Cost Optimisation in Certification of Software Product Lines

Ricardo J. Rodríguez
Babel Group, DLSIIS, ETSINF
Universidad Politécnica de Madrid
Research Institute of Applied Sciences in Cybersecurity
University of León, Spain
Email: rj.rodriguez@unileon.es

Sasikumar Punnekkat
School of Innovation, Design and Engineering
Mälardalen University, 721 23 Västerås, Sweden
Email: sasikumar.punnekkat@mdh.se

***Abstract*—Safety-critical systems (such as those in automotive, avionics, railway domains), where a failure can result in accidents with fatal consequences, need to certify their products as per domain-specific safety standards. Safety certification is not only time consuming but also consumes the project budget. Adopting a reuse oriented development and certification paradigm can be highly beneficial in such systems. Though there had been several research efforts on cost models in the context of software reuse as well as software product lines, none of them have addressed the certification related costs.**

In this paper, we present a cost model for software product lines, which incorporates certification costs as well. We first propose a mathematical model to represent a Software Product Line and then present an approach to compute, using optimisation theory, the set of artifacts that compose a new product assuring an expected level of confidence (that is, a certain Safety Integrity Level) at an optimised cost level. The proposed approach can help developers and software project managers in making efficient design decisions in relation to the choice of the components for a new product variant development as part of a product line.

Keywords*—*software product lines, cost model,safety certification, certification costs, integer programming

I. Introduction

Software Product Line Engineering (SPLE) is becoming a widely adopted approach to develop software-intensive systems using platforms and mass customisation [1]. A platform (also called core asset base) defines a collection of artifacts that can be reused across a company portfolio. The differences between product platforms are usually expressed in terms of features, where a set of features are seen as the specification of a product variant, i.e., a particular member of the product line. One of the benefits of SPLE is the reuse of software components as much as possible, thus saving production costs. This is specially important in industrial domains such as automotive industry, where software has become the key enabler for innovations [2]. For instance, the software development costs were estimated to reach 13% of the production cost of a vehicle in 2010 [2].

An SPLE approach can also be adopted in safety-critical systems, where a failure can result in accidents with fatal consequences [3]–[5]. Furthermore, engineers developing these systems (industrial, automotive or avionics domains, among others) need to perform safety certification of their products before they are used, produced or sold. Safety certification process is not only time consuming but also substantially increases the overall development cost [6]. However, the adoption of SPLE for safety-critical systems can rise several challenges [7], and cost optimisation might be considered as a non-essential issue in advance. Reuse of the already developed software components together with the reuse of the associated parts of the safety evidences and argument fragments in the construction of a new safety case is considered as a potential approach for cost savings and is being addressed by several EU-funded projects such as SafeCer [8].

In this paper, we introduce a cost model for Software Product Lines that addresses the certification aspects as well. Specifically, we propose a mathematical model to represent a Software Product Line as well as a heuristic strategy to compute, using optimisation theory, the platform members that compose a new product variant assuring an expected level of confidence (that is, a certain Safety Integrity Level (SIL) [9]). This approach can help to make efficient design decisions in relation to the choice of the components for a new product variant development as part of a product line.

The main contributions in this paper are: 1) a mathematical model for represnting a software product line, 2) an extended cost model for Software Product Lines that takes certification into account, based on the cost model given in [10], and 3) an approach to compute, using optimisation theory, the set of artifacts for a given product portfolio that compose a new product variant with an expected confidence level at a minimum cost.

The outline of this paper is as follows. Section II reviews related works. Section III introduces a cost model that accounts for certification, taking as basis the model given in [10], and a formal definition of a Software Product Line (SPL). Then, Section IV proposes a heuristic strategy to minimise the cost of creating a new product variant in a given SPL using optimisation theory, and evaluates its performance under different scenarios. Section V concludes the paper and briefly states the future work.

II. Related Work

Software reuse allows to improve the software quality at a lower cost. A wide review of cost models for software reuse and reusability can be found in [11], [12]. Software cost estimation models are mainly categorised in three categories [13]:

Expert judgment models, that consider experience and knowledge of one or more experts; algorithmic (or parametric) models, which use input parameters to cost estimation; and machine learning approaches, which approximate accurately cost generalising the obtained knowledge and dynamically adjusting to the conditions under study.

In this work, we focus on parametric cost models, such as SLIM [14], [15], COCOMO-II [16], FPA [17] or SEEM-SEM (System Evaluation and Estimation of Resources - Software Estimation Model). They normally rely on a wide range of project attributes to estimate project costs, such as the development effort, complexity, or experience, among others. Namely, COCOMO-II [16] is an open model that states an equation involving development for reuse, multistage development, risk plans and team cohesion [18]. However, certification issue is not considered. Similarly, other non-open model such as SLIM [14], [15] considers several parameters, as the one commented before, but lacks certification aspects. Any of the above cost models could easily add certification issue as one more factor into its cost estimation equation. However, they finally estimate the effort, schedule and costs of a software project in an algorithmical way, but without considering the best suitable options to minimise cost.

Other cost estimation models have been also proposed in Software Product Lines [10], [19], although safety certification is neither considered. It is worth mentioning the above models do not consider certification cost issues as they were not initially planned for safety-critical systems. In theory, they could also be extended to contemplate these issues. In this paper, we revise the latter model provided by Bockle et al. and refine it by taking product variant certification into account.

Regarding optimisation in SPL, it is worth mentioning the work by Oleachea et al. [20] where a tool based on Alloy language is presented that performs exact, discrete multi-objective optimization in the context of variability models. To the best of our knowledge, no prior works deal with cost optimisation in cost estimation models for SPL. In this paper, we also provide an algorithm to optimise the cost of adding a new product to a company portfolio considering also certification issues.

III. A Cost Model for Certification

In this section, we firstly define an SPL in a formal way, which will be later used to mathematically express the cost optimisation problem in an SPL, and then refine the Bockle et al.'s cost model [10] for a family of n products in a Software Product Line (SPL) by considering safety certification.

A. Software Product Line Definition

Definition 1: A Software Product Line (SPL) is a tuple $\mathcal{S} = \langle \mathcal{X}, \mathcal{P}, \mathcal{F}, \mathcal{R}, \mathcal{E}, \mathcal{D} \rangle$, where:

- $\mathcal{X} = \{x_1, x_2 \ldots, x_n\}$ is the set of components in the SPL;
- $\mathcal{P} = \{p_1, p_2 \ldots, p_m\}$ is the set of product variants in the SPL;
- $\mathcal{F} = \{f_1, f_2 \ldots, f_u\}$ is the set of features available in the SPL;
- $\mathcal{R} : \mathcal{P} \times \mathcal{F} \rightarrow \{0, 1\}$ relates the set of features provided by a product.
- $\mathcal{E} : \mathcal{F} \times \mathcal{X} \rightarrow \{0, 1\}$ relates the set of components that provide a feature. That is, all components that has a value of 1 for a given feature f are interchangeable, i.e. $x, x' \in \mathcal{X}, \mathcal{E}(f, x) = \mathcal{E}(f, x') = 1$ means that we can pick either x or x' to provide the feature f.
- $\mathcal{D} : \mathcal{X} \times \mathcal{X} \rightarrow \{0, 1\}$ defines the dependencies among the components. Note that when a component x_i depends of some other x_j, then x_j (or its dependencies) cannot depend of other component x_k such that $\mathcal{D}(x_i, x_k) = 1$. Otherwise, a cyclic dependency will arise.

For convenience, in the sequel we denote $\mathcal{R}, \mathcal{E}, \mathcal{D}$, in matrix form, i.e., $\mathcal{R} \in \{0, 1\}^{|\mathcal{P}| \times |\mathcal{F}|}, \mathcal{E} \in \{0, 1\}^{|\mathcal{F}| \times |\mathcal{X}|}, \mathcal{D} \in \{0, 1\}^{|\mathcal{X}| \times |\mathcal{X}|}$.

B. A Certification Cost Model

The Bockle et al.'s cost model [10] states the cost of establishing an SPL with n product variants in an organization as:

$$C = C_{org} + C_{cab} + \sum_{i=1}^{n} (C_{unique}(p_i) + C_{reuse}(p_i)) \qquad (1)$$

where C_{org} is the cost to an organization for adopting the software product line approach for its set of products; C_{cab} is the cost to develop a core asset base to support the product line being adopted; C_{unique} is the cost to develop unique software for a new artifact p_i; and C_{reuse} is the cost to reuse a core asset p_i.

A safety-critical system typically has a set of mandatory safety requirements that needs to be fulfilled to assure a certain level of safety. Safety requirements are usually specified by industrial standards, such as IEC 61508 [9], ISO 26262 [21] or DO-178C [22] in the industrial equipment, road vehicles or avionics domains, respectively. A system is said to be *safety-certified* (or just *certified*) when it fulfils the safety requirements enforced by a safety-related standard.

Consider a company where an SPL has been adopted and develops a set of products for safety-critical systems that needs to be certified. Note that certification of a product p_i, composed by a set of components $\{x_1, \ldots, x_m\}$, implies the certification of each one of its components $x_i, 1 \leq i \leq m$. In safety-critical systems, the safety standards typically specifies multiple levels of safety performance for a safety related function, which are termed as Safety Integrity Level (SIL) [9]. Though different standards name them by numbers or alphabets etc, we could as well assume them to be a non-zero integer and are attributable to a component that implements the concerned safety function. The SIL specifies a target level of risk reduction, and in a safety-critical system forces the minimum SIL required by its components (or product variants) and sub-components. That is, a component x_i that depends of a set of other components and has a SIL of 2 enforces all dependent components to have a SIL level greater than or equal to 2.

Therefore, Definition (1) of an SPL can then be redefined in the context of safety-critical systems by considering the initial SIL of each component, i.e., by adding a vector $\mathcal{I}$ to previous definition, where $\mathcal{I} : \mathcal{X} \rightarrow [1, SIL_{max}]$ is a vector that assigns a SIL to each component in the SPL, and SIL_{max} is the maximum achievable SIL in the system. Unfortunately, certification process does not come at zero cost. A product variant p_i that is initially certified (C_{cert}) can also needs to be re-certified in a different context, which of course adds an extra cost (C_{recert}). Thus, we define a function cost $\mathcal{C}_{recert} : \mathcal{X} \times \mathcal{I} \rightarrow c, c \in \mathbb{R}^{+}_{\geq 0}$, that assigns a cost $c \geq 0$ of re-certifying to a given safety integrity level s for a component x, and a cost $\mathcal{C}_{other} : \mathcal{X} \rightarrow c, c \in \mathbb{R}^{+}_{>0}$, that considers the other cost of a component, such as cost of reuse and cost of unique development. If a component x_i cannot achieve a certain s SIL, its cost of re-certification to that level is set to zero, i.e. $\mathcal{C}_{recert}(x_i, s) = 0$. Considering all this, Equation (1) can be refined as follows:

$$C' = C_{org} + C_{cab} + \sum_{p' \in \mathcal{P}'} C_{cert}(p') + \sum_{i=1}^{n} (C_{unique}(p_i) + C_{reuse}(p_i) + C_{recert}(p_i, s')) \tag{2}$$

where $\mathcal{P}' \subseteq \mathcal{P}$ is the set of products that needs to be initially certified to a given SIL, and $C_{recert}(p_i, s')$ is the cost of re-certification of a product p_i to an s' level.

The recertification cost $C_{recert}(p_i, s')$ of a product p_i to a s' level is indeed equal to the cost of recertification of each component that composes the product p_i, and its subcomponents, to s' level, i.e. $C_{recert}(p_i, s') = \sum_{x \in \mathcal{X}'_i} \left(\mathcal{C}_{recert}(x, s') + \sum_{\forall x' \in \mathcal{D}(x,x')=1} \mathcal{C}_{recert}(x', s') \right)$, where $\mathcal{X}'_i = \{x | x \in \mathcal{X} \wedge f \in \mathcal{F}, \mathcal{R}(p_i, f) = 1 \Rightarrow \mathcal{E}(f, x) = 1\}$ for a given p_i.

In the sequel, we develop a heuristic strategy to solve, in a given scenario and considering the terms of Equation (2), the best set of components that minimises the cost of developing a new product variant in an SPL.

IV. A Cost Optimisation Problem for Certification

Algorithm 1 shows our heuristic strategy to minimise the cost of adding a new product variant to an SPL. As input, it needs the SPL $\mathcal{S}$, as previously defined in Section III, the vector $\mathcal{I}$ that assigns a SIL level to each current component, the cost functions $\mathcal{C}_{other}$ and $\mathcal{C}_{recert}$, the expected SIL of the new product variant SIL_{new} and the set of features $\mathcal{F}' \subseteq \mathcal{F}$ that must be provided by such a new product variant. As output, it provides the set of eligible components $\mathcal{Z}$ that minimise the cost. Note that $\mathcal{Z}$ can be empty, which means that with the current configuration there is no possible solution.

Step 1 computes the number of components that can be recertified as the non-null elements contained in $\mathcal{C}_{recert}$. Steps 2 – 5 properly initialise the variables used by the heuristic algorithm, such as a copy of matrix $\mathcal{D}$, $\mathcal{X}$, $\mathcal{I}$ and the vector

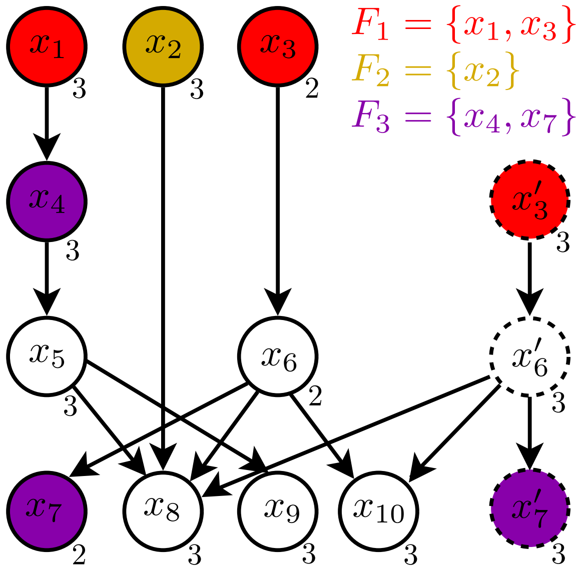

Figure 1. Component dependencies (and SIL) and features of the hypothetical example.

of costs $\mathcal{C}'$, which is later used as optimization function.Steps 6 – 12 iterates for each component x in the SPL $\mathcal{S}$ that can be certified to a level s, and it has not yet been processed. First, a temporary path is created by recursively iterating over the dependencies of x until all non-dependent components are reached, checking whether subcomponents can reach an SIL $\geq s$. Otherwise, the path is not valid. Recall that a component x_i that depends of a set of other components enforces all its dependencies to have an SIL greater than or equal to the level of x_i. When such a path is possible, these updated components are added to the set of components $\mathcal{X}'$, setting its SIL level properly and marking the tuples (x, s) as processed (step 8). The costs of re-certifying these new components are summed up to the cost of each old component x (step 9). Lastly, steps 10 and 11 set the features provided by the new components as the old components, and the same with component dependencies. After that, step 13 adjusts the diagonal elements of $\mathcal{D}'$ for each component as the negated sum of dependencies, i.e. $\forall x \in D', D'(x, x) = -\sum_{x' \neq x} D'(x, x')$. Step 14 computes the set of components $\mathcal{A}$ that minimises the cost as solution of the BIP problem (3). The first constraint relates the set of features that must be provided by the selected components. The second constraint relates the component dependencies, while the third constraint assures that the selected components have a SIL greater than the given one as input parameter. The last constraint restricts the values of components to binary values (that is, selected or not selected). Finally, the set of selected components is created (step 15).

Let us remark that the BIP problem (3) enables to compute a solution under two different scenarios: Either for adding a new product variant, or either for re-certifying an existing product variant in the SPL to a higher SIL. In both scenarios, the selected components that minimise the cost are computed by solving the BIP problem (3).

We illustrate the applicability of Algorithm 1 by an hypothetical example under both scenarios. Consider a small com-

```
Input: $\mathcal{S} = \langle \mathcal{X}, \mathcal{P}, \mathcal{F}, \mathcal{R}, \mathcal{E}, \mathcal{D} \rangle, \mathcal{I}, \mathcal{C}_{other}, \mathcal{C}_{recert}, SIL_{new}, \mathcal{F}'$
Output: $\mathcal{Z}$
1  Get the number of elements $n$ that can be re-certified as the number of non-null elements of $\mathcal{C}_{recert}$
2  Create a matrix $\mathcal{D}' = 0_{|\mathcal{X}|+n,|\mathcal{X}|+n}$, and $\forall x, x' \in \mathcal{X}, \mathcal{D}'(x, x') = \mathcal{D}(x, x')$
3  Create a vector $\mathcal{X}' = \mathcal{X}$
4  Create a vector $\mathcal{I}' = 0_{|\mathcal{X}|+n}$, and $\forall x \in \mathcal{X}, \mathcal{I}'(x) = \mathcal{I}(x)$
5  Create a vector $\mathcal{C}' = \{0\}^n$, and $\forall x \in \mathcal{X}, \mathcal{C}'(x) = \mathcal{C}_{other}(x)$
6  foreach $(x, s) \in \mathcal{C}_{recert}, \mathcal{C}_{recert}(x, s) \neq 0$ and $(x, s)$ not yet processed do
7      Create a temporary path $\mathcal{P}'$ by recursively iterating over the dependencies of $x$ until all components without
       dependencies are reached, checking whether subcomponents can reach a SIL $\geq s$
8      When such a path is possible, add these new components to $\mathcal{X}'$, and set its SIL level properly marking each $(x, s)$ as
       processed
9      Add the cost of re-certifying each one of these components $x'$ to $s$ level, i.e. $\mathcal{C}'(x') = \mathcal{C}'(x) + \mathcal{C}_{recert}(x, s)$
10     Set the features provided by each one of these upgraded components as the not upgraded components
11     Set the dependencies of each one of these upgraded components as the not upgraded components
12 end
13 For each component, set the diagonal elements of $\mathcal{D}'$ equal to the number of dependent elements, i.e.
   $\forall x \in \mathcal{X}', D'(x, x) = -\sum_{x' \neq x} D'(x, x')$.
14 Compute $\mathcal{A}$ as solution to the following Binary Integer Programming problem:
```

$$\begin{aligned} \mathcal{A} = \mathit{minimize}\, & \mathcal{C}' \cdot \mathcal{X}'^{\top} \\ \mathit{subject\ to}\, & \mathcal{E} \cdot \mathcal{X}'^{\top} = \mathbf{1} \\ & \mathcal{D}' \cdot \mathcal{X}'^{\top} \geq \mathbf{0} \\ & (\mathcal{E} \cdot \mathcal{I}') \cdot \mathcal{X}'^{\top} \geq \mathbf{1} \cdot SIL_{new} \\ & x \in \{0, 1\}, \forall x \in \mathcal{X}' \end{aligned} \tag{3}$$

```
   where $\mathcal{E}' \subseteq \mathcal{E}, \forall f \in \mathcal{F}', \mathcal{E}'(f, :) = \mathcal{E}(f, :)$
15 Create the set of selected components $\mathcal{Z} = \{x \in \mathcal{X}', \mathcal{A}(x) = 1\}$
```

Algorithm 1: A strategy to minimise the cost of a new product variant in an SPL.

pany in the road vehicle domain that builds vehicle equipments (such as seat belt fasteners, buckles, or airbags) and where an SPL has been recently adopted. Assume that this SPL has a total of $|\mathcal{X}| = 10$ components. For the sake of simplicity, let us focus in one product, e.g. an airbag equipment. Suppose that in accordance with a new safety state legislation, a new airbag equipment must be built, with a Safety Integrity Level (SIL) of 3. Let the airbag equipment be composed of three features, i.e., $\mathcal{R} = \{1, 1, 1\}$, and $\mathcal{F} = \{f_1, f_2, f_3\}$, and the matrix $\mathcal{E} = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$, that is, f_1 is provided either by x_1 or x_3, f_2 is only provided by x_2 and f_3 is either provided by x_4 or x_7 components.

The dependencies in such an SPL between a component x_i and x_j, textually represented as $x_i \rightarrow x_j$, are the following: $x_1 \rightarrow x_4; x_2 \rightarrow x_8; x_3 \rightarrow x_6; x_4 \rightarrow x_5; x_5 \rightarrow \{x_8, x_9\}; x_6 \rightarrow x_7, x_8, x_{10}$. Figure 1 (left-hand side) shows the component dependencies, with its SIL, and features of this hypothetical example.

Lastly, let us suppose that the initial SIL level is $\mathcal{I} = \{3, 3, 2, 3, 3, 2, 2, 3, 3, 3\}$, and the cost of reuse and the cost of unique development for the components is $\mathcal{C}_{other} = \{\$1500, \$1000, \$1000, \$1200, \$1200, \$900, \$750, \$1500, \$1350, \$1000\}$.

Let us firstly consider the first scenario, where given the SPL $\mathcal{S}$ and the rest of input parameters as described above, re-certification is not possible. Suppose first a desired SIL in the new product variant as the minimum, i.e., $SIL_{new} = 1$. The solution provided by Algorithm 1 is $\{x_2, x_3, x_6, x_7, x_8, x_{10}\}$, having a cost of \$6150. Thus, functionality f_1 is provided by x_3, f_2 by x_2 and f_3 by x_7. The rest of components appear by dependencies among them. Suppose now that the desired SIL level is $SIL_{new} = 3$. In such a case, the solution is $\{x_1, x_2, x_4, x_5, x_8, x_9\}$ with a cost of \$7750, being the functionalities f_1, f_2 and f_3 provided by x_1, x_2 and x_4, respectively.

Let us now consider the second scenario, where re-certification is possible. In the above scenario description, we have the chance to recertificate x_3, x_6, x_7 components upto SIL 3 at the same cost (for the sake of simplicity), i.e., $\mathcal{C}(x_3, 3) = \mathcal{C}(x_6, 3) = \mathcal{C}(x_7, 3) = 200$. The new components certified to SIL 3 are termed as x_3', x_6', x_7'. Figure 1 (right-hand side) shows these new components and its relationships with the others. As it can be seen, a temporary path has been properly achieved, and then such new components are considered as valid. In the second scenario, the solution provided by Algorithm 1 is $\{x_2, x_3', x_6', x_7', x_8, x_{10}\}$, having a cost of \$6750. Thus, functionality f_1 is provided by x_3', the old component x_3 re-certified to 3 SIL, f_2 by x_2 and f_3 by x_7', the old component x_7 re-certified to 3 SIL. As before, the other components appear by dependencies among them.

A. Performance Evaluation

Herein, we evaluate the performance of Algorithm 1 in complex scenarios. Our main goal is to evaluate how the BIP problem (3) performs in relation to the number of components and the number of features.

A random generator of SPLs have been developed, following the definition given in Section III-A. Both generator and the Algorithm (1) have been implemented on MATLAB [23], and experiments have been run in a machine with a 2GHz Intel Core i7 processor and 8GB 1600MHz DDR3 RAM. MATLAB uses a linear programming (LP)-based branch-and-bound algorithm to solve binary integer programming problems. Recall that integer programming is known to be NP-complete [24].

We have performed a sensitivity analysis of execution time varying the number of features between 100 and 3000, with a step of 100, and the number of components in the SPL between 100 and 500, with a step of 50. The number of components implementing the same feature and the number of component dependencies have been randomly selected between 1 and 10, and between 1 and 20, respectively. We have measured a total of 100 computations of the Algorithm 1 for each pair of number of features and number of components, and then computed the average execution time.

Figure 2 plots the average execution time (in seconds) of the Algorithm 1 varying the number of components and the number of features. The results show that for small number of components, and independently of the number of features, the execution time is relatively near to 0. However, the average execution time rapidly increases with respect to the number of components, reaching its maximum at the extreme point (500 components with 3000 features). It can be seen that the execution time of the BIP problem (3) clearly has a strong dependence on the number of components, almost independently on the number of features.

The results also show that execution time is relatively small (less than 0.5 seconds in the experiments). Thus, our approach can handle complex scenarios with a large number of components with a good performance, being its running time negligible compared to the time needed to gather the data. We aim at developing a tool for making the gathering of data from the user (i.e. description of an SPL) easier and more intuitive, as well as providing the feedback in a more human-readable way.

V. Conclusions

A Software Product Line (SPL) defines a set of artifacts that can be reused across a company portfolio. The reuse of software products allows to save production cost to companies. An SPL approach can also be adopted in safety-critical systems (such as those in automotive, avionics or railway domains) that must also comply with safety standards. This safety certification process is not only time consuming but also has expensive production costs. In this paper, we introduce a cost model for SPLs that addresses the certification aspects. Besides, we also formally define an SPL and propose a heuristic strategy that makes use of optimisation theory (i.e., a Binary Integer Programming problem) to computes the set of artifacts in an SPL that conforms a new product variant at an optimised cost and assures a certain level of confidence (normally called Safety Integrity Level, SIL).

Our cost model proposal takes certification cost of products into account, unlike other cost models that can be found in the literature. Our heuristic strategy offers a good trade-off between accuracy and time complexity, and helps to make efficient design decisions during development process in an SPL for safety-critical systems. We have evaluated the performance of our optimisation algorithm for cost optimisation in a randomly generated SPL varying the number of features and components, and results show that our approach is able to converge in a reasonable time for large SPL systems, and the execution time is clearly depending on the number of variables to be optimised (i.e., the number of components). In fact, execution time is negligible compared to the time needed to gather the data in the formal definition that we propose.

To overcome the latter issue, as future work we aim at developing a tool to collect data, solve the problem and report an understandable feedback to the final users. We also aim at evaluating our approach with real industrial case studies, and at extending our proposal by eliminating the independence assumption on SIL.

Acknowledgments

This work was partially supported by ARTEMIS Joint Undertaking nSafeCer Project under grant agreement no. 295373, by the SSF funded Synopsis Project, and by the Spanish National Institute of Information Technologies (INTECO) accordingly to the rule 19 of the Digital Confidence Plan (Digital Agency of Spain) and the University of León under the contract X43. Part of this work was done while Dr. Rodríguez was a visiting researcher at Mälardalen University, Västerås (Sweden).

References

[1] K. Pohl, G. Böckle, and F. J. Linden, *Software Product Line Engineering: Foundations, Principles and Techniques*. Springer, 2005.

[2] B. Hardung, T. Kölzow, and A. Krüger, "Reuse of Software in Distributed Embedded Automotive Systems," in *Proceedings of the 4th ACM International Conference on Embedded Software (EMSOFT)*. New York, NY, USA: ACM, 2004, pp. 203–210.

[3] H. Koziolek, T. Goldschmidt, T. de Gooijer, D. Domis, and S. Sehestedt, "Experiences from Identifying Software Reuse Opportunities by Domain Analysis," in *Proceedings of the 17th International Software Product Line Conference (SPLC)*. New York, NY, USA: ACM, 2013, pp. 208–217.

[4] C. Dumitrescu, R. Mazo, C. Salinesi, and A. Dauron, "Bridging the Gap Between Product Lines and Systems Engineering: An Experience in Variability Management for Automotive Model Based Systems Engineering," in *Proceedings of the 17th International Software Product Line Conference (SPLC)*. New York, NY, USA: ACM, 2013, pp. 254–263.

[5] M. Schulze, J. Mauersberger, and D. Beuche, "Functional Safety and Variability: Can It Be Brought Together?" in *Proceedings of the 17th International Software Product Line Conference (SPLC)*. New York, NY, USA: ACM, 2013, pp. 236–243.

[6] S. Baumgart, J. Froberg, and S. Punnekkat, "Towards Efficient Functional Safety Certification of Construction Machinery Using a Component-Based Approach," in *3rd International Workshop on Product Line Approaches in Software Engineering (PLEASE)*, June 2012, pp. 1–4.

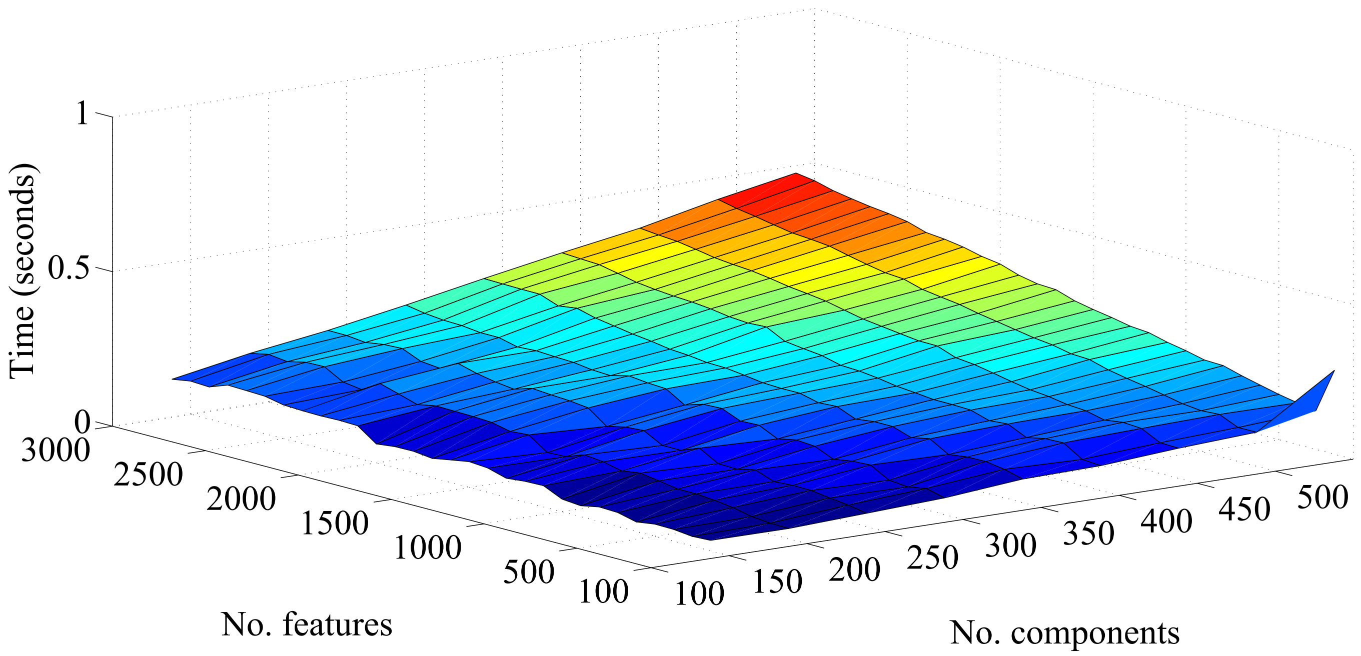

Figure 2. Average execution time of the Algorithm (1) w.r.t. the number of features and components.

[7] M. Becker, S. Kemmann, and K. C. Shashidhar, "Integrating Software Safety and Product Line Engineering using Formal Methods: Challenges and Opportunities," in *Workshop Proceedings of the 14th International Conference on Software Product Lines (SPLC)*. Lancaster University, 2010, pp. 129–136.

[8] SafeCer, "Safety Certification of Software-Intensive Systems with Reusable Components, EU-Artemis JU funded project," http://www.safecer.eu.

[9] *IEC 61508: Functional safety of electrical/electronic/programmable electronic safety-related systems*, International Electrotechnical Comission Std., 2010.

[10] G. Bockle, P. Clements, J. D. McGregor, D. Muthig, and K. Schmid, "Calculating ROI for Software Product Lines," *IEEE Software*, vol. 21, no. 3, pp. 23–31, May 2004.

[11] W. Frakes and C. Terry, "Software Reuse: Metrics and Models," *ACM Comput. Surv.*, vol. 28, no. 2, pp. 415–435, June 1996.

[12] A. Mili, S. Chmiel, R. Gottumukkala, and L. Zhang, "An Integrated Cost Model for Software Reuse," in *International Conference on Software Engineering*, 2000, pp. 157–166.

[13] E. Papatheocharous, H. Papadopoulos, and A. S. Andreou, "Feature Subset Selection for Software Cost Modelling and Estimation," *CoRR*, vol. abs/1210.1161, 2012.

[14] L. Putnam, "A General Empirical Solution to the Macro Software Sizing and Estimating Problem," *IEEE Transactions on Software Engineering*, vol. SE-4, no. 4, pp. 345–361, July 1978.

[15] L. Putnam and W. Myers, *Measures For Excellence: Reliable Software On Time, Within Budget*, ser. Yourdon Press Computing Series. Yourdon Press, 1992.

[16] 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, no. 1, pp. 57–94, 1995.

[17] A. Albrecht and J. E. Gaffney, "Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation," *IEEE Transactions on Software Engineering*, vol. SE-9, no. 6, pp. 639–648, November 1983.

[18] I. Sommerville, *Software Engineering: Seventh Edition*. Pearson Education, 2004.

[19] B. Boehm, A. Brown, R. Madachy, and Y. Yang, "A Software Product Line Life Cycle Cost Estimation Model," in *Proceedings of the International Symposium on Empirical Software Engineering (ISESE)*, August 2004, pp. 156–164.

[20] R. Olaechea, S. Stewart, K. Czarnecki, and D. Rayside, "Modelling and Multi-objective Optimization of Quality Attributes in Variability-rich Software," in *Proceedings of the Fourth International Workshop on Nonfunctional System Properties in Domain Specific Modeling Languages (NFPinDSML)*. New York, NY, USA: ACM, 2012, pp. 2:1–2:6.

[21] *ISO 26262: Road vehicles – Functional safety*, International Organization for Standardization Std., 2011.

[22] *RTCA DO-178C. Software Considerations in Airborne Systems and Equipment Certification*, Radio Technical Commission for Aeronautics Std., 2012.

[23] The MathWorks, "Matlab, http://www.mathworks.com/," 2010, version R2010a.

[24] R. Karp, "Reducibility among Combinatorial Problems," in *Complexity of Computer Computations*, ser. The IBM Research Symposia Series. Springer US, 1972, pp. 85–103.