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 a feature-oriented approach

N. Amougou*, M. Fouda

Higher Teacher Training College, University of Yaounde I, P.O. Box 337, Yaoundé, Centre Region, 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 require Commercial Off-The-Shelf Products (COTS). Therefore, evaluating the cost of a system in a Software Product Line (SPL) is difficult because many COTS evolve in features at different levels of the feature model. Currently, several approaches have been proposed to evaluate software project costs, including algorithmic methods such as the Constructive Cost Model, Software Life-Cycle Management (SLIM), and Functional Point Analysis (FPA), as well as non-algorithmic methods such as Delphi, rule-based, and learning-based approaches. However, COTS components are independent of the project, and their costs are subject to change. In this study, we suggest a methodology and propose algorithms to approximate the cost of a product belonging to a given software product line, including the minimal and maximal costs. The novelty of this work lies in enriching feature business components by adding new knowledge during domain analysis, such as required COTS and their associated costs for each feature at different levels of the feature model. This enhanced model enables the development of algorithms for more effective 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. Introduction

In software engineering, estimating the resources, effort, time, and cost required for a software product is difficult for project managers. To address this concern, software product line engineering [1] has been proposed to construct applications more easily, quickly, and with better quality. In [2], a Software Product Line (SPL) is defined 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 challenging in software project management because predicting costs is difficult in the early stages of software development [3]-[4]. Moreover, many factors affect software cost estimation, including implementation efficiency and the number of reviews and studies conducted before the development stage. Software effort and cost are strongly related, as effort is the primary driver of software cost [5]-[8]. Most existing techniques estimate the cost, effort, and duration. These techniques are also easy to understand and apply to software projects; however, they can fail and produce inaccurate estimates [9]. Hybrid techniques can improve estimation by combining two or more methods to increase accuracy. Therefore, the accuracy of cost estimation techniques is a significant challenge. Cost estimation faces more challenges in software product lines than in traditional systems. The first reason is that multiple COTS may be involved in implementing features at different levels. Second, COTS are independent of the current project, and their costs may change. In this study, the base feature models [10] are hierarchical models that capture the commonality and variability of a Product Line (PL) by defining its features, their dependencies, and the constraints between them.

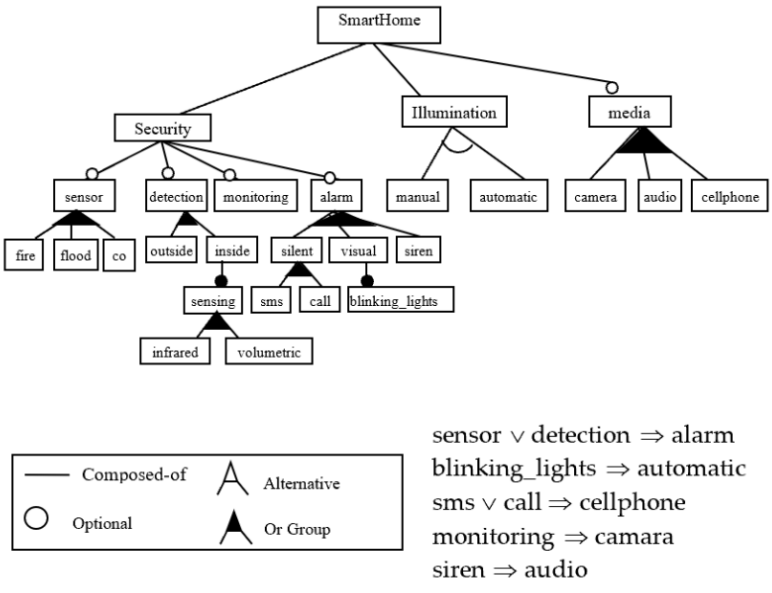


Figure 1. Extract of Smart Home feature model with constraints

*CORRESPONDING AUTHOR | N. Amougou | ✉ ngoumoua@yahoo.fr
© 2026 The Author(s). Published by Universiti Malaysia Pahang Al-Sultan Abdullah Press. This is an open access article under the CC BY-NC 4.0 license

Figure 1 shows an example feature model with constraints for a Smart Home PL. This example is interesting because it shows all sorts of features, such as mandatory, optional, alternative, and/or group features. The Feature-Oriented Reuse Method with Business Component Semantics (FORM/BCS) [11]-[14] allows us to extend feature models. Each system belonging to the product line is derived from a valid selection of feature combinations [15], and product configuration is the name of this process [16]-[17]. To estimate the SPL-based system cost, we propose a coupled methodology. Our method formally enriches the concept of feature business components in the FORM/BCS method to add new knowledge when analysing a domain. Additional knowledge includes the required COTS and their costs for each feature at different levels of the feature business component solution part, which is a feature model. This added information enables us to propose algorithms to estimate the cost of a system in a software product line and identify the system with the minimal cost and the product with the maximal cost. The remainder of this paper is organised as follows. In Section 2, we review the current approaches to software cost estimation. Section 3 presents our proposal. Section 4 presents a case study, and Section 5 concludes the study and provides perspectives for future research directions.

2. Related Works

In Software Cost Estimation, techniques fall into two principal groups. These are those that use source lines of code as their input and others that do not. In [18], the author describes the mechanisms and features of five approaches. This important technique uses an algorithmic formula to estimate the software cost [19]. Organisations use a combination of algorithmic methods (COCOMO), as indicated in references [20]-[23], to estimate cost, effort, and schedule when planning new software development activities. The approach, called the feed-forward neural network with Principal Component Analysis, is a reduction technique proposed by the authors in [24]. The main objective of this study was to use this technique to measure the accuracy of a software cost estimation model. The proposed approach combines both algorithmic and non-algorithmic methods. Therefore, to estimate software project costs, the authors combined an algorithmic method with a non-algorithmic method (Artificial Neural Networks). The Software Life-Cycle Management (SLIM) approach, developed by Larry Putnam of Quantitative Software Management in the late 1970s, is an experimental software effort-testing model. Details of this approach are provided in [25]-[28]. This technique estimates the effort and time required for a project of a given size. Software life cycle management is closely related to the COCOMO model's software parameters. [18] lists the advantages and disadvantages of SLIM. This method measures software complexity and size based on the functions it provides to the user. Allan Albrecht at IBM developed this approach and proposed it to the scientific community [29]-[30]. Counting Function Points requires two steps [26]: counting the various user functions and adjusting for processing complexity. The FPA has many advantages, as described in [31]-[32].

To estimate the software development effort, the authors employed a WNN [33]. Experiments were conducted on two variants of WNN: the Morlet wavelet function [34] and the Gaussian function, and a threshold acceptance training algorithm for the wavelet neural network was proposed. The results were compared with other computational intelligence methods, such as the Multilayer Perceptron [35], Radial Basis Function Network [36], Multiple Linear Regression [37], Dynamic Evolving Neuro-Fuzzy Inference System [38], and Support Vector Machine [39]. All these comparisons were computed in terms of the Mean Magnitude Relative Error (“MMRE” applied to both the Canadian Financial “CF” dataset and the IBM Data Processing Services “IBMDPS ”) dataset. Recent efforts have proposed hybrid approaches for software project cost estimation. In [40], a hybrid cost estimation method for planning software projects using fuzzy logic and machine learning was suggested. In [41], a hybrid machine learning method for estimating software project costs was proposed. In [42], a hybrid approach to software cost estimation in global software development was presented. These approaches could substantially impact software cost estimation. However, they deferred the implementation to future studies. A first comparison of these approaches was performed in [18] using criteria such as Ease of Use, Adaptability, Accuracy, Consistency, Interpretability, Automability, Tool Supportability, Empirical Validation, Sensitivity, and Handling Imprecision and Uncertainty. The author classifies these methods from most to least important as follows: COCOMO Model, Hybrid approach (based on both algorithmic and non-algorithmic methods), Putman Model, Function Point Analysis, and Wavelet Neural Network. Another comparative study of software estimation techniques was conducted in [9]. In this second comparison, COCOMO, SLIM, and Robust performed well in terms of accuracy. Across these approaches, none considers the existence of COTS systems.

3. SPL-Based System Cost Estimation

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 [43]. Within the SPLE field, dynamic software product lines (DSPLs) have emerged as a promising approach for developing reusable, dynamically reconfigurable core assets. Service-oriented dynamic software product lines (SO-DSPLs) represent a class of DSPLs built on services and service-oriented architectures (SOAs) [43]. SO-DSPL supports user needs and expectations in a continuously changing environment. SO-DSPL combines services in various configurations and contexts, simplifying the deployment of product variants based on the same core services but tailored to different customer needs. Feature modelling is the main activity for representing and managing PL requirements as reusable assets, allowing users to derive customised product configurations. Product configuration refers to the decision-making process of selecting an optimal set of features from the PL that comply with feature model (FM) constraints. To estimate cost in an SPL-based system using a feature-oriented approach, we propose a coupled methodology with COCOMO, a widely used software cost estimation technique. This choice is supported by the fact that no single approach in the scientific community achieves 100% accuracy, and none accounts for COTS, whose cost depends on many attributes and is not well understood. The proposed methodology comprises five steps.

Step 1: Perform a 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 the COCOMO approach.
Step 5: Estimate the system cost by adding the costs of the COTS involved in the system to the cost of each feature.

To understand the previous steps, it is necessary to understand what feature business components are, how COTS can be integrated into feature business components, and how to estimate the cost of SPL-based systems. The following subsections provide more details to help readers understand the proposed methodology.

3.1 Features Business Components

A feature business component is a type of software asset in which the body part is a feature realisation [11]. It has an identifier, description, and body. It can be decomposed into sub-feature business components and uses other feature business components, as listed in 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 the feature business components using the following schema: 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, which is the context. Table 2 provides a better understanding of the study.

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

Table 2. Feature business component description structure

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

The 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** |]

Table 3. Feature business component realisations structure

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

Further specifications for the following concepts: Domain, Process, Business Activity, Context & Context-awareness, can be found in [13]. Feature realisation: A set of effectively reusable software products constitutes the realisation of a feature business component. These reusable software products are business analysis models composed of a solution, which is a feature, and a sequence of adaptation points. An adaptation point is a pair consisting of a feature and the sequence of features that implement it. Table 3 presents this structure to improve understanding of feature realisation.

We model feature realisations using the following schema:

FeatureRealization = = [solution: Feature ;
adaptationpoints : $\mathbb{F}$ {(Feature, $\mathbb{F}$ Feature)} │
$\forall$ fr: FeatureRealization, (f, V) $\in$ adaptationpoints(fr) $\Rightarrow$ f$\in$ featurefragments(solution(fr)) $\wedge$ V = realizations(f)]

In FORM/BCS, a feature is a property visible to users and application developers in the domain. Features are classified into two groups: generic features (or services) and contextual features. Generic features are those of the capacity level of a domain. This means that a generic feature literally marks a distinct service, operation, or function. Contextual features, however, relate to the operating environment, domain technology, and implementation techniques. System performance is a non-functional feature that is also classified in the second group. A feature model of a domain provides the "intention" of that domain. This is why feature models are created only with generic features. To formalise this concept, we consider a generic feature as an activity caused by an event applied to a target object set. Features have generalisation and decomposition. A feature decomposition has three types 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 captured the features using the following schema:

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)* = $\varnothing$ *]*

In the absence of an ambiguous context, we establish the following correspondence:

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 the business activities for the management of evolution.

BusinessActivity == [name: **Name;**
decomposition: [mandatory: $\mathbb{F}$ **BusinessTask**;
optional: $\mathbb{F}$ **BusinessTask**;
alternative: $\mathbb{F}$ $\mathbb{F}$ **BusinessTask**;

or: 𝔽 𝔽 **BusinessTask]**;
clashing: 𝔽 **BusinessTask;**
primitive: **Logic**⎪]
BusinessActivity == [name: **Name**;
decomposition: [mandatory: 𝔽 **BusinessTask**;
optional: 𝔽 **BusinessTask**;
alternative: 𝔽 𝔽 **BusinessTask**;
or: 𝔽 𝔽 **BusinessTask]**;
clashing: 𝔽 **BusinessTask;**
primitive: **Logic**⎪]

In the absence of an 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 • *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 the business tasks for the management of evolution.

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

Similarly, with a non-ambiguous context, we establish the following correspondence:

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 = 𝔽 *Class*

A class is characterised by its name, attributes, and operations.

Class = = *[name :* ***Text***
attributs : 𝔽 ***Attribut***
operations : 𝔽 ***Operation***⎪ *]*

In the previous Z schema, we used Unified Modelling Language types *Attribut*, Operation, and well-known types such as ***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 its evaluation attributes. Using these attributes and their values, you can compare items within the same class. Although several of these attributes are common across classes, some are specific to a single class. Common attributes include price, market share, licence type, and compatibility issues. Examples of attributes for Database Management Systems (DBMS) are transaction speed and scalability issues. For each attribute, we

must define its measurement scale, which can be nominal, ordinal, ratio, or absolute. Table 4 presents examples of evaluation attributes applicable to several classes.

Table 4. COTS evaluation attributes

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 divided into two parts: the cost of acquiring the product or its licence and the cost of support, maintenance, and setup. The diffusion of a COTS product can be defined in terms of market size and share. Market size is an absolute number that represents the number of people who use items similar to a given item. For instance, the market size of Google Chrome is the number of web browser users. Market share is the fraction of the market that uses a specific product. For instance, the market share of Internet Explorer is the fraction of the market that uses Internet Explorer. Another interesting attribute for evaluating a software product is the type of licence under which it is sold, for example, single user, open source, or GNU. We formalise COTS with the following Z schema, where, for simplicity, we treat ratio attributes as integers.

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 [43]

3.2.2 COTS in feature business component

COTS can be essential for implementing the features of a feature business component or for their runtime. Therefore, to clearly specify a feature, we must show the COTS involved in the features within the solution part of the feature realisations. Thus, we improve the feature specification with the Z schema as follows:

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 feature *f*, we establish the following correspondence:

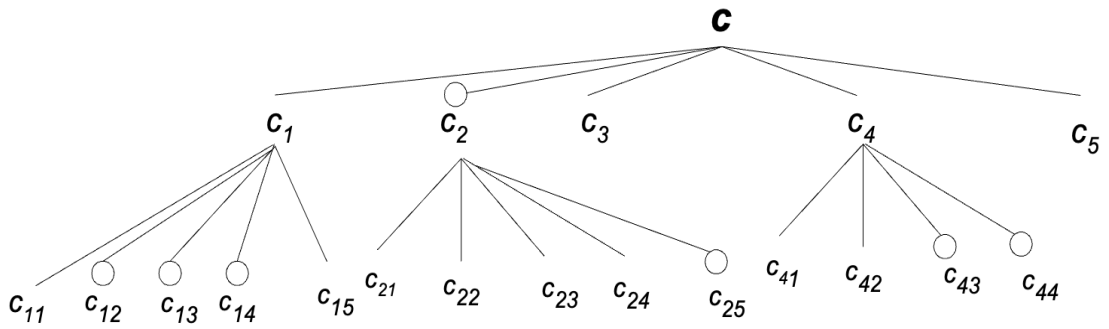
cost(f) corresponds to direct_cost(f) + cost(decomposition(f))
with
direct_cost(f) = $\sum_{c \in cots(f)} cost(c)$ + COCOMO (activity(f))
and
cost(decomposition(f)) = $\sum_{g \in decomposition(f)} cost(g)$

COCOMO (activity (f)) represents the software cost estimate using the COCOMO approach for the direct activity of feature *f*. The following is a feature business component skeleton that defines the Computerised System for the Integrated Management of State Personnel and Salaries (MSPS) of a country, as illustrated in Figure 2.

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

3.3 SPL-Based System Cost Estimation

System cost: Given a system belonging to a software product line, SPL, we estimate the cost to implement this system, CE, using the following algorithm called System Cost Estimation. In this algorithm, the variable CEC is an intermediate variable used to contain, at every step, 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 an SPL-based system: Given a software product line, *SPL*, for which the feature business component is *fbc*, we call the minimal cost of *an SPL* the minimal cost to implement a system belonging to *SPL*. To estimate this cost, we estimate the cost for every software system in the *SPL* and choose the minimum of these costs. To achieve this, we used the following algorithm. In this algorithm, *MCC* is an intermediate variable that stores the minimum current cost at each step, and *MC* stores the minimum cost at the end of the algorithm.

```
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
```

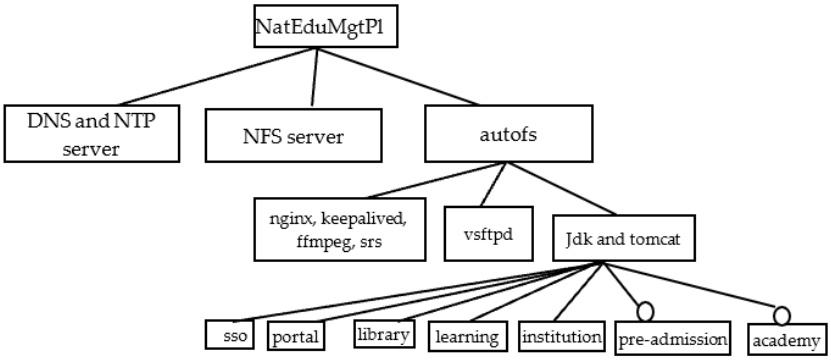
Maximal cost for an SPL-based system: Given a software product line, *SPL*, for which the feature business component is *fbc*, we call the maximal cost of the *SPL* the maximal cost to implement a system belonging to the *SPL*. To estimate this cost, we estimate the cost for every software system in the *SPL* and choose the maximum of these costs. To achieve this, we used the following algorithm. In this algorithm, *MCC* is an intermediate variable that stores the maximum current cost at each step, and *MC* stores the maximum cost at the end of the algorithm.

```
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. Application Case Studies

The first case study is an organisation software for tertiary institutions in an undesignated country. The Ministry of Higher Education in Malaysia initiated the product line, referred to as the National Educational Management Product Line (NatEduMgtPl). The product line's vision is to supply software products to state universities, other higher institutions, and Enterprise Resource Planning (ERP) vendors. Educational institutions in the country implement the BMP (Bachelor, Master, and PhD) system, which makes their core operations largely the same; hence, they have a product line. Let us note that to develop and deploy such a system, many COTS are required, such as the Domain Name System (DNS) server,

Network Time Protocol (NTP) server, Network File System (NFS) server, and Automount File System (AutoFS). Some are free and open-source software, such as AutoFS, but many have an acquisition cost, licensing fees, and costs for support, maintenance, and setup. Figure 3 shows the solution part of the feature business component realisation for this product line. By running the minimal and maximal cost estimation algorithms, we see that the minimal-cost system for this software product line excludes pre-admission and academy features, while the maximal-cost system includes one of the two mutually exclusive features along with all mandatory and optional features.



$$\text{pre-admission} \Rightarrow \neg\, \text{academy}$$
$$\text{academy} \Rightarrow \neg\, \text{pre-admission}$$

Figure 3. Partial feature model of the national educational management product line

The second case study is a country's Computerised System for the Integrated Management of State Personnel and Salaries (MSPS). To develop and deploy this system, we need COTS, notably a Database Management System and Reports Delivery System (RDS). In addition, some COTS, such as the DBMS PostgreSQL, are free and open source, but many others, such as the RDS SAP Crystal Reports, have an acquisition cost or licence fee, in addition to support, maintenance, and setup costs. Figure 4 shows the solution part of the realisation for the feature business component of the Integrated Management of State Personnel and Salaries (IMSPS) product line. Running the minimal and maximal cost estimation algorithms, we observe that the minimal-cost system for this software product line is the system without the salary management feature, and the maximal-cost system includes this feature along with all mandatory and optional features.

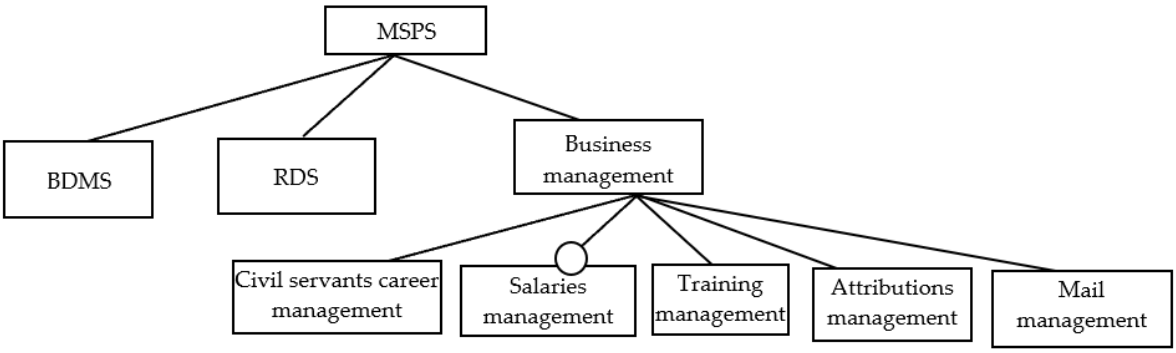


Figure 4. Partial feature model of the integrated management of state personnel and salaries product line

5. Conclusions

Software Cost Estimation is an unavoidable and vital aspect of software development projects, particularly in SPL-based systems. Many approaches have been proposed, but none have achieved 100% accuracy. Therefore, no single technique or model should be preferred over others. Between approaches that use source lines of code as their input and others that do not, in this paper, we propose considering another variable: Commercial Off-The-Shelf Products, and we suggest a five-step methodology to approximate the cost of SPL-based systems. To estimate the cost of COTS in software product lines, we place it in the solution part, within the realisation section on feature business components. This integration allows us to propose an algorithm that takes as input a system belonging to an SPL and estimates its cost, as follows: We can also estimate the minimal and maximal costs for a system in that SPL. The novelty of this study is that we enriched feature business components by adding new knowledge, such as the required COTS and their costs for each feature. The new model allows for practical system cost estimation in SPL. We have applied the new approach to the National Educational Management Product Line and plan to extend the SPL to others. It is also beneficial to estimate software costs by combining our approach with other approaches that differ from 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 for further development. We will investigate this further in future studies.

Acknowledgements
The authors thank the Cameroonian Ministry of Higher Education for inspiring the partial feature model and the University of Yaounde, Cameroon, for providing the infrastructure required to carry out this research.

Funding
This study was not supported by any grants from funding bodies in the public, private, or not-for-profit sectors.

Declaration of Competing Interest
The authors declare no conflict of interest.

Credit Authorship Contribution Statement
N. Amougou: Methodology; Data curation; Writing - original draft; Resources
M. Fouda: Conceptualization; Formal analysis; Visualisation; Supervision; Writing- reviewing & editing

Availability of Data and Materials
Data sharing is not applicable to this article, as this study generated or analysed no new datasets.

Ethics Declarations
This study did not involve human participants or animal subjects. Therefore, this study did not require ethical approval.

Generative Artificial Intelligence Declarations
No generative artificial intelligence tools were used to prepare this manuscript. All intellectual content is original and is solely attributable to the authors.

References
[1] K. Pohl, G. Böckle, and F. van der Linden, *Software Product Line Engineering: Foundations, Principles and Techniques*. Berlin, Germany: Springer, 2005, https://doi.org/10.1007/3-540-28901-1
[2] P. Clements and L. Northrop, *Software Product Lines: Practices and Patterns*. Boston, MA, USA: Addison-Wesley Professional, 2001. ISBN: 978-0-201-70332-0.
[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, https://doi.org/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, https://doi.org/10.22581/muet1982.2002.18
[5] R. R. Venkataraman and J. K. Pinto, *Cost and Value Management in Projects*. Hoboken, NJ, USA: Wiley, 2008, https://doi.org/10.1002/9780470261033
[6] O. Benediktsson and D. Dalcher, "Estimating size in incremental software development projects," *IEE Proceedings — Software*, vol. 152, no. 6, pp. 253–259, 2005, https://doi.org/10.1049/ip-sen:20050019
[7] B. W. Boehm, "Software cost estimation meets software diversity," in *Proceedings of the 2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*, Buenos Aires, Argentina, 2017, pp. 495–496, https://doi.org/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 *Proceedings of the 2010 IEEE/ACS International Conference on Computer Systems and Applications (AICCSA)*, Hammamet, Tunisia, 2010, pp. 1–6, https://doi.org/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, https://doi.org/10.5121/ijsea.2017.8603
[10] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, and A. S. Peterson, "Feature-oriented domain analysis (FODA) feasibility study," Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, USA, Tech. Rep. CMU/SEI-90-TR-021, 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] 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.
[13] 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.
[14] 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.

[15] A. Umar and J. Lee, "A model-based approach to managing feature binding time in software product line engineering," in *Proc. MODELS 2018 Workshops: ModComp*, CEUR Workshop Proceedings, 2018, pp. 18–23. [Online]. Available: https://ceur-ws.org/Vol-2245/

[16] 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, https://doi.org/10.1109/TLA.2022.9693556

[17] T. Kehrer, T. Thüm, A. Schultheiß, and P. M. Bittner, "Bridging the gap between clone-and-own and software product lines," in *Proceedings of the 2021 IEEE/ACM 43rd International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*, Madrid, Spain, 2021, pp. 21–25, https://doi.org/10.1109/ICSE-NIER52604.2021.00013

[18] I. M. Keshta, "Software cost estimation approaches: A survey," *Journal of Software Engineering and Applications*, vol. 10, no. 10, pp. 824–842, 2017, https://doi.org/10.4236/jsea.2017.1010046

[19] B. W. Boehm, *Software Engineering Economics*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1981. ISBN: 978-0-13-822122-5.

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

[21] 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 (METRICS 2002)*, Ottawa, ON, Canada, 2002, pp. 13–20, https://doi.org/10.1109/METRIC.2002.1011321

[22] 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, no. 4, pp. 1–6, 2005, https://doi.org/10.1145/1082983.1083171

[23] H. Rastogi, S. Dhankhar, and M. Kakkar, "A survey on software effort estimation techniques," in *Proceedings of the 2014 5th International Conference — Confluence: The Next Generation Information Technology Summit*, Noida, India, 2014, pp. 826–830, https://doi.org/10.1109/CONFLUENCE.2014.6949367

[24] 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, https://doi.org/10.5120/12970-0099

[25] L. H. Putnam, "A general empirical solution to the macro software sizing and estimating problem," *IEEE Transactions on Software Engineering*, vol. 4, no. 4, pp. 345–361, 1978, https://doi.org/10.1109/TSE.1978.231521

[26] C. F. Kemerer, "An empirical validation of software cost estimation models," *Communications of the ACM*, vol. 30, no. 5, pp. 416–429, 1987, https://doi.org/10.1145/22899.22906

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

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

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

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

[31] L. Wu, "The comparison of the software cost estimating methods," DCU School of Computing, Dublin, Ireland, 1997. [Online]. Available: http://www.computing.dcu.ie/~renaat/ca421/LWu1.html. Accessed: Nov. 1, 2025.

[32] 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, no. 3, pp. 121–127, 2012.

[33] 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, no. 11, pp. 1853–1867, 2008, https://doi.org/10.1016/j.jss.2007.12.793

[34] O. Rioul and M. Vetterli, "Wavelets and signal processing," *IEEE Signal Processing Magazine*, vol. 8, no. 4, pp. 14–38, 1991, https://doi.org/10.1109/79.91217

[35] 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, https://doi.org/10.9781/ijimai.2016.415

[36] 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, https://doi.org/10.1162/neco.1989.1.2.281

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

[38] 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, no. 2, pp. 144–154, 2002, https://doi.org/10.1109/91.995117

[39] V. N. Vapnik, *Statistical Learning Theory*. New York, NY, USA: Wiley, 1998. ISBN: 978-0-471-03003-4.

[40] 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.

[41] 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, https://doi.org/10.5281/zenodo.8017130

[42] 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. S4, pp. 1–25, 2022.

[43] 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, no. 1, pp. 3–23, 2014, https://doi.org/10.1016/j.jss.2013.12.038