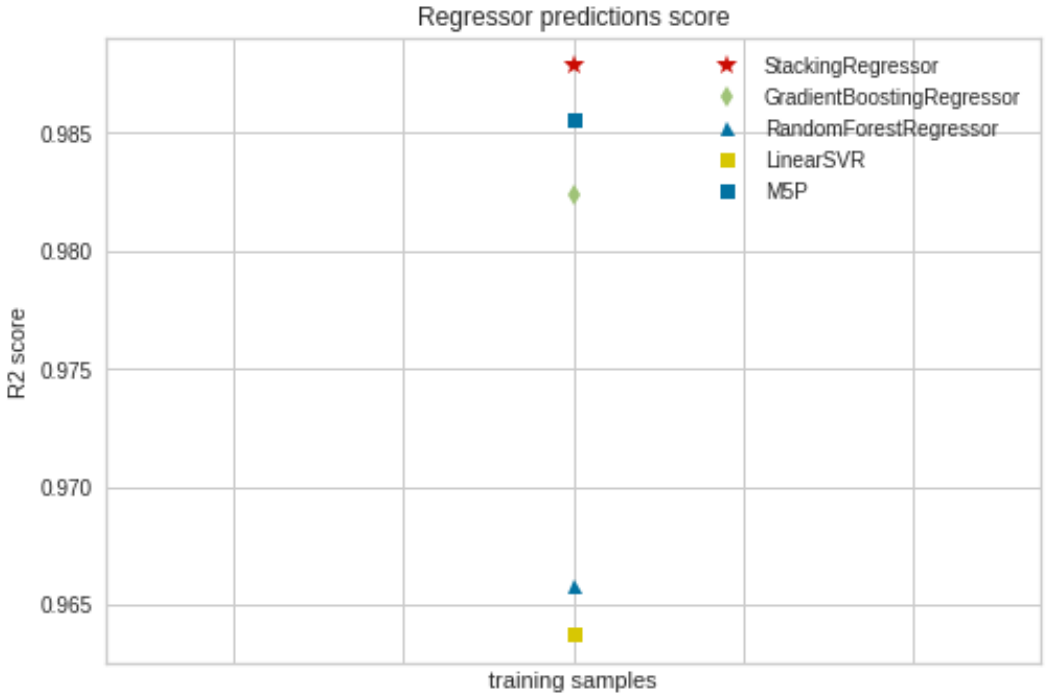

Figure 5.6 – ML techniques Performance Assessment

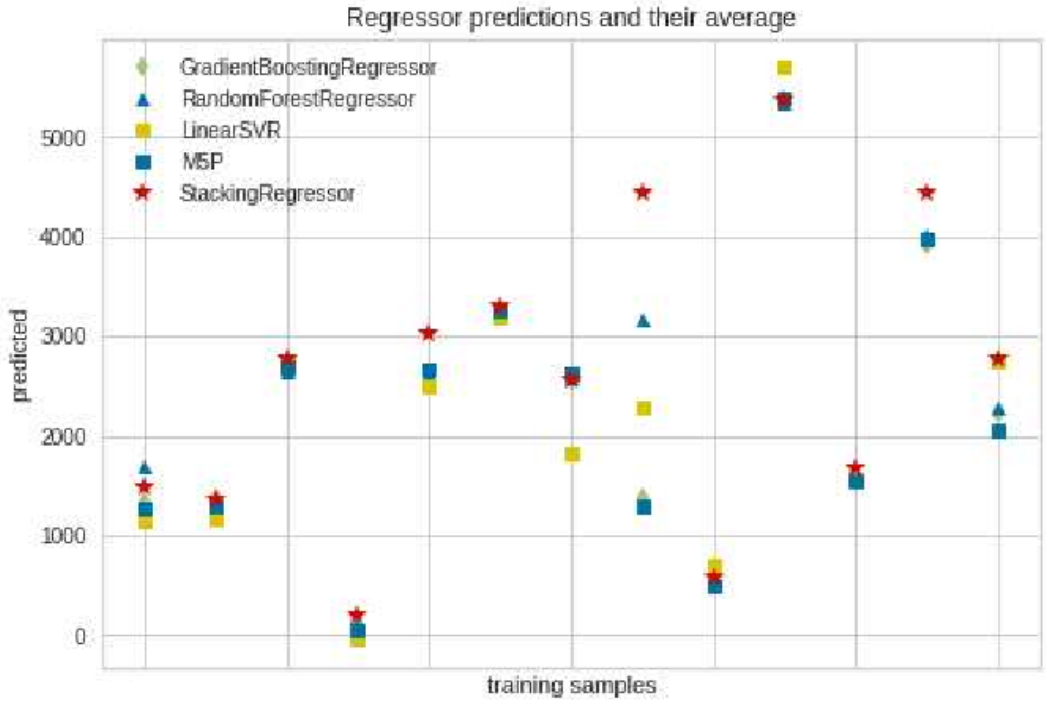

Figure 5.7 – ML techniques accuracy

of the Functional Process. Indeed, using the existing approximation size measurement method is useful in this situation [33] [34].

5.6 Automatically SEEE through a ERWebApp

Recall that our research goal is to help estimators efficiently estimate the effort of an enhancement in both traditional and scrum projects. Regarding this goal, we found that using the stacking ensemble model is the most accurate approach. However, using this approach manually is time-consuming. Manual solutions are not practical. For this reason, we propose to develop an ERWebApp to rapidly make SEEE. The proposed ERWebApp is designed to first generate the enhancement functional size and then estimate its corresponding effort. To create a user-friendly interface, we follow human-computer interaction design guidelines [125]. The ERWebApp is developed using Bootstrap[4], Anvil Platform[5], and Python[6]. Python is used in the backend to create the prediction model that maps the input and output data based on the ensemble model, while Anvil's Platform and Bootstrap are used in the frontend to display content. In order to transfer content between the ERWebApp and the prediction model, the use of the Anvil platform appeared to be beneficial. It is used to help in the visualization of the prediction model by creating and hosting the prediction web page, which is entirely written in Python using predictable and minimal resources (CPU, memory, threads). The ERWebApp is styled using Bootstrap, which adds responsiveness to the interface as well as cross-browser compatibility.

5.6.1 ERWebApp Users

The ERWebApp is designed to meet the needs of the three Scrum roles: Product Owner, Scrum Master, and Development team members. It will enable estimators to express the ER in the form of the US. Of course, described in the textual form. Based on the description of a US, its functional size will be generated in terms of CFP units. The estimator then receives the estimated effort based on ontology and ML techniques without extensive knowledge beforehand. This implies that the three scrum roles would be able to perform the following actions in the web interface:

1. The Scrum Master: is responsible for the revised planning and deciding on the execution of the ER.
2. The Product Owner: Submit the ER description.

4. https://getbootstrap.com/
5. https://anvil.works/open-source
6. https://www.python.org/

3. The Development Team: reformulate the ER in three steps: (1) specify a formal description of the ER, (2) generate the Functional size of the ER in CFP units, and (3) estimate the effort required to implement the ER.

The main part of the ERWebApp is described throughout three pages of interfaces including a projects overview (for Scrum Master), a submit ER form (for Product owner) and regulate ER form (for the development team). Focusing on making the ERWebApp easy to use, we created three sessions for the three scrum roles. Indeed, we created three login sessions/profiles for the three roles. Figure 5.8 displays an example of the Product owner login session page.

5.6.2 Product Owner Interface: Submit ER

When a new enhancement occurs in an existing project, the ER description must be submitted by the Product owner. As shown in Figure 5.9, ER have to be implemented by the Development team. Once the ER was approved, a detailed description can be developed (in other words, going from an informal to a formal ER description). Using this interface page, the Product Owner can express and submit an enhancement requested in natural language.

5.6.3 Development Team Interface

We concentrated on the development team session because that is the team in charge of managing and implementing the enhancement. The development team will provide the COSMIC sizing of an enhancement as well as an estimate of the corresponding effort. Figure 5.10 depicts the output via the user interface.

5.6.3.1 Enhancement Request Details

The web interface page of the Development team includes the enhancement details. Figure 5.10, contains two buttons: the blue-button downloads the ER description validated by the Product Owner, and the green button generates a formal explanation of a specific ER. An ER's formal description is written as described in section 4.3.1.1 of Chapter 4.

As an <Actor>

I want to <Goal>

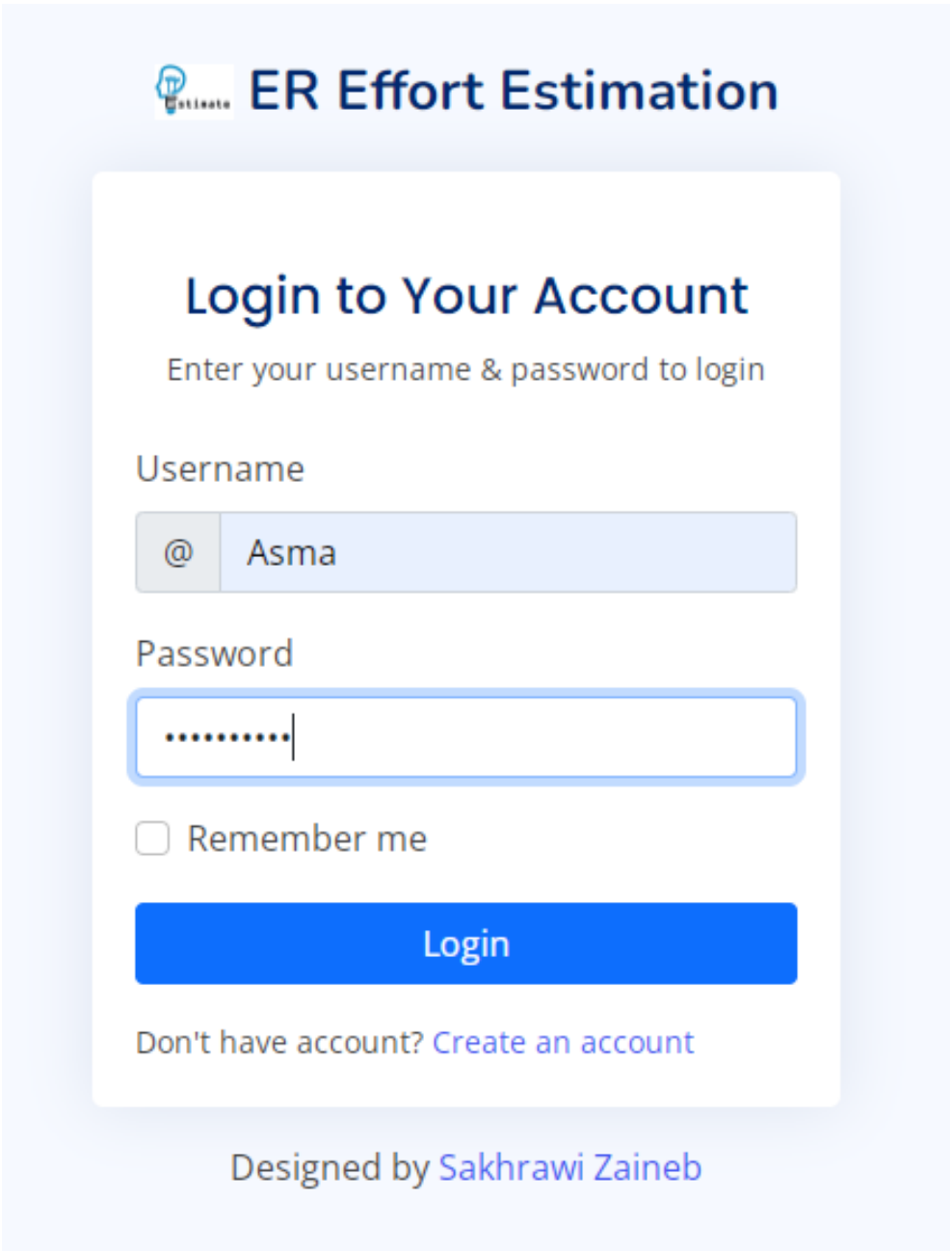

Figure 5.8 – Login Page Product Owner

so that <value or expected benefit>

Figure 5.11 shows a full formal description of a selected ER. The functional size of the specified ER is also represented in terms of the CFP units. A button was created to illustrate how the functional size of an enhancement, noted as FS(FC), can be found and used to estimate the effort required to complete this enhancement. When the button "click to reveal in detail" in Figure 5.11 is selected, the details of FS(FC) are generated as shown in Figure 5.12 (details of FSM are provided in section 4.3.1.2.3 of Chapter 4).

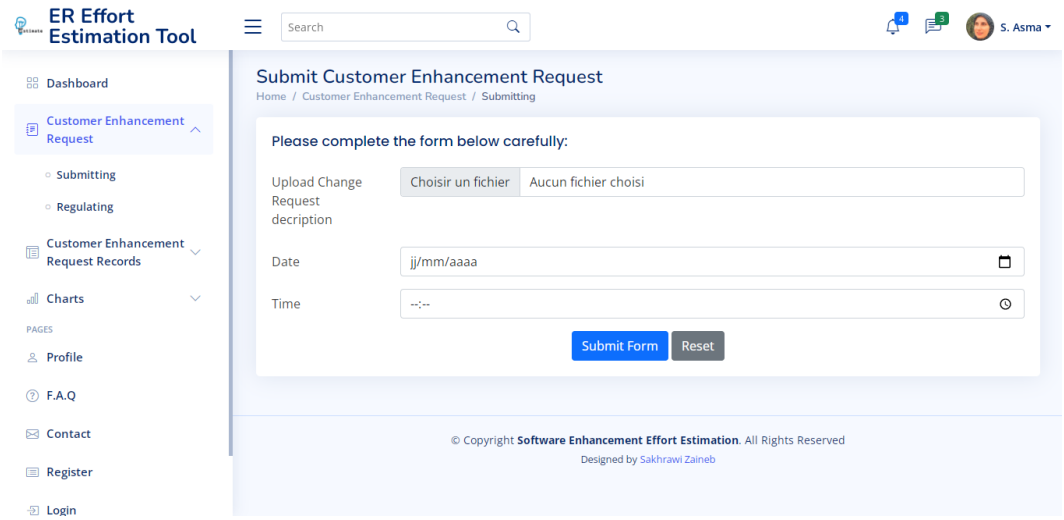

Figure 5.9 – Submit ER

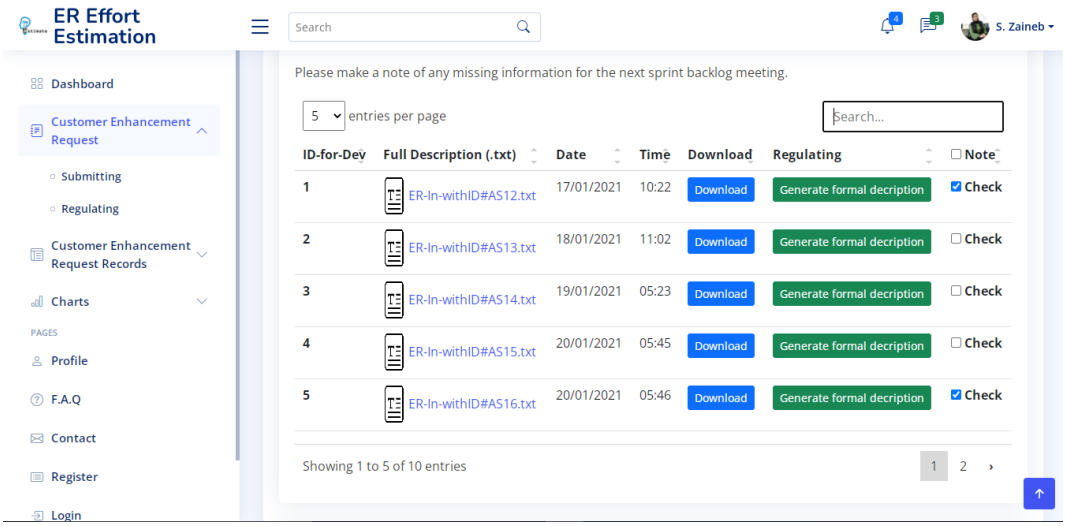

Figure 5.10 – Development Team Interface

5.6.3.2 A Web page for SEEE

As shown in Figure 5.14 using the Anvil platform, we created a web page for SEEE. Through the Anvil platform, we can share a private link to the web page. Therefore, in the Bootstrap template, we included this link to be used by the Development team (see Figure 5.13). The Anvil platform is also used to connect the model built-in Google Collab to the input variable, which is the functional size of the ER. The SEEE web page interface includes two sessions: the train session constructed on the backend with the Goggle Collab and the prediction session designed for the Developer team with the Anvil platform. As shown in Figure 5.14, the Development team needs to select the model to estimate the effort of the ER. The team can choose one of the two SEEE models: A model

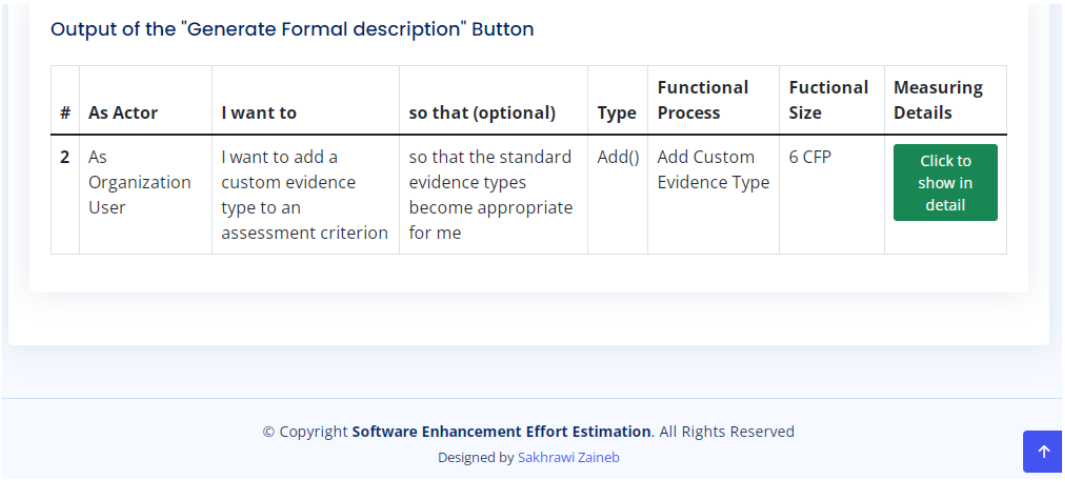

Figure 5.11 – ER Details

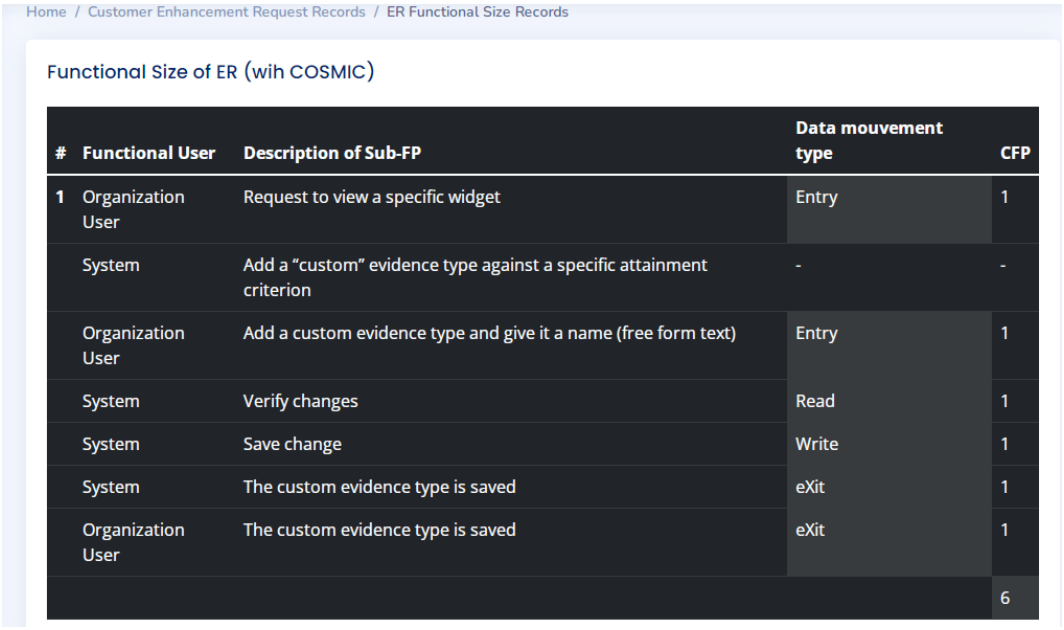

Figure 5.12 – Regulate ER page

based on the SVR algorithm which gives the accurate estimation results as described in chapter 4 within the scrum context. Or a model based on the ensemble model that uses the SVR algorithm as the final estimator. Also, the Development teams have to provide the ER Functional size (approximated) in CFP units as an input. So that the developed ERWebApp generates the SEEE.

5.6.3.3 Scrum Master Web Page

Figure 5.15 depicts the Scrum Master's web page interface. We did not put much emphasis on the role of the Scrum master because the estimation process is handled by

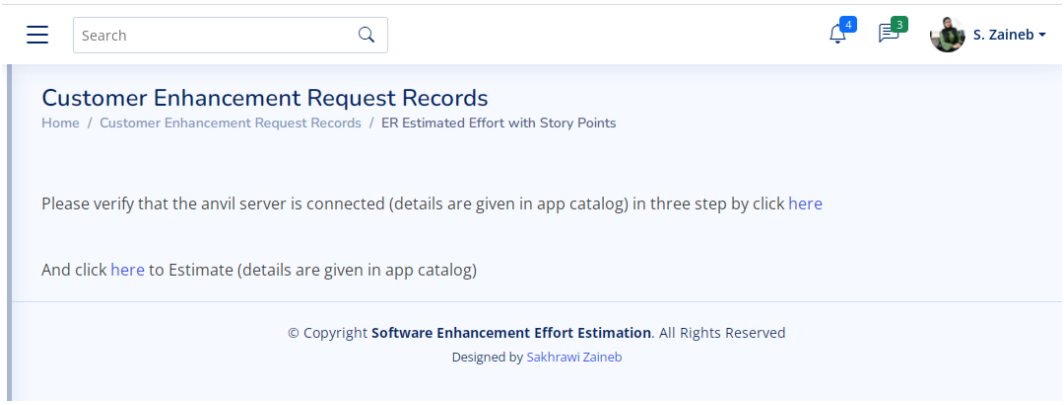

Figure 5.13 – Estimating ER Effort

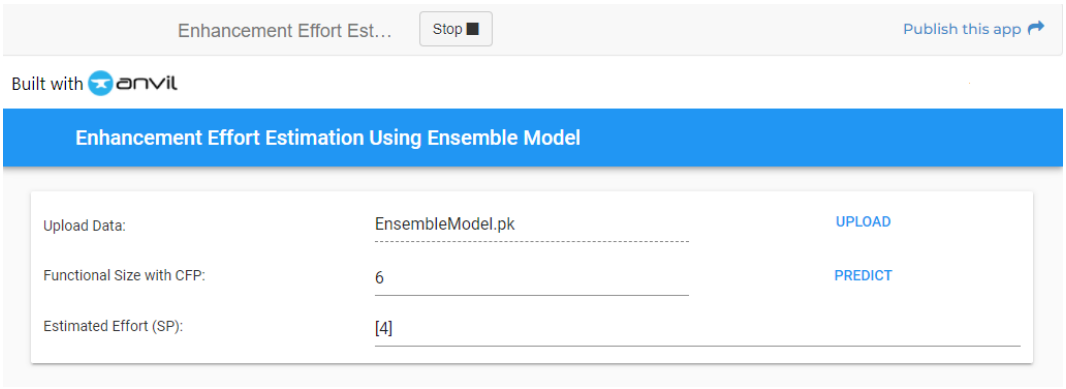

Figure 5.14 – Estimating ER Effort with Anvil

the Developer team. However, it will be improved in the future.

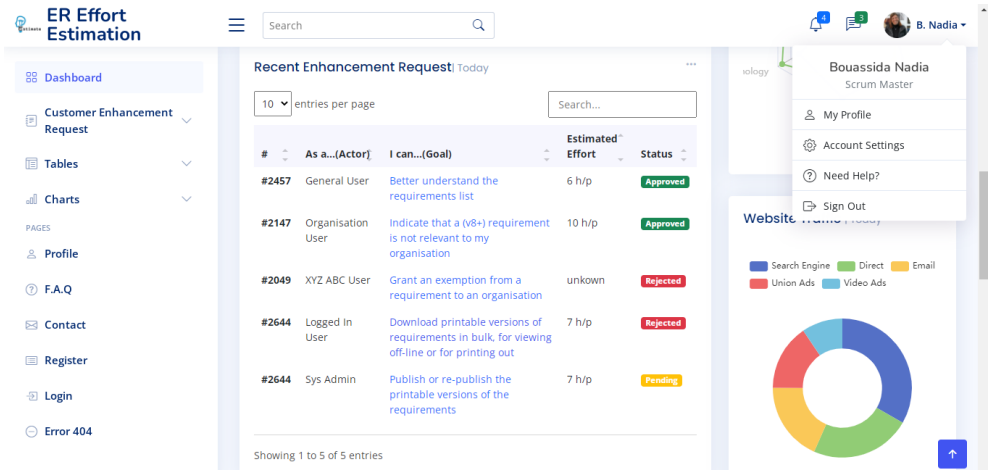

Figure 5.15 – Scrum Master

5.7 Conclusion

The following are the results of the experiment studies investigated in this chapter:

— The correlation coefficient calculated between enhancement functional size and enhancement effort has a value of 0.5 indicating that there is a strong relationship. As a result, the enhancement functional size was selected as the primary independent variable.

— M5P is more accurate with small MAEs= 0.0612 and with quite good performance of 99% compared to GBRerg, LinearSVR, and RFR.

— The stacking ensemble method (combining GBRerg, LinearSVR, and RFR) is more accurate with small MAEs= 0.0383 and R2 Score=0.987 compared to M5P algorithm.

We also built an ERWebApp for SEEE. The main purpose of the ERWebApp is to facilitate the prediction of the effort required to complete an enhancement. Of course, it will evolve re-actively in response to competition across many organizations by building a web application.

Conclusion and Perspectives

This chapter highlights the research's contributions, addresses some of its limitations, and offers future research directions based on the findings.

Recall Thesis Contributions

This thesis proposed an approach for estimating software enhancement effort. It assessed and analyzed the impact of using the COSMIC FSM method on the accuracy of SEEE. The proposed approach is intended to help stakeholders in effectively managing Enhancement Requests.

For this purpose, we started by an SMS, presenting our first contribution, which is divided into six steps: (1) Defining the mapping questions, (2) Finding primary studies, (3) Screening studies, (4) Abstract key-wording, (5) Data extraction, and (6) Mapping Results. Based on the findings obtained in this SMS, estimators should be aware that SEEE using ML techniques as part of non-algorithmic models demonstrated increased accuracy prediction over the algorithmic models. When employing the enhancement functional size as independent variables, ML techniques often achieve a reasonable level of accuracy. In the SMS, we proved that the majority of the research studies using FSM methods focused mainly on the development phase. In addition, we investigated the use of FSM methods (IFPUG and COSMIC) to identify their impact on improving SEEE performance. The results of this investigation showed that the functional size of an enhancement is useful when it is used as an independent variable for SEEE. Our main proposal in this thesis is to investigate the use of the powerful FSM method, which is the COSMIC ISO 19761 method. Taking into account the enhancement size, we estimate the software enhancement effort using ML techniques. The result of this contribution is

published in [12].

Our second contribution is split into two parts. First, we proposed to develop an ontology-based model for classifying ER as either FC or TC. We used the Protégé editor to classify ER (user reviews) from the PROMISE repository and create a comprehensive representation of the ER. Thus, each ER classified as FC is associated with its corresponding effort using expert judgment. Second, considering the output of the first part, we investigate the effectiveness and performance of four ML techniques: ABR, GBR, LinearSVR, and RFR to provide an accurate SEEE. The results of this contribution are published in [13] and [14].

Our third contribution investigated the impact of using the first and second FSM generations (*i.e.,* IFPUG and COSMIC FSM methods) on the accuracy of SEEE. The CFS algorithm is used to select the appropriate features. The enhancement FS is used as an independent variable to the SEEE model in the context of traditional projects. Following that, a comparison of IFPUG with COSMIC is conducted to provide the effectiveness of the FSM method. In this contribution, we also select the use of the appropriate FSM method, that is COSMIC sizing. In this case, we used the SVR model to provide a more accurate SEEE not only in traditional projects but also in software agile (scrum) projects. The results of this contribution are published in [15] and [16].

Our fourth contribution evaluated and analyzed the use of the stacking ensemble method over the individual models in improving the accuracy of the SEEE within the scrum context. We compared two ML-based models for SEEE: the M5P (as an individual ML technique) and stacking as an ensemble method combining different regression models (GBRegr, LinearSVR, and RFR) using the ISBSG dataset. Regarding experimental results, we found that using the stacking ensemble model is the most accurate model. However, using this model manually is time-consuming. Of course, manual solutions are not practical. For this reason, we propose to develop an ERWebApp to rapidly and automatically make SEEE. The ERWebApp is designed to first generate the enhancement functional size and then estimate its corresponding effort. The results of this contribution are published in [17] and [18].

Threats to Validity

This thesis proposed an approach to provide a more accurate SEEE using ML techniques and taking into account the COSMIC functional size of an ER as an independent variable.

The validity of this thesis's results is pertinent to internal validity and external validity.

— The internal validity threats are related to three issues:

  — The first issue affecting the internal validity of our approach is its dependence on a detailed description of the ER classified as FC. Such information may not always be available. There are so many different styles of writing US today that each company has its own. In this thesis, enhancement requests are identified and classified using an Ontology model. Nonetheless, the FSM method used in this thesis is independent of the used format to describe the enhancement request (use cases, activities, user stories, etc.).

  — The second issue, affecting the internal validity, is the use of a high-level ontology model. Some COSMIC functional processes may appear to be implemented as a new requirement while others do not implement new requirements (adaptive maintenance) and need existing software (perfective maintenance). As a result, all the measurement results must be always updated following the identified functional processes as an enhancement.

  — The third issue is that the scope of this study is limited to the ER classified as FC. While the FUR is more likely to change during the SMLC, the NFR and PRC may also change. For example, technical debt can result in more time spent developing per functional process. As a result, dealing with technical debt may necessitate a significant amount of effort in terms of code restructuring. In this thesis, however, we used the COSMIC FSM method to measure the functional size of an ER. COSMIC method, in general, measures the functional size of software based on the FUR. Whereas, some software engineering researchers' used COSMIC FSM to measure the NFR size as well. Especially, when NFR evolves into FUR.

  — The fourth issue is related to the use of the CFS algorithm. In fact, even though the selected CFS algorithm gives improved experiment results, it is still compulsory to test other CFS algorithms.

— On the other hand, external validity threats deal with the possibility to generalize the results of this study.

  — The first issue is the limited number of both traditional and agile software projects to test the proposed SEEE approach. In fact, for the traditional

development method, only six private projects and a single PROMISE dataset containing enhancement requests are used to evaluate enhancement requests semantically. A single popular ISBSG dataset containing measurement in COSMIC Function Points is used to investigate the impact of using an ER size in order to improve the accuracy of the SEEE. For Scrum projects, we used a single public dataset to measure the FS of an ER and estimate the corresponding enhancement effort. This limited number of software projects is insufficient to generalize the results of our study. Thus, testing the proposed approach in an industrial environment is required.

— Concerning the SEEE, we used individual and ensemble ML techniques, and also the Expert judgment approach. Regarding the scrum context, the experts' judgment estimation is the closest one to reality. Expert-based estimation taking into consideration if the changed functionality is implemented or not does not have a major value. However, experts' estimation may cause less transparency about how they found their results. In our study, we showed that it is possible to use ML models for estimating the desired ER Effort. The problem is that the choice of the estimation model depends on many factors like the dataset, and therefore the software company domains, etc. Using a given SEEE model, software stakeholders can make acceptable decisions that will contribute to ultimate project success.

Perspectives

The perspectives of this work can be summarized in the following main points:

— Sizing enhancement request at different levels of detail so that SEEE can be determined at different levels of accuracy. We believe that approximate and rapid ER evaluation is required, particularly for an urgent ER (*i.e.*, emergency maintenance). When there was not enough time to carry out the entire COSMIC measurement process. As a result, we propose that for future work, we use an approximation method to measure the enhancement size as proposed by the enhancement requester in natural language.

— For future work, we propose to focus not only on the different types of software maintenance, but also on an in-depth analysis of ER using the structural size mea-

surements, as proposed by Hakim et al. [126].

— Analyze the use of different feature selection algorithms. We are currently working on analyzing and evaluating the relationship between dataset attributes using the OLS regression algorithm. The results of this study have been accepted in the Innovation and system and software engineering Journal 2021 [127].

— Investigate the enhancement which is categorized as TC. Even though most of NFR evolves into FUR, we believe that the conjunction of both the ISO 25000 series of software quality model and the COSMIC ISO 19761 can be used to control and evaluate the software enhancement project progress.

— Analyze SEEE in depth. SEEE can be presented through five attributes: Plan effort, Specify Effort, Design Effort, Implement Effort, and Test effort. Hence, we are currently working on identifying the significant drivers (one of the five sub-step effort) of the total required effort for SEEE. We assume that estimating the enhancement (or test cases) effort for testing activity (regression testing) is also a key activity of software enhancement project.

Bibliography

[1] Petersen, Kai, Robert Feldt, Shahid Mujtaba, and Michael Mattsson. "Systematic mapping studies in software engineering." In 12th International Conference on Evaluation and Assessment in Software Engineering (EASE) 12, pp. 1-10. 2008.

[2] Cheikhi, Laila, and Alain Abran. "PROMISE and ISBSG Software Engineering data repositories: A survey." In 2013 Joint Conference of the 23rd International Workshop on Software Measurement and the 8th International Conference on Software Process and Product Measurement, pp. 17-24. IEEE, 2013.

[3] P. Tripathy, K. Naik, Software Evolution and Maintenance, Wiley, 2014

[4] ISO/IEC (2022) ISO/IEC/IEEE international standard for software engineering - software life cycle processes - maintenance. ISO/IEC/IEEE 14764:2022(E) IEEE 2022.

[5] García-Floriano, Andrés and López-Martín, Cuauhtémoc and Yáñez-Márquez, Cornelio and Abran, Alain. Support vector regression for predicting software enhancement effort, Information and Software Technology,97, pp. 99–109, 2018.

[6] Bourque, Pierre, and Richard E. Fairley. "Guide to the Software Engineering-Body of Knowledge. 2014." Online in Internet: URL: http://www. swebok. org [Stand: 12.01. 2005].

[7] Aljamaan, H., Elish, M.O. and Ahmad, I., An ensemble of computational intelligence models for software maintenance effort prediction. In International Work-Conference on Artificial Neural Networks, Springer, pp.592-603, 2013.

[8] Ali, Syed Sarmad and Zafar, Muhammad Shoaib and Saeed, Muhammad Tallal, “Effort Estimation Problems in Software Maintenance – A Survey”, 2020 3rd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET), 2020

[9] ERTUĞRUL, EGEMEN and Baytar, Zakir and ÇATAL, ÇAĞATAY and MURATLI, ÖMER CAN. Performance tuning for machine learning-based software development effort prediction models, Turkish Journal of Electrical Engineering & Computer Sciences,27, pp. 1308–1324, 2019.

[10] Rahaman, Syed Mujib, and V. Valli Kumari. "A Model for Corrective Software Maintenance Effort Estimation after Privacy Leak Detection in Social Network." In 2020 International Conference on Artificial Intelligence and Signal Processing (AISP),IEEE,pp.1-10, 2020.

[11] Singh, Chamkaur, Neeraj Sharma, and Narender Kumar. "Analysis of software maintenance cost affecting factors and estimation models." International Journal of Scientific and Technology Research 8, no. 9 ,pp. 276-281, 2019.

[12] Sakhrawi, Zaineb, Asma Sellami, and Nadia Bouassida. "Software Enhancement Effort Prediction Using Machine-Learning Techniques: A Systematic Mapping Study." SN Computer Science 2, no. 6, pp.1-15, 2021.

[13] Sakhrawi, Zaineb, Asma Sellami, and Nadia Bouassida. "Requirements Change Requests Classification: An Ontology-Based Approach." In International Conference on Intelligent Systems Design and Applications, pp. 487-496. Springer, Cham, 2019.

[14] Sakhrawi, Zaineb, Asma Sellami, and Nadia Bouassida. "Software Enhancement Effort Estimation using Machine Learning Regression Methods."V 12,pp. 412-423, 2020.

[15] Sakhrawi, Zaineb, Asma Sellami, and Nadia Bouassida. "Investigating the Impact of Functional Size Measurement on Predicting Software Enhancement Effort Using Correlation-Based Feature Selection Algorithm and SVR Method." In International Conference on Software and Software Reuse, pp. 229-244. Springer, Cham, 2020.

[16] Sakhrawi, Zaineb, Asma Sellami, and Nadia Bouassida. "Support vector regression for enhancement effort prediction of Scrum projects from COSMIC functional size." Innovations in Systems and Software Engineering, pp. 1-17, 2021.

[17] Sakhrawi, Zaineb, Asma Sellami, and Nadia Bouassida. "An Improved Prediction of Software Enhancement Effort using Correlation-based Feature Selection and M5P ML algorithm." In 2020 IEEE/ACS 17th International Conference on Computer Systems and Applications (AICCSA), pp. 1-8, IEEE, 2020.

[18] Sakhrawi, Zaineb, Asma Sellami, and Nadia Bouassida. "Software enhancement effort estimation using correlation-based feature selection and stacking ensemble method." Cluster Computing pp. 1-14, 2021.

[19] Basri, Sufyan and Kama, Nazri and Ibrahim, Roslina, «A Novel Effort Estimation Approach for Requirement Changes during Software Development Phase,» International Journal of Software Engineering and Its Applications, pp. 237-252, 2015.

[20] Fairley, Richard E. Managing and leading software projects. John Wiley Sons, 2011.

[21] Boehm, Barry W and Abts, Chris and Brown, A Winsor and Chulani, Sunita and Clark, Bradford K and Horowitz, Ellis and Madachy, Ray and Reifer, Donald J and Steece, Bert, «Software Cost Estimation with COCOMO II,» Prentice Hall PTR, 2000.

[22] S.D. Sheetz, D. Henderson and L. Wallace, "Understanding developer and manager perceptions of function points and source lines of code", The Journal of Systems and Software, Elsevier, vol. 82, pp. 1540-1549, 2009.

[23] González-Ladrón-de-Guevara, Fernando and Fernández-Diego, Marta and Lokan, Chris. The usage of ISBSG data fields in software effort estimation: A systematic mapping study, Journal of Systems and Software,113, pp.188–215, 2016.

[24] IFPUG International Function Point Users Group, A Functional Size Measurement Method,COSMIC and IFPUG Glossary of terms,2011.

[25] IFPUG International Function Point Users Group, Common Software Measurement International Consortium, COSMIC and IFPUG Glossary of terms,2015.

[26] Ali, Syed Sarmad, Muhammad Shoaib Zafar, and Muhammad Tallal Saeed. "Effort Estimation Problems in Software Maintenance–A Survey." In 2020 3rd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET), pp. 1-9. IEEE, 2020.

[27] Pressman, Roger S. Software engineering: a practitioner's approach. Palgrave macmillan, 2005.

[28] Hugo, Dionisio Ramón. "Practical software measurement. Objective information for decision makers." Journal of Computer Science and Technology 3, no. 2, p.70, 2003.

[29] Rook, Paul. "Controlling software projects." Software engineering journal 1, no. 1, pp. 7-16, 1986.

[30] Symons, C. "A comparison of the key differences between the IFPUG and cosmic functional size measurement methods." Common Software Measurement International Consortium, 2011.

[31] Alain Abran, Jean-Marc Desharnais, Barbara Kitchenham, Dylan Ren, Charles Symons,Steve Woodward, Diana Baklizky, Peter Fagg, Arlan Lesterhuis, Luca Santillo, Frank Vogelezang, Carol Buttle, Cigdem Gencel, Roberto Meli, Hassan Soubra, and Chris Woodward, Guideline on Non-Functional and Project Requirements: How to Consider non-functional and Project Requirements in Software Project Performance Measurement, Benchmarking and Estimating,2015.

[32] Abran, Alain. Software project estimation: the fundamentals for providing high quality information to decision makers, Information and Software Technology, 2015.

[33] Mall, Rajib. Fundamentals of software engineering. PHI Learning Pvt. Ltd., 2018.

[34] Alain Abran, Bernard Londeix , Marie O'Neill, Luca Santillo, Frank Vogelezang, Jean-Marc Desharnais, Pam Morris et al. "The cosmic functional size measurement method." Measurement Manual, Version 4, no. 1, 2015.

[35] Mahmood, Yasir, Nazri Kama, Azri Azmi, Ahmad Salman Khan, and Mazlan Ali. "Software effort estimation accuracy prediction of machine learning techniques: A systematic performance evaluation." Software: Practice and Experience 52, no. 1, pp. 39-65, 2022.

[36] Nerkar, L. R., and P. M. Yawalkar. "Software Cost Estimation using Algorithmic Model and Non-Algorithmic Model a Review." Int J Comput App 2, pp. 4-7, 2014.

[37] Sangwan, Om Prakash. "Software effort estimation using machine learning techniques." In 2017 7th International Conference on Cloud Computing, Data Science Engineering-Confluence, pp. 92-98. IEEE, 2017.

[38] Zhang, Xian-Da. "Machine learning." In A Matrix Algebra Approach to Artificial Intelligence, pp. 223-440. Springer, Singapore, 2020.

[39] Kulmanov, Maxat, Fatima Zohra Smaili, Xin Gao, and Robert Hoehndorf. "Semantic similarity and machine learning with ontologies." Briefings in bioinformatics 22, no. 4, pp. bbaa199, 2021.

[40] Rashwan, Abderahman, Olga Ormandjieva, and René Witte. "Ontology-based classification of non-functional requirements in software specifications: a new corpus and

svm-based classifier." In 2013 IEEE 37th Annual Computer Software and Applications Conference, pp. 381-386. IEEE, 2013.

[41] Blessie, E. Chandra, and E. Karthikeyan. "Sigmis: A feature selection algorithm using correlation based method." Journal of Algorithms Computational Technology 6, no. 3, pp. 385-394,2012.

[42] Hall, Mark Andrew, "Correlation-based feature selection for machine learning",Citeseer,113, pp.1–8, 1999.

[43] Breiman, Leo. Random forests, Machine learning,45, pp. 5–32, 2019.

[44] Cortes, Corinna and Vapnik, Vladimir. Support-vector networks, Machine learning,20, pp.273–297, 1995.

[45] Freund, Yoav and Schapire, Robert E. A desicion-theoretic generalization of online learning and an application to boosting, European conference on computational learning theory,5, pp. 23–37, 1995.

[46] Freund, Yoav and Schapire, Robert and Abe, Naoki. A short introduction to boosting, Journal-Japanese Society For Artificial Intelligence,14, pp. 771-780, 1999.

[47] Hidmi, Omar and Sakar, Betul Erdogdu, Robert and Abe, Naoki.Software development effort estimation using ensemble machine learning, International Journal Computer Communication and Instrument Engineering,no. 4, pp. 143–147, 2017.

[48] Quinlan, John R, "Learning with continuous classes," I5th Australian joint conference on artificial intelligence, vol. 92, pp. 343–348, 1992.

[49] Idri, Ali and Hosni, Mohamed and Abran, Alain, Systematic literature review of ensemble effort estimation,Journal of Systems and Software, vol.118, pp.151–175,2016

[50] Shukla, Suyash and Kumar, Sandeep, A Stacking Ensemble-based Approach for Software Effort Estimation, ENASE, pp.205–212, 2021

[51] Alsolai, Hadeel, and Marc Roper. "A systematic literature review of machine learning techniques for software maintainability prediction." Information and Software Technology 119, pp. 106214, 2020.

[52] Devulapally, Gopi Krishna, Agile in the context of Software Maintenance A Case Study, Thesis no: MSSE-2015-12,2015.

[53] Choudhari, Jitender, and Ugrasen Suman. "Iterative maintenance life cycle using extreme programming." In 2010 International Conference on Advances in Recent Technologies in Communication and Computing, pp. 401-403. IEEE, 2010.

[54] Merzouk, Soukaina, Abdessamad Cherkaoui, Abdelaziz Marzak, Nawal Sael, and Fatima-Zahra Guerss. "The proposition of Process flow model for Scrum and eXtreme Programming." In Proceedings of the 4th International Conference on Networking, Information Systems Security, pp. 1-6. 2021.

[55] Arora, Mohit and Verma, Sahil and Chopra, Shivali and others, A Systematic Literature Review of Machine Learning Estimation Approaches in Scrum Projects, Cognitive Informatics and Soft Computing, pp.573-586, 2020.

[56] David J. Anderson, Kanban: Successful Evolutionary Change for Your Technology Business,Blue Hole Press, 261 pages, 2010.

[57] Cervone, H Frank, Understanding agile project management methods using Scrum, OCLC Systems and Services: International digital library perspectives, 2011

[58] Choudhari, Jitender and Suman, Ugrasen, points based effort estimation model for software maintenance, Procedia Technology, 4, pp.761–765,2012.

[59] Grenning, James, Planning poker or how to avoid analysis paralysis while release planning, Hawthorn Woods: Renaissance Software Consulting, 2002.

[60] Lavazza, Luigi, and Sandro Morasca. "On the evaluation of effort estimation models." In Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering, pp. 41-50. 2017.

[61] Richardson, Gary L. Project management theory and practice. Crc Press, 2010.

[62] Nassif, Ali Bou, Mohammad Azzeh, Ali Idri, and Alain Abran. "Software development effort estimation using regression fuzzy models." Computational intelligence and neuroscience 2019 (2019).

[63] Ulziit, Bayarbuyan, Zeeshan Akhtar Warraich, Cigdem Gencel, and Kai Petersen. "A conceptual framework of challenges and solutions for managing global software maintenance." Journal of Software: Evolution and Process 27, no. 10 pp. 763-792, 2015.

[64] Heričko, Marjan and Živkovič, Aleš, The size and effort estimates in iterative development, Information and Software Technology, vol. 50, no. 7, pp. 772-781, 2008.

[65] Malhotra, Ruchika, A systematic review of machine learning techniques for software fault prediction, Applied Soft Computing, vol. 27, pp. 504-518, 2015.

[66] Kitchenham, Barbara, O. Pearl Brereton, David Budgen, Mark Turner, John Bailey, and Stephen Linkman. "Systematic literature reviews in software engineering–a

systematic literature review." Information and software technology 51, no. 1, pp. 7-15, 2009.

[67] López-Martín, Cuauhtémoc. Predictive accuracy comparison between neural networks and statistical regression for development effort of software projects, Applied Soft Computing, 27, pp. 434–449 , 2015.

[68] Ku, Yan, Jing Du, Ye Yang, and Qing Wang. "Estimating software maintenance effort from use cases: An industrial case study." In 2011 27th IEEE International Conference on Software Maintenance (ICSM), pp. 482-491. IEEE, 2011.

[69] Wu, Hong, Lin Shi, Celia Chen, Qing Wang, and Barry Boehm. "Maintenance effort estimation for open source software: A systematic literature review." In 2016 IEEE International Conference on Software Maintenance and Evolution (ICSME), pp. 32-43. IEEE, 2016.

[70] De Lucia, Andrea, Eugenio Pompella, and Silvio Stefanucci. "Assessing effort estimation models for corrective maintenance through empirical studies." Information and Software Technology 47, no. 1, pp. 3-15, 2005.

[71] Leung, Hareton KN. "Estimating maintenance effort by analogy." Empirical Software Engineering 7, no. 2, pp. 157-175, 2002.

[72] Fioravanti, Fabrizio, and Paolo Nesi. "Estimation and prediction metrics for adaptive maintenance effort of object-oriented systems." IEEE Transactions on software engineering 27, no. 12, pp. 1062-1084, 2001.

[73] Jorgensen, Magne. "Experience with the accuracy of software maintenance task effort prediction models." IEEE Transactions on software engineering 21, no. 8, pp. 674-681, 1995.

[74] Ramil, Juan F and Lehman, Meir M, "Metrics of software evolution as effort predictors—a case study," Proceedings International Conference on Software Maintenance: IEEE Computer Society Press: Los Alamitos CA, pp. 163-172, 2000

[75] Agrawal, Manish and Chari, Kaushal, "Software effort, quality, and cycle time: A study of CMM level 5 projects," IEEE Transactions on software engineering, vol. 33, n1IEEE, pp. 145–156, 2007.

[76] Riaz, Mehwish and Mendes, Emilia and Tempero, Ewan, "A systematic review of software maintainability prediction and metrics," 2009 3rd International Symposium on Empirical Software Engineering and Measurement, pp. 367-377, 2009.

[77] Quah, Tong-Seng and Thwin, Mie Mie Thet, "Application of neural networks for software quality prediction using object-oriented metrics," Journal of systems and software, vol. 76, pp. 147–156, 2005.

[78] Zhou, Yuming and Leung, Hareton, "Predicting object-oriented software maintainability using multivariate adaptive regression splines," Journal of systems and software, vol. 80, pp. 1349–1361, 2007.

[79] Shukla, Ruchi and Misra, AK, "Ai based framework for dynamic modeling of software maintenance effort estimation," 2009 International Conference on Computer and Automation Engineering, IEEE, pp. 313–317, 2009.

[80] Bhatnagar, Roheet and Bhattacharjee, Vandana and Ghose, Mrinal Kanti, "Software development effort estimation–neural network vs. regression modeling approach," International Journal of Engineering Science and Technology, pp. 2950–2956, 2010.

[81] Stojanov, Zeljko and Dobrilovic, Dalibor and Stojanov, Jelena and Jevtic, Vesna, "Estimating software maintenance effort by analyzing historical data in a very small software company," Scientific Bulletin of The Politehnica University of Timioara, Transactions on Automatic Control and Computer Science, p. 2, 2013

[82] Malhotra, Ruchika and Chug, Anuradha, "Software maintainability prediction using machine learning algorithms," Software Engineering: An International Journal (SEIJ), 2012.

[83] Ahmed, Moataz A and Al-Jamimi, Hamdi A, "Machine learning approaches for predicting software maintainability: a fuzzy-based transparent model," IET software, pp. 317–326, 2013.

[84] Malhotra, Ruchika and Lata, Kusum, "An exploratory study for predicting maintenance effort using hybridized techniques," Proceedings of the 10th Innovations in Software Engineering Conference, pp. 26–33, 2017

[85] Malhotra, Ruchika and Lata, Kusum, "On the Application of Cross-Project Validation for Predicting Maintainability of Open Source Software using Machine Learning Techniques," 2018 7th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO), pp. 175–181, 2018.

[86] Shukla, Ruchi and Shukla, Mukul and Misra, Arun Kumar and Marwala, Tshilidzi and Clarke, WA, "Dynamic software maintenance effort estimation modeling using

neural network, rule engine and multi-regression approach," International Conference on Computational Science and Its Applications, vol. 15, p. 157–169, 2012.

[87] Shukla, Ruchi and Misra, Arun Kumar, "Estimating Software Maintenance Effort - A Neural Network Approach," Proceedings of 1st India Software Engineering Conference, pp. 107-112, 2008.

[88] Song, Tae-Hoon and Yoon, Kyung-A and Bae, Doo-Hwan. An approach to probabilistic effort estimation for military avionics software maintenance by considering structural characteristics, 14th Asia-Pacific Software Engineering Conference (APSEC'07), pp. 406–413, 2007.

[89] Yu, Liguo,"Indirectly predicting the maintenance effort of open-source software," Journal of Software Maintenance and Evolution: Research and Practice,vol. 18, pp. 311–332,2006.

[90] Cerón-Figueroa, Sergio and López-Martín, Cuauhtémoc and Yáñez-Márquez, Cornelio. Stochastic gradient boosting for predicting the maintenance effort of software-intensive systems, IET Software,pp. 99–109 , 2019.

[91] Rijwani, Poonam and Jain, Sonal, "Enhanced Software Effort Estimation using Multi Layered Feed Forward Artificial Neural Network Technique," Twelfth International Multi-Conference on Information Processing-2016 (IMCIP-2016), pp. 307–312, 2016.

[92] Jane Huffman Hayes, Sandip C. Patel, Liming Zhao,A Metrics-Based Software Maintenance Effort Model, Proceedings of IEEE Eighth European Conference on Software Maintenance and Reengineering (CSMR'04), pp. 254-258, 2004.

[93] Kitchenham, Barbara and Pfleeger, Shari Lawrence and McColl, Beth and Eagan, Suzanne, "An empirical study of maintenance and development estimation accuracy,"The Journal of Systems and Software, Elsevier, pp. 57-77, 2002.

[94] Abdallah, Ammar and Abran, Alain, Enterprise Architecture Measurement: An Extended Systematic Mapping Study, I.J. Information Technology and Computer Science, 9, pp. 9-19, 2019.

[95] Chua, Bee Bee and Bernardo, Danilo Valeros and Verner, June, "Criteria for Estimating Effort for Requirements Changes," conf/eurospi/2008, pp. 36-46, 2008.

[96] Zielczynski, Peter, "Requirements Management Using IBM Rational RequisitePro," IBM Press, 2007.

[97] De Andrés, Javier and Landajo, Manuel and Lorca, Pedro, "Using Nonlinear Quantile Regression for the Estimation of Software Cost," HAIS2018, Oviedo, Spain, June 20-22, 2018, Proceedings, pp. 422–432, 2018.

[98] Nassif, Ali Bou, Luiz Fernando Capretz, and Danny Ho. "Analyzing the non-functional requirements in the desharnais dataset for software effort estimation." arXiv preprint arXiv:1405.1131, 2014.

[99] Nguyen, Vu and Boehm, Barry and Danphitsanuphan, Phongphan, "A controlled experiment in assessing and estimating software maintenance tasks",Information and software technology,53,6,pp. 682–691,2011.

[100] Ahn, Yunsik and Suh, Jungseok and Kim, Seungryeol and Kim, Hyunsoo, "The software maintenance project effort estimation model based on function points," Journal of Software Maintenance and Evolution: Research and Practice, p. 71–85, 2003.

[101] Sammut, Claude and Webb, Geoffrey I, "Encyclopedia of machine learning," Springer Science and Business Media, 2011.

[102] Jović, Alan, Karla Brkić, and Nikola Bogunović. "A review of feature selection methods with applications." In 2015 38th international convention on information and communication technology, electronics and microelectronics (MIPRO), pp. 1200-1205. Ieee, 2015.

[103] Elmidaoui, Sara and Cheikhi, Laila and Idri, Ali and Abran, Alain, "Machine learning techniques for software maintainability prediction: Accuracy analysis", Journal of Computer Science and Technology, V.35,Num.5,pp.1147–1174,2020

[104] Minku, Leandro L and Yao, Xin, "A Principled Evaluation of Ensembles of Learning Machines for Software Effort Estimation," Proceedings of the 7th International Conference on Predictive Models in Software Engineering, pp. 1–10, 2011.

[105] Group, International Software Benchmarking Standards, "Glossary of terms for software project development and enhancement," version 5.17, vol. 113, pp. 188–215, 2018.

[106] Bajwa, Sohaib-Shahid. Investigating the nature of relationship between software size and development effort, International Journal of Computer Applications,2008.

[107] Jones, Karen Sparck. A statistical interpretation of term specificity and its application in retrieval. Journal of documentation 28(1), pp. 11-20, 1972.

[108] Desharnais, Jean-Marc and Buglione, Luigi and Kocatürk, Buğra, Using the COSMIC method to estimate Agile user stories, Proceedings of the 12th International Conference on Product Focused Software Development and Process Improvement, pp.68-73, 2011

[109] Bala, Abdalla and Abran, Alain, "Use of the multiple imputation strategy to deal with missing data in the ISBSG repository",Journal of Information Technology & Software Engineering, vol.6, pp. 171, 2016.

[110] Biesiada, Jacek and Duch, Wlodzisław, Feature selection for high-dimensional data—a Pearson redundancy based filter, Computer recognition systems, pp.242-249, 2007.

[111] ,Shepperd, Martin, and Steve MacDonell. "Evaluating prediction systems in software project estimation." Information and Software Technology 54, no. 8, pp. 820-827,2012.

[112] Bhardwaj, Mridul and Ajay, Rana, Estimation of Testing and Rework Efforts for software Development Projects, Asian Journal of Computer Science and Information Technology, 5, 5, pp.33-37, 2015

[113] Sellami, Asma, Mariem Haoues, Nour Borchani, and Nadia Bouassida. "Towards an Assessment Tool for Controlling Functional Changes in Scrum Process." In IWSM-Mensura, pp. 34-47. 2018.

[114] ,Moritz, Steffen and Bartz-Beielstein, Thomas, time series missing value imputation in R, The R Journal, 9, 1, pp.1-12,2017

[115] Cooper, Robert G and Sommer, Anita F, Agile-Stage-Gate: New idea-to-launch method for manufactured new products is faster, more responsive,Industrial Marketing Management,59, pp.167–180,2016

[116] Angara, Jayasri and Prasad, Srinivas and Sridevi, Gutta, Towards Benchmarking User Stories Estimation with COSMIC Function Points-A Case Example of Participant Observation, International Journal of Electrical and Computer Engineering (IJECE), pp. 3076-3083, 2018.

[117] Charles Symons, Alain Abran, Christof Ebert,Frank Vogelezang, Measurement of Software Size: Advances Made by the COSMIC Community, International Conference on Software Process and Product Measurement (IWSM-MENSURA), 2016.

[118] Yohannese, Chubato Wondaferaw, Tianrui Li, and Kamal Bashir. "A three-stage based ensemble learning for improved software fault prediction: an empirical comparative study." International Journal of Computational Intelligence Systems 11, no. 1, pp. 1229-1247,2018

[119] Field, Andy. Discovering statistics using IBM SPSS statistics. sage, 2013.

[120] Yadav, Sanjay and Shukla, Sanyam, Analysis of k-fold cross-validation over hold-out validation on colossal datasets for quality classification,2016 IEEE 6th International conference on advanced computing (IACC), vol.6, pp.78–83, 2016.

[121] Gian Antonio Susto, Andrea Schirru, Simone Pampuri, Machine Learning for Predictive Maintenance:A Multiple Classifier Approach, ieee transactions on industrial informatics,pp. 812-820 , 2015.

[122] Wen, Jianfeng, Shixian Li, Zhiyong Lin, Yong Hu, and Changqin Huang. "Systematic literature review of machine learning based software development effort estimation models." Information and Software Technology 54, no. 1, pp.41-59, 2012.

[123] Wang, Leizhi and Zhu, Zhenduo and Sassoubre, Lauren and Yu, Guan and Liao, Chen and Hu, Qingfang and Wang, Yintang, Improving the robustness of beach water quality modeling using an ensemble machine learning approach,Science of The Total Environment, vol.765, pp.142-760,2021

[124] Haoues, Mariem and Sellami, Asma and Ben-Abdallah, Hanêne,"Towards functional change decision support based on COSMIC FSM method,"Information and Software Technology, vol. 110, pp.78–91,2019.

[125] Guney, Zafer.Considerations for human-computer interaction: user interface design variables and visual learning in IDT." Kıbrıslı Eğitim Bilimleri Dergisi 14, no. 4 pp. 731-741,2019

[126] Sellami, Asma, Hela Hakim, Alain Abran, and Hanêne Ben-Abdallah. "A measurement method for sizing the structure of UML sequence diagrams." Information and Software Technology 59, pp. 222-232, 2015

[127] Labidi Taher, Sakhrawi Zaineb, Sellami Asma, Achraf Mtibaa and Bouassida Nadia, “On the Use of OLS regression algorithm and Pearson correlation algorithm for improving the SLA establishment process”, Innovations in Systems and Software Engineering, pp.1-15, 2022

Appendix A: Primary Studies

Appendix A - Quality assessment results

Study	**MQ 1**	**MQ 2**	**MQ 3**	**MQ 4**	**MQ 4.1**	**MQ 4.2**	**MQ 4.3**	**MQ 4.4**	**Total score**	**Rating**
S1 [93]	Y	Y	Y	Y	Y	Y	P	P	7	Excellent
S2 [68]	Y	P	N	Y	P	P	P	N	4	Good
S3 [99]	Y	N	N	Y	P	N	Y	N	3.5	Good
S4 [89]	Y	P	Y	Y	P	P	N	P	5	Good
S5 [92]	Y	P	Y	Y	P	P	P	P	5.5	Good
S6 [100]	Y	Y	P	P	P	P	Y	Y	6	Good
S7 [71]	Y	P	P	P	P	P	N	N	3.5	Good
S8 [72]	Y	N	P	P	Y	P	P	N	4	Good
S9 [73]	Y	P	Y	P	Y	P	P	P	5.5	Good
S10 [74]	Y	P	P	P	P	Y	P	P	5	Good
S11 [90]	Y	P	Y	Y	Y	P	Y	Y	7	Excellent
S12 [5]	Y	P	Y	Y	Y	P	Y	Y	7	Excellent
S13 [91]	Y	P	Y	Y	Y	P	Y	Y	7	Excellent

Appendix B: Example of Ontology's class and its corresponding customer's review

Appendix B - Example of Ontology's class and its corresponding customer's review from PROMISE

Class	Customer Enhancement Request (from PROMISE Review)
FUNCTIONAL	product shall update existing conference rooms
FUNCTIONAL	product able delete conference rooms
FUNCTIONAL	product shall able delete room equipment
FUNCTIONAL	product shall maintain list players
CONSTRAINT	product must comply Sarbanes Oxley
CONSTRAINT	product shall comply estimatics laws relating recycled parts usage
CONSTRAINT	product shall comply insurance regulations regarding claims processing
CONSTRAINT	System shall meet applicable accounting standards final version System must successfully pass independent audit performed certified auditor
COMPATIBILITY	Dont waste money installed threeGs took forever load tap blank white screen pissed
MAINTAINABILITY	product shall expected operate least five years customer installation
PERFORMANCE	System shall administrator add remove categories website five minutes
PORTABILITY	product shall interface ChoiceParts system day approximately
RELIABILITY	movies shall streamed demand time
SECURITY	updates data files database must initiated Disputes System
USABILITY	product shall make users want use eighty percent users surveyed report regularly using product first two weeks

Appendix C: User story Functional sizing example

Appendix C - COSMIC functional change Size and its corresponding effort in terms of Story points

US Id	User Story description			COSMIC Functional Change description						
	As an actor...)	I can ...(Goal)	Effort (Story Points)	Type (add, delete, or modify)	FC description	E	R	W	X	Total Size (CFP)
1	O-User User	Add a custom evidence type to an assessment criterion (because the standard evidence types are not appropriate for me)	4	ADD	Add Custom Evidence Type	2	1	1	2	6

2	XYZ ABC User	Create bulk emails to be sent to users	5	ADD	Create Bulk Email	2	1	1	2	6
4	O-User	Add an improvement action against a specific attainment criterion	5	ADD	Add Improvement Action	3	1	1	2	7
5	Assessment Completer	Add and amend general comments for my requirement response	3	ADD	Add Widget Comments	1	1	1	2	5
6	General User	View an organisation's Assessment Report with various enhancements	4	MODIFY	View Improved Assessment Report	4	1	0	3	8

7	O-User	Create, amend and delete users within my organisation user (using the simplified interface)	3	MODIFY	Maintain Users (Simplified Interface)	2	1	1	2	6
8	XYZ ABC User	Delete a pending bulk email	1	DELETE	Delete Bulk Email	2	1	0	2	5
9	General User	View Improved Assessment Report - HTML Version	3	ADD	Create an HTML version of the assessment report	2	1	0	2	5
Sprint 1 total effort			28	Sprint 1 total Functional change size						48

*Note: O-User refers to as Organisation User

Publications

[1] Labidi, Taher, Zaineb Sakhrawi, Asma Sellami, and Achraf Mtibaa. "An Ontology-based approach for preventing incompatibility problems of quality requirements during cloud SLA establishment." In International Conference on Computational Collective Intelligence, pp. 663-675. Springer, Cham, 2019. *Rank B*

[2] Sakhrawi, Zaineb, Asma Sellami, and Nadia Bouassida. "Requirements Change Requests Classification: An Ontology-Based Approach." In International Conference on Intelligent Systems Design and Applications, pp. 487-496. Springer, Cham, 2019. *Rank C*

[3] Sakhrawi, Zaineb, Asma Sellami, and Nadia Bouassida. "Software Enhancement Effort Estimation using Machine Learning Regression Methods." Volume 12, 2020, pp. 412 - 423. *SJR: Q4*

[4] Sakhrawi, Zaineb, Asma Sellami, and Nadia Bouassida. "An Improved Prediction of Software Enhancement Effort using Correlation-based Feature Selection and M5P ML algorithm." In 2020 IEEE/ACS 17th International Conference on Computer Systems and Applications (AICCSA), pp. 1-8. IEEE, 2020. *Rank C*

[5] Sakhrawi, Zaineb, Asma Sellami, and Nadia Bouassida. "Investigating the Impact of Functional Size Measurement on Predicting Software Enhancement Effort Using Correlation-Based Feature Selection Algorithm and SVR Method." In International Conference on Software and Software Reuse, pp. 229-244. Springer, Cham, 2020. *Rank B*

[6] Sakhrawi, Zaineb, Asma Sellami, and Nadia Bouassida. "Software Enhancement Effort Prediction Using Machine-Learning Techniques: A Systematic Mapping Study." SN Computer Science 2, no. 6, pp. 1-15, 2021. *RANK 2020; indexed dblp and google scholar*

[7] Sakhrawi, Zaineb, Asma Sellami, and Nadia Bouassida. "Support vector regression for enhancement effort prediction of Scrum projects from COSMIC functional size." Innovations in Systems and Software Engineering, pp. 1-17,2021. *RANK 2021; SJR:Q3*

[8] Sakhrawi, Zaineb, Asma Sellami, and Nadia Bouassida. "Software enhancement effort estimation using correlation-based feature selection and stacking ensemble method." Cluster Computing, pp.1-14,2021. *RANK 2021; SJR:Q2,IF:1.8*

[9] Labidi Taher, Sakhrawi Zaineb, Sellami Asma, Achraf Mtibaa and Bouassida Nadia, “On the Use of OLS regression algorithm and Pearson correlation algorithm for improving the SLA establishment process”, Innovations in Systems and Software Engineering, pp. 1-15, 2022. *RANK 2021; SJR:Q3*

[10] Sakhrawi, Z.; Sellami, A. and Bouassida, N. "Software Enhancement Effort Estimation using Stacking Ensemble Model within the Scrum Projects: A Proposed Web Interface". In Proceedings of the 17th International Conference on Software Technologies, ISBN 978-989-758-588-3, ISSN 2184-2833, pages 91-100, 2022. *Rank B*