An Analysis of Domain and Application Engineering Co-evolution for Software Product Lines based on Cladistics: A Case Study

Anissa Benlarabi
IMS Team, SIME Laboratory
ENSIAS, Mohamed V Souissi University
Rabat, Morocco
a.benlarabi@gmail.com

Amal Khtira
IMS Team, SIME Laboratory
ENSIAS, Mohamed V Souissi University
Rabat, Morocco
amalkhtira@gmail.com

Bouchra El Asri
IMS Team, SIME Laboratory
ENSIAS, Mohamed V Souissi University
Rabat, Morocco
elasri@ensias.ma

***Abstract*—Software product line engineering is a discipline for large scale reuse, its main advantage is the ability to reuse a set of domain assets in the development of a large number of products. In order to achieve this benefit, the software product line must cope with business requirements evolution. When dealing with evolution, the most effort must be granted to the understanding of the change and the identification of its impact because changes happening to products must be propagated to domain artifacts that are used for the whole family, and if the impact is not studied, each product will evolve separately from the domain assets. Many techniques were proposed to facilitate the impact analysis, such as evolution traceability or documentation. However, they consider only the change on the domain assets level and they underestimate issues raised by the fact when products evolve separately from the domain assets, which decreases the ability of the software product line to derive all the products features. In this paper, we tackle this issue by analyzing the co-evolution of software product lines and their products. We use cladistics classification, which was used in biology to construct their evolutionary trees, then we compare the trees using mathematical analysis and we propose a solution to restore the perfect co-evolution of the software product line and its products. We carried out a case study on a Mobile Media software product line to illustrate our approach.**

***Index terms*—** Software product lines; Co-evolution; Cladistics.

I. Introduction

Software product line engineering [1] is a software engineering discipline centered on reuse. It consists in developing a set of domain assets, which can be reused to derive a set of products for a particular market [2]. Its main goal is the reduction of costs and time to market, which can only be achieved by the continuous adaption of the domain assets to the ever-changing user requirements. Hence, to maximize benefits from the software product line common platform, the evolution activity must be the pivot activity of the software product line process development.

The primary aim for the evolution activity is the protection of the software from the aging problem, which pictures the fact of having a vital software for the organization but which cannot be evolved [3]. Unlike single software, software product line aging problem is not only caused by the loss of knowledge but also by the inability of the software product line to support all the features of the old and the new products. This happens especially when the changes happening to the products are not propagated to the domain artifacts, in this case each product evolves separately from the domain artifacts and the software product line will no longer be able to derive all the features. Hence, instead of having a software product line we will have a set of independent products.

The approach presented here aims at improving the understanding of how the software product line and its products evolved in time and how they influenced each other during their evolution. It focuses on analyzing the co-evolution of the software product lines and their products. The change in software product lines has two levels, the level of domain engineering and the level of application engineering, the evolution of each level impact the evolution of the other, our co-evolution analysis helps identifying how the evolution of products and the evolution of the core assets impacted each other. In this paper, we focus on the impact of the changes happening to products on the core assets because it was less tackled by the researchers than the domain engineering evolution impact. Co-evolution was extensively studied in biology [4] to show how organisms influence each other during their evolution. The co-evolution of host-parasite is a famous example from biology [5]. Beside biology, the co-evolution was studied also in software engineering [6] [7] [8].Similarly to co-evolution in biology, we will study the co-evolution of many populations of software. Our work consists in a co-evolution model for software product lines based on cladistics classification [9], which identifies the evolution path of a group of organisms based on their shared characters and classifies them in evolutionary tree. We start by establishing the evolutionary trees of the software product line and its products, then we perform a mathematical analysis to correct divergences between their evolution paths. We illustrate our approach through a case study on the mobile media software product line [10], we started by applying the approach on one product but we intend to experiment it on other products and compare the obtained results. Currently, we consider that all the products features must be derived from the domain engineering; we do not consider the products specific features that are not intended to be part of the platform.

In Section 2, we explain the co-evolution in biology, the we present some co-evolution studies in software engineering and we introduces the co-evolution of domain and application assets. In Section 3, we propose present our approach through a case study; we firstly study the evolution courses of the

 ISBN: 978-1-61208-367-4

software product line and its products through their evolutionary trees established using cladistics classification. Secondly, we compare these resulted trees to extract their similarities and divergences then we correct these divergences using a mathematical analysis. We give a conclusion in Section 4.

II. Related Works

According to our literature search, works done to understand how the evolution of domain engineering and the evolution of application engineering influence each other rely mainly on traceability links between the artifacts of the two levels. A framework for traceability was proposed by Anquetil et al. [11], the framework allows for tracing links between the different artifacts and present them in a graphical view, the developers can use the graphical view to know the impacted artifacts by a change. Ajila and Kaba [12] proposed a tool which gives operation instances modification, operations for consistency checking, and operations for change impact analysis. The tool calculates the impact of a change on the basis repositories that involved the software product line artifacts and their relationships. Goknil proposed a meta-modeling approach for requirements traceability management [13]. He focuses on post-requirements traceability, in particular between requirements models and architectural models, the goal is to determine which architectural components are impacted by a requirement change. Traceability approaches consider the traceability of links between the domain assets and the links between the domain and the application assets as a basis for the change impact analysis activity. However, they rely on human knowledge which is too expensive and error prone. In addition, they consider that the change happens only in the domain assets. Our work allows for defining the impact of the change by identifying the hidden links between the reference and the application assets, it also improves the change understanding through a synthesis of the history of the software product line evolution and help predicting future changes by considering changes that were implemented at a product level and may be propagated to the reference assets and then to the other products. Instead of relying on the human knowledge, we use the evolution histories of the software product line and its products and we analyze their co-evolution using cladistics classification.

III. Software Product Lines Co-evolution

In this section, we present the co-evolution principal, which was used mainly in biology, and we give an insight on some works that deal with the co-evolution in software engineering context. Thus, we introduce the co-evolution of domain engineering and application engineering in software product lines.

A. Co-evolution in Biology

Co-evolution of species in biology describes the situation when an evolution of a population of species can affect another population of species, and consequently induces its evolution. It consists in a mutual evolutionary influence between two populations [4]. A population in biology represents any group of descendants of the same ancestor that appeared due to changes of the ancestor characteristics. Understanding how populations co-evolve allows for determining how environmental changes impact directly their evolution. The co-evolution of host-parasite is a famous example of biological co-evolution. Because parasites cause damages to their hosts, hosts develop new capacities to resist to their parasites however parasites also develop capacities to overcome this resistance [5]. Therefore, a clearer understanding of hostparasite co-evolution will point to new possibilities for organic farming and reduce the application of ecologically harmful chemicals.

B. Co-evolution in Software Engineering

Co-evolution was tackled in other fields, such as software engineering; we present here some works that showed the necessity to take into consideration the co-evolution between different layers of a solution to preserve its consistency and correctness and also to reduce evolution costs.

Ruscio et al. [6] addressed the co-evolution of meta-models and their related entities: models, transformations and tools, especially the automated adaptation of these entities in order to preserve their correctness and consistency. The authors introduced a set of basic ingredient a co-adaptation solution must provide, and they point out on the necessity to have a unique technique for meta-models co-evolution regardless the related entity type. They proposed the EMFMigrate tool, which applies a set of migration rules on the related entities depending on the change type and the relation between the metamodel and the entity, because in some relations meta-models changes may be independent and do not require a co-evolution.

Kster and Trifu [7] tackled the problem of traceability between the requirements and the architecture incited by the fact that an important part of evolution costs are spent to locate the impacted elements. He presents a case study on the co-evolution between requirements and architectural design from which he extracted a set of requirements for a solution of co-evolution of architectural model and requirements model. Then he proposed a solution using graphs in which elements from both models are linked by decisions. The graph is dynamically navigable, and helps identifying the change impact easily.

Seidl et al. [8] introduced the co-evolution of software product lines. He stated that evolution of SPLs can harm the mapping between features and realization artefacts, for example if an implementation asset is deleted and a mapping to it remains in the system, products that include features mapped to this missing item will be invalid. For this reason, proposed an approach to co-evolve the features mapping and the system models, more accurately the feature model and the realization artefacts. He made a classification of evolution scenarios either in problem space (insert feature, delete feature, Split feature, etc.) or in solution space (replace method, rename method, etc.) into two groups: interspatial evolutions that reaches beyond the boundaries of the originating space and intraspatial

 ISBN: 978-1-61208-367-4

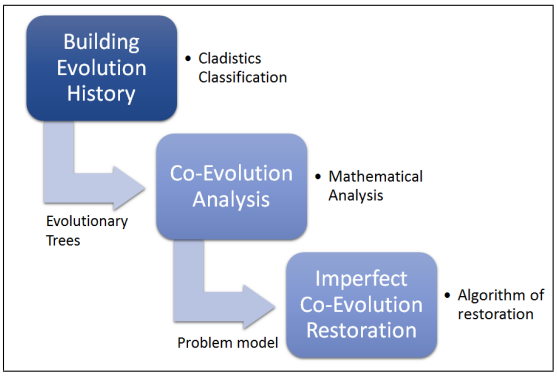

Figure 1. Co-evolution approach for SPL

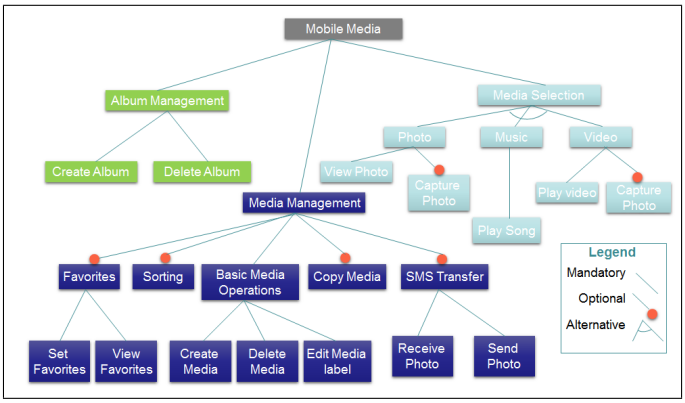

Figure 2. Mobile Media SPL features model

evolutions that impact only the space they are originating from. Thus, he extended Eclipse by a set of remapping operators to maintain the consistency of features mapping. These operators will be sequentially executed after each interspatial evolution.

C. Domain and Application Assets Co-evolution in Software Product Lines : a cladistics based approach

In this paper, we introduce the concept of co-evolution of domain and application assets in software product lines, which consists in comparing the evolution paths of the domain assets and the application assets and then deducing if application assets were changed independently from their domain assets. Organisms co-evolution analysis relies mainly on the visible characters of these organisms, in software the visible characters are its features. Hence, we will consider only features models; thus, we study the co-evolution of the domain features model and the application features models.

To deal with such co-evolution, we propose an approach based on Cladistics [9], which is a biological technique used to understand how organisms evolve over time (see Fig. 1.), it builds an evolutionary tree for a population by classifying its members in a tree on the basis of the evolution of their physical characters or the evolution of their behavior. The steps of our approach are as follows:

- Building evolution history: in this step, we use the data about evolution in order to establish the evolution path of the software product line and each derived product. We use cladistics classification, which gives a classification of the members of a population based on their shared characters. In the case of software product lines, the characters of software are its features
- Co-evolution Analysis: in this step, we perform a mathematical analysis of the domain and applications assets co-evolution through sets, we introduce the hypothesis of our analysis and we represent the perfect co-evolution by means of mathematical equalities
- Imperfect co-evolution correction: in this step, we present mathematically the imperfect co-evolution on the basis of the analysis we did in step 2 and we propose an algorithm to correct.

In the following subsections, we will explain in more details our approach and the techniques used in each step.

IV. Domain and Application Engineering Co-evolution Analysis: a Case Study

In this section, we present our approach co-evolution approach through a case study on the mobile media software product line with one of its derived products. In the first subsection, we present the mobile media software product line and the derived product features, in the second subsection, we construct their cladograms using cladistics classification, a cladogram is a branching diagram which represents the evolution path of a group of organisms based on their shared characters. In the third section, we compare their cladograms and we correct the detected imperfections using a mathematical analysis.

A. Mobile Media software product line

The Mobile Media software product line manipulates photo, music, and video on mobile devices, such as mobile phones and it has 200 derived products [10]. Many evolution scenarios were performed on the mobile media software product line, we take into consideration in our case study six evolution scenarios. Hence, we have a population P1 formed by seven releases of the software product line and a population P2 formed by the seven releases of a derived product. We present the feature model of the seventh release of the mobile media software product line in Fig. 2.

B. Building Evolution History

In order to build the evolution history for the populations of the software product line and the product, we use cladistics classification, which is a biological technique which classifies a set of organisms derived from the same ancestor in a evolutionary tree. In the following subsections, we give more details about this technique and its application on the mobile media software product line.

1) Cladistics Classification: Cladistics classification was used in biology to construct evolutionary trees that shows the evolutionary relationships among various biological organisms, on the basis of the similarities and differences in

their physical or genetic characteristics [9]. It assumes that in a population of organisms, a new organism appeared due to the change of the group characters. Hence, it identifies the evolution path of these organisms based on their shared characters. In the case when more than one possible tree can be generated for one group we must choose the most parsimonious evolutionary tree which is the shortest one. The length of a tree is obtained by calculating the sum of all the characters fits where the fit of a character is the number of its occurrences on the tree. In addition to identifying the evolution path of a taxonomic group, cladistics classification helps identifying which character change is responsible for the appearance of each organism and also the characters that mostly participate to the evolution of the group. The steps of the cladistics classification are:

- Select the population to be classified
- Identify the characters of the population and their different states
- Classify the group on the basis of their shared characters in an evolutionary tree called cladogram
- When having more than one cladogram in result, an analysis of parsimony is required.

2) Evolutionary trees of mobile media and its product: We follow the mentioned steps for the cladistics classification.

The first step consists in defining populations for which we will study the co-evolution. a population is constructed by a set of organisms derived from the same ancestor by adding new characters or capabilities. in order to compare the evolution path of the software product line mobile media with the evolution path of its derived product, we must build their evolution paths. We constructed two populations, the first one P1 is formed by seven versions of the software product line mobile media, while the second one P2 is formed by seven versions of the derived product. In Tables 1 and 2, respectively, we give a detailed description of the populations P1 and P2, which we constructed on the basis of the feature models of the different releases of P1 and P2:

After we defined the two populations P1 and P2, the second step consists in defining the characters of each population. In biology, the characters of organisms of a population represent their visible traits, which could be physical or behavioral characteristics. Hence, for each population, we will identify its behavioral characters that are its features. By assuming the hypothesis $H0$ of features Independence, the set of characters of a population will be composed by independent features. In Tables 1 and 2, respectively, we formulated the vectors of features of the populations P1 and P2 as follow, the number of features are 19 and 16 for $A1$ and $A2$, respectively:

$$A1 = \{F_{1,i} \text{ while } i \in N, i \leq 19\},$$
$$A2 = \{F_{2,i} \text{ while } i \in N, i \leq 16\}$$

In order to classify the versions of each population, we will construct in the third step the features states matrices that illustrate the states of features in each version. Each feature has two states, the primitive state, which denotes the nonexistence of the feature and it is represented by 0, and the derived state, which denotes its existence and it is represented by 1. The features state matrices of our populations P1 and P2 are presented in Tables 3 and 4, respectively. We construct cladograms on the basis of these matrices by grouping versions together based on their shared characters. In this steps we used the tools PHYLIP to generate the coordinates of the evolutionary trees of P1 and P2 from their features state

TABLE I. MOBILE MEDIA SOFTWARE PRODUCT LINE POPULATION

Version	Description
V1.0	The first release of the mobile media software product line, this release encompasses the following features: Manage photos ($F_{1,1}$), Create album ($F_{1,2}$), Delete album($F_{1,3}$), Create media ($F_{1,4}$), Delete media ($F_{1,5}$), View media ($F_{1,6}$), Sort media ($F_{1,7}$), Edit media label ($F_{1,8}$)
V1.1	The second release of the mobile media software product line, in which the following features were added: Set favorites ($F_{1,9}$) and See favorites ($F_{1,10}$)
V1.2	The third release of the mobile media software product line, in which the feature Copy media ($F_{1,11}$) was added
V1.3	The fourth release of the mobile media software product line, in which the following features were added: Send media ($F_{1,12}$) and Receive media ($F_{1,13}$)
V1.4	The fifth release of the mobile media software product line, in which the feature Add music media management ($F_{1,14}$) was added
V1.5	The sixth release of the mobile media software product line, in which the following features were added: Add video media management ($F_{1,15}$), Capture videos ($F_{1,16}$) and Capture photos ($F_{1,17}$)
V1.6	The seventh release of the mobile media software product line, in which the following features were added: Play videos ($F_{1,18}$) and Play music ($F_{1,19}$)

TABLE II. DERIVED PRODUCT POPULATION

Version	Description
V2.0	The first release of the product, this release encompasses the following features: Manage photos ($F_{2,1}$), Create album ($F_{2,2}$), Delete album($F_{2,3}$), Create Photo ($F_{2,4}$), Delete Photo ($F_{2,5}$), View Photo ($F_{2,6}$), Sort media ($F_{2,7}$), Edit media label ($F_{2,8}$)
V2.1	The second release of the product, in which the following features were added: Set favorites ($F_{2,9}$) and See favorites ($F_{2,10}$)
V2.2	The third release of the product, in which the feature Copy media ($F_{2,11}$) was added
V2.3	The fourth release of the mobile media software product line, in which the following features were added: Send media ($F_{2,12}$) and Receive media ($F_{2,13}$)
V2.4	The fifth release of the product, in which the feature Print photo ($F_{2,14}$) was added
V2.5	The sixth release of the product, in which, the feature Capture photos ($F_{2,15}$) was added
V2.6	The seventh release of the product, in which, the feature Share photo in social websites ($F_{2,16}$) was added

 ISBN: 978-1-61208-367-4

TABLE III. FEATURES STATES MATRIX B OF THE MOBILE MEDIA SPL

B	$F_{1,1}$ .. $F_{1,8}$	$F_{1,9}$ $F_{1,10}$	$F_{1,11}$	$F_{1,12}$ $F_{1,13}$	$F_{1,14}$	$F_{1,15}$ .. $F_{1,17}$	$F_{1,18}$ $F_{1,19}$
V0	1	0	0	0	0	0	0
V1	1	1	0	0	0	0	0
V2	1	1	1	0	0	0	0
V3	1	1	1	1	0	0	0
V4	1	1	1	1	1	0	0
V5	1	1	1	1	1	1	0
V6	1	1	1	1	1	1	1

TABLE IV. FEATURES STATES MATRIX C OF THE PRODUCT

C	$F_{2,1}$ .. $F_{2,8}$	$F_{2,9}$ $F_{2,10}$	$F_{2,11}$	$F_{2,12}$ $F_{2,13}$	$F_{2,14}$	$F_{2,15}$	$F_{2,16}$
V0	1	0	0	0	0	0	0
V1	1	1	0	0	0	0	0
V2	1	1	1	0	0	0	0
V3	1	1	1	1	0	0	0
V4	1	1	1	1	1	0	0
V5	1	1	1	1	1	1	0
V6	1	1	1	1	1	1	1

matrices. In Fig. 3. we present the example of the input file of P1. The output file of Phylip contains the coordinates of the evolutionary tree of the population P1, we used the online tool Phyfi which we present in Fig. 4 in order to compile this file and generate the cladogram. The resulted cladograms of P1 and P2 are illustrated in Fig. 5 and Fig. 6, respectively.

C. Co-evolution Analysis

In biology, the perfect co-evolution can be restored by identifying the branches that cause this divergence and extending the cladograms by them. However, by assuming the hypothesis $H1$ that features of the software product line are sufficient but not necessary to derive all the features of the derived products, we will eliminate the imperfection caused by branches that exist in the software product line cladogram and are absent from the products cladogams. Our hypothesis is based on the fact that the software product line feature model take into consideration commonality and also variability of products. In this subsection, we will formulate our hypothesis about the perfect co-evolution in software product lines. Thus, we verify

```
1      7    19
2  V1.0     1111111100000000000
3  V1.1     1111111110000000000
4  V1.2     1111111111000000000
5  V1.3     1111111111110000000
6  V1.4     1111111111111100000
7  V1.5     1111111111111111100
8  V1.6     1111111111111111111
```

Figure 3. Input file for drawing the cladogram of P1

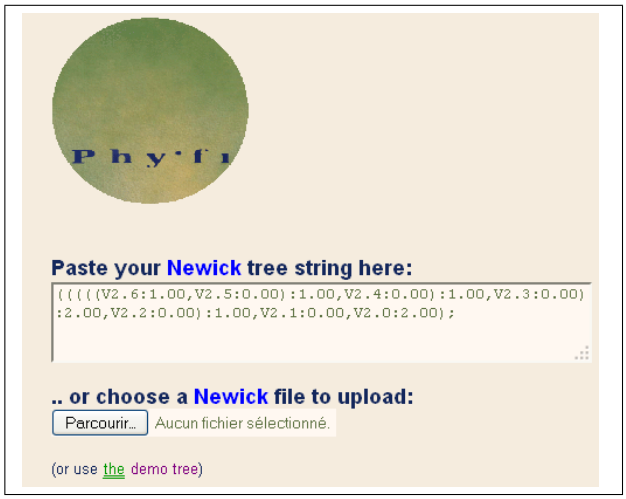

Figure 4. The coordinates of the cladogram P1

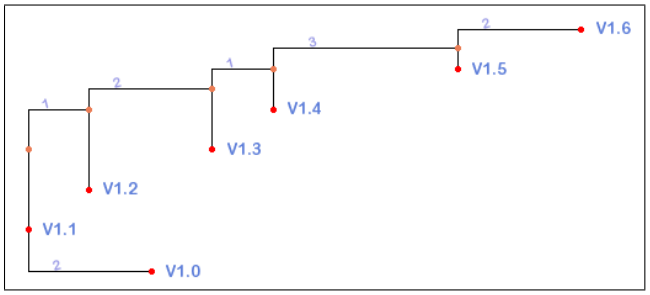

Figure 5. The cladogram of the Mobile Media SPL

these hypothesis for the two populations and we propose an algorithm to correct the extracted imperfections

1) Perfect co-evolution modeling: We set three hypothesis for the software product line, $H0$ and $H1$ are already explained above, in addition we formulated a new hypothesis $H2$ on the basis of $H0$ and $H1$:

(H0) features independence: In the set of features $A1$ and $A2$, the features are independent from each other

$$\forall i, j \in N, i \leq 19, j \leq \text{n}, F_{1,i} \neq F_{1,j}$$
$$\forall i, j \in N, i \leq 16, j \leq \text{m}, F_{2,i} \neq F_{2,j}$$

(H1) domain features sufficiency: Each feature of the set $A2$ has a corresponding feature in the set $A1$

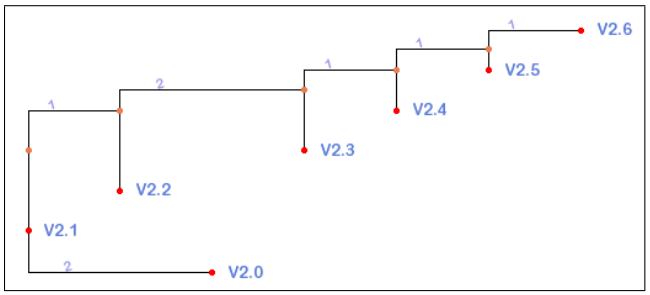

Figure 6. The cladogram of the derived product

$$\forall j \in N, j \leq 16, \exists i \in N, i \leq 19 / F_{2,j} = F_{1,i}$$

(H2) features exclusion: This hypothesis is deduced from the combination of **H0** and **H1**. Each feature of $A2$ has only one corresponding feature in $A1$

$$\begin{cases} F_{2,i}, F_{2,j}/i, j \leq 16\} \subset \{F_{1,x}, F_{1,y}/x, y \leq 19\} \\ F_{2,i} = F_{1,x} \end{cases} \Rightarrow F_{2,j} = F_{1,y}$$

On the basis of the hypotheses **H0** and **H2**, we deduce that the relationship between the two cladogram of P1 and P2 must respect the following inequality:

$$B \times A1 \geq C \times A2 \quad (1)$$

This inequality means that for each couple of leafs $L1i, L2i/0 < i \leq \mathrm{k}$ of the cladograms of P1 and P2, the number of features of $L1i$ must be superior to the number of features of $L2i$. Assuming the hypothesis **H1**, the inequality can be reduced to the following equality. The vector $A3 = \{F_{3,i}$ while $i \in N, i \leq 3\}$ represents the features of the software product line that are not supported by the derived product, and the entries di, j of the matrix D are equal to 0 or 1 depending on weather the features exist in the product or no :

$$B \times A1 - C \times A2 = \begin{pmatrix} d_{1,1} & \cdots & d_{1,s} \\ d_{2,1} & \cdots & d_{2,s} \\ \vdots & \ddots & \vdots \\ d_{k,1} & \cdots & d_{k,s} \end{pmatrix} \times \begin{pmatrix} F_{3,1} \\ F_{3,2} \\ \vdots \\ F_{3,i} \\ \vdots \\ F_{3,s} \end{pmatrix} \quad (2)$$

After the calculation of this equality we will obtain seven equalities as follow:

$$\begin{cases} \sum_{i=1}^{19} b_{1,i} \times F_{1,i} - \sum_{j=1}^{16} c_{1,j} \times F_{2,j} = \sum_{j=1}^{3} d_{1,j} \times F_{3,j} \\ \sum_{i=1}^{19} b_{2,i} \times F_{1,i} - \sum_{j=1}^{16} c_{2,j} \times F_{2,j} = \sum_{j=1}^{3} d_{2,j} \times F_{3,j} \\ \sum_{i=1}^{19} b_{3,i} \times F_{1,i} - \sum_{j=1}^{16} c_{3,j} \times F_{2,j} = \sum_{j=1}^{3} d_{3,j} \times F_{3,j} \\ \sum_{i=1}^{19} b_{4,i} \times F_{1,i} - \sum_{j=1}^{16} c_{4,j} \times F_{2,j} = \sum_{j=1}^{3} d_{4,j} \times F_{3,j} \\ \sum_{i=1}^{19} b_{5,i} \times F_{1,i} - \sum_{j=1}^{16} c_{5,j} \times F_{2,j} = \sum_{j=1}^{3} d_{5,j} \times F_{3,j} \\ \sum_{i=1}^{19} b_{6,i} \times F_{1,i} - \sum_{j=1}^{16} c_{6,j} \times F_{2,j} = \sum_{j=1}^{3} d_{6,j} \times F_{3,j} \\ \sum_{i=1}^{19} b_{7,i} \times F_{1,i} - \sum_{j=1}^{16} c_{7,j} \times F_{2,j} = \sum_{j=1}^{3} d_{7,j} \times F_{3,j} \end{cases} \quad (3)$$

2) Perfect Co-evolution for the Mobile Media Software Product Line: By considering the features states matrices of the mobile media and its product, the inequality (1) can be expressed as follows:

$$\begin{pmatrix} 1 & 1 \cdots & 0 & 0 \\ 1 & 1 \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \\ 1 & 1 \cdots & 1 & 1 \end{pmatrix} \times \begin{pmatrix} F_{1,1} \\ F_{1,2} \\ \vdots \\ F_{1,10} \\ \vdots \\ F_{1,19} \end{pmatrix} \geq \begin{pmatrix} 1 & 1 \cdots & 0 \\ 1 & 1 \cdots & 0 \\ \vdots & \ddots & \vdots \\ 1 & 1 \cdots & 1 \end{pmatrix} \times \begin{pmatrix} F_{2,1} \\ F_{2,2} \\ \vdots \\ F_{2,10} \\ \vdots \\ F_{2,16} \end{pmatrix}$$

TABLE V. NEW FEATURES STATES MATRIX B' OF THE MOBILE MEDIA SPL

B'	$F_{1,1}$ .. $F_{1,8}$	$F_{1,9}$ $F_{1,10}$	$F_{1,11}$	$F_{1,12}$ $F_{1,13}$	$F_{1,14}$	$F_{1,15}$ .. $F_{1,17}$	$F_{1,18}$ $F_{1,19}$	$F_{2,14}$	$F_{2,14}$
V0	1	0	0	0	0	0	0	0	0
V1	1	1	0	0	0	0	0	0	0
V2	1	1	1	0	0	0	0	0	0
V3	1	1	1	1	0	0	0	0	0
V4	1	1	1	1	1	0	0	1	0
V5	1	1	1	1	1	1	0	1	0
V6	1	1	1	1	1	1	1	1	1

We calculate the equalities (3) for the mobile media software product line and its derived product in order to deduce the results of their co-evolution:

$$\begin{cases} \sum_{i=1}^{19} b_{1,i} \times F_{1,i} - \sum_{j=1}^{16} c_{1,j} \times F_{2,j} = 0 \\ \sum_{i=1}^{19} b_{2,i} \times F_{1,i} - \sum_{j=1}^{16} c_{2,j} \times F_{2,j} = 0 \\ \sum_{i=1}^{19} b_{3,i} \times F_{1,i} - \sum_{j=1}^{16} c_{3,j} \times F_{2,j} = 0 \\ \sum_{i=1}^{19} b_{4,i} \times F_{1,i} - \sum_{j=1}^{16} c_{4,j} \times F_{2,j} = 0 \\ \sum_{i=1}^{19} b_{5,i} \times F_{1,i} - \sum_{j=1}^{16} c_{5,j} \times F_{2,j} = \underline{F_{1,14} - F_{2,14}} \\ \sum_{i=1}^{19} b_{6,i} \times F_{1,i} - \sum_{j=1}^{16} c_{6,j} \times F_{2,j} = \underline{F_{1,14} - F_{2,14} + F_{1,15} + F_{1,16}} \\ \sum_{i=1}^{19} b_{7,i} \times F_{1,i} - \sum_{j=1}^{16} c_{7,j} \times F_{2,j} = \underline{F_{1,14} - F_{2,14} + F_{1,15} + F_{1,16} + F_{1,18} + F_{1,19} - F_{2,16}} \end{cases}$$

We notice that two imperfections was detected after the calculation. They are underlined, the first in the fifth equality and the second is in the last equality. These imperfections are caused by the two features $F_{2,14}$ "Print photo" and $F_{2,16}$ "Share photo in social websites". The two features exist in the product and are absent from the software product line. The vector $A3$ is composed by the following features :($F_{1,14}$ "Add music media management", $F_{1,15}$ "Add video media management", $F_{1,16}$ "Capture videos", $F_{1,18}$ "Play videos", $F_{1,19}$ "Play music", $F_{2,14}$ "Print photo", $F_{2,16}$ "share photo in social websites"). From the seven equalities we deduce the matrix D of the inequality (1):

$$B \times A1 - C \times A2 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 1 & 1 & 1 & 0 & 0 & -1 & 0 \\ 1 & 1 & 1 & 1 & 1 & -1 & -1 \end{pmatrix} \times \begin{pmatrix} F_{1,14} \\ F_{1,15} \\ F_{1,16} \\ F_{1,18} \\ F_{1,19} \\ F_{2,14} \\ F_{2,16} \end{pmatrix}$$

3) Imperfect Co-evolution Correction: In order to correct imperfections represented by the negative entries in the matrix D we propose the following algorithm which will restore the missing features to the software product line. By applying this algorithm to the mobile media software product line, the two features $F_{2,14}$ "Print photo" and $F_{2,16}$ "Share photo in social websites" of he derived product will be added to mobile media software product line states features matrix, in Table 5 we present the new matrix B' of the software product line.

Our approach allows for restoring the integrity of the software product line, by propagating features that were developed

Algorithm 1 Imperfect Co-evolution Correction Algorithm

```
//s the number of features of the SPL
//k the number of versions of the SPL
t ← s
for i ← 0 to k do
    while A3[i] ≠ A1[j] & j ≤ s do
        Imperfect ← True
        j ← j + 1
    end while
    if Imperfect = True then
        t ← t + 1
        ResizeA1(t)
        A1[t] ← A3[i]
        for r ← 0 to k do
            B[r][n] ← |D[r][n]|
        end for
    end if
end for
```

Figure 7. Imperfect co-evolution correction algorithm

on the products level to the domain engineering level. In the presented case study the two features "Print photo" and "Share photo in social websites" of the derived product were propagated to the mobile media software product line. The approach aims at preserving the software product line from the aging phenomenon by correcting the divergences between products and the software product line that happened during their evolution.

Our motivation is conducted by the main principal of software product line engineering which is the ability of the domain feature model to support all the features of the derived products. Furthermore, cladistics classification technique allowed us to restore the missing features to the corresponding versions of the software product line in order to achieve the perfect co-evolution of the software product line with its products. This approach enables also the extension of the software product line capabilities, for our example, the two added features can be propagated to the other derived products that manage photos. Thereby, it helps predicting new features and anticipating new requirements, for example the feature "Share photo in social websites" which we propagated to the mobile media software product can also be adapted to include other media such as songs and videos.

V. Conclusion

In this paper, we introduced the co-evolution of domain and application engineering in software product lines, which consists in identifying the evolution paths of the domain assets and the application assets and finding if they are similar or different. Our purpose is to preserve the ability of domain engineering assets to derive the application assets even during their evolution. This purpose can be achieved by propagating the changes that happened to the products only to the software product line. Co-evolution was extensively studied in biology in order to understand how organisms influence each other during their evolution. The co-evolution relies basically on the physical characters or behaviors of organisms. Since the features of a software represent its visible characters, we consider only the co-evolution of feature models in this paper.

We used the biological Cladistics classification to build the evolutionary trees of the software product line and its derived products, then we perform a mathematical analysis to extract similarities and differences between these trees. Thereby, we propose an algorithm to propagate the missing features that cause divergence between the evolutionary trees to the domain feature model. We applied our approach to the software product line of mobile media applications that manage media such as songs, photos and videos on mobile devices. We compared the evolution paths of the software product line and one of its derived products, then we applied our analysis to propagate the features that exists in the product but are missed from the software product line to the domain features model. As a consequence, we restored the ability of the software product line to derive all the products features, and also we extended its capabilities by the new features.

References

[1] K. Pohl, G. Bckle, and F. J. van der Linden, Software product line engineering: foundations, principles and techniques, Springer, 2005.

[2] P. Clements, L. Northrop, and B. W. Boehm, "Software product lines : practices and patterns", Fondo Xavier Clavigero, S.J. ITESO, 2002.

[3] D. L. Parnas, "Software aging", in Proc. The 16th international conference on Software engineering, 1994, pp. 279-287.

[4] P. R. Ehrlich and P. H. Raven, Butterflies and plants: a study in coevolution Evolution, JSTOR, 1964, pp. 586-608.

[5] R. M. Anderson and R. M.May, "Coevolution of hosts and parasites", Parasitology, 1982, vol. 85, no 02, pp. 411-426.

[6] D. Di Ruscio, L. Iovino, and A. Pierantonio, "What is needed for managing co-evolution in MDE?". In Proc. The 2nd International Workshop on Model Comparison in Practice, 2011, pp. 30-38.

[7] M. Kster and M. Trifu, "A case study on co-evolution of software artifacts using integrated views". In Proc. The WICSA/ECSA, 2012, pp. 124-131.

[8] C. Seidl , F. Heidenreich, and U. Amann, "Co-evolution of models and feature mapping in software product lines". In Proc. The 16th International Software Product Line Conference, 2012, pp. 76-85.

[9] Brinkman, S.L. Fiona, and D. D. Leipe, Bioinformatics: a practical guide to the analysis of genes and proteins, Vol. 43, John Wiley Sons, 2004.

[10] L. P. Tizzei, M. Dias, C. M. Rubira, A. Garcia, and J. Lee, "Components meet aspects: assessing design stability of a software product line", Information and Software Technology, Elsevier, 2011, vol. 53, no 2, pp. 121-136.

[11] N. Anquetil, U. Kulesza, R. Mitschke, A. Moreira, J. Royer, A. Rummler, and A. Sousa, "A model-driven traceability framework for software product lines", Software Systems Modeling, Springer, 2010, 9, pp. 427-451.

[12] S. A. Ajila and A. B. Kaba, "Evolution support mechanisms for software product line process", Journal of Systems and Software, Elsevier, 2008, vol. 81, no 10, pp. 1784-1801.

[13] A. Goknil, I. Kurtev, K. van den Berg, and J. Veldhuis, "Semantics of trace relations in requirements models for consistency checking and inferencing", Software Systems Modeling, Springer, 2011, 10, pp. 31-54.

 ISBN: 978-1-61208-367-4