INTERNATIONAL JOURNAL OF SOFTWARE ENGINEERING AND COMPUTER SYSTEMS
E-ISSN: 2180-0650
VOLUME 12, ISSUE 1, 2026, pp. 27 – 38
DOI: https://doi.org/10.15282/ijsecs.12.1.2026.3.0149

RESEARCH ARTICLE

System Cost Estimating in Software Product Lines using Feature-Oriented Approach

N. Amougou[1,] *, M. Fouda[1]

[1]Higher Teacher Training College, University of Yaounde I, PO Box 337 Yaounde, Cameroon

ABSTRACT - The feature business component of a software product line in its solution part shows services provided by that software product line in a feature model which is an AND/OR graph. To be implemented, these services can require Commercial Off-The-Shelf Products (COTS). Therefore, evaluate the cost of a system in Software Product Line (SPL) is difficult since many COTS are evolved in features at different levels of the feature model. Currently, many approaches had been proposed to evaluate software project cost such as algorithmic methods (e.g.: Constructive Cost Model (COCOMO), Software Life-Cycle Management (SLIM), Functional Point Analysis (FPA), etc.) and non-algorithmic Methods (e.g.: Delphi, rule-based, learning, etc.). However, COTS are independent of the project and their costs are dynamic. In this work, we suggest a methodology and propose algorithms to approximate cost of a product belonging to a given software product line, the minimal cost and the maximal cost. The novelty of this work lies in the fact that we enrich feature business components by adding new knowledge when analyzing business in a given domain, like required COTS and their costs for each feature at different levels of the feature model. The new model of feature business components allows proposing algorithms for system cost estimation.

ARTICLE HISTORY
Received : 03 August 2025
Revised : 18 December 2025
Accepted : 24 January 2026
Published : 04 May 2026

KEYWORDS
System cost estimation
Software Product Line
Domain analysis
Reuse
Feature-orientation

1.0 INTRODUCTION

In software engineering, master resources, effort and time needed as well as the cost of the software product is not easy for the project manager. To tackle this concern, software product line engineering [1] has been proposed in order to construct applications more easily, more quickly and with a better quality. In [2], a Software Product Line (SPL) is defined by authors as "a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way". However, software cost estimation is a very challenging activity in the project management of software because predicting the cost is a difficult process at the early stage of the software's development [3, 4]. Moreover, the estimation of the software's cost is impacted by many factors, including the implementation's efficiency, and the number of reviews and studies done prior to the development stage cost. There is clearly a strong relationship that exists between the estimation of software effort and software cost, as it can be said that the effort is the primary driving factor for the software's cost [5-8].

Most existing techniques are used to estimate cost, effort, and duration. These techniques are also easy to understand and apply to software projects, but their failure can lead to inaccurate and wrong estimates [9]. Estimation techniques can be enhanced by using hybrid and combined two and more techniques to improve the accuracy. Therefore, the accuracy of cost estimation techniques is a major challenge. For a system in a software product line, cost estimation faces more challenges than traditional systems. The first reason is that multiples COTS can be involved to perform features at different levels and secondly COTS are independent for the current project and their costs are dynamic. In this work, the base is featuring models [10] which are hierarchical models that capture the commonality and variability of a Product Line (PL) by defining its feature, their dependencies and constraint between them. An example of a feature model with constraints associated to a Smart Home PL is shown in Figure 1. This example is interesting since it shows all sorts of features such as mandatory features, optional features, alternative features and or group features.

The elaboration of the Feature-Oriented Reuse Method with Business Component Semantics (FORM/BCS) [11-16] allowed us to extend feature models. Each system belonging to the product line is derived from a selection of a valid combination of features [17], product configuration is the name of this process [18, 19]. To estimate SPL-Based system cost, we propose a coupled methodology. Activities of our research method involve enriching formally the concept of feature business components in the FORM/BCS method, to add new knowledge when analyzing a domain. Additional knowledge includes required COTS and their costs for each feature at different levels of the feature business component solution part which is a feature model. The added new information allows propose algorithms to estimate the cost for a system belonging to a software product line and to determine the system with the minimal cost and the product with the maximal cost.

***CORRESPONDING AUTHOR | N. Amougou** | ✉ 200204006@hbtu.ac.in

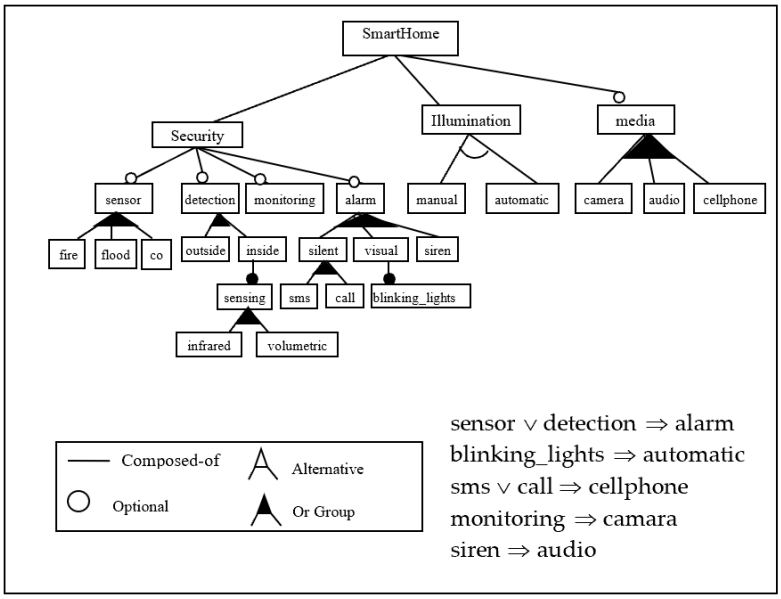


Figure 1. Extract of Smart home feature model with constraints

The article is organized as follows. We review, in section 2 current approaches to software cost estimation. Section 3 exposes our proposal. A case study is presented in Section 4 and finally section 5 is devoted for the conclusion and perspectives of this research.

2.0 RELATED WORKS

In Software Cost Estimation, techniques can be divided in two principal broad groups. These are those that use the source lines of codes (SLOC) as their input and others that do not. In [20], the author describes the mechanisms and features of five approaches. This very important technique employs an algorithmic formula in order to estimate the software's cost [21]. Organizations use COCOMO, as indicated in references [22-25] to estimate cost, effort, and schedule when planning new software development activities. The approach which is called the feed-forward neural network with Principal Component Analysis (PCA) is a reduction technique proposed by authors in [26]. The main objective of authors is to use this technique in order to measure the accuracy of the software cost estimation model. The proposed approach is based on both algorithmic and non-algorithmic methods. So, to estimate the software project's costs, authors used a combination of algorithmic method (COCOMO) and non-algorithmic one (Artificial Neural networks).

The SLIM approach, developed by Larry Putnam of Quantitative Software Management in the late 1970s is considered as an experimental software effort testing model. Details for this approach are given in [27-30]. This technique elaborates the effort and time that is supposed to be applied for a project of specified size. Software life-cycle management (SLIM) is closely related in software parametric to COCOMO model. Advantages and drawbacks of SLIM are given in [20]. This method allows measuring complexity and size of software, in terms of which functions it is able to provide to the user. This approach was developed by Allan Albrecht at IBM and was proposed to the scientific community the first time in the year 1976 [31, 32]. Two steps are necessitated in counting the Function Points [28], which are counting the various user functions and making adjustments for processing the complexity. FPA has many advantages which can be find in [33, 34].

To make an estimation of the software development effort, authors employed Wavelet Neural Network (WNN) in [35]. Experiments were made on two variants of WNN, the Morlet wavelet function [36], the Gaussian function, and a threshold acceptance training algorithm for wavelet neural network (TAWNN) that had been proposed. The results of these were compared with other computational intelligent methods, such as the Multilayer Perceptron (MLP) [37], the Radial Basis Function Network (RBFN) [38], the Multiple Linear Regression (MLR) [39], the Dynamic Evolving Neuro-Fuzzy Inference System (DENFIS) [40] and the Support Vector Machine (SVM) [41]. All of these comparisons were computed in terms of the Mean Magnitude Relative Error "MMRE" applied on both the Canadian Financial "CF" dataset and the IBM Data Processing Services "IBMDPS" dataset.

Recent efforts are made to propose hybrid approaches for software project cost estimation notably in [42], a hybrid cost estimation method for planning software projects using fuzzy logic and machine learning is suggested, in [43], a hybrid machine learning method for estimating software project cost is proposed, and in [44], a software cost estimation in global software development using hybrid approach is presented. All these approaches have the potential to have a substantial impact on software cost estimation. However, their implementation is deferred to future works.

A first comparison of the previous approaches was done in [20] following certain criteria such as: Ease of Use, Adaptability, Accuracy, Consistency, Interpretable, Automatable, Tool Supported Supportability, Empirical Validation,

Sensitivity and Handling Imprecision and Uncertainty. The author classifies these methods from the most important to the least important one as follows: COCOMO Model, Hybrid approach this is based on both algorithmic and non-algorithmic methods, Putman Model, Function Point Analysis and Wavelet Neural Network. Another comparison study of software estimation techniques is done in [9]. For this second comparison, if we consider the accuracy criteria, COCOMO, SLIM and Robust approaches are good. A general observation for all these approaches is that none of them takes into account the existence of COTS.

3.0 SPL-BASED SYSTEM COST ESTIMATION USING FEATURE-ORIENTED APPROACH

Software product line engineering is a software reuse paradigm that aims to develop a family of products with reduced time to market and improved quality [45]. Within the SPLE field, dynamic software product lines (DSPLs) have emerged as a promising means to develop reusable and dynamically reconfigurable core assets. Service-oriented dynamic software product lines (SO-DSPL) represent a class of DSPLs that are built on services and service-oriented architectures (SOAs) [45]. SO-DSPL support user needs and expectations in a continuously changing environment. SO-DSPL combine services in various configurations and contexts, simplifying the deployment of product variants that are mostly based on the same core services but tailored to meet different customers' needs. Feature modeling is the main activity to represent and manage PL requirements as reusable assets by allowing users to derive customized product configurations. Product configuration refers to the decision-making process of selecting an optimal set of features from the PL that comply with the feature model (FM) constraints.

To estimate cost in SPL-Based system using feature-oriented approach, we propose to implement scrupulously a coupled methodology with COCOMO approach, which is a very important technique to evaluate a software cost estimate. Our choice is sustained by the fact that no single one of the proposed approaches in the scientific community gives a hundred percent accuracy and none of them takes into account the existence of COTS while the cost of these components depends on many attributes and is not mastered. The methodology we suggest is composed of five steps:

Step 1: Perform detailed business analysis of the target domain;

Step 2: Identify all COTS in feature business components;

Step 3: Evaluate the cost of each identified COTS;

Step 4: Evaluate the cost of each feature using COCOMO approach;

Step 5: Estimate the cost of the system by adding the costs of the different COTS involved in the system with the costs of each feature.

To understand the previous steps, it is necessary to understand what feature business components are, how COTS can be integrated in feature business components and how to estimate the cost of SPL-Based systems. The following subsections give more details to better understand the suggested methodology. It is recommended that the authors provide adequate information to enable the work to be replicated. Methods that have previously been published should be referenced, and only relevant modifications should be mentioned

3.1 Features Business Components

A feature business component is a kind of software asset in which the body part is a feature realization [11]. It has an identifier, a description and a body. It can be decomposed in sub feature business components and uses other feature business components such as Table 1.

Table 1. Feature business component's structure

Name	A simple text
Is_composed_of	A finite set of sub feature business components
Uses	A finite set of used feature business components
Description	A composition of the intention, the engineering activity, the target business and the environment
Body	The realization part of the feature business component

Hence, we capture features business components by the following schema;

FeatureBusinessComponent = = [name: **TEXT**;
is_composed_of: 𝔽 **FeatureBusinessComponent**
uses: 𝔽 **FeatureBusinessComponent**
description: **Description**
body: **FeatureRealization** |]

Description: The description part of a feature business component is composed of its intention, the engineering activity the descriptor plans to perform, its target, the target business and the environment that is the context. Table 2 below is given for a better understanding.

Table 2. Feature business component description's structure

Intention	The engineering activity that is planned to perform
Target	The target business
Environment	The context

Feature business component descriptions are specified by the following Z notation schema:

Description = = [intention : **EngineeringActivity** ;
target : **Business;**
environment: **Context** |]

EngineeringActivity = = **AnalysisActivity** | **DesignActivity** | **ImplementationActivity**
AnalysisActivity = {analyze, …}
DesignActivity = {design, decompose, describe, specify, …}
ImplementationActivity = {implement, …}

Business = = [domain: **Domain**; processes: 𝔽 **Process** |]

Further specification for the following concepts: Domain, Process, Business Activity, Context & Context-awareness can be found in [15].

Feature realization: A set of effectively reusable software products constitute the realization part of a feature business component. These reusable software products are business analysis models, and it is composed by a solution which is a feature and a sequence of adaptation points. An adaptation point is a couple formed of a feature and a sequence of features which implement it. To improve the understanding of feature realization we give table 3.

Table 3. Feature Business Component Realizations structure

Solution	A feature model of a domain
Adaptation_points	A finite set of couples. For each couple, the first element is a feature in the solution and the second element is the finite set of features that implement the first element.

We model feature realizations by the following schema.

FeatureRealization = = [solution: Feature ;
adaptationpoints : 𝔽 {(Feature, 𝔽 Feature)} |
∀ fr: FeatureRealization, (f, V) ∈ adaptationpoints(fr) ⇒ f∈ featurefragments(solution(fr)) ∧ V = realizations(f)]

In FORM/BCS, a feature is a property visible by users and application developers of the domain. Features are classified in two groups: generic features (or services) and contextual features. Generic features are those of the capacity level of the domain. This means that a generic feature literally marks a distinct service, operation or function. Contextual features however are those of the operating environment, the domain technology and the implementation techniques. Performances of a system are non-functional features which are also classified in that second group.

A feature model of a domain gives the "intention" of that domain. This is why feature models are made only with generic features. In order to formalize this concept, we consider that a generic feature is an activity caused by an event which is applied to a target objects set. Features have generalization and decomposition. A feature's decomposition has three kinds of disjoint subcomponents:

1. The set of common features which indicate reuse opportunity,
2. The set of optional features;
3. The set of groups of alternate features.

Hence, we capture features by the schema below:

Feature = =*[* name: **TEXT**;
activity: BusinessActivity ;
objects: BusinessObjects ;
decomposition:[mandatory: $\mathbb{F}$ *Feature;*
optional: $\mathbb{F}$ *Feature;*
alternative: $\mathbb{F}\ \mathbb{F}$ *Feature]*
generalization: $\mathbb{F}$ *Feature* |
$\forall$ *f : Feature, optional(decomposition(f))* $\cap$ *mandatory(decomposition(f))* $= \varnothing$
$\forall$ *f : Feature,* $(\cup\ (A \in alternative(decomposition(f)))) \cap$
mandatory(decomposition(f)) $= \varnothing$
$\forall$ *f: Feature, optional(decomposition(f))* $\cap\ (\cup(A \in$
alternative(decomposition(f)))) $= \varnothing$
$\forall$ *f : Feature , A1, A2* $\in$ *alternative(decomposition(f)), A1*$\cap$ *A2* $= \varnothing$
$\forall$ *f : Feature , (mandatory(decomposition(f))* $\cup$ *optional(decomposition(f))* $\cup$
$(\cup\ (A \in alternative(decomposition(f)))))\ \cap$ *generalization(f)* $= \varnothing$ *]*

With a no ambiguous context, we establish the following correspondences:

mandatory (f) corresponds to *mandatory (decomposition(f))*
optional(f) corresponds to *optional(decomposition(f))*
alternative(f) corresponds to *alternative(decomposition(f))*
or(f) corresponds to *or(decomposition(f))*
decomposition (f) corresponds to *mandatory (f)* $\cup$ *optional (f)* $\cup\ (\cup\ (A \in alternative(f)))$

We say that a feature *f* is abstract if *decomposition (f)* $= \varnothing$.
We define the set
Abstract_Feature $= \{f:Feature \bullet decomposition\ (f) = \varnothing\}$

A business activity has a set of "mandatory" tasks, a set of "optional" tasks, a set of "alternative" tasks, a set of "or" tasks and a set of "clashing" tasks. It can be primitive or not. The following schema specifies business activities for the management of evolutions.

BusinessActivity == [name: **Name**;
decomposition: [mandatory: $\mathbb{F}$ **BusinessTask** ;
optional: $\mathbb{F}$ **BusinessTask**;
alternative: $\mathbb{F}\ \mathbb{F}$ **BusinessTask**;
or: $\mathbb{F}\ \mathbb{F}$ **BusinessTask]**;
clashing: $\mathbb{F}$ **BusinessTask;**
primitive: **Logic** |]
BusinessActivity == [name: **Name**;
decomposition: [mandatory: $\mathbb{F}$ **BusinessTask** ;
optional: $\mathbb{F}$ **BusinessTask**;
alternative: $\mathbb{F}\ \mathbb{F}$ **BusinessTask**;
or: $\mathbb{F}\ \mathbb{F}$ **BusinessTask]**;
clashing: $\mathbb{F}$ **BusinessTask;**
primitive: **Logic** |]

With a no ambiguous context, we write:

mandatory (ba) for *mandatory (decomposition(ba))*
optional(ba) for *optional(decomposition(ba))*
alternative(ba) for *alternative(decomposition(ba))*
or(ba) for *or(decomposition(ba))*
decomposition (ba) for mandatory (ba) $\cup$ *optional (ba)* $\cup\ (\cup\ (A \in alternative(ba)))$
We say that a business activity *ba* is abstract if *decomposition (ba)* $= \varnothing$.
We define the set
Abstract_Business_Activity $= \{ba:Business\_Activity \bullet decomposition\ (ba) = \varnothing\}$

A business task has a set of "mandatory “operations, a set of "optional" operations, a set of "alternative" operations, a set of "or" operations and a set of "clashing" operations. It can be primitive or not. The following schema specifies business tasks for the management of evolutions.

BusinessTask == [name: **Name**;
decomposition: [mandatory: ℙ **BusinessOperation** ;
optional: ℙ **BusinessOperation**;
alternative: ℙ ℙ **BusinessOperation**;
or: ℙ ℙ **BusinessOpeartion**];
clashing: ℙ **BusinessOperation;**
primitive: **Logic** |]

In a similar manner, with a no ambiguous context, we establish following correspondences:

mandatory (bt) corresponds to *mandatory (decomposition(bt))*
optional(bt) corresponds to *optional(decomposition(bt))*
alternative(bt) corresponds to *alternative(decomposition(bt))*
or(bt) corresponds to *or(decomposition(bt))*
decomposition (bt) corresponds to *mandatory (bt)* $\cup$ *optional (bt)* $\cup$ ($\cup$ ($A \in$ *alternative(bt)*))
We say that a business task *bt* is abstract if *decomposition (bt)* = $\varnothing$.
We define the set
Abstract_Business_Task = {*bt:Business_Task* • *decomposition (bt)* = $\varnothing$}

BusinessObjects = ℙ *Class*

A class is characterized by his name, attributes and operations:

Class = = *[name :* ***Text***
attributs : ℙ ***Attribut***
operations : ℙ ***Operation***| *]*

In previous Z schema, we use Unified Modeling Language types *Attribut*, *Operation* and well-known types like ***Boolean*** and ***Text*** which are basic.

3.2 Feature Business Component with COTS

3.2.1 COTS Classification and evaluation

Each COTS product can be distinguished by evaluation attributes. By help of these attributes and their values it will be possible to compare items belonging to the same class. While several of these attributes are common across the classes, some attributes are specific of a single class. Examples of common attributes are: price, market share, license type, compatibility issues, etc. Examples of attributes for Data Base Management Systems (DBMS) are transaction speed, scalability issue, etc. For each attribute, we must define its measurement scale, which can be: nominal, ordinal, ratio, and absolute. Examples of evaluation attributes, which can be applied to several classes, are presented in the following Table 4.

Table 4. COTS évaluation attribut

Attribute	**Scale**
Acquisition cost	Ratio
Ownership cost	Ratio
Market size	Ratio
Market share	Ratio
License type	Nominal

The cost of a software product can be split into two parts: the cost of acquisition of the product or of its license and the cost derived from its support, maintenance, and setup. The diffusion of a COTS product can be defined in terms of market size and market share. The market size is an absolute number representing the number of people that use items similar for the given one. For instance, the market size for Google Chrome is the number of user web browsers. Market share is the fraction of the market that uses the specific product. For instance, the market share for Internet explorer is the number of users of Internet Explorer itself. Another interesting attribute for evaluating a software product is the type of license under which it is sold, e.g. single user, open source, or GNU. We formalize COTS with the following Z schema in which, for simplification reasons, we consider ratio attributes as integer.

COTS = = *[name :* ***TEXT***
acquisition_cost : INTEGER
ownership_cost: INTEGER
market_size: INTEGER
market_share: INTEGER
licence_type: TEXT|]

With a no ambiguous context we write:
cost(c) for acquisition_cost(c) + ownership_cost(c)

For further explicit explanations on how COTS costs are identified and updated, look in [46]

3.2.2 COTS in Feature Business Component

COTS can be essential for the implementation of features of a feature business component or for their running time. That why for a clear specification of feature, we must show COTS involved in features inside the solution part of feature realizations. Thus, we improve feature's specification with the Z schema below.

Feature = = *[name: TEXT*
verb: Activity ;
objects: Target
decomposition:[common: 𝔽 Feature;
optional: 𝔽 Feature;
variabilities: 𝔽 𝔽 Feature]
generalization: 𝔽 Feature
cots: 𝔽 COTS |]

When the context is understandable, for a feature *f*, we establish the following correspondence:

cost(f) corresponds to direct_cost(f) + cost(decomposition(f))
with
direct_cost(f) = $\sum_{c \in cots(f)} cost(c)$ + COCOMO (activity(f))
and
cost(decomposition(f)) = $\sum_{g \in decomposition(f)} cost(g)$

COCOMO (activity (f)) represents software cost estimate using COCOMO approach for the direct activity of feature *f*.

Example: The following feature business component skeleton defines the Computerized System for the Integrated Management of State Personnel and Salaries (MSPS) of a country.

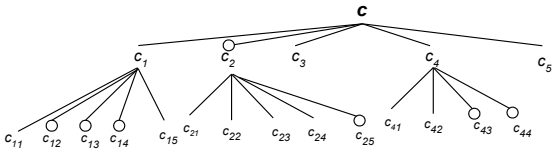
Name *: Feature Business Component of a Computerized System for the Integrated Management of State Personnel and Salaries of a country*
Is_composed_of *: {civil servant career management feature business component, salaries management feature business component, training management feature business component, attributions management feature business component, mail management feature business component}*
Uses *: {Database Management System feature business component, Reports delivery system feature business component}*
Description *:*
Intention *: Analyze*
Target *:*
Domain : C = (manage)$_{ACTION}$(State personals and salaries)$_{TARGET}$
Process : C_1 = (manage)$_{ACTION}$(civil servants career)$_{TARGET}$
C_2 = (manage)$_{ACTION}$(salaries)$_{TARGET}$
C_3 = (manage)$_{ACTION}$(training)$_{TARGET}$
C_4 = (manage)$_{ACTION}$(attributions)$_{TARGET}$
C_5 = (manage)$_{ACTION}$(mail)$_{TARGET}$
/ sub-process of C_1 */*
C_{11} = (manage)$_{ACTION}$(decisions, personnels governed by the general status or the labor code)$_{TARGET}$
C_{12} = (manage)$_{ACTION}$(decisions, magistrates)$_{TARGET}$
C_{13} = (manage)$_{ACTION}$(decisions, university lecturers)$_{TARGET}$
C_{14} = (manage)$_{ACTION}$(decisions, police officers)$_{TARGET}$
C_{15} = (transfer)$_{ACTION}$(decisions)$_{TARGET}$
/ sub-process of C_2 */*
C_{21} = (transfer)$_{ACTION}$(decisions)$_{TARGET}$
C_{22} = (calculate)$_{ACTION}$(salaries)$_{TARGET}$

C_{23} = *(manage)*$_{ACTION}$*(workstation)*$_{TARGET}$
C_{24} = *(manage)*$_{ACTION}$*(profiles, workstations)*$_{TARGET}$
C_{25} = *(manage)*$_{ACTION}$*(connections, workstations)*$_{TARGET}$
/ sub-process of* C_4 **/*
C_{41} = *(manage)*$_{ACTION}$*(workstations)*$_{TARGET}$
C_{42} = *(manage)*$_{ACTION}$*(profiles, workstations)*$_{TARGET}$
C_{43} = *(manage)*$_{ACTION}$*(connections, workstations)*$_{TARGET}$
C_{44} = *(manage)*$_{ACTION}$*(transactions, workstations)*$_{TARGET}$

Context *: All the country's regulatory texts governing the management of state personnel and their salaries*

Body :

Solution *:*



Adaptation points *: {*

(c, {{c_1*,* c_3*,* c_4*,* c_5*}, {*c_1*,* c_2*,* c_3*,* c_4*,* c_5*}}),*
(c_1*, {{*c_{11}*,* c_{15}*}, {*c_{11}*,* c_{15}*,* c_{12}*}, {*c_{11}*,* c_{15}*,* c_{13}*}, {*c_{11}*,* c_{15}*,* c_{14}*}, {*c_{11}*,* c_{12}*,* c_{13}*,* c_{15}*}, {*c_{11}*,* c_{13}*,* c_{14}*,* c_{15}*}, {*c_{11}*,* c_{12}*,* c_{14}*,* c_{15}*}, {*c_{11}*,* c_{12}*,* c_{13}*,* c_{14}*,* c_{15}*}}),*
(c_2*, {{*c_{21}*,* c_{22}*,* c_{23}*,* c_{24}*}, {*c_{21}*,* c_{22}*,* c_{23}*,* c_{24}*,* c_{25}*}}),*
(c_4*, {{*c_{41}*,* c_{42}*}, {*c_{41}*,* c_{42}*,* c_{43}*},{*c_{41}*,* c_{42}*,* c_{44}*},{*c_{41}*,* c_{42}*,* c_{43}*,* c_{44}*}})*

}

Figure 2. Feature business component skeleton for the Management of State Personnel and Salaries of a country

3.3 SPL-Based System Cost Estimation

System cost

Given a system, system, belonging to a software product line, SPL, we estimate the cost to implement this system, CE, by using the following algorithm called System Cost Estimation. In this algorithm, the variable CEC is an intermediate variable used to contain every time the cost of the current system.

Algorithm: System Cost Estimation
Result: CE: REAL
system: FeatureBusinessComponent ;
f: Feature;
CEC : REAL
CEC := direct_cost(solution(realization(system)))
If decomposition(solution(realization(system))) != Ø **then**
  For each *f* **in decomposition(solution(realization(system)))**
    CEC := CEC + direct_cost(*f*) + cost(decomposition(*f*))
  end
end
CE := CEC

Minimal cost for SPL-Based system

Given a software product line, *SPL*, for which the feature business component is *fbc*, we call minimal cost of *SPL*, the minimal cost to implement a system, *system* belonging to *SPL*. To estimate this cost, we estimate the cost for every software system in *SPL* and we choice the minimal cost of all these costs. To do this we use the following algorithm. In this algorithm, the variable *MCC* is an intermediate variable used to contain every time the minimal current cost and the variable *MC* contain at the end of the algorithm the minimal cost.

Algorithm: Minimal Cost Estimation
Result: *MC*: REAL
fbc: FeatureBusinessComponent ;
system: FeatureBusinssComponent;
MCC: REAL
***MCC* := cost (solution(realization(s ∈ systems(*fbc*))))**
For each *system* **in systems (*fbc*) \ {s}**
  MC := cost (solution(realization(*system*)))
  If *MC* < *MCC* **then**
    MCC* := *MC
  end
end
MC := *MCC*

Maximal cost for SPL-Based system

Given a software product line, *SPL*, for which the feature business component is *fbc*, we call maximal cost of *SPL*, the maximal cost to implement a system, *system* belonging to *SPL*. To estimate this cost, we estimate the cost for every software system in *SPL* and we choice the maximal cost of all these costs. To do this we use the following algorithm. In this algorithm, the variable *MCC* is an intermediate variable used to contain every time the maximal current cost and the variable *MC* contain at the end of the algorithm the maximal cost.

Algorithm: Maximal Cost Estimation
Result: *MC*: REAL
fbc: FeatureBusinessComponent ;
system: FeatureBusinssComponent;
MCC: REAL
***MCC* := 0**
For each *system* **in systems (*fbc*)**
MC := cost (solution(realization(*system*)))
If *MC* > *MCC* **then**
MCC* := *MC
end
end
MC := *MCC*

4.0 APPLICATION CASE STUDIES

The first case study is an organism software for tertiary institutions of an undesignated country. The product line, referred to as National Educational Management Product Line (NatEduMgtPl), was initiated by the Ministry of Higher Education in that country. The vision of the product line is to supply software products to state universities, other higher institutions, and Enterprise Resource Planning (ERP) vendors. The educational institutions in the country implement the BMP (Bachelor, Master and PhD) system - which makes their core operations largely the same- hence a product line. Let us notice that to develop and deploy such a system, we need many COTS, such as the Domain Name System (DNS) server, the Network Time Protocol (NTP) server, the Network File System (NFS) server, Automount File System (AutoFS), etc. Certain of them are free and open-source software like AutoFS but many have a cost of acquisition or its license and the cost for its support, maintenance, and setup.

Figure 3 shows the solution part of the realization for the feature business component of this product line. Executing the minimal cost estimation algorithm and the maximal cost estimation algorithm, we can observe that the system with the minimal cost for this software product line is the system without pre-admission and academy features and the maximal cost system includes one of the two mutually exclusive features along with all mandatory and optional features.

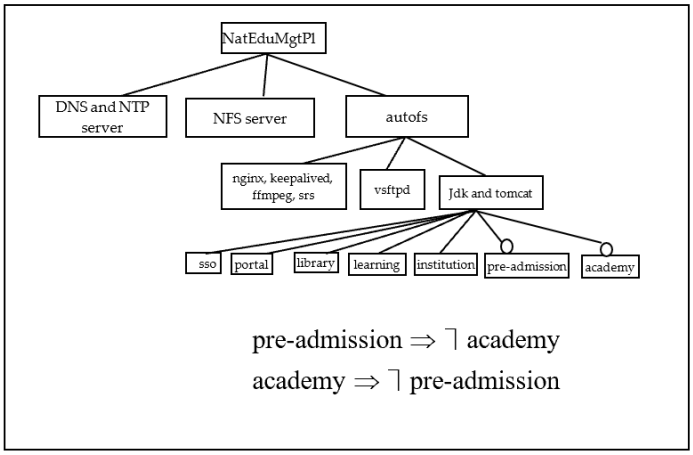


Figure 3. Partial feature model of the National Educational Management Product Line

The second case study here is the Computerized System for the Integrated Management of State Personnel and Salaries (MSPS) of a country. To develop and deploy this system, we need COTS notably a Database Management System (DBMS) and Reports Delivery System (RDS). Also, certain COTS like the DBMS PostgreSQL are free and open source but many like the RDS SAP Crystal Reports have a cost of acquisition or its license and the cost for its support, maintenance, and setup. Figure 4 shows the solution part of the realization for the feature business component of the Integrated Management of State Personnel and Salaries product line. Running the minimal cost estimation algorithm and the maximal cost estimation algorithm, we can observe that the system with the minimal cost for this software product line is the system without salaries management feature and the maximal cost system includes this feature along with all mandatory and optional features.

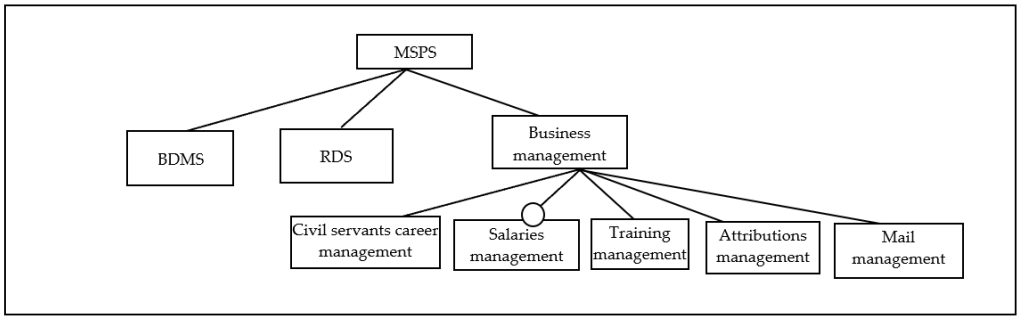


Figure 4. Partial feature model of the Integrated Management of State Personnel and Salaries Product Line

5.0 CONCLUSIONS AND FUTURE RESEARCH

Software Cost Estimation is an unavoidable and a vital aspect in software development projects and particularly in SPL-Based system. Many approaches have been proposed, but no single one gives a hundred percent accuracy. That is why one technique and model should not be preferred over all others. Between approaches that use the source lines of codes (SLOC) as their input and others that do not, in this paper, we propose to take into account another variable that is Commercial Off-The-Shelf Products (COTS) and we suggested a methodology composed of five steps to approximate the cost of SPL-Based systems. To estimate the cost of COTS, in software product lines, we put them within the solution part inside the realization section of feature business components. This integration allows propose an algorithm that takes as input a system belonging to a SPL and estimate the cost of this system. We can also estimate the minimal cost for a system in that SPL and the maximal cost. The novelty is that we enriched feature business components by adding new knowledge like required COTS and their costs for each feature. The new model allows practical system cost estimation in SPL. We have applied the new approach to the National Educational Management Product Line and plan to extend to others SPL. It is also beneficial to estimate software cost by combining our approach with other approaches different to COCOMO. However, our approach still faces limitations such as scalability and reliance on accurate COTS data. Combining our method with machine learning approaches would allow further development. This is our further investigation.

ACKNOWLEDGEMENTS

Here, we thank the Cameroonian Ministry of Higher Education for inspiring the partial feature model in Figure 3.

AUTHORS CONTRIBUTION

M. Fouda (Conceptualization; Formal analysis; Visualisation; Supervision)

N. Amougou (Methodology; Data curation; Writing - original draft; Resources)

CONFLICT OF INTEREST

The authors declare no conflicts of interest.

REFERENCES

[1] K. Pohl, G. Böckle, and F. J. van der Linden, Software Product Line Engineering: Foundations, Principles and Techniques. Berlin: Springer Science & Business Media, 2005, doi:10.1007/3-540-28901-1.

[2] P. Clements and L. Northrop, Software Product Lines: Practices and Patterns. Boston: Addison-Wesley Professional, 2001.

[3] S. Beiranvand and M. A. Zare-Chahooki, “Accuracy improvement in software cost estimation based on selection of relevant features of homogeneous clusters,” Journal of Artificial Intelligence and Data Mining, vol. 11, no. 3, pp. 453–476, 2023, doi:10.22044/jadm.2023.12750.2429.

[4] J. Rashid, M. W. Nisar, T. Mahmood, A. Rehman, and S. Y. Arafat, “A study of software development cost estimation techniques and models,” Mehran University Research Journal of Engineering and Technology, vol. 39, no. 2, pp. 413–431, 2020, doi:10.22581/muet1982.2002.18.

[5] R. R. Venkataraman and J. K. Pinto, Cost and Value Management in Projects. Hoboken, NJ: John Wiley & Sons, 2011.

[6] O. Benediktsson and D. Dalcher, “Estimating size in incremental software development projects,” IEE Proceedings Software, vol. 152, pp. 253–259, 2005, doi:10.1049/ip-sen:20050019.

[7] B. W. Boehm, "Software cost estimation meets software diversity," in Proceedings of the 39th International Conference on Software Engineering Companion, 2017, pp. 495–496, doi:10.1109/ICSE-C.2017.159.

[8] S. Aljahdali and A. F. Sheta, "Software effort estimation by tuning COCOMO model parameters using differential evolution," in IEEE/ACS International Conference on Computer Systems and Applications, 2010, pp. 1–6, doi:10.1109/AICCSA.2010.5586985.

[9] M. Aljohani and R. Qureshi, "Comparative study of software estimation techniques," International Journal of Software Engineering & Applications, vol. 8, no. 6, pp. 39–53, 2017, doi:10.5121/ijsea.2017.8603.

[10] J. A. Hess, W. E. Novak, K. C. Kang, S. G. Cohen, and A. S. Peterson, Feature-Oriented Domain Analysis (FODA) Feasibility Study. Pittsburgh, PA: Carnegie Mellon University, 1990.

[11] M. Fouda and A. Ngoumou, "The feature-oriented reuse method with business component semantics," International Journal of Computer Science and Applications, vol. 6, no. 4, pp. 63–83, 2009.

[12] M. Fouda and N. Amougou, "Product lines' feature-oriented engineering for reuse: A formal approach," International Journal of Computer Science Issues, vol. 7, no. 5, pp. 382–393, 2010.

[13] M. Fouda and N. Amougou, "Transformational variability modelling approach to configurable business system application," in Software Product Line – Advanced Topic, A. O. Elfaki, Ed. London: IntechOpen, 2012, pp. 43–68.

[14] N. Amougou and M. Fouda, "Feature-relationship models: A paradigm for cross-hierarchy business constraints in SPL," International Journal of Computer Science and Information Security, vol. 16, no. 9, pp. 112–124, 2018.

[15] N. Amougou and M. Fouda, "Context metamodel in pervasive systems for dynamic software product lines," Journal of Software Engineering & Intelligent Systems, vol. 5, no. 3, pp. 124–137, 2020.

[16] N. Amougou and M. Fouda, "Extended dynamic software product line architectures for context integration and management," Journal of Software Engineering & Intelligent Systems, vol. 6, no. 1, pp. 28–41, 2021.

[17] A. Z. Umar and J. Lee, "A model-based approach to managing feature binding time in software product line engineering," in MODELS Workshops, 2018.

[18] G. P. Espinel-Mena, J. L. Carrillo-Medina, M. Flores-Calero, and M. Urbieta, "Software configuration management in software product lines: Results of a systematic mapping study," IEEE Latin America Transactions, vol. 20, no. 5, pp. 718–730, 2022, doi:10.1109/TLA.2022.9693556.

[19] T. Kehrer, A. Schultheiß, T. Thüm, and P. M. Bittner, "Bridging the gap between clone-and-own and software product lines," in IEEE, 2021, doi:10.1109/ICSE-NIER52604.2021.00013.

[20] I. M. Keshta, "Software cost estimation approaches: A survey," Journal of Software Engineering and Applications, vol. 10, pp. 824–842, 2017, doi:10.4236/jsea.2017.1010046.

[21] B. Boehm, Software Engineering Economics. Englewood Cliffs, NJ: Prentice Hall, 1981.

[22] B. Boehm, B. Clark, E. Horowitz, C. Westland, R. Madachy, and R. Selby, "Cost models for future software life cycle processes: COCOMO 2.0," Annals of Software Engineering, vol. 1, pp. 57–94, 1995, doi:10.1007/BF02249046.

[23] P. Musilek, W. Pedrycz, N. Sun, and G. Succi, "On the sensitivity of COCOMO II software cost estimation model," in Proceedings of the 8th IEEE Symposium on Software Metrics, 2002, pp. 13–20, doi:10.1109/METRIC.2002.1011321.

[24] Z. Chen, T. Menzies, D. Port, and B. Boehm, "Feature subset selection can improve software cost estimation accuracy," ACM SIGSOFT Software Engineering Notes, vol. 30, pp. 1–6, 2005, doi:10.1145/1083165.1083171.

[25] H. Rastogi, S. Dhankhar, and M. Kakkar, "A survey on software effort estimation techniques," in 5th International Conference on Confluence, 2014, pp. 826–830, doi:10.1109/CONFLUENCE.2014.6949367.

[26] R. M. Waghmode, L. V. Patil, and S. D. Joshi, "A collective study of PCA and neural network based on COCOMO for software cost estimation," International Journal of Computer Applications, vol. 74, no. 16, pp. 25–30, Jul. 2013, doi:10.5120/12970-0099.

[27] L. H. Putnam, "A general empirical solution to the macro software sizing and estimating problem," IEEE Transactions on Software Engineering, vol. 4, pp. 345–361, 1978, doi:10.1109/TSE.1978.231521.

[28] C. F. Kemerer, "An empirical validation of software cost estimation models," Communications of the ACM, vol. 30, pp. 416–429, 1987, doi:10.1145/22899.22906.

[29] H. Leung and Z. Fan, "Software cost estimation," in Handbook of Software Engineering, pp. 1–14, 2002, doi:10.1142/9789812389701_0014.

[30] D. R. Jeffery and V. R. Basili, "Validating the TAME resource data model," in Proceedings of the 10th International Conference on Software Engineering, 1988, pp. 187–201, doi:10.1109/ICSE.1988.93700.

[31] A. J. Albrecht, "Measuring application development productivity," in Proceedings of the Joint SHARE/GUIDE/IBM Application Development Symposium, 1979, pp. 83–92.

[32] A. J. Albrecht, AD/M Productivity Measurement and Estimate Validation. Purchase, NY: IBM Corp., 1984.

[33] L. Wu, "Software cost estimation," [Online]. Available: http://www.computing.dcu.ie/~renaat/ca421/LWu1.html

[34] N. Sharma, A. Bajpai, and M. R. Litoriya, "A comparison of software cost estimation methods: A survey," International Journal of Computer Science and Applications, vol. 1, 2012.

[35] K. V. Kumar, V. Ravi, M. Carr, and N. R. Kiran, "Software development cost estimation using wavelet neural networks," Journal of Systems and Software, vol. 81, pp. 1853–1867, 2008.

[36] O. Rioul and M. Vetterli, "Wavelets and signal processing," IEEE Signal Processing Magazine, vol. 8, pp. 14–38, 1991, doi:10.1109/79.91217.

[37] H. Ramchoun, M. A. J. Idrissi, Y. Ghanou, and M. Ettaouil, "Multilayer perceptron: Architecture optimization and training," International Journal of Interactive Multimedia and Artificial Intelligence, vol. 4, no. 1, pp. 26–30, 2016, doi:10.9781/ijimai.2016.415.

[38] J. Moody and C. J. Darken, "Fast learning in networks of locally-tuned processing units," Neural Computation, vol. 1, no. 2, pp. 281–294, 1989, doi:10.1162/neco.1989.1.2.281.

[39] G. K. Uyanik and N. Güler, "A study on multiple linear regression analysis," Procedia - Social and Behavioral Sciences, vol. 106, pp. 234–240, 2013, doi:10.1016/j.sbspro.2013.12.027.

[40] N. K. Kasabov and Q. Song, "DENFIS: Dynamic evolving neural-fuzzy inference system and its application for time-series prediction," IEEE Transactions on Fuzzy Systems, vol. 10, pp. 144–154, 2002, doi:10.1109/91.995117.

[41] V. N. Vapnik, Statistical Learning Theory. New York: Wiley, 1998.

[42] A. Jaiswal, J. Raikwal, and P. Raikwal, "A hybrid cost estimation method for planning software projects using fuzzy logic and machine learning," International Journal of Intelligent Systems and Applications in Engineering, vol. 12, no. 1, pp. 696–707, 2024.

[43] B. O. Akumba, I. Agaji, N. Blamah, and E. Ogalla, "A hybrid machine learning method for estimating software project cost," International Journal of Innovative Science and Research Technology, vol. 8, no. 3, pp. 2681–2687, 2023.

[44] A. Gani, A. Akhunzada, and M. Junaid, "Software cost estimation in global software development using hybrid approach," Journal of Management Information and Decision Sciences, vol. 25, no. 4, pp. 1–25, 2022.

[45] R. Capilla, J. Bosch, P. Trinidad, A. Ruiz-Cortés, and M. Hinchey, "An overview of dynamic software product line architectures and techniques: Observations from research and industry," Journal of Systems and Software, vol. 91, pp. 3–23, 2014, doi: 10.1016/j.jss.2013.12.038.