Measures to report the Location Problem of Model Fragment Location*

Manuel Ballarín
SVIT Research Group
Universidad San Jorge
Zaragoza, Spain
mballarin@usj.es

Ana C. Marcén
SVIT Research Group
Universidad San Jorge
Zaragoza, Spain
acmarcen@usj.es

Vicente Pelechano
pele@dsic.upv.es
Centro de Investigación en Métodos de Producción de Software
Universitat Politècnica de València
Valencia, Spain

Carlos Cetina
SVIT Research Group
Universidad San Jorge
Zaragoza, Spain
ccetina@usj.es

ABSTRACT

Model Fragment Location (MFL) aims at identifying model elements that are relevant to a requirement, feature or bug. Many MFL approaches have been introduced during the last years, addressing the identification of the model elements that correspond to a specific functionality. However, there is a lack of detail when the measurements about the search space (models) and the measurements about the solution to be found (model fragment) are reported. Generally, the only reported measure is the model size. In this paper we propose to use five measurements (size, volume, density, multiplicity and dispersion) in order to report the location problems. These measurements are the result of analyzing 1.308 MFLs over a family of industrial models during the last four years. Through two MFL approaches we emphasize the importance of these measurements in order to compare results. Our work not only proposes to improve the reporting of the location problem, but we also provide real measurements of location problems, being these measurements useful to other researchers during the design of synthetic location problems.

KEYWORDS

Model Fragment Location, Feature Location, Traceability Link Recovery, Bug Location

ACM Reference Format:
Manuel Ballarín, Ana C. Marcén, Vicente Pelechano, and Carlos Cetina. 2018. Measures to report the Location Problem of Model Fragment Location. In *Proceedings of International Conference on Model Driven Engineering Languages and Systems (MODELS'18).* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

*Produces the permission block, and copyright information

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
MODELS'18, October 2018, Copenhagen, Denmark
© 2018 Copyright held by the owner/author(s).
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

From the timeless traceability activity [21] to recent research efforts on feature location [16], [6], [7] and Bug Location [2], Model Fragment Location (MFL) has been gaining momentum. Overall, these MFL approaches address the identification of the model elements that are relevant to a requirement, feature or bug.

Current MFL approaches have leveraged Information Retrieval, Linguistic, and Search-based techniques to achieve the location of relevant model fragments. These approaches provide in detail the algorithms and the parameters used to tune those algorithms. Nonetheless, there is a lack of detail when the measurements about the search space (models) and the measurements about the solution (model fragment) are reported. Generally, the only reported measure is the model size. However, in most of the cases, the model sizes values are not comparable among different works since different models are measured in different manners.

In this paper we propose to use five measurements (size, volume, density, multiplicity and dispersion) in order to report the location problems during MFL. On the one hand, size and volume measure the search space. On the other hand, density, multiplicity and dispersion measure the solution to be located. Our proposed measures are the result of analyzing 1.308 MFLs performed during the last four years, in the models of industrial dimensions of CAF [1].

Properly reporting the location problem is important, because otherwise it is not possible to compare the performance of different approaches against each other. It is not the same challenge to locate a large model fragment in a small model than to locate a small and scattered model fragment over several large models. We illustrate this phenomenon comparing the performance of two MFL approaches (in terms of precision and recall, which are performance measures widely used by the research community).

Our work not only proposes to improve the reporting of the location problem, but we also provide real measures of location problems during MFL over industrial models. The aim of these values is twofold: (1) that researchers who create synthetic location problems have a reference from real world problems, and (2) to

[1] http://www.caf.net/en

raise awareness among researchers of MFL approaches about the profile of real world MFL problems.

The reminder of the paper is structured as follow. Section 2 provides an overview of MFL. Section 3 clarifies the way to count model elements, and introduces our measurements for the location problem. Section 4 presents the values in the CAF case study. Section 5 performs a statistical analysis to provide evidence of the significance of the results. Section 6 discusses the outcomes of the paper. Section 7 gathers the works related to this one. Finally, Section 8 concludes the paper.

2 OVERVIEW OF MODEL FRAGMENT LOCATION

Traceability Links Recovery (TLR) is one of the most common activities performed during the software system maintenance phase. TLR is concerned with establishing the model fragment that implements a particular natural language requirement. TLR among requirements and models is one type of MFL. Next, we illustrate MFL with TLR.

Figure 1 shows an excerpt, taken from a real-world train of a Train Control and Management Language (TCML) model. TCML is the domain specific language used by our industrial partner, designed following UML conventions. TCML has the expressiveness required to describe both the iteration between the main pieces of equipment installed in a train unit and the non-functional aspects related to regulation.

For TLR, the input query is a natural language requirement, for which the model fragment must be retrieved. The example depicted in Figure 1 shows a natural language requirement describing the high voltage functionality of one of the projects of our industrial partner. This requirement is as follows: "The PLC will command the raise of pantograph (AT_PANT_RAISE_ORDER) if the AT_PANTO_RAISE push button is enabled in the active cabin while the pantograph is down (AT_PANT_DOWN), being the HSCB disabled (AT_HSCB_OFF)".

On TLR, as in other MFL activities such as feature location or bug location, a search space (a model defining the set of all possible solutions) and a solution (model elements to be found) exist. Focusing on the example shown in Figure 1, the search space corresponds to the model (see the top part of the figure) and the solution corresponds to the model fragment (see the bottom right part of the figure).

The top part of Figure 1 shows a train unit furnished with multiple pieces of equipment. The equipments showed are: Knife switch, Cabin A, Pantograph, ACR and HSCB. Each equipment (e.g. PANTOGRAPH) in a train unit has a collection of properties (for instance, AT_PANT_UP) and a collection of orders (for instance, AT_INHIB_PANT_ORDER). The communications among equipments are addressed through the state machines. For instance, to go from a pantograph in down state to a pantograph in up state, the transition named CABIN_PANTO_RAISE is triggered.

The bottom right part of Figure 1 shows the model fragment to be located. The model fragment comprises several model elements including the *HSCB* and its property, the *PANTOGRAPH* with its corresponding property and order and finally the *CABIN A* with its order. Also, the model fragment comprises a part of a state machine which represents the functionality associated with the raising of the pantograph.

3 COUNTING MODEL ELEMENTS

In this paper we propose to use five measurements (size, volume, density, multiplicity and dispersion) in order to report the location problems. On these measurements the count of model elements are important. Currently there are approaches in charge of carrying out this task. [21].

- **C1 Counting the number of elements in a model:** This approach is computed as the number of elements of the model. However, this approach neglects both the complexity of the elements and differences among diagrams.
- **C2 Weight factors per model element:** This approach measures the size of a model taking into account the complexity of the elements because the complexity and information that is provided by the diagram elements are not the same for all the elements. Based on the findings of [12], [19] proposes to define complexity levels (e.g., simple, medium, and large) where each complexity level is related to a weight (e.g., simple = 1, medium = 1.5, large = 2). From these weights, this approach is computed as the number of elements weighed by complexity. Unfortunately, this approach neglects the inherent differences among diagrams.
- **C3 Weight factors per model element per diagram type:** This approach measures the size of a model taking into account not only the complexity of the elements but also the differences among diagrams. To do this, in [19], the weight of each diagram element is calculated as $weight(e) = log_2(|E|)$, where e is the element whose weight is being calculated and E is the class of the element according to the diagram.

After applying these three approaches to a set of models, [19] came to the conclusion that their results are extremely correlated. In fact, none of these approaches yields significant better results than the other ones. Therefore, although any of the presented approaches can be used to count model elements, C1 is the easiest to implement and compute, so it is strongly recommended to use C1.

Facing the different ways to count model elements, we propose to report the size of models by means of C1. However, the size of a model is not enough to measure the search space and the solution in MFL. In the following section, we propose measurements to report the location problem.

3.1 Measurements for Model fragment Location

This section presents the measurements that we propose to report the location problem of MFL. Therefore, some of these measurements focus on measuring the search space, and the rest of them focus on measuring the solution to be searched. The Figure 2 shows the measurements of the search space (Size, and Volume) and the measurements of the solution (Density, Multiplicity, and Dispersion).

In order to measure the search space, we propose the following two measurements:

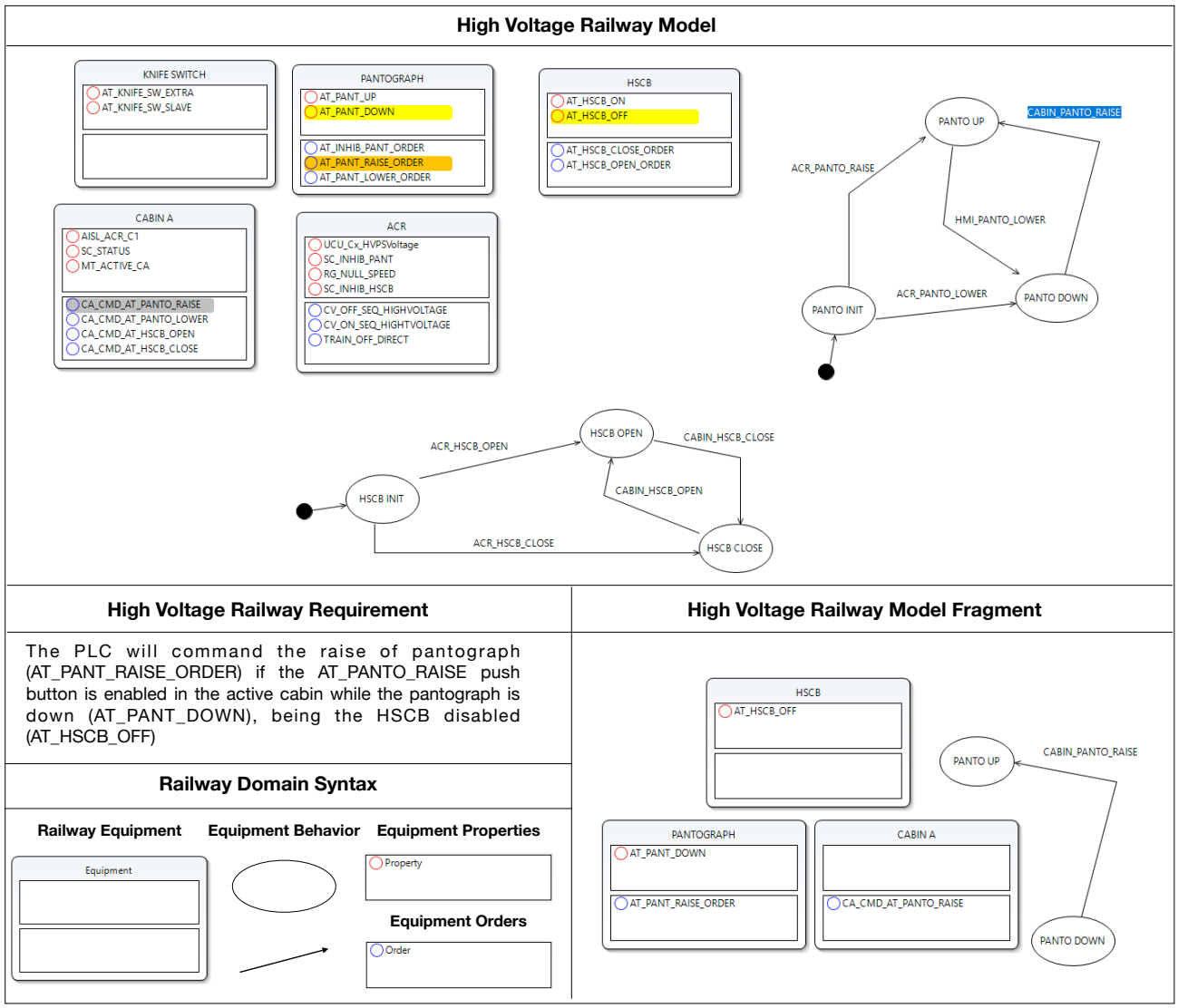

Figure 1: Example of a CAF model and model fragment

- The **Size** measures the number of elements that the model contains from the metric C1. Since the larger the model the larger the search space, this measurement determines how complex the search space ends up being to find the solution.

  The first row of Figure 2 represent conceptually this measurement, where a solution is searched in two different models. The first one is smaller than the second one, so the search space for the first model is smaller than the search space for the second model.

  Given the example of Figure 1, the model is composed of five equipments (knife switch, pantograph, HCSB, cabin, and ACR). Moreover, the state diagrams of the model contains six states and eight transactions. Therefore, the size of the model is computed as the addition of all these elements, so the size is 19 model elements. This measurement is used frequently in most MFL works. However, although this measurement is not a novelty for our work, we include it for the sake of completeness.
- The **Volume** measures the amount of models which compose the search space where a solution is searched. Since the larger the number of models the larger the search space, this measurement determines how large the search space becomes according to the number of models.

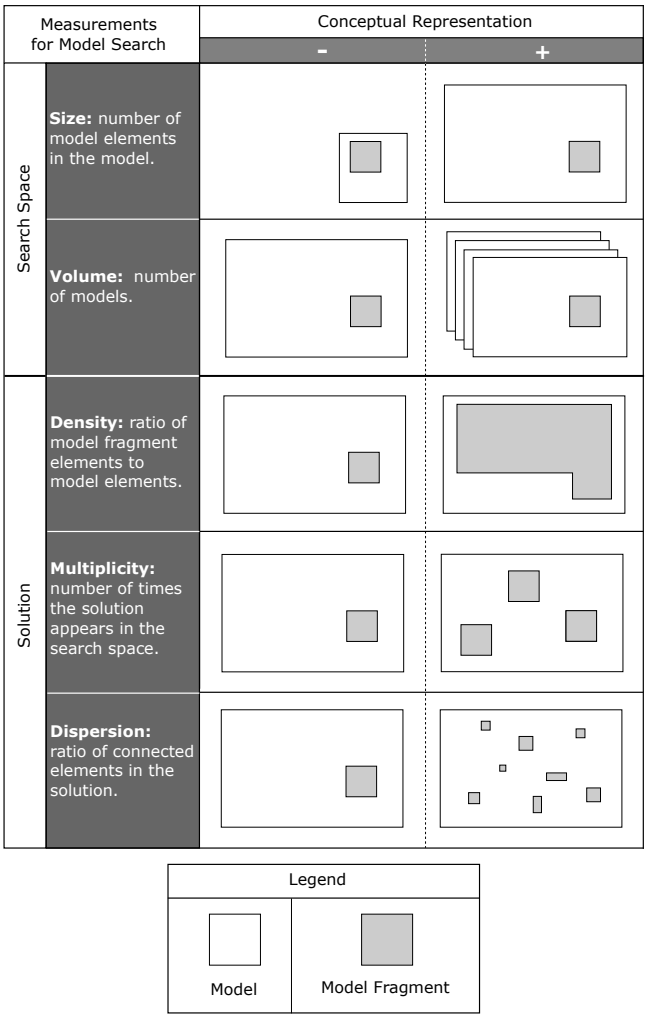

Figure 2: Conceptual representations of the measurements

The second row of Figure 2 represents conceptually this measurement, where a solution is searched in two different search spaces. The first search space is composed of a model, in contrast, the second search space is composed by several models.

In the example of Figure 1, there is only one model. Therefore, the search space is only composed of 1 model, so the volume in this example is equal to 1 model.

In order to measure the solution, we propose the following three measurements:

- The **Density** measures the percentage of model elements which realize a solution. In other words, the model fragment is composed by the model elements that realize the solution, so the density is computed as the ratio of model fragment elements to model elements. Since the larger the model fragment the larger the density, this measurement determines how large the solution ends up being in contrast to the model.

  The third row of Figure 2 represents conceptually this measurement, where a model realizes two different solutions. The first solution is realized by a model fragment which contains a few model elements, in contrast, the second solution is realized by a model fragment which contains the majority of the model elements.

  In the example of Figure 1, the size of the model is equal to 19, and the size of the model fragment can be computed as the number of elements that are pointed out so the size of the model fragment is equals to five. Since the density is equals to 26.31% model elements are part of the solution model fragment.

- The **Multiplicity** measures the number of times the solution appears in the search space. Since the more solutions are found, the greater the multiplicity, this measurement determines how complex the search ends up being, according to the number of solutions that the search space contains for the same solution.

  The fourth row of Figure 2 represents conceptually this measurement, where a solution is searched in two models. The search in the first model reveals a model fragment as the solution. In contrast, the search in the second model reveals three model fragments, so there are three solutions in the model.

  In the example of Figure 1, none of the model elements are repeated, so it is no possible to find two model fragments with the same elements. A bigger model may contain more complex state machines, so it might be possible to find patterns that are repeated. However, in the Figure 1, none of the model elements are repeated, so the multiplicity is equal to 1.

- The **Dispersion** measures the ratio of connected elements in the solution. Specifically, a model fragment is composed of the model elements which realize a solution, but these elements may be connected or not in the model, so the elements of the model fragment can be divided into different groups in the model. The Dispersion is computed as the ratio between the number of groups and the number of elements and its value is from 0 to 1, where values around 0 indicate a strong connection among the solution elements and values around 1 indicate a strong dispersion among the solution elements. Since the more the groups are found the larger the dispersion, this measurement determines how complex the search ends up being if model elements which compose a model fragment are linked or not.

  The fourth row of Figure 2 represents conceptually this measurement, where a solution is realized by two models. In the first model, all the elements that realize the solution are linked, so the model fragment is a unity. However, in the second model, the elements that realize the solution are

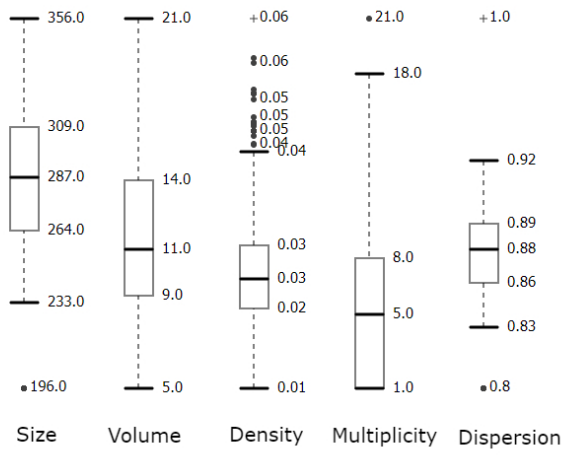

Figure 3: Maximum, Minimum and Mean values of the Case study Measurements

disconnected, so the model fragment is divided into groups where each group is composed by one or more model elements.

In the example of Figure 1, none of the elements of the model fragment are connected to the others, so there are so number of groups as number of elements in the model fragment. Since the model fragment is composed of the elements that are pointed out, the model fragment contains five elements and the number of groups is equals to five. Therefore, when we compute the dispersion as the ratio between the number of groups and the number of elements, we obtain a dispersion value equals to 1, which means a strong dispersion among the model fragment elements.

4 VALUES IN THE CAF CASE STUDY

This section presents the resulting values after applying MFL over the industrial models of CAF as part of their migration process between their model set to a Model-based Software Product Line. We analyze the result of applying 1.308 MFLs. We provide that results in Figure 3 and Table 1 as reference from real world MFL problems.

Figure 3 depicts a box plot with the values including, for each of the proposed measurements, the maximum, the minimum and the mean of the values. Each column in the plot corresponds to one of the proposed measurements (correspondingly named in the bottom of the plot). The results obtained are as follows:

- The **Size** measurement of the search space is between 196.0 model elements and 356.0 model elements. In other words, the largest search space used through the case study is composed by 356 model elements. By contrast, the smallest search space used is composed by a total of 196 model elements. In addition, most MFLs were performed in search spaces with size between 264 model elements and 309 model elements, being the search space with a size of 287 model elements the most frequent.
- The **Volume** measurement of the search spaces is between 5 models and 21 models. This means that the search spaces where MFL approach were performed are among these values. Most MFLs was performed in search spaces with a volume between 9 models and 14 models, being 11 models the most commonly volume among all search spaces.
- The **Density** measurement of the solutions is between 1% and 6%, being the most common density interval between 2% model elements and 3% model elements. In the case study, the most repeated solution has a density of 3% model elements.
- The **Multiplicity** measurement of the solutions is between 1 time and 21 times, oscillating between 1 time and 8 times the most repeated range for each of the solutions. This mean that, given a solution, the total of the times that the solution can be located is between 1 time and 8 times as maximum. the most frequent multiplicity of our solutions has a value of 5 times.
- The **Dispersion** measurement of the solutions is between 0.8 groups / elements and 1.0 groups / elements. The range is between 0.83 groups / elements and 0.92 groups / elements, meaning that among these values are the total of connected elements. In the case study, the most repeated number of connected model elements for each solution is 0.88 groups / elements.

It is important to emphasize that the MFL is not performed on all existing models of our industrial partner. Domain experts are involved during the MFL approach, contributing with their domain knowledge. A way to contribute with this domain knowledge is by restricting the location approach to those models in which the domain experts estimate that the solution will be found (in the case of the initialization of the Software Product Lines, the features).

Table 1 shows the relationship between the value obtained of a particular measurement and the number of MFLs that use this value. In other words, Tab.1 overviews the top frequently MFLs carried out over the industrial models. Each column in the table identifies one of the proposed measurements in this work, while the columns below these correspond to the most frequent value obtained (named Value in Table 1) regarding to the amount of MFLs (named MFLs in Table 1).

According to the values shown in the table, 272 model elements are the most frequent *Size* of the search spaces of the case study, with a total of 82 MFLs carried out. According to the *Volume*, the most frequent value is 10, meaning that 10 is the amount of models which compose the search space where a model fragment is located.

Regarding to the solution measurements (*Density*, *Multiplicity* and *Dispersion*), 2.67 % is them most common percentage of model elements which realize the solutions, obtained in a total of 34 MFLs. Also, 1 time is the most commonly *Multiplicity* obtained during 608 MFLs, and 0.857 is the most frequent ratio of groups/elements, obtaining that value during 425 MFLs.

4.1 Experimental Setup

The goal of this experiment is to evaluate if MFL is influenced by the presented measurements. In order words, this experiment

Table 1: TOP 10 frequently results during Model Fragment Location over Industrial Models

Search Frequency										
	Size		**Volume**		**Density**		**Multiplicity**		**Dispersion**	
	Value	MFLs	Value	MFLs	Value	MFLs	Value	MFLs	Value	MFLs
1	272	82	10	240	0.0267	34	1	608	0.857	425
2	321	82	9	189	0.0257	29	7	129	0.889	229
3	262	80	11	176	0.028	29	5	128	0.875	200
4	287	73	14	168	0.0249	28	6	111	0.833	162
5	279	48	8	104	0.0218	27	8	87	1.0	114
6	281	44	12	96	0.0297	25	9	74	0.9	60
7	322	43	16	64	0.03	25	10	51	0.8	53
8	292	43	13	52	0.0197	24	11	32	0.909	50
9	233	42	15	45	0.0265	21	13	27	0.923	10
10	332	41	7	42	0.0244	21	12	25	0.917	5

empowers us to determine if the results of MFL depend on the values of measurements or MFL obtains the same results whatever the values of the measurements are. To do this, the experiment addressed the searches in the models of the CAF Case Study by means of two approaches that obtain the best results to recover Traceability between requirements and models [21]. The first one [18] is a Linguistic Rule-Based (Linguistic) approach that is based on Parts-of-Speech (POS) Tagging and Traceability rules. The second one [4] is an Information Retrieval (IR) approach that is based on Latent Semantic Indexing (LSI) and Singular Value Decomposition (SVD). The results were analyzed depending on the measurement:

- The Size is already studied in other researches, in fact, this measurement is used frequently in most MFL works. Therefore, although this measurement is included in our research for the sake of completeness, we did not evaluate the relevance of this measurement in this work.
- The Volume measures the number of models, so to evaluate the impact of this measurement, we took into account searches where the search space is composed of one or more models and only one of them contained the solution. Moreover, the search spaces had to have models with a similar size, the solutions had to have similar density values and dispersion values, and the multiplicity values had to be equals to one, which means the solution is just one in the search space. Therefore, the values for the other measurements (Size, Density, Multiplicity, and Dispersion) are similar or equal for all the searches.
- The Density measures the ratio of model fragment elements to model elements, so to evaluate the impact of this measurement, we took into account searches where the search space was composed of the same model and the solutions has different number of elements so that the density was not the same for all the searches. Given that the search space was the same for all the searches, the size and the volume values were the same for all the searches. Moreover, the solutions had to have similar multiplicity and dispersion values. Therefore, the values for the other measurements (Size, Volume, Multiplicity, and Dispersion) are similar or equal for all the searches.
- The Multiplicity measures the number of times the solution appears in the search space, so to evaluate the impact of this measurement, we took into account searches where each search space contains a different model and the searched solution is the same. Moreover, the search spaces had to have similar volume values and size values, and the solutions had to have similar density values, and similar dispersion values. Therefore, the values for the other measurements (Size, Volume, Density, and Dispersion) are similar or equal for all the searches.
- The Dispersion measures the ratio of connected elements in the solution, so to evaluate the impact of this measurement, we took into account searches where the search space was composed by the same model and the elements of the solutions were scattered in the model to a greater or lesser extent so the dispersion values were not the same for all the searches. Given that the search space was the same for all the searches, the size and the volume values were the same for all the searches. Moreover, the solutions had to have similar density values and multiplicity values. Therefore, the values for the other measurements (Size, Volume, Density, and Multiplicity) are similar or equal for all the searches.

For each search, each approach generates a model fragment as a possible solution. Then, we compare this possible solution against the real solution, which we know beforehand according to the CAF Case Study. Once the comparison is performed, a confusion matrix is calculated. Therefore, we obtain two confusion matrices, one for the Linguistic approach and one for the IR approach.

A confusion matrix is a table that is often used to describe the performance of a classification model (in this case, the approaches) on a set of test data (the resulting model fragments) for which the true values are known (from the CAF Case Study). In our case, each solution that generated by the approaches is a model fragment that is composed of a subset of the model elements that are part of the model (where the solution is being searched). Since the granularity

will be at the level of model elements, the presence or absence of each model element will be considered as a classification. The confusion matrix distinguishes between the predicted values and the real values by classifying them into four categories:

True Positive (TP): values that are predicted as true (in the solution) and are true in the real scenario (the oracle).

False Positive (FP): values that are predicted as true (in the solution) but are false in the real scenario (the oracle).

True Negative (TN): values that are predicted as false (in the solution) and are false in the real scenario (the oracle).

False Negative (FN): values that are predicted as false (in the solution) but are true in the real scenario (the oracle).

Then, some performance metrics are derived from the values in the confusion matrix. Specifically, we will create a report that includes four performance metrics (precision, recall, the F-measure, and MCC) for each of the searches for each approach.

Precision measures the number of elements from the solution that are correct according to the ground truth (the solutions in CAF Case Study) and is defined as follows:

$$Precision = \frac{TP}{TP + FP}$$

Recall measures the number of elements of the solution that are retrieved by the proposed solution and is defined as follows:

$$Recall = \frac{TP}{TP + FN}$$

The F-measure corresponds to the harmonic mean of precision and recall and is defined as follows:

$$F - measure = 2 * \frac{Precision * Recall}{Precision + Recall} = \frac{2 * TP}{2TP + FP + FN}$$

However, none of these previous measures correctly handle negative examples (TN). The **Matthews Correlation Coefficient (MCC)** is a correlation coefficient between the observed and predicted binary classifications that takes into account all the observed values (TP, TN, FP, FN), and is defined as follows:

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

Recall values can range between 0% (which means that no single model element from the solution from the CAF Case Study is present in the model fragment of the obtained solution) to 100% (which means that all the model elements from the CAF Case Study are present in the obtained solution). Precision values can range between 0% (which means that no single model element from the obtained solution is present in the solution from CAF Case Study) to 100% (which means that all the model elements from the obtained solution are present in the solution from CAF Case Study). A value of 100% precision and 100% recall implies that both the obtained solution and the solution from the oracle are the same. MCC values can range between −1 (which means that there is no correlation between the prediction and the solution) to 1 (which means that the prediction is perfect). Moreover, a MCC value of 0 corresponds to a random prediction.

4.2 Results

In Table 2, we outline the results aggregated for each of the approaches. Each row shows the Precision, Recall, F-measure, and MCC obtained through each approach taking into account the selected searches for each new measurement.

The IR approach achieves the best precision results for all the measurements. However, the best results for the other performance indicators depend on what measurement is evaluated. According to the Density, The IR approach achieves the best results for all the performance indicators, providing a mean precision value of 66.22%, a recall value of 48.08%, a combined F-measure value of 52.09%, and a MCC value of 0.53. In contrast, according to the Multiplicity and the Volume, the Linguistic approach achieves the best results for recall, F-measure, and MCC. Moreover, both approaches achieve the best precision results for searches which evaluate the Multiplicity, providing mean precision values up to 68%.

5 STATISTICAL ANALYSIS

To properly compare the different searches, the data resulting from the empirical analysis was analyzed using statistical methods.

5.1 Statistical Significance

A statistical test must be run to assess whether there is enough empirical evidence to claim that the new measurements have an impact on the results so we have to consider their values to search in models. To achieve this, two hypotheses for each new measurement are defined: the null hypothesis H_0, and the alternative hypothesis H_1. The null hypothesis H_0 is typically defined to state that there is no impact on the searches although the measurement have different values, whereas the alternative hypothesis H_1 states that there is an impact on the searches depending on whether the measurement values differ. In such case, a statistical test aims to verify whether the null hypothesis H_0 should be rejected.

The statistical tests provide a probability value, $p - Value$. The $p-Value$ obtains values between 0 and 1. The lower the $p-Value$ of a test, the more likely that the null hypothesis is false. It is accepted by the research community that a $p-Value$ under 0.05 is statistically significant [3], and so the hypothesis H_0 can be considered false.

The test carried out depends on the properties of the data. Since our data does not follow a normal distribution in general, our analysis requires the use of nonparametric techniques. There are several tests for analyzing this kind of data; however, the Quade test is the most powerful when working with real data [8].

Table 3 shows the Quade test statistic and $p - Values$ for recall and precision. The values that are pointed out indicate which are statistically significant, so the hypothesis H_0 can be considered false. At least one of the values for recall or precision for one of the approaches is statistically significant. Consequently, we can state that the new measurements have an impact on the searches, although the degree of this impact is an issue that remain as future work.

Table 2: Mean Values and Standard Deviations for Precision, Recall and F-Measure for the approaches depending of the measurement evaluated

		Precision	Recall	F-Measure	MCC
Volume	Linguistic	26.84 ± 18.78	50.75 ± 17.12	30.46 ± 17.72	0.32 ± 0.16
	IR	34.52 ± 27.29	34.19 ± 20.26	27.28 ± 12.57	0.29 ± 0.13
Density	Linguistic	22.96 ± 22.92	43.90 ± 29.45	25.89 ± 22.25	0.25 ± 0.23
	IR	66.22 ± 27.19	48.08 ± 18.85	52.09 ± 18.46	0.53 ± 0.18
Multiplicity	Linguistic	68.43 ± 21.06	55.56 ± 0.00	60.15 ± 7.96	0.60 ± 0.10
	IR	77.08 ± 25.66	18.06 ± 8.27	27.41 ± 9.30	0.35 ± 0.08
Dispersion	Linguistic	23.50 ± 21.86	48.48 ± 25.96	27.25 ± 20.59	0.27 ± 0.21
	IR	66.41 ± 28.25	43.47 ± 19.57	49.28 ± 19.61	0.51 ± 0.20

Table 3: Quade test statistic and $p - Values$

			Precision	Recall
Volume	Linguistic	p-Value	**0.013**	0.629
		Statistic	**18.00**	0.27
	IR	p-Value	**0.013**	**0.031**
		Statistic	**18.00**	**10.59**
Density	Linguistic	p-Value	0.561	**0.024**
		Statistic	0.62	**0.024**
	IR	p-Value	**0.020**	**0.047**
		Statistic	**6.58**	**4.59**
Multiplicity	Linguistic	p-Value	0.882	0.125
		Statistic	0.02	NaN*
	IR	p-Value	0.769	**0.015**
		Statistic	0.10	**25**
Dispersion	Linguistic	p-Value	0.220	0.700
		Statistic	1.63	0.15
	IR	p-Value	**0.003**	0.451
		Statistic	**12.03**	0.60

*The recall values are equals for all the searches in this case.

6 DISCUSSION

Results highlight that the proposed measurements have an impact on the outcomes of the studied approaches. Through this work, we have identified a series of facts that serve as a starting point for discussing why the proposed measurements and the provided values are significant for the research community. Through the following paragraphs, said facts are discussed:

1 From the results, it is possible to conclude that size reports do not provide enough information on the search problem. This is due to the inability of the size measurement to accurately represent the inherent challenge levels of models.

This issue is better illustrated through an example taken from the case study. In the example, there are two models of sizes 37 and 113 elements (respectively) and a feature that is present in both models. At first glance, one may expect it to be easier to find the feature in the smaller model. However, in the first model, the feature is implemented through a model fragment containing only two elements, and all the elements in the model contain similar texts and word patterns. In the second model, the feature is implemented through 28 elements, which are clearly differentiated from the rest of the elements in the model in terms of text and word patterns. In the depicted scenario, both approaches are able to find the second model fragment with a much greater accuracy than the first model fragment, rendering the size of the model insufficient to depict the search problem.

2 Search problems in models are relatively new when compared to search problems in other kinds of documents such as the web or source code. Hence, search problems in models have inherited measurements accepted by the community in similar fields. However, the novelty of applying searches to models is the models, and it is not possible to directly apply measurements to models from webs and code. In that sense, our work puts into perspective that such novel search problem requires more attention on how to report it in a correct manner so performance results can be properly evaluated.

3 Among the research community, in the face of the absence of real work datasets, synthetic datasets are very common

and popular. These synthetic datasets are useful to test extreme scenarios and situations. Since our findings show that the proposed measurements are significant and impactful with regard to the performance of search approaches, we provide their real world values so designers can carry out a real world search problem profiling process when designing synthetic test cases and extreme search problem scenarios. Therefore, the reference values for the measurements that have been presented through this paper can be useful for other researchers.

4 Through the results, it has become clear that the defined measurements have an impact on the results of the approaches. However, it may be possible to define new measurements or derive other measurements from the presented ones. Moreover, it may be possible to try and identify patterns in the results of the approaches based on the values of the measurements. Finally, we have not studied how different approaches are affected by each measurement, or in other words, some approaches may be more sensitive to the values of certain measurements while not being affected by other measurements. All these possibilities for further exploration of the measurements and their impact remain as future work.

To sum up, the results of our work are promising, and along with the facts identified and exposed through the prior paragraphs, open the door for the study of the particularities of reporting the location problem of MFL.

7 RELATED WORK

We focus our research on Model Fragment Location (MFL), so the most important knowledge areas where our research can be applied are: Bug Location, Feature Location, Traceability Links Recovery. For this reason, through this section, we analyze some of the existing works in these areas, and compare our work with them.

Most of the existing works focus on searches in source code, therefore their measurements are oriented to measure the number of code lines or the number of methods in the code. For instance, [22] present a systematic literature survey of bug location over source code. In [5, 17] the authors present systematic literature surveys of feature location techniques over source code. Javed et al. [11] present a systematic literature review to discover the existing traceability approaches and tools between software architecture and source code. By contrast to them, our work focuses on models instead of source code, so the measurements that are presented in this work are oriented to measure models (search space) and model fragments (solution space).

7.1 Related works on MFL over models

Some recent works center their efforts on MFL on models. Winkler et al. [21] classify several approaches that have been created during the past 15 years which try to optimize automatic identification of traces over models. De Lucia et al. [4] present a Traceability Links Recovery method and tool based on Latent Semantic Indexing (LSI) which includes models. Spanoudakis et al. [18] present a linguistic rule-based approach to support the automatic generation of Traceability Links between requirements and models. In [10, 15, 20, 23, 24] the authors focus on the location of features over models by comparing the models with each other to formalize the variability among them in the form of a Software Product Line. For instance, in [10] the authors present an improved version of a family mining approach for automatically discovering commonality and variability between related system variants. They apply their approach to function block diagrams used to develop automation software and show its feasibility by a manufacturing case study.

Wille et al. [20] present an approach to analyze related models and determine the variability among them. Their analysis provides crucial information about the variability (changed parts, additional parts, and parts without any modification) between the models in order to create family models. In [23] an approach for synthesizing a software product line using model comparison is presented. During model difference detection, the presented approach applies EMF Compare, a generic model comparison tool. For specifying the variability, the approach applies the Common Variability Language (CVL) [9], a generic language for expressing variability. Based on the comparison results, a preliminary product line model (CVL model) can be automatically induced and the SPL developer may further enhance the product line model. Just like us, the authors applied their research in the railway domain to illustrate their work.

Zhang et al. [24] present an approach for automating the augmentation of product lines using model comparison and variability modeling techniques. Their approach aims to reduce manual effort involved in this process by automatically suggesting a tentative augmented product line model. The approach applies CVL Compare, a generic approach for automating the synthesis of a CVL-based product line from a set of existing product models. Martinez et al. [15] introduced a generic and extensible framework for bottom-up approach to Software Product Line Engineering. They presented the approach' principles with the objective to reduce the current high up-front investment required for a systematic reuse end-to-end adoption.

All these works (see top of Table 4) are evaluated by means of different case studies which are measured to a greater or lesser extent. The most popular measurement reported is the size. However, none of them take into account the same measurements to work with models and the measurements are selected by the researchers according to their own judgment. In contrast, we propose a set of measurements which are strongly related with models to measure the search space and the solution space.

7.2 Our previous related works on MFL over models

Finally, some of our previous works [7], [6], [1], [14], [13] present Feature Location approaches to discover software artifacts that implement the feature in models.

Marcen et al. [14] propose a feature location approach to discover model elements that implement the feature functionality in a model. Through a model and a feature description, model fragments extracted from the model and the feature description are encoded based on a domain ontology. Then, a Learning to Rank algorithm is used to train a classifier that is based on the model fragments and

Table 4: Overview of the related works regarding with their searches and the five presented measurement: Size (S), Volume (V), Density(DE), Multiplicity (M), and Dispersion (DI)

	Related Works	Searches in Models	Measurements				
			Search Space		Solution		
			S	V	DE	M	DI
Other works	[22]	X	-	-	-	-	-
	[5]	X	-	-	-	-	-
	[17]	X	-	-	-	-	-
	[11]	X	-	-	-	-	-
	[21]	√	X	X	X	X	X
	[4]	√	X	X	X	X	X
	[18]	√	√	X	X	X	X
	[20]	√	√	X	X	X	X
	[10]	√	X	X	X	X	X
	[23]	√	√	X	X	X	X
	[24]	√	X	X	X	X	X
	[15]	√	X	X	X	X	X
Our previous works	[7]	√	√	√	X	X	X
	[6]	√	√	√	X	X	X
	[1]	√	X	X	X	X	X
	[14]	√	√	√	X	X	X
	[13]	√	√	√	X	X	X

feature description encoded. Lapeña et al. [13] presented Computer Assisted Clone-and-Own form Models (CACAO4M), an approach to rank relevant model fragments for the development of particular requirements for a new product. Through their approach, the authors aim to prioritize the model fragments that are easier to understand from the perspective of a software engineer.

Arcega et al. [1] propose an approach that combines architecture models at run-time and information retrieval for feature location. Specifically, their approach uses a scenario that executes the desired feature to be located. Also, the approach ranks all of the model elements that are executed to extract the model elements that are related to the feature. Font et al. [6] presented a Genetic Algorithm to Feature Location. They provide a custom encoding that enable the genetic algorithm to work with model fragments and a set of genetic operations that can be applied to individuals following that encoding. In addition, they present a fitness function, a parent selection operator, a crossover operation and a mutation operation.

Font et al. [7] propose and compare five search algorithms to locate features over a family of models: Evolutionary Algorithm (EA-MFL), Random Search (RS-MFL) used as a sanity check, steepest Hill Climbing (HC-MFL), Iterated Local Search with random restarts (ILS-MFL), and a hybrid between Evolutionary and Hill Climbing (EHC-MFL). They applied Latent Semantic Analysis (LSA) as the fitness function. Their results show that Search-based Software Engineering techniques can be applied to locate features in product models. They evaluate their work over two families of industrial models, demonstrating that Search-based Software Engineering for feature location at the model level can be applied in real world environments.

Most of these works are evaluated by means of case studies which take into account the size of the models or the volume of models (see bottom of Table 4). However, none of the case studies take into account all the measurements which are presented in this paper: size, volume, density, multiplicity, and dispersion.

8 CONCLUSIONS

Traceability Links Recovery, Feature Location and Bug Location are popular activities in the context of software maintenance. When the artifacts where the requirements, features, or bugs are located are models, approaches focus on identifying relevant sets of model elements (Model Fragments). These Model Fragment Location (MFL) approaches leverage Information Retrieval, Linguistic, and Search-Based Software Engineering techniques to locate the model fragments.

However, there is a lack of detail in the reporting of the measurements of both the search space and the solution, with model size being the only reported measure. Since different models are measured in different ways, model size values are not a valid comparison.

Through this paper, we proposed the usage of five measurements to report the results of MFL techniques. Apart from the size measurement, we introduced four novel measurements: volume, density, multiplicity, and dispersion. From the five measurements, size and volume measure the search space, while density, multiplicity, and dispersion measure the solution.

In order to determine the relevance of the proposed measurements, we studied whether the values of the measurements have an impact on the results provided by two distinct MFL approaches. To that extent, we evaluated said approaches in terms of precision and recall, and analyzed their outcomes with regard to the measurements of the case study.

The results presented in the paper show that all the proposed measurements have a direct impact on the results of the MFL approaches, so we strongly recommend their study and reporting. Furthermore, we also provide real measures of location problems during MFL over industrial models. These values can be a reference for researchers who create synthetic location problems.

ACKNOWLEDGMENTS

This work has been partially supported by the Ministry of Economy and Competitiveness (MINECO) through the Spanish National R+D+i Plan and ERDF funds under the project Model-Driven Variability Extraction for Software Product Line Adoption (TIN2015-64397-R).

REFERENCES

[1] Lorena Arcega, Jaime Font, Øystein Haugen, and Carlos Cetina. 2016. Feature Location through the Combination of Run-Time Architecture Models and Information Retrieval. In *International Conference on System Analysis and Modeling*. Springer, 180–195.

[2] Lorena Arcega, Jaime Font, Øystein Haugen, and Carlos Cetina. 2017. On the Influence of Models at Run-Time Traces in Dynamic Feature Location. In *Modelling Foundations and Applications*, Anthony Anjorin and Huáscar Espinoza (Eds.). Springer International Publishing, Cham, 90–105.

[3] Andrea Arcuri and Lionel Briand. 2014. A Hitchhiker's Guide to Statistical Tests for Assessing Randomized Algorithms in Software Engineering. *Software Testing, Verification and Reliability* 24, 3 (2014), 219–250.

[4] Andrea De Lucia, Fausto Fasano, Rocco Oliveto, and Genoveffa Tortora. 2004. Enhancing an Artefact Management System with Traceability Recovery Features. In *Proceedings of the 20th IEEE International Conference on Software Maintenance*. IEEE, 306–315.

[5] Bogdan Dit, Meghan Revelle, Malcom Gethers, and Denys Poshyvanyk. 2013. Feature Location in Source Code: a Taxonomy and Survey. *Journal of software: Evolution and Process* 25, 1 (2013), 53–95.

[6] Jaime Font, Lorena Arcega, Øystein Haugen, and Carlos Cetina. 2016. Feature Location in Model-Based Software Product Lines Through a Genetic Algorithm. In *International Conference on Software Reuse*. Springer, 39–54.

[7] J. Font, L. Arcega, ÃŸ. Haugen, and C. Cetina. 2017. Achieving Feature Location in Families of Models through the use of Search-Based Software Engineering. *IEEE Transactions on Evolutionary Computation* (2017), 1–1. https://doi.org/10.1109/TEVC.2017.2751100

[8] Salvador García, Alberto Fernández, Julián Luengo, and Francisco Herrera. 2010. Advanced Nonparametric Tests for Multiple Comparisons in the Design of Experiments in Computational Intelligence and Data Mining: Experimental Analysis of Power. *Information Sciences* 180, 10 (2010), 2044–2064.

[9] ÃŸystein Haugen, Andrzej Wasowski, and Krzysztof Czarnecki. 2013. CVL: Common Variability Language. 2 (08 2013).

[10] Sönke Holthusen, David Wille, Christoph Legat, Simon Beddig, Ina Schaefer, and Birgit Vogel-Heuser. 2014. Family Model Mining for Function Block Diagrams in Automation Software. In *18th International Software Product Lines Conference*.

[11] Muhammad Atif Javed and Uwe Zdun. 2014. A systematic literature review of traceability approaches between software architecture and source code. In *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*. ACM, 16.

[12] Kurt Koffka. 2013. *Principles of Gestalt Psychology*. Vol. 44. Routledge.

[13] Raúl Lapeña, Jaime Font, Carlos Cetina, and Óscar Pastor. 2017. Model Fragment Reuse Driven by Requirements. In *Proceedings of the Forum and Doctoral Consortium Papers Presented at the 29th International Conference on Advanced Information Systems Engineering, CAiSE*. 12–16.

[14] Ana C. Marcén, Jaime Font, Óscar Pastor, and Carlos Cetina. 2017. Towards Feature Location in Models Through a Learning to Rank Approach. In *Proceedings of the 21st International Systems and Software Product Line Conference - Volume B (SPLC '17)*. ACM, New York, NY, USA, 57–64. https://doi.org/10.1145/3109729.3109734

[15] Jabier Martinez, Tewfik Ziadi, Tegawendé F. Bissyandé, Jacques Klein, and Yves Le Traon. 2015. Bottom-up Adoption of Software Product Lines: a Generic and Extensible Approach. In *Proceedings of the 19th International Conference on Software Product Lines*.

[16] J. Martinez, T. Ziadi, T. F. BissyandÃĺ, J. Klein, and Y. l. Traon. 2015. Automating the Extraction of Model-Based Software Product Lines from Model Variants (T). In *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. 396–406. https://doi.org/10.1109/ASE.2015.44

[17] Julia Rubin and Marsha Chechik. 2013. A Survey of Feature Location Techniques. In *Domain Engineering*. Springer, 29–58.

[18] George Spanoudakis, Andrea Zisman, Elena Pérez-Minana, and Paul Krause. 2004. Rule-Based Generation of Requirements Traceability Relations. *Journal of Systems and Software* 72, 2 (2004), 105–127.

[19] Harald Störrle. 2014. On the Impact of Layout Quality to Understanding UML Diagrams: Size Matters. *MODELS* (2014).

[20] David Wille, Sönke Holthusen, Sandro Schulze, and Ina Schaefer. 2013. Interface Variability in Family Model Mining. In *17th International Software Product Line Conference*.

[21] Stefan Winkler and Jens Pilgrim. 2010. A Survey of Traceability in Requirements Engineering and Model-Driven Development. *Software and Systems Modeling (SoSyM)* 9, 4 (2010), 529–565.

[22] W Eric Wong, Ruizhi Gao, Yihao Li, Rui Abreu, and Franz Wotawa. 2016. A Survey on Software Fault Localization. *IEEE Transactions on Software Engineering* 42, 8 (2016), 707–740.

[23] Xiaorui Zhang, Øystein Haugen, and Birger Møller-Pedersen. 2011. Model Comparison to Synthesize a Model-Driven Software Product Line. In *Proceedings of the 15th International Conference on Software Product Lines*.

[24] Xiaorui Zhang, Øystein Haugen, and Birger Møller-Pedersen. 2012. Augmenting Product Lines. In *19th Asia-Pacific Software Engineering Conference*.