Determining the relative importance of features for influencing software product similarity matching

Mannion, Mike; Kaindl, Hermann

Published in:
2023 IEEE 47th Annual Computers, Software, and Applications Conference (COMPSAC)

DOI:
10.1109/COMPSAC57700.2023.00253

Publication date:
2023

Document Version
Author accepted manuscript

Determining the Relative Importance of Features for Influencing Software Product Similarity Matching

Mike Mannion
Dept of Computing
Glasgow Caledonian University
Glasgow, UK
m.a.g.mannion@gcu.ac.uk

Hermann Kaindl
Institute
TU Wien
Vienna, Austria
hermann.kaindl@tuwien.ac.at

***Abstract*— As a software product line evolves a significant management challenge is comparing existing products to each other or planned products. The approach to product comparison will vary according to its purposes. One solution includes the representation of a configured product as a weighted binary string where 1 represents a feature's presence, 0 represents its absence, and the weight represents the different levels of *relative importance* to the product that a feature is perceived to have. Relative importance values influence similarity matching so that the features considered important are the ones that primarily influence what is judged to be similar. A binary string similarity metric supports product comparison (a *product similarity metric*). For a product line that contains thousands of features the allocation of relative importance values is only practical when done automatically. This paper proposes a novel algorithm for automatically determining the relative importance of each feature. A feature tree can represent a product line in which a feature is a node in the tree and a relationship between features is an edge. A feature's relative importance is calculated as a function of local and global tree structural measures. The local measures are the number of input and output nodes to which a feature is connected and the variability property of each of these nodes. The global measure is the distance of the feature from the root node. A mobile phone worked example illustrates the feasibility of the algorithm.**

Keywords*— *Feature reuse, product similarity, binary strings

I. INTRODUCTION

Software-intensive product line engineering (SPLE) is a discipline to systematically plan, construct, evolve and manage a set of products that share a set of software features and assets that satisfy the specific needs of one or more target markets. SPLE is an alternative approach to building a product line than cloning one or more existing products. The principal stages are product scoping, domain engineering and application engineering. Over time, the range of products and features in each product can evolve for many reasons. These include supplier sales, profit motives, customer demand for more personalized products, customer confusion, personnel changes, market competition, brand positioning, organisational restructures, mergers, outsourcing or changing legislation. As the volume, variety, and velocity (i.e. speed into different markets) of products increases, a significant management challenge is comparing how existing products in the product line are similar to each other or to planned products to be derived from the product line. The approach to product comparison will vary according to its purposes, and the subsequent choice of goals, methods and metrics. In SPLE, a feature model sets out the structural relationships between features in the product line. The process of creating a product from the feature model is product configuration. Feature models are large and complex when product lines have hundreds or thousands of features. They may include abstract features that aid modelling but are unimplemented. They may consist of other product line feature models e.g. a mobile phone product line feature model may include a camera product line feature model.

In [1], to aid the process of product comparison, two products configured from a product line's feature model were represented as weighted binary strings, where 1 represents a feature's presence, 0 its absence, and the weight represents the different levels of relative importance to the product that a feature is perceived to have. A comparison between the two products using binary string metrics [2] was illustrated with the Jaccard Coefficient [3]. However, a method for the automatic allocation of a weight to each feature was not identified. This is a significant issue for feature models with hundreds and thousands of features. Here, our research question is:

RQ1: How can a feature's relative importance in a software product line be calculated automatically for influencing similarity scores between products?

We address this question by considering a product-line feature model as a tree with a root node feature, in which each node is a feature and each edge is a relationship between features. We focus on a feature's relative importance within the feature tree topology. We propose a novel algorithm for calculating a feature's relative importance as a function of the number of input and output nodes to which the feature is connected, the variability property of each of these input and output nodes, and the distance of the feature from the root node. It is an algorithm designed for the comparison of two products so that more important features more strongly influence similarity scores than less important ones.

Section II discusses the elements of product comparison. Section III describes the representation of a product line as a feature tree. Section IV explains the use of weighted binary strings for product comparison where each weight represents a feature's relative importance. Section V sets out an algorithm to calculate a feature's relative importance within a feature tree's topology. Section VI shows the feasibility of the algorithm using a mobile phone worked example. Section VII describes some threats to the validity of the approach. Section VII discusses related work. Section IX presents some conclusions.

II. PRODUCT COMPARISON

Product comparison varies according to its purposes, and the

subsequent choice of goals, methods and metrics.

Comparison Purposes: Business objectives drive different comparison purposes. For example, the consideration of entering a new target market involves understanding how an existing product compares to the products in that market. Avoiding legal liability involves assessing if a product falls within the legislative and regulatory boundaries. Improving product development involves regular reviews of the shape and size of different product clusters. In practice, comparison and decision-making criteria vary. Although they often include product feature similarity, other criteria are also used e.g. sales, costs of maintenance, and profits.

Comparison Goals: Typical goals might be:

1. Is Product X similar to Product Y?
2. Is Product X similar to *any existing* product?
3. Is Product X similar to *any possible* product (that can be generated from the product line)?
4. Is Product X part of a group of similar products based on a selected set of features?

The latter three strategies can be computationally expensive when the numbers of features run into thousands. Our concern is with the task of automatic allocation of feature weights where the goal is to compare Product X to Product Y.

Comparison Methods: There are different methods for implementing the comparison including:

- Compare only selected individual features e.g. features of interest, the most influential.
- Compare only the total number of features common to both products – which reduces the search time but a complicating factor is whether to include or exclude negative matches i.e. does the absence of features advance the case for similarity or not?
- Compare the total number of features that are different between the two products.

Our interest is a comparison across some or all features common to both products.

Comparison Metrics: Some binary string metrics measure feature similarity, others dissimilarity [2]. Both use ratios in which the denominator is a function of the total number of features. However, the total varies depending upon the inclusion or exclusion of features absent in both products. The numerator in similarity metrics is often the total number of features common to each product, whereas in dissimilarity metrics it is the total number of features that are different in each product. In this paper, we focus on similarity metrics.

Relative importance: Some features are perceived with different importance than others and merit a different weight in metric calculations. Relative importance can be allocated based on whether a feature has a unique selling point, its importance to a key customer or other stakeholders, its importance to other features, internal organizational structures, process design or implementation considerations, or planned outsourcing arrangements. Our focus in this paper is on a feature's relative importance to other features within the feature tree topology.

Knowing the relative importance of different features can affect how comparisons are undertaken. In some comparisons, when comparing across all features, including relative importance is useful to minimize the risk of two products viewed as similar when one product is missing features of high relative importance but makes up the feature deficit with features of low relative importance. In other comparisons, only features with relative importance above a certain threshold are considered.

The process of allocation of relative importance to a feature is undertaken before product comparison takes place and depends on the purpose of the product comparison. It involves getting the agreement of organizational stakeholders, each of whom may have different perspectives. In practice, several factors make the consistent agreement on the allocation of weights to features difficult to maintain. They include when a product line model grows to hundreds or even thousands of features when the people who work on the product line come and go over time, or when there are many incremental changes. They motivate seeking an automatic weight allocation method. A starting point for this work is to represent a product line as a form of a graphical network and run algorithms over the topology of the network. The most popular graphical representation is a feature model as a *tree* with additional crosscutting relationships between nodes (e.g. Fig 1).

III. Product Line Feature Models

Different factors inform the choice of a feature model representation including its purpose, its ease of comprehension and navigation, the simplicity of encoding and analysing features and their properties, and the complexity of product configuration. Many feature models, languages, notations and support tools have been proposed [4-17]. Product configuration relies on making selection decisions from a feature model. Each feature has a property *variability type*. This property determines whether a feature must be selected or not during configuration. Table I shows four commonly used variability types [18].

TABLE I. Description of Variability Type

Variability Type	Description
Mandatory	'mandatory' feature automatically selected.
Exclusive-OR	set of mutually exclusive choices, only one must be selected
Inclusive-OR	set of choices, some or all must be selected
Option	single choice which may or may not be selected

When a feature's variability type is mandatory, it is selected when its parent is selected. When a feature's variability type is Exclusive-OR then there can be many mutually exclusive features but only one is selected. When a feature's variability type is Inclusive-OR one or more values are selected. When a feature's variability type is optional, one may select it or not. Crosscutting selection relationships between nodes with different parents are more difficult to express and often done separately in text. In [18] such relationships are restricted to *requires* and *excludes* i.e. one feature requires another feature to have a specific value or one feature requires another feature to *not* have a specific value. In practice, these relationships can be more complex. For example, in Figure 1 the Pokemon Game needs a High Definition Screen and a RAM value that excludes 4Gb. In [12] a product-preserving transformation is presented

that converts complex constraints into the elements of the basic feature modelling language by introducing abstract features where appropriate. This makes the resultant feature model larger but easier to process.

IV. Comparing Products using Weighted Binary Strings

Binary string similarity metrics can be constructed from a small number of binary string count variables. Table II defines four variables for use when comparing two binary strings, B1 and B2, each with N digits.

TABLE II. Binary String Variables

B_{11}	the number of binary digits where feature$_i$ in B1 is 1 and feature$_i$ in B2 is 1.
B_{00}	the number of binary digits where feature$_i$ in B1 is 0 and feature$_i$ in B2 is 0.
B_{01}	the number of binary digits where feature$_i$ in B1 is 0 and feature$_i$ in B2 is 1.
B_{10}	the number of binary digits where feature$_i$ in B1 is 1 and feature$_i$ in B2 is 0.

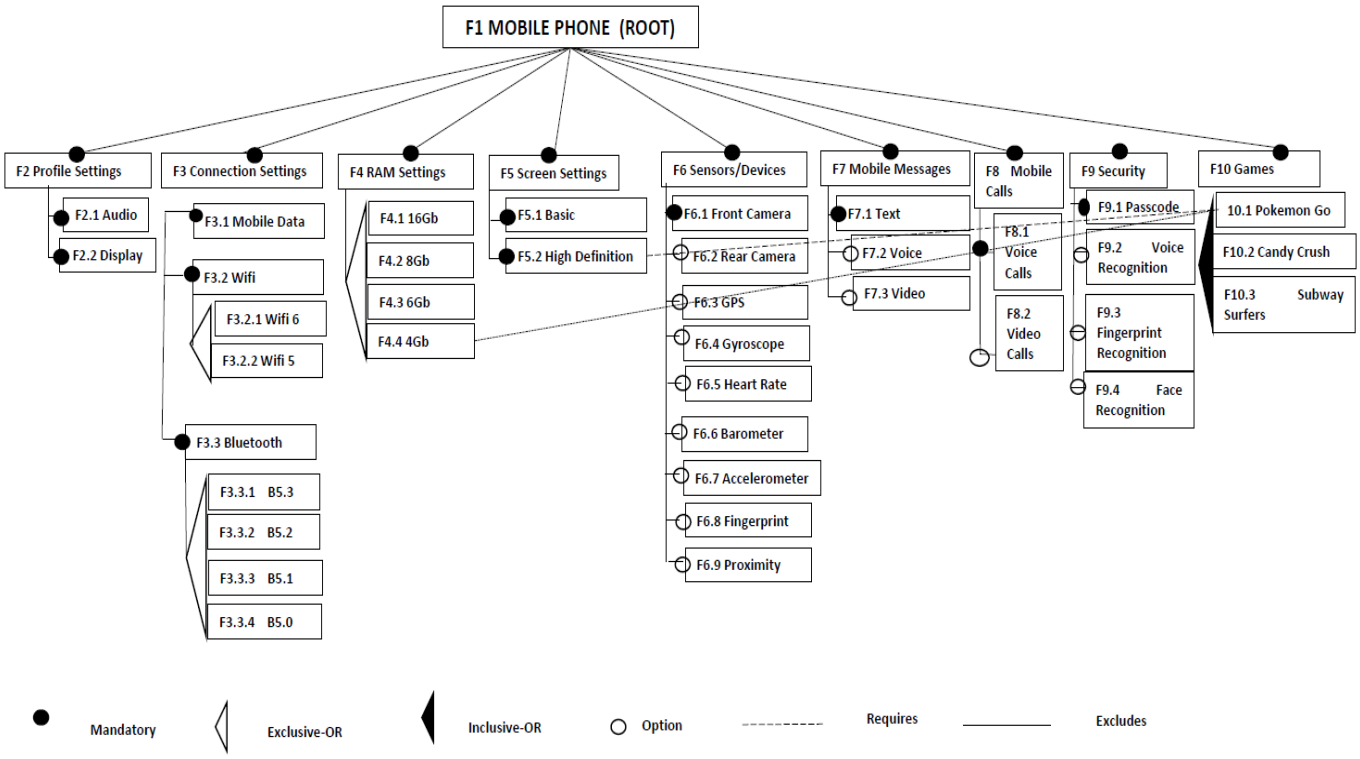

Fig. 1. Feature Model of Mobile Phone Product Line and Four Derived Products

Eq (1)−(4) show calculations for each string variable, where PA_i is the value of the digit at the ith position of the binary string representing product PA, and PB_i is the value of the digit at the ith position of the binary string representing product PB, and w_i is the weight of a feature. Eq (5) shows the calculation for the Jaccard Coefficient.

$$B_{11} = \sum_{i=0}^{N} w_i PA_i \mid (PA_i=1, PB_i=1) \quad \text{Eq (1)}$$

$$B_{00} = \sum_{i=0}^{N} w_i PA_i \mid (PA_i=0, PB_i=0) \quad \text{Eq (2)}$$

$$B_{01} = \sum_{i=0}^{N} w_i PA_i \mid (PA_i=0, PB_i=1) \quad \text{Eq (3)}$$

$$B_{10} = \sum_{i=0}^{N} w_i PA_i \mid (PA_i=1, PB_i=0) \quad \text{Eq (4)}$$

The Jaccard Coefficient is defined as

$$JC = (B_{11}) / (B_{11} + B_{01} + B_{10}) \quad \text{Eq (5)}$$

Example: From Figure 1, Table III shows the connection settings (F3) for two product configurations A and B. The weighted binary string for Product A is 110110001, and for Product B is 111011010. Let the weight of each feature be 1. Table IV shows the binary string variables for comparing A and B. From Eq (5) the Jaccard Coefficient = 4 / (4+2+2) = 0.5 i.e. there is a 50% similarity between A and B for F3. Whilst this example shows a comparison of a subset of features the approach can be deployed for holistic product comparison using all features in one long product string.

TABLE III. PRODUCTS A AND B AS BINARY STRINGS

Feature	Product A	Product B
F3	1	1
F3.1	1	1
F3.2	1	1
F3.2.1	0	1
F3.2.2	1	0
F3.3	1	1
F3.3.1	0	1
F3.3.2	0	0
F3.3.3	0	0
F3.3.4	1	0

V. DETERMINING A FEATURE'S RELATIVE IMPORTANCE

We focus here on a feature's relative importance within the feature tree topology. When features are numerous, determining the relative importance of each feature by getting an all-stakeholder agreement is untenable. An alternative approach is to leverage the topology of the feature tree, recognizing that topology design itself is a political process, and can evolve.

Social network analysis is deployed in many applications e.g. city networks, biological networks, social media, entertainment, viral marketing. The merits of applying different combinations of social network analysis metrics have long been studied. Freeman [19] described a set of measures for computing the global importance of each node in a network based on its connectivity. White [20] defined a general framework for evaluating different properties of different network analysis algorithms. Various aspects of social network construction, analysis and mining are set out in [21]. Network *centrality* is used to describe the relative importance of a node in a network. It is normally a measure of connectivity to other nodes. There can be *local* and *global* measures of centrality corresponding to the notion that a node in a network might be well connected in its local environment but that environment might not be well connected to the rest of the network. We propose a feature's *relative importance* for product comparison is calculated using local and global measures and is a function of the number of features it is locally connected to, the strength of each of these connections and the distance between the feature and the root.

The *degree* of each feature vertex is a local centrality measure that is the sum of the total number of edges going into the vertex (in-degree) and the total number of edges coming out of the feature vertex (out-degree). In Figure 1, Table V shows the degree of each vertex for each connection setting under F3. We chose to determine the *connection strength* of a feature as a function of the variability type of the two connected features. For example, when two features are both of variability type *Mandatory*, the strength of the relationship between the two nodes is very strong. When two features are of variability type *Option*, the strength of the relationship between the two features is weaker. Table VI sets out a proposed set of weights for different combinations of feature variability types. Each weight has been normalised to arrive at a value between 0 and 1. For example, from Figure 1, since the variability type of F3.2 (start vertex) is *Mandatory* and that of F3.2.1 (end vertex) is *XOR*, the weight value of the edge joining F3.2 and F3.2.1 is 12/16.

To calculate the overall connection strength of a feature F_i, we need to add the connection strengths of all its inputs and outputs from other features. To normalise this value between 0 and 1, we can divide it by the degree of the feature. This is represented in Eq (6) where F_i(*connection strength*) is the total connection strength of a feature F_i, where F_iG_{ij}(*connection strength*) is the connection strength of each feature G_j to which F_i is connected, and where D_{Fi} is the degree of the feature F_i.

TABLE IV. DEGREES FOR FEATURES UNDER F3

Feature	In-Degree	Out-Degree	Degree
F3	1	3	4
F3.1	1	0	1
F3.2	1	2	3
F3.2.1	1	0	1
F3.2.2	1	0	1
F3.3	1	4	5
F3.3.1	1	0	1
F3.3.2	1	0	1
F3.3.3	1	0	1
F3.3.4	1	0	1

TABLE V. CONNECTION STRENGTHS BETWEEN FEATURES WITH DIFFERENT VARIABILITY TYPES

START Feature	Mandatory	12/16	13/16	14/16	16/16
	Exclusive-OR	10/16	12/16	13/16	14/16
	Inclusive-OR	8/16	10/16	12/16	13/16
	Option	4/16	8/16	10/16	12/16
		Option	Inclusive-OR	Exclusive -OR	Mandatory
	END Feature				

$$F_i(\text{connection strength}) = \{\sum_{j=0}^{D} F_iG_{ij}(\text{connection strength}) \quad \text{Eq (6)}$$

For example, in Figure 1, the connection strength of F3 is

F3(connection strength)
= Σ connection strengths to F1, F3.1, F3.2, F3.3
={16/16 + 16/16 + 16/16 + 16/16}
= 4 Eq (7)

One global measure of centrality is the *shortest distance* between a feature and the root of the feature tree. This distance is computed by counting the number of edges when traversing a path between them. In many feature trees, the most prominent features are located near the root. We considered that for structural product comparison, the closer to the root a feature was located, the more weight it carried. For the example in Figure 1, Table VII shows the distances between F1, the root feature, and the connection settings features under F3.

TABLE VI. DISTANCES BETWEEN F1 AND FEATURES UNDER F3

Features	Distance
F1 to F3	1
F1 to F3.1	2
F1 to F3.2	2
F1 to F3.2.1	3
F1 to F3.2.2	3
F1 to F3.3	2
F1 to F3.3.1	3
F1 to F3.3.2	3
F1 to F3.3.3	3
F1 to F3.3.4	3

In summary, the relative importance of a feature is governed

by the size and the strength of its connections to the features in its immediate proximity, moderated by their distance from the root. Hence,

F(relative importance)=
F(connection strength)/F(root distance) Eq (8)

For example. using Eq (7) and Eq (8)

F3(relative importance)
= F3(connection strength)/F3(root distance) Eq (9)
= 4 / 1 = 4

TABLE VII. MOBILE PHONE PRODUCT LINE FEATURE MODEL AND DERIVED PRODUCTS

Feature	Feature	Variability Type	Connected Strength	Root Dist	Relative Importance	Basic	Business	Leisure	Gold
F2	*Profile Settings*	Mandatory	3.00	1	3.00	✓	✓	✓	✓
F2.1	Audio	Mandatory	1.00	2	0.50	✓	✓	✓	✓
F2.2	Display	Mandatory	1.00	2	0.50	✓	✓	✓	✓
F3	*Connection Settings*	Mandatory	4.00	1	4.00	✓	✓	✓	✓
F3.1	Mobile Data	Mandatory	1.00	2	0.50	✓	✓	✓	✓
F3.2	Wi-Fi	Mandatory	2.75	2	1.38	✓	✓	✓	✓
F3.2.1	Wifi-6	Exclusive-OR	0.88	3	0.29				✓
F3.2.2	Wifi-5	Exclusive-OR	0.88	3	0.29	✓	✓	✓	
F3.3	Bluetooth	Mandatory	4.50	2	2.25	✓	✓	✓	✓
F3.3.1	5.3	Exclusive-OR	0.88	3	0.29				✓
F3.3.2	5.2	Exclusive-OR	0.88	3	0.29		✓		
F3.3.3	5.1	Exclusive-OR	0.88	3	0.29			✓	
F3.3.4	5.0	Exclusive-OR	0.88	3	0.29	✓			
F4	*RAM Settings*	Mandatory	4.50	1	4.50	✓	✓	✓	✓
F4.1	16Gb	Exclusive-OR	0.88	2	0.44				✓
F4.2	8Gb	Exclusive-OR	0.88	2	0.44		✓		
F4.3	6 Gb	Exclusive-OR	0.88	2	0.44			✓	
F4.4	4Gb	Exclusive-OR	1.63	2	0.81	✓			
F5	*Screen Settings*	Mandatory	3.00	1	3.00	✓	✓	✓	✓
F5.1	Basic	Mandatory	1.00	2	0.50	✓	✓	✓	✓
F5.2	High Definition	Optional	1.81	2	0.91		✓	✓	✓
F6	*Sensors, Devices*	Mandatory	8.00	1	8.00	✓	✓	✓	✓
F6.1	Front Camera	Mandatory	1.00	2	0.50	✓	✓	✓	✓
F6.2	Rear Camera	Optional	0.75	2	0.38		✓	✓	✓
F6.3	GPS	Optional	0.75	2	0.38	✓	✓	✓	✓
F6.4	Gyroscope	Optional	0.75	2	0.38				✓
F6.5	Heart rate	Optional	0.75	2	0.38		✓	✓	✓
F6.6	Barometer	Optional	0.75	2	0.38				✓
F6.7	Accelerometer	Optional	0.75	2	0.38				✓
F6.8	Fingerprint	Optional	0.75	2	0.38		✓	✓	✓
F6.9	Proximity	Optional	0.75	2	0.38				✓
F7	*Mobile Messages*	Mandatory	3.50	1	3.50	✓	✓	✓	✓
F7.1	Text Message	Mandatory	1.00	2	0.50	✓	✓	✓	✓
F7.2	Voice Message	Optional	0.75	2	0.38	✓	✓	✓	✓
F7.3	Video Message	Optional	0.75	2	0.38		✓	✓	✓
F8	*Mobile Calls*	Mandatory	2.75	1	2.75	✓	✓	✓	✓
F8.1	Voice Call	Mandatory	1.00	2	0.50	✓	✓	✓	✓
F8.2	Video Call	Optional	0.75	2	0.38		✓	✓	✓
F9	*Security*	Mandatory	4.25	1	4.25	✓	✓	✓	✓
F9.1	Passcode	Mandatory	1.00	2	0.50	✓	✓	✓	✓
F9.2	Voice Recognition	Optional	0.75	2	0.38		✓	✓	✓
F9.3	Fingerprint Recognition	Optional	0.75	2	0.38		✓	✓	✓
F9.4	Face Recognition	Optional	0.75	2	0.38		✓		✓
F10	*Games*	Optional	3.44	1	3.44	✓	✓	✓	✓
F10.1	Pokemon Go	Inclusive-OR	0.81	2	0.41	✓	✓	✓	✓
F10.2	Candy Crush	Inclusive-OR	0.81	2	0.41			✓	✓
F10.3	Subway Surfers	Inclusive-OR	0.81	2	0.41			✓	✓

VI. MOBILE PHONE WORKED EXAMPLE

Table VIII shows the features of four different mobile phone products derived from the feature tree in Figure 1. The Basic phone enables telephone calls or text messages. The Business Phone offers high-quality communication tools. The Leisure phone is a communication and entertainment tool. The Gold phone has the most features. There are several sub-trees: Profile Settings, Connection Settings, RAM Settings, Screen Settings, Sensors and Devices, Mobile Messages, Mobile Calls, Security, and Games. Table VIII also shows each feature's variability type, its connected strength calculated from Eq 6, its distance to

the root and its relative importance, calculated from Eq 7.

Table IX shows the binary string variable values and Jaccard Coefficients for comparing the Basic, Business and Leisure phones against the Gold. Note that for feature F4.4 (end feature), the connection strength is calculated from its connections with feature F4 (start feature) and feature F10.1 (start feature). Similarly, for feature F5.2 (end feature), its connection strength is calculated from its connections with feature F5 (start feature) and Feature F10.1 (start feature). In calculating connection strength we did not distinguish between *contain*, *requires* or *exclude* relationships. We considered these relationships as informing tree traversal methods during the product configuration process rather than having value for post-configuration comparison.

The three Jaccard Coefficient values for the comparisons between Basic vs. Gold, Leisure vs. Gold and Business vs. Gold when weights are not attributed are 0.59, 0.72, 0.75, suggesting that the Leisure and Business phones are closer in similarity to the Gold phone than the Basic Phone. The three Jaccard Coefficient values for the comparisons between Basic vs. Gold, Leisure vs. Gold and Business vs. Gold when weights are attributed are 0.89, 0.92, 0.93. This suggests that all three phones have the same level of similarity to the Gold phone. It reflects that across the three phones, there is much similarity across those features near the tree's root. The features with the largest relative influence are F2, F3, F4, F5, F6, F7, F8, F9 and F10.

Table X unpacks the detail of these results across specific elements of each product, showing where there are product differences and the same functionality. The shading shows the differences e.g. the greatest areas of difference between the Basic phone and the Gold phone are in F4 (*RAM Settings, 78% similarity*) and F5 (*Screen Settings, 79% similarity*).

This sort of result can inform judgements about the product assortment strategy e.g. to what extent do levels of similarity affect brand reputation, customer confusion, sales, profits, product roadmaps, pricing strategies and online vs offline channel mix? Are there similarity thresholds beyond which these variables are negatively affected? what effect will new product variations have on the company's market share?

TABLE VIII. BINARY STRING VARIABLE VALUES FOR THREE PRODUCT COMPARISONS

ALL Features	**Basic vs. Gold**		**Leisure vs. Gold**		**Business vs. Gold**	
	No Weights	*Weights*	*No Weights*	*Weights*	*No Weights*	*Weights*
B_{11}	22.00	45.22	31.00	49.13	33.00	49.94
B_{01}	12.00	4.40	9.00	3.33	8.00	2.52
B_{10}	3.00	1.40	3.00	1.02	3.00	1.02
B_{00}	4.00	1.83	4.00	1.83	4.00	1.83
JC	0.59	0.89	0.72	0.92	0.75	0.93

TABLE IX. INDIVIDUAL FEATURE COMPARISONS FOR THE BASIC, LEISURE & BUSINESS PHONES AGAINST THE GOLD PHONE

	Feature	**Basic vs. Gold**	**Leisure vs. Gold**	**Business vs. Gold**
F2	Profile Settings	1.0	1.0	1.0
F3	Connection Settings	0.87	0.87	0.87
F4	RAM Settings	0.78	0.84	0.84
F5	Screen Settings	0.79	1.0	1.0
F6	Sensors, Devices	0.82	0.89	0.89
F7	Mobile Messages	0.95	1.0	1.0
F8	Mobile Calls	0.93	1.0	1.0
F9	Security	0.87	1.0	1.0
F10	Games	0.82	0.82	1.0

VII. DISCUSSION AND THREATS TO VALIDITY

The method for calculating relative importance depends on the purpose for which a product comparison is undertaken. We focused on a feature's relative importance within the feature tree topology. This may be sufficient for a product comparison based only on the presence and absence of features. For other types of comparison, other properties of a feature may be required e.g. although we did use one intrinsic feature property (*variability type*) we recognise that others might be helpful too.

The measures used to calculate the relative importance were a combination of a local measure, *degree*, for connection strength and a global measure, *closeness centrality*, for distance from the root. However, other combinations of local and global measures can be explored. As an alternative global measure to closeness centrality one might use *betweenness centrality* which measures the number of times a feature falls on the shortest path between two other features. Another idea is whether an individual feature merits being attributed some additional "reputational" value in a larger significant group of features. We also signalled that a feature further away from the root carried less weight. Some argue that as leaf features are often closer to the actual implementation their weights should be greater.

The automatic weight allocation algorithm benefits from being straightforward to calculate although the time to complete the set of shortest distance calculations will be proportionate to the number of features in the graph. We showed a comparison of unweighted versus weighted examples, but we recognise that we have not sense-checked, for a larger example, the outcome of an automated method with a manual method in which some weights are independently assigned by experts.

We do not claim that the Jaccard Coefficient (JC) is the best metric choice - other binary string metrics can be lines of investigation. We used the JC as a metric that excludes negative matches. There are benefits and limitations to including or excluding negative matches. Much depends on the application context and the required level of granularity for decision-making e.g. if two similarity metrics generate values within a small margin and a larger margin is sufficient for decision-making

then metrics choice is less important.

In determining the range of values selected in Table VI, we applied some broad principles. These were: (i) the lowest value was to be when an optional feature was connected to an optional feature (ii) the highest value was to be when a mandatory feature was connected to a mandatory feature (iii) exclusive-OR was regarded as a more powerful connection than inclusive-OR (iv) the values needed to lie between 0 and 1. We recognise that the size of the range between the lowest value and the highest value may have an impact on the emergent values of connected strength and hence relative importance. We acknowledge that different values for this table might be conceived and explored.

The design of the topology of a tree will affect the relative importance allocated to each feature. Changes made to the tree will require the algorithm to be re-run. Computationally this is straightforward. Such change is a natural part of the evolution of a product line as the relative importance of features ebbs and flows. For example, over time, the variability type of a feature can change e.g. an option in a high-end version of a product line can, years later, be mandatory in all versions of the product line.

VIII. Related Work

The discovery of important nodes in a graph is a popular theme. Influential nodes are typically nodes on which the graph structure depends to maintain its cohesiveness or nodes that can spread a message widely in the network. Application contexts inform different approaches. In the context of Machine Learning "Relative Feature Importance (RFI)" helps identify the importance of one feature relative to a subset of features [22]. RFI is cast as a generalisation of Permanent Feature Importance, which quantifies the overall reliance of the model on a feature and Conditional Feature Importance, which quantifies its unique contributions to all remaining features. A review of techniques for analysing world city networks is in [23]. In [24] a novel algorithm decreases the computational complexity of closeness centrality calculations by applying a community detection algorithm to extract community structures of the network. A best node becomes the critical node for each community and after reviewing links between communities, another best node as a gateway node is found. Other similar work includes [25-29].

In search-based software engineering, much works centres on the value of using similarity-matching processes and tools to enhance different aspects of software development. These efforts vary in how they address the different comparison goals listed in Section 2. In [30] a Hamming distance metric was used to reduce product-testing times to prioritize the testing order of a subset of products. Other work along the same lines includes [31-34]. A wide range of similarity data management solutions is proposed in [35] for different application domains. To support software development effort estimation a new machine learning software description similarity model was trained [36] using a paragraph vector algorithm on software project descriptions that are available in the GitHub repository. To address the challenge that software developers have in searching cross-language open repositories, a machine learning tool was developed [37] to detect similar applications written in different languages. In [38] a method is presented for mining prior feature interactions in a software product line's artefacts to predict unwanted interactions between a new feature and existing features. In [39] many different methods, techniques and tools approaches are described about how to tackle the re-engineering of software-intensive legacy systems into software product lines.

Automatic feature weight allocation is of interest for classification and clustering algorithms. Weight adjustments are made to accommodate the error between the observed and predicted similarity. Many algorithms have been designed to enhance accuracy and performance. In [40] integrating differential evolution and dynamic differential evolution strategies are explored. Other work [41] examined if dominant features skew the result of similarity functions. In [42] the use of an inter-feature correlation coefficient can minimize the impact of redundancy of a feature's contribution to classification – greater correlation leads to greater redundancy. In [43], entity matching and image classification in building a multi-modal model for similarity search to support e-commerce companies manage product portfolios.

IX. Conclusion

The contribution of this paper is an algorithm for computing scores for the relative importance of a feature in a software product line as a function of the *degree* of each feature, the variability property of each feature and each feature's distance from the root. It is an algorithm designed for the comparison goal context of "How similar is Product X to Product Y?", so that more important features more strongly influence similarity scores than less important ones. The determination of relative importance is separate from the type of product comparison being undertaken. However, to illustrate the value of the algorithm we presented an example of undertaking a structural product comparison of two configured products based on feature presence and feature absence. This comparison showed the feasibility of the algorithm, letting the more important features influence the similarity scores in such a way that they became higher and emphasise plausible similarities more strongly. Other types of comparison purpose might include the cost-benefits of alternative implementations of a set of features or the extent of change that has occurred in different products to the same feature. These comparisons may, in turn, be a characteristic of different development stages e.g. strategic planning, architectural design, managing the technical debt that arises when prioritizing a quick fix release to address a product bug. We recognize for these types of comparisons, that different metrics beyond binary string metrics may be required.

Here, the "importance" of a feature relative to others is defined to influence similarity matching. Future work will investigate whether such importance scores may be used for other purposes. This may entail however defining importance slightly differently.

References

[1] M Mannion and H Kaindl, "Using Binary Strings for Comparing Products from Software-Intensive Systems Product Lines", Proc. 24th International Conference on Enterprise Information Systems", Prague, Czech Republic, Apr 26-28 2021, pp. 257–266.

[2] S Choi, S Cha, and C Tappert, "A Survey of Binary Similarity and Distance Measures" in Systemics, Cybernetics and Informatics, 8, 1, 2010, pp 43-48.

[3] P Jaccard, “Distribution de La Flore Alpine dans Le Bassin Des Dranses et Dans Quelques Régions Voisines” in Bulletin de la Société Vaudoise des Sci Naturelles, 37, 1901, pp. 241-272.

[4] K C Kang, S Cohen, J Hess, W E Novak, and A. Peterson, “Feature-Oriented Domain Analysis (FODA) Feasibility Study”, Carnegie-Mellon Software Engineering Institute Tech Report CMU/SEI-90-TR-21, 1990.

[5] K C Kang, S Kim, J Lee, K Kim, G J Kim and E Shin, “FORM: A Feature-Oriented Reuse Method with Domain-Specific Reference Architectures”, Annals of Software Engineering 5, 1, 1998, pp. 143–168.

[6] D. Batory, “Feature Models, Grammars, and Propositional Formulas”, Proc. Int’l Software Product Line Conf. (SPLC). Springer, Berlin, Heidelberg, 2005, pp. 7–20.

[7] D. Benavides, A. Ruiz-Cortés, and P. Trinidad, “Automated Reasoning on Feature Models” in Proc. Int’l Conf. Advanced Information Systems Engineering (CAiSE), 2005, pp. 491–503.

[8] K. Czarnecki, S. Helsen, and U. Eisenecker, “Formalizing Cardinality-Based Feature Models and Their Specialization2, Software Process: Improvement and Practice, 10, 2005, pp. 7–29.

[9] P-Y. Schobbens, P. Heymans, and J-C. Trigaux, in “Feature Diagrams: A survey and a formal semantics”, in Proceedings of the IEEE International Conference on Requirements Engineering. IEEE Computer Society, 2006, pp. 139–148.

[10] P-Y. Schobbens, P. Heymans, J-C. Trigaux and Y. Bontemps, “Generic Semantics of Feature Diagrams”, Computer Networks, 51, 2, 2007, pp. 456-479.

[11] M Mannion and H Kaindl, “Using Parameters and Discriminants for Product Line Requirements’, Systems Engineering, 11, 1, 2008, pp 61-80.

[12] A. Knüppel, T. Thüm, S. Mennicke, J. Meinicke and I. Schaefer, “Is There a Mismatch between Real-World Feature Models and Product-Line Research”, ESEC/FSE’17, September 04-08, 2017, Paderborn, Germany, pp. 291-302.

[13] Q. Boucher, A. Classen, P. Faber, and P. Heymans, “Introducing TVL, a Text-based Feature Modelling Language” in Proceedings of the Fourth International Workshop on Variability Modelling of Software-intensive Systems (VaMoSâĂŹ10), Linz, Austria, January. 27–29, pp. 159-162.

[14] M. Rosenmüller, N. Siegmund, T. Thüm and G. Saake. „Multi-Dimensional Variability Modeling” in Proc 5th Inernational Workshop on Variability Modeling of Software-Intensive Systems, Jan 2011 pp. 11–22.

[15] C. Kästner, T. Thüm, G. Saake, J. Feigenspan, T. Leich, F. Wielgorz, and S. Apel. “FeatureIDE: A Tool Framework for Feature-Oriented Software Development”, Proc. Int’l Conf. Software Engineering (ICSE). IEEE, Washington, DC, USA, 2009, pp. 611–614.

[16] M. Mendonça, M. Branco and D. Cowan, “S.P.L.O.T.: Software Product Lines Online Tools” in Proceedings of. Conference on. Object-Oriented Programming, Systems, Languages and Applications (OOPSLA). ACM, New York, NY, USA, 2009, pp. 761–762.

[17] D. Beuche, H. Papajewski, and W. Schröder-Preikschat, “Variability Management with Feature Models” Science of Computer Programming, 2004, 53 (3), pp.333-352.

[18] BS ISO/IEC 26558:2017: Software and systems engineering. Methods and tools for variability modelling in software and systems product line

[19] L. Freeman, “Centrality in social networks. conceptual clarification”, Social Networks, 1979, 1, pp. 215-239.

[20] S. White and P Smyth, “Algorithms for Estimating Relative Importance in Networks”, in 9th ACM SIGKDD International Conf o nKnowledg Discovery and Data Mining . Aug 24-27, 2003 Washington D.C., USA, pp.266-275.

[21] R. lhajj and J. Rokne, (eds) Encyclopedia of Social Network Analysis and Mining, Springer, New York, 2018.

[22] G. Konig. C. Molnar, B. Bischl, and M. Grosse-Wentrup, “Relative Feature Importance” in Proceedings of 25th International Conference on Pattern Recognition, Milan, Italy, Jan 10-15 2021, pp 9318-9325.

[23] S. Xue, L Xiong, Z. Lu, and J. Wu, “Graph-theoretic node importance mining in world city networks: methods and applications”, Information Discovery and Delivery, 45/2, 2017, pp. 57-65.

[24] C. Salavati, A. Abdollahpouri, and M. Zhaleh, “Ranking nodes in complex networks based on local structure and improving closeness centrality”, Neurocomputing (Amsterdam), 2019 Vol 336, pp. 36-45.

[25] Z. Wang, C. Sun, J. Xi, and X. Li, “Influence maximization in social graphs based on community structure and node coverage gain”, Future Generation Computer Systems, 118, 2021, pp. 327-338.

[26] A. Zareie and A. Sheikhahmadi, “A hierarchical approach for influential node ranking in complex social networks”, Expert Systems with Applications, Vol 93, Mar 2018, pp. 200-211.

[27] Y. Ou, Q. Guo, J-L. Xing, and J-G. Liu, “Identification of spreading influence nodes via multi-level structural attributes based on the graph convolutional network”, Expert Systems with Applications, Vol 203, Iss 3 Oct 2022.

[28] B. Yu, Y Zhang, Y Xie, C. Zhang, and K. Pan, “Influence-aware graph neural networks”, Applied Soft Computing Journal, Vol 104, 2021,.

[29] T. Wen and Y. Deng, Identification of influencers in complex networks by local information dimensionality, Information Sciences, V 512, Feb 2020, pp 549-562.

[30] M. Al-Hajjaji, T. Thüm, M. Lochau, J. Meinicke, and G.Saake, “Effective product-line testing using similarity-based product prioritization”, Softw. & Systems Modeling, 18, 1, 2019, pp. 499-521.

[31] C. Henard, M. Papadakis, G. Perrouin, J. Klein, P. Heymans, and Y.L. Le Traon, “By Passing the Combinatorial Explosion: Using Similarity to Generate and Prioritize T-Wise Test Configurations for Software Product Lines”, IEEE Trans. Softw. Eng., 40, 7, pp. 650–670.

[32] A. Sanchez, S. Segura, and A. Ruiz-Cortes, “A Comparison of Test Case Prioritization Criteria for Software Product Lines”, Int. Conf. on Softw. Testing, Verification and Validation, 2014, pp. 41-50.

[33] X. Devroey, G. Perrouin, A. Legay, P-Y. Schobbens, and P. Heymans, “Search-based Similarity-driven Behavioural SPL Testing”, 10th Int’l Workshop on Variability Modelling of Software-intensive Systems, 2016, (VaMoS’16), pp. 89-96.

[34] M. Sahak, D. Jawai, and S. Halim, “An Experiment of Different Similarity Measures on Test Case Prioritization for Software Product Lines”, Journal of Telecommunications, Electronics & Computer Engineering, 2017, 9, 3-4, pp. 177-185.

[35] S. Satoh, A Zimek, I Bartolini B. Jonsson, L. Vadicamo, F. Carrara, M. Aumulle, and R, Pagh (Eds), “Similarity Search and Applications, 13th International Conference, SISAP 2020 Copenhagen, Denmark, September 30 – October 2, 2020 Proceedings.

[36] R. Kapur and B. Sodhi, “OSS Effort Estimation using Software Features Similarity and Developer Activity-Based Measures”, ACM Transactions on Software Engineering Methodology, 2022, Vol. 31 (2), pp. 1-35.

[37] F. Ullah, M. Naeem, H Naeem, X. Cheng, and M. Alazab, “CroLSSim: Cross-language software similarity detector using a hybrid approach of LSA-based AST-MDrep features and CNN-LSTM model”, International Journal of Intelligent Systems, 2022, Vol 37 (9), pp. 5768-5795.

[38] S. Khoshmanesh and R. Lutz, “Feature Similarity: A Method to Detect Unwanted Feature Interactions Earlier in Software Product Lines”, in 12th International Conference on Similarity and Search Applications, 2019, pp. 356-361.

[39] R. Lopez-Herrejon, J. Martinez, W. Assuncao, T Ziadi, M Acher, and S Vergilio, “Handbook of Re-Engineering Software Intensive Systems into Software Product Lines”, Springer, 2023.

[40] C-R Dong, W Ng, W Y Wing, X-Z Wang; P Chan, and D S Yeung, “An improved differential evolution and its application to determining feature weights in similarity-based clustering”, Neurocomputing, 146, 2014, pp.95-103.

[41] H Yazdani, D Ortiz-Arroyo, and H. Kwasnicka, “New Similarity Functions”, in Proceedings of the 3rd IEEE International Conference on Artificial Intelligence and Pattern Recognition, 2016, pp. 47–52.

[42] X Peng and Y Zhu, “A Novel Strategy to Adjust Feature Weights for Data Classification” in Proceedings of the 12th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), 10, 2019, pp. 1-5.

[43] M Hong Le and A Hinneburg, “An Application of Learned Multi-modal Product Similarity to E-Commerce”, in T. Skopal et al. (Eds.): SISAP 2022, LNCS 13590, pp. 25–39, 2022.