# The Effect of Feature Characteristics on the Performance of Feature Location Techniques

Abdul Razzaq, Anthony Ventresque, Rainer Koschke, Andrea De Lucia, and Jim Buckley

**Abstract**—*Feature Location (FL)* is a core software maintenance activity that aims to locate observable functionalities in the source code. Given its key role in software change, a vast array of Feature Location Techniques (FLTs) have been proposed but, as more and more FLTs are introduced, the *selection of an appropriate FLT* is an increasingly difficult problem. One consideration is the *characteristics of the features* being sought. For example, in the code associated with the feature, programmers may have named identifiers consistently, and with meaningful naming conventions, or not, and this may impact on the suitability of different FLTs. The suggestion that such characteristics matter has implicit support in the literature: An analysis of existing FLT empirical studies reveals that the system under study can often have a stronger impact on FLT performance than differing FLTs themselves. To understand this interaction between feature characteristics and FLTs better, this paper proposes *a suite of feature-characteristic metrics* that are postulated to control FLTs' performance, holistically across FLTs and impacting on individual FLTs to different degrees. To evaluate the suite, a controlled experiment is performed, using 878 features, to probe the relationship between the metrics and the performance of four FTL techniques: three commonly-used techniques and one state-of-the-art technique. The evaluation is performed using four commonly used evaluation measures and extended by employing 41 other established source-code metrics as extraneous variables. Results of the empirical evaluation suggest that the feature-metric suite presented impacts FLT performance holistically, and impacts different FLTs to different degrees. Thus, this paper moves towards the more standard selection of appropriate FLTs, with respect to the prominent feature characteristics in the software systems under study, and more rigorous consideration of the features selected to compare FLTs.

**Index Terms**—Feature Location, Bug Localization, Software Maintenance, Software Recommendation, Software Characteristics.

## 1 Introduction

Many software maintenance tasks mandate the location of the code that implements some user functionality, for example debugging, or evolving code [1], [2]. Locating the observable functionalities in source code is referred to as Feature location (FL) [3], [4] and because it has such a key role in the software change process [5], a wide range of Feature Location Techniques (FLTs) have been proposed [5]. These predominantly rely on source code structural analysis, textual analysis and (increasingly) hybrid techniques [3], [5].

It is difficult to determine relative FLTs' performance [4], [5], towards the identification of the best FLT. For example, some studies suggest that the Vector Space Model (VSM) outperforms other techniques [6], while others claim that Latent Semantic Indexing (LSA) outperforms VSM [7]. Others still, claim that Latent Dirichlet Allocation (LDA) outperforms both LSA and VSM [8]. This suggests that other, uncharted factors, impacting FLTs performance, are at play in these studies.

One possibility is that, because different systems were under study in these evaluations, the variability might be in part due to the differing characteristics of the features those systems contain. For example, one system may have features that are highly localized in the code whereas, in another, they may be highly delocalized. The belief that such feature characteristics might impact FLT performance is further strengthened by the results of empirical studies where multiple FLTs are compared using more than one system and thus, more than one set of features (where a 'feature set' is considered the set of features available for each system). In these studies [6], [8], [9], [10], [11], [12], [13], the impact of testing the techniques against different sets of features, from different systems, often seems to dominate the difference of applying differing FLTs. This contrasts with the implicit position in FLT evaluation literature that the performance of an FLTs is a function of the FLTs themselves [4], [9], [10], [11], [14], and/or those FLTs' configurations (i.e. the parameter settings employed by an FLT) [6], [12], [13], [15], [16]. However, if true, it suggests that careful consideration should be given to the sets of features used as test cases in the empirical evaluation of FLTs.

The different feature characteristics, in (different) feature sets, can be said to control the performance of FLTs if there is strong evidence of causality over FLTs' performance [17] and/or they impact different FLTs' performance to different degrees [18], [19]. To evaluate this thesis, we are guided by software metric research where the role of code characteristics, measured via software metrics, has been been used to indicate software engineering quality [20], [21]. Analogously, this paper proposes a feature metric suite that characterizes features towards assessing those characteristics as indicators of *overall FLTs' performance* and *performance differences between pairs of FLTs*.

In this paper several feature characteristics, captured by metrics, are proposed, and how they are postulated to impact FLT performance, is described. Then, to empirically evaluate if the suite of proposed metrics has an impact on the FLTs, a controlled experiment is performed, where different sets of features are characterized using the met-

• *A. Razzaq is part of the Lero-Irish Software Research Centre. E-mail: abdul.razzaq@lero.ie*

rics proposed, and the impact of those differing feature characteristics on FLTs is identified. The contribution of this research is that it moves towards explaining FLTs' performance in relation to feature characteristics. This will ultimately allow a more standard selection of systems and feature sets for FLT evaluation, cognisant of their impact on FLT performance. It may also inform the design of recommender systems, facilitating selection of FLTs, given knowledge of the feature set being sought.

A replication package, which includes the intermediate results, scripts to obtain those results, and implementations of FLTs and feature sets, is available through the web-resource: https://www.lero.ie/research/datasets/feature_location/featuremetrics. In addition, we have provided a website http://metriccalculator.lero.ie/FeatureMetricCalculator/ to allow researchers to submit a feature set for characterization with respect to our metrics.

The remaining sections of the paper are as follows: Section 2 presents FL in overview and describes the empirical basis of FLT evaluations. Section 3 presents a proposed feature-metric suite, relates it to possible performance aspects of FLTs, and presents an associated theoretical framework, to characterize the potentially moderating impact of features' characteristics. Section 4 presents the research questions and details the empirical designs used to answer each question: The associated results are presented in Section 5. Section 6 discusses the results, presents practice-guidance informed by the results, and enumerates the lessons learned from the findings. Threats to validity are discussed in Section 7. Finally, conclusions and future work are presented in Section 8.

```
1.  float calculateAverage(){
2.    return calculateSum()/stack-size
3.  }
4.
5.  int findMaximum(){
6.   max=0
7.    while(extractElement()<>NULL)
8.        if (element > max)
9.            max<-element
10.   return max
11.   }
12.
13.  int calculateSum(){
14.    sum=0
15.     while(extractElement()<>NULL)
16.         sum<-sum + element
17.   return sum
18.  }
19.
20.  int extractElement(){
21.    if (current_location == -1)
22.        return NULL
23.     return stack[current_location--]
24.    }
```

Fig. 1. Features in a Stack Data-structure

## 2 Background and Related Work

A feature is an observable system functionality that can be triggered by users [4]. Take for example, the pseudo code presented in Figure 1. Here four methods are implemented using a stack data-structure: 1) *calculateAverage()*, 2) *findMaximum()*, 3) *calculateSum()*, and 4) *extractElement()*: In this case, it is reasonable to assume that three features may be available to the user: calculating the average of a set of numbers (on the stack), calculating the total of those numbers and finding the largest number in the set.

When exercising the 'calculating average' feature, the user invokes the *calculateAverage()*, *calculateSum()*, and *extractElement()* methods. Similarly, when exercising the 'calculating sum' feature the user invokes the *calculateSum()*, and *extractElement()* methods. These examples illustrate that a feature could map to multiple methods and, in larger systems multiple parts of multiple methods. Likewise a method, or part of a method can map to multiple features. Ultimately this means that a feature can be scattered throughout the code of a software system.

The task of feature location (FL) then is to locate source code elements related to a specified feature [5]. This activity is intrinsically associated with software maintenance and evolution, which frequently mandate the location of a feature's code to document, configure, add, remove, debug or improve on functionalities [4]. In the case of the 'calculating average' feature, the location of the feature is lines 1-3, lines 13-18 and lines 20-24. The task of a Feature Location Techniques (FLT) is to locate the code related to specified features on demand. As the 'calculating average' feature suggests, the location of a feature can be delocalized and so, in large code bases, can be very difficult to determine.

### 2.1 Classification of FLTs

FLTs can be classified by two distinguishing factors. Analysis type is the most commonly used factor, classifying the FLTs into textual, structural, historical and dynamic types [5]. Textual analysis presumes that the comments and identifier-names in the source code encode feature-related domain knowledge and can be exploited using textual searches [12], [15], [22]. Structural analysis allows developers to identify the extent of the features [10] by following data or control-flow dependencies, typically from a known feature-related source code element [23], [24]. In historical analysis, feature-related code elements are identified by leveraging re-enactment artifacts [11], [25], [26]. Here relationships like co-change and change descriptions are leveraged to help in FL. Finally, in dynamic analysis, the feature is exercised and traced at execution time [1]. More recently, to compensate for the limitations of individual analysis types [27], [28], [29], FLTs have moved towards refining those approaches or leveraging a combination of the approaches [10], [11], [14], [22], [23], [25], [26], [27], [30], [31]. These latter FLTs, based on combinations of existing techniques, are referred to as *hybrids* [14], [25].

The second important distinguishing factor, even though seldom explicitly described in the literature, is the goal of the FLTs: Razzaq et al.'s systematic literature review of the

field classifies FLTs as either being directed at obtaining an initial feature-foothold location in the code, or near-full feature location [3]. When a developer is interested in locating a single code element entry-point into the feature, possibly to begin impact analysis or debugging, then the goal of the FLT can be considered *foothold location*. Conversely, if developers are interested in identifying the full extent of the feature (for example, to identify feature variants between different versions of a system with a view to recovering its product line [28]), then the goal of FLT is *near-full feature location*.

### 2.2 FLTs' Evaluation Components

Usually, FLTs are empirically evaluated by case studies or experiments [3], [5]. To characterize the empirical design of such studies, Dit et al. [5] proposed three essential components: the software system (under investigation), evaluation metrics, and user-input. As FLTs act on user inputs to determine the feature-relevant source code in a software system, Dit et al. [5]'s listing of empirical components can be expanded to include the source code locations of the features correctly corresponding to the user's input. The correct code locations for a set of features is referred to as the gold set [11]. In the case of the stack example in Figure 1, for example, *calculateSum()*, *culculateAverage()* and *extractElement()* are the gold set methods related to the 'calculating average' feature.

Practically, gold sets are often obtained for the evaluation of FLTs using re-enactment, where textual descriptions of historically-made changes are considered as a proxy for user input/descriptions, and the locations of the resultant code changes are considered as a proxy for the features' locations. Cumulatively, the set of user inputs describing the features and the associated code locations for those features are known as the feature set in this study.

### 2.3 FLT Evaluation and Feature Characteristics

Here we define a *feature characteristic* as a measurable attribute of the code elements or user input associated with a feature. For example, one feature characteristic may be the number of code elements that make up a feature. In the case of the *calculate sum* and *calculate average* features shown in 1, the number-of-code-elements may be measured by their number of constituent methods (2 and 3 respectively) or by their lines of code (11 and 14 respectively).

Razzaq et al. [3], in their systematic review of FLT empirical studies, noted that these studies do not consider the characteristics of the features in their work: instead, they noted a concentration on studies that evaluate single FLTs [7], [24], [27], studies that compare FLTs (for example [6], [8], [9], [10], [11], [14]), and studies that have looked at FLT configurations. For example, several studies have focused on whether to use code identifiers, comments, literals or combinations of these, to represent source code during IR-based FL [12], [13], [29]. Another branch of research has looked at whether to use the summary, description or both (as obtained from bug-reports or feature requests) as the query for the search [6], [12], [15], [16]. In general, empirical evaluations in this field have neglected feature characteristics as a line of consideration.

#### 2.3.1 The Potential Impact of Feature Characteristics

This study aims to assess if causality, defined as the relationship between FLTs and FLT performance, may be partly, or significantly, accounted for by feature characteristics. The literature basis for such a study is illustrated in Table 1. It presents a listing of FLT empirical studies that compare more than one FLT with more than one feature set, these studies being identified during the literature reviews reported on in [3] and [32]. The second-last column details the biggest performance difference between any of the different FLTs evaluated when compared to the same feature set in each study. The last column details the biggest performance difference between the same FLTs on different feature sets in each study. The table is ordered by study and then by the difference between these last two columns.

What is striking here is that the feature sets seem to have more effect than the differing FLTs in most (three-quarters) cases. Since the characteristics of each feature are intrinsic to the overall effect of the feature set, Table 1 implies that feature characteristics often explain FLT performance more than the different FLTs studied. In addition to larger differences, statistics of central tendency measures (e.g. mean, median) show similar results, as evident from the studies that provided their statistics [8], [23], [25], [33], [34], [36].

This pattern persists even when many FLTs are compared in a study. For example, Binkley et al. [8] compared seven FLTs, and while they did differ significantly in their performance, individual FLT's performance on the different feature sets employed differed by the same order. Likewise, Thomas et al. [6] employed 3172 different configurations for three FLTs. Even though it was again shown that different configurations have a significant impact on FLTs' performance, the largest difference (within configuration) for different feature sets was again of comparable scale. These findings, mirrored across the result sets of these empirical studies, suggest the strong effect of characteristics of the features employed in FLTs evaluations. Hence, the research presented here argues that it is important to help researchers to select appropriate feature sets which provide coverage over the relevant feature characteristics that impact performance [5] by assessing FLTs against specific feature set characteristics. This research is also important for practitioners who, given a specific characteristic profile for their system's features (as determined by previous software evolution activity perhaps), can then target appropriate FLTs. This research moves towards these goals by probing FLT performance, cognisant of feature set characteristics.

## 3 Feature Metric Suite

Metrics [21] are useful, in the context of this study, to quantitatively reflect feature characteristics. For a metric to be selected in this research, it should be well-formed and feature-specific. As such, we propose the following properties:

1) *Constancy across FLTs* – The metrics should be exclusively defined by the feature, independent of the FLTs applied. This criterion will allow the impartial selection of the metrics for the FLT under investigation. Specifically, this study sets out to assess mod-

TABLE 1
Performance Difference between FLTs and Benchmarks in Comparison Studies

| Study | Evaluation Measure | Number of FLTs Compared | Number of feature sets Employed | FLTs' Impact** | Feature Set's Impact*** |
|---|---|---|---|---|---|
| **Razzaq et al. [32]** | Recall | 8 | 12 | 0.354 | **0.796** |
| | F-Measure | 8 | 12 | 0.136 | **0.411** |
| | $MRR^1$ | 8 | 12 | 0.734 | **0.999** |
| | Precision | 8 | 12 | 0.085 | **0.284** |
| | $MAP^2$ | 8 | 12 | 0.233 | **0.266** |
| **Lee et al. [33]** | MAP | 6 | 50 | 0.67 | **0.96** |
| | MRR | 6 | 50 | 0.67 | **0.95** |
| **Li et al. [23]** | Precision | 3 | 16 | 0.38 | **0.62** |
| | Recall | 3 | 16 | 0.36 | **0.4** |
| **Gethers et al. [27]** | Average Precision | 9 | 6 | 0.11 | **0.32** |
| **Wang et al. [25]** | MAP | 5 | 4 | 0.17 | **0.33** |
| | MRR | 5 | 4 | 0.18 | **0.26** |
| **Kim et al. [26]** | Average Rank | 4 | 8 | 2.51 | **3.9** |
| **Shi et al. [34]** | MRR | 5 | 4 | 0.29 | **0.39** |
| **Binkley et al. [8]** | MRR | 7 | 5 | 0.3 | **0.33** |
| **Corley et al. [35]*** | MRR | 4 | 6 | 0.036 | **0.048** |
| **Corley et al. [36]*** | MRR | 3 | 6 | 0.06 | **0.068** |
| **Dit et al. [14]** | Mean Effectiveness | 7 | 3 | 19320 | **19326** |
| **Thomas et al. [6]** | Top-20 | 4 | 3 | **0.353** | 0.337 |
| **Ye et al. [11]** | MAP | 4 | 6 | **0.41** | 0.34 |
| | MRR | 4 | 6 | **0.45** | 0.35 |
| **Mahmoud et al. [37]*** | MAP | 9 | 3 | **0.3** | 0.28 |
| **Mahmoud et al. [38]** | MAP | 5 | 3 | **0.4** | 0.21 |
| | Lag | 5 | 3 | **50** | 30 |

* Multiple analyses have been presented in these studies
** Largest Performance Difference between FLTs on a feature set
*** Largest Performance Difference between feature sets when using the same FLT
[1] Mean Reciprocal Rank, [2] Mean Average Precision

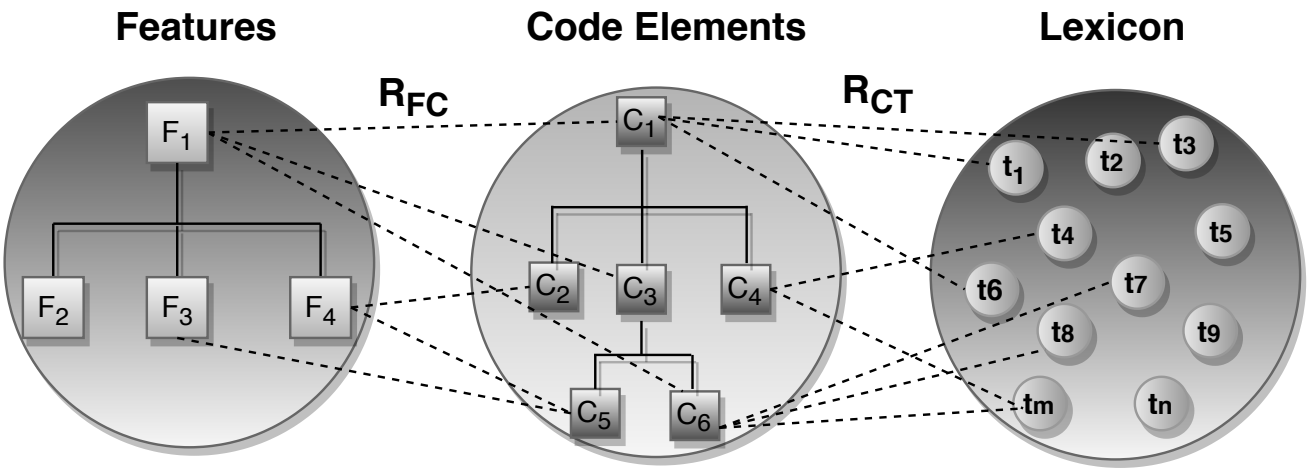


Fig. 2. Direct Relation of Features to Code Elements and Indirect Relation of Features to System' Lexicon

TABLE 2
Feature Metrics

| Type of Characterist. | Feature Metric | Description |
|---|---|---|
| **Gold-set** | Relative Feature Size | Number of source code elements which constitute a feature, relative to the total number of elements in the software system |
| | Tangled Elements | The number of code elements shared by a feature with at least one other feature |
| | Crosscutting in Feature | For a feature, the number of other features that intersect that feature by at least one code element |
| | Unique Lexical Coverage | Unique lexical size of the feature, relative to the unique lexical size of the system |
| | Lexical Saturation | Lexical size of the feature, relative to lexical size of the system when repetitions of words allowed |
| **User Input** | Query (Unique) Size | Number of unique terms that are used to constitute the feature-query in the feature set |

eration impact, and a moderator has to be constant across independent variables by definition [18].

2) *Focused on a specific feature* – A metric's value should be exclusively defined by the associated feature. This criterion will disassociate the characteristics of one feature from others and is required to keep the measurement of metric values at feature-level granularity.
3) *Compliance with validation properties* – The feature metrics must comply with the five mathematical measurement properties proposed by Briand et al. [39]: non-negativity, null value, monotonicity, merging of modules, and disjoint module additivity.

Note that the values of all the metrics are calculated after employing the best empirical practice for configurations identified by several case studies [6], [12], [13], [15], [16] (best practice will be presented in Section 4.1.2).

### 3.1 Feature Metrics

A feature set belonging to a system under investigation can be represented as a tuple $\widehat{\mathcal{F}} = (\mathcal{C}, \mathcal{L}, \mathcal{CS}, \mathcal{R}_{\mathcal{FC}}, \mathcal{R}_{\mathcal{CT}})$. $\mathcal{C}$ is the set of code elements that implement $\widehat{\mathcal{F}}$. These code elements are coded using a lexicon $\mathcal{L}$. Here $\mathcal{CS}$ is the set of structural relationships between code elements. For example, a *number of children* relationship between a parent and children classes is a structural relation. Finally, $\mathcal{R}_{\mathcal{FC}} = ((f, c) \mid f \in \widehat{\mathcal{F}}, c \in \mathcal{C})$ is the mapping between specific features and their code elements and $\mathcal{R}_{\mathcal{CT}} = ((c, t) \mid c \in \mathcal{C}, t \in \mathcal{L})$ is the mapping between code elements and their lexicons. This is illustrated in Figure 2. For a feature, say $\mathcal{F}_1$, the task of the FLT is to find all elements $\mathcal{C}_{\mathcal{F}_1} \in \mathcal{C}$ which implement $\mathcal{F}_1$ through the relation $R_{fc}(\in \mathcal{R}_{\mathcal{FC}})$ to that feature. In this instance, the code elements are $C_1$, $C_3$, and $C_6$. In the context of textual FLTs, lexicons $t_1$, $t_3$, $t_6$, $t_7$, $t_8$, and $t_m$ can be leveraged in this identification task through the $\mathcal{R}_{\mathcal{CT}}$ relations. Table 2 presents a summary of the feature metrics used in this research on FLTs, and the remaining portion of this section provides a more detailed description of these metrics.

***Metric 1: Relative Feature Size***

*Relative Feature Size* ($\mathcal{RFS}$) is the number of source code elements that constitute a feature relative to the total number of elements in the software system under study. This can be represented as follows:

$$\mathcal{RFS}(\mathcal{F}) = \frac{|\mathcal{C}_{\mathcal{F}}|}{|\mathcal{C}|}$$

where $\mathcal{C}_{\mathcal{F}}$ is the set of code elements that implement the feature $\mathcal{F}$. In Figure 2, $F_1$ is made up of three (i.e. $C_1$, $C_3$, and $C_6$) from six code elements, hence its $\mathcal{RFS}$ value is 3/6.

*Rationale for inclusion* – FLTs attempt to assign a score to code elements according to their relevancy to the feature-query [12], [14], [15], [16]. Code elements are then placed in a list ranked according to their relevance scores. During the assessment of FLTs, this ranked-list is used to measure the FLT performance, in line with some evaluation measures.

A feature with large $\mathcal{RFS}$ would have a larger extent in relation to the system, an attribute that would facilitate FLTs in placing at-least some of the feature related code elements towards the top in the ranked-list. However, it is likely that such a large feature is more structurally delocalized in the code, leading to more effort for FLTs attempting to find the full feature extent: this may result in the FLT missing some feature-related code elements towards the top of the list.

Accordingly, it seems intuitive that, if the goal of the feature location is feature-foothold, a large feature footprint might help but that it may hinder locating the full feature extent. That is: $\mathcal{RFS}$ may have a moderation effect on measures of FLTs' performance depending on the FLT goal.

***Metric 2: Tangled Elements***

*Tangled Elements* ($\mathcal{TE}$) count the total number of code elements in a feature, shared by at least one other feature. For example, for a feature $\mathcal{F}$, $\mathcal{TE}$ can be defined as follows:

$$\mathcal{TE}(\mathcal{F}) = |\{c \in \mathcal{C}_{\mathcal{F}} : foreach_{(\mathcal{F}' \in \widehat{F})} \mathcal{F}' \neq \mathcal{F} \Longrightarrow c \in \mathcal{C}_{\mathcal{F}'}\}|$$

Where $\mathcal{C}_{\mathcal{F}}$ is the set of code elements related to $\mathcal{F}$ and $\widehat{\mathcal{F}}$ is the complete set of features. As an example of $\mathcal{TE}$, consider Figure 2. Because $F_1$ is an exclusive feature that shares no code elements with other features, its value is zero. On the other hand, because $F_4$ and $F_3$ share a code element $C_5$, their $\mathcal{TE}$ value is 1.

*Rationale for inclusion* - A higher $\mathcal{TE}$ for a feature means that the feature contains more elements which are shared across features. Our speculation is that the more a feature is *contaminated* with tangled elements (a higher $\mathcal{TE}$), the more it loses its individuality and the more difficult it is for FLTs to discriminate that feature.

***Metric 3: Crosscutting in Features***

This metric is derived from *concern-interlacing* [40], [41] measures, which assess the degree to which concerns overlap. For a feature the *Crosscutting in Feature*($\mathcal{CiF}$) metric measures the number of other features that share at least one code element with that feature. Thus it is similar to $\mathcal{TE}$, but instead of the overlap being quantified in terms of the code associated with the feature, it quantifies overlap in terms of the overlapping features. For a feature F, it can be represented as follows:

$$\mathcal{CiF}(\mathcal{F}) = |\{\mathcal{F}' \in (\widehat{\mathcal{F}} - \mathcal{F}) | \mathcal{C}_{\mathcal{F}} \cap \mathcal{C}_{\mathcal{F}}{}' \neq \emptyset\}|$$

Again consider Figure 2 where $F_4$ and $F_3$ share some code. Hence they are cross-cutting features with a $\mathcal{CiF}$ value of 1.

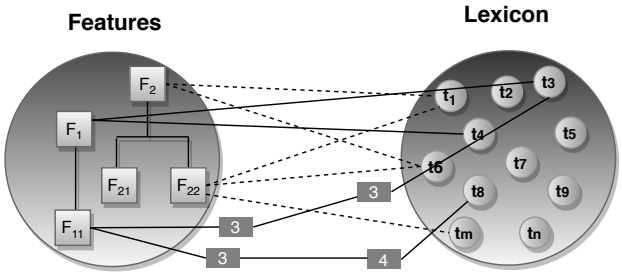


Fig. 3. Impact of Feature Relations on their Lexicon Sharing

*Rationale for inclusion* - As mentioned above, $\mathcal{CIF}$ seems quite related to the Tangled element metric presented and so a higher $\mathcal{CIF}$ would seem to imply greater FLT difficulty. However, when finding a feature foothold in the code base, having several overlapping features may facilitate the location of that foothold. This intuition is based on the assumption that the more other features cross-cut a feature, the more it represents basic/core functionality. Take, for example, a menu which offers the user a range of feature/menu options. All these features share this common menu and it provides a foothold to all.

$\mathcal{RFS}$, $\mathcal{CIF}$ and $\mathcal{TE}$ all exploit the $\mathcal{R}_{\mathcal{FC}}$ relationships. The metrics presented from here, concentrate on the $\mathcal{R}_{\mathcal{CT}}$ relationships.

***Metric 4: Unique Lexical Coverage***

*Unique Lexical Coverage* $(\mathcal{ULC})$ is the number of unique words that are associated with a feature F, relative to the number of unique words in the system. For example, in Figure 3, where features are directly related to the lexicons in the feature-associated code, $(\mathcal{ULC})$ for *F2* is 2/11.

*Rationale for inclusion* - This metric measures the part of the overall application domain knowledge [42] encoded in a system belonging to a feature. Our conjecture about $(\mathcal{ULC})$ is that it assesses the amount of potential vocabulary a domain expert can leverage in their query: the more potential domain vocabulary in the system a feature has, the more likely they will be to get a hit; that is, the more likely a developer is to formulate a query that will result in a hit. But against that, the more potential vocabulary from the domain, the more overlaps might happen because of lexical tangling and so the less discrimination for the feature sought, meaning that lots of other hits will be of lesser relevance. $(\mathcal{ULC})$ varies from 0 to 1. In the case of a $(\mathcal{ULC})$ of 0, a feature encodes none of the overall domain knowledge. However, in the case of a $(\mathcal{ULC})$ of 1, the complete domain knowledge is covered by a feature. This is possible when a composite feature contains the code of component features, which is not exceptional [28], [43]. We anticipate a sweet spot between these two extreme: one that maximizes discrimination, possibly differing across FLTs.

***Metric 5: Lexical Saturation***

If $\widehat{\mathcal{F}}$ is the set of features in a system and $\mathcal{L}$ is the lexicon that code all the elements related $\widehat{\mathcal{F}}$, then, *Lexical Saturation* $(\mathcal{LS})$ of a feature $(\mathcal{F})$ belonging to $\widehat{\mathcal{F}}$ is described as follows:

$$\mathcal{LS}(\mathcal{F}) = \frac{\sum_{(l,m)\in\mathcal{L}'_{\mathcal{F}}} m}{\sum_{(l,n)\in\mathcal{L}'} n}$$

where $\mathcal{L}'$= {(l,n) $\forall$ l$\in$L $\wedge$ n is the number of occurrences of l in the source code $\mathcal{C}$ } and $\mathcal{L}'_{\mathcal{F}}$= {(l,m) $\forall$ l$\in$L$_F$ $\wedge$ m is the number of occurrences of l in the source code $\mathcal{C}_{\mathcal{F}}$} where $\mathcal{L}'_{\mathcal{F}}$ is the corpus belonging to feature F. Note that $\mathcal{LS}(\mathcal{F})$ incorporates repetitions of words.

Consider that in Figure 3, the lexicons might be repeated more than once in the code base, as represented by the numbers towards the right-hand-side of the edges. They might then appear more than once in the code associated with a feature (represented by the numbers towards the left hand side of the edges). So, in this case, the *Lexical Saturation* of feature $F_{11}$ is 6/16, because a total of 6 non-unique terms occur in feature $F_{11}$ out of a total population of 16.

*Rationale for inclusion* - The discriminating factor between $\mathcal{LS}$ and $\mathcal{ULC}$ is vocabulary repetition: it directly indicates the feature size, relative to the system size in terms of the lexical corpus, measuring the degree to which a feature is integral to the implementation body of all of the feature set in a system. It is different from the $\mathcal{ULC}$ in that a feature with more unique words can have a smaller size in terms of $\mathcal{LS}$ if there is less repetition of words. Thus, this metric assesses the degree to which vocabulary emphasis (in terms of vocabulary repetition) can impact on the performance of FLTs. Like $\mathcal{ULC}$, $\mathcal{LS}$ also varies from 0 to 1.

***Metric 6: Query Size***

*Query Size* $(\mathcal{QS})$ counts the number of unique terms in the feature query.

*Rationale for inclusion* - There is strong empirical evidence for the impact of the query's characteristics on performance of FLT in FL literature [6], [12], [15], [16]. We select the query size, which is specifically important for IR-based FLTs, as it relates the feature-query to the source code elements that implement the queried feature. $(\mathcal{QS})$ intrinsically quantifies the lexical space which an FLT leverages to locate the implementation body of a feature. The intuition is that a higher number of unique terms in a feature-query provides higher potential to locate more source code elements related to the feature. However, this could also have a confounding effect, where the terms defined in the query map to other source code elements, not related to the feature.

### 3.2 Theoretical Evaluation of Metric Suite

Our presented metrics comply with the five mathematical measurement properties proposed by Briand et al. [39]. Both our $\mathcal{R}_{\mathcal{CT}}$ and $\mathcal{R}_{\mathcal{FC}}$ based feature metrics assume *non-negative* values. $\mathcal{RFS}$, $\mathcal{TE}$, and $\mathcal{CIF}$ are based on the cardinality of the source code elements or other features and therefore their minimum possible value is zero. In the case of $\mathcal{TE}$ and $\mathcal{CIF}$, when there is no relationship between two features, these metrics return a measurement of zero, meeting the *null value* property. All other metrics are based on the text size of the feature-related source code and query, and hence always produce a value equal to, or greater than zero. To obey the *monotonicity* (i.e. non-decreasing) property, when a new method is added to a feature, $\mathcal{RFS}$ increases

and if the added method is shared by another feature, $\mathcal{TE}$ and $\mathcal{CiF}$ can only increase. Similarly, the addition of a method can only increase $\mathcal{LS}$, and in the case of the newly added method also adding some unique terms, $\mathcal{ULC}$ increases. In the same way, the addition of new terms in a query ensures the non-decreasing property of $\mathcal{QS}$. Finally, since no metric is obtained after merging two (or more) features (i.e. features are isolated), the final two properties are also adhered to.

### 3.3 Feature Characteristics as Moderators

The primary reason for significant interest in FLTs, in general, is the belief that *FLTs can reduce the extensive effort required in the software maintenance phase by automating the process of FL* [5]; a belief assessed by empirical assessment of FLTs' accuracy. This section provides a theoretical basis for the inclusion of feature characteristics in FLT assessments.

The validity of any causal inference depends on how evidence of causality is established in experiments [44]. Clearly identifying the variables to be controlled, in order to evaluate a hypothesis, is a core question during experimental design [45]. However, variables that are not controlled also often influence the outcome, affecting the validity of the causality assessment [44]. Such variables have a confounding effect [19], [44] and their manipulation/evaluation can improve empirical findings by scoping/contextualizing the results more accurately [19], [44], [45].

In this case, suppose that $X$ (FLT/Configuration) is the independent variable and $Y$ (Performance) is the dependent variable. Then, a linear regression model to quantitatively analyze the effect of $X$ on $Y$ can be described by the following equation:

$$Y = \beta_0 + \tau X + \varepsilon \tag{1}$$

Here $\tau$ represents the relationship between $X$ and $Y$ and $\beta_0$ and $\varepsilon$ are the population regression intercept and residual respectively. The path diagram showed in Figure 4 (A) graphically illustrates this causal belief that '$X$ causes $Y$': that an FLT and its configurations are the sources of causality for the FLTs' performance.

Now consider that an investigator is interested in predicting the performance of several FLTs, given a known set of features. It is possible that the features' characteristics in the feature set (e.g., the number of lines of code comprising a feature) affect the FLTs' performance. In experimental theory, the feature set is known as a *moderator* or *covariate* [18], [19]. A moderator can help better predict (or explain) the dependent variable by moderating the relationship between the independent and dependent variables [19]. To accommodate this in (1), a third variable W is included, which quantifies some feature characteristics and can better explain the outcome $Y$, alone or in interaction with the independent variable $X$ [18], [19]. Then (1) can be re-written:

$$Y = \beta_0 + \tau' X + \beta W + \varepsilon \tag{2}$$

Note that $W$ is a between-participant variable which is not measured repeatedly. Note also, from equation 2, that a simultaneous change in $X$ and $W$ will have an additive effect on $Y$ [19] which can increase the power of causation. This is

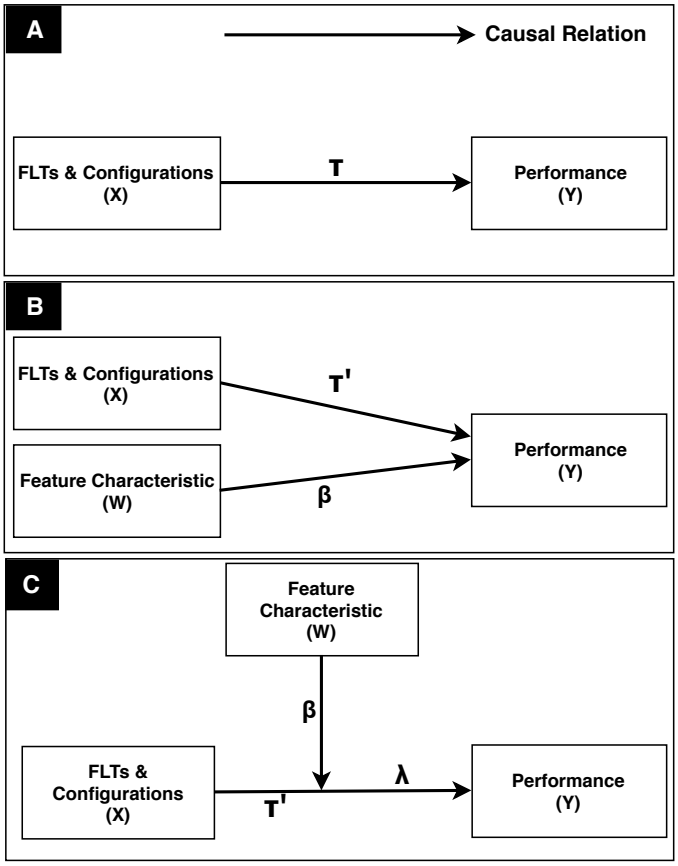


Fig. 4. Path Diagram Illustrating the Causation and Moderation Effect of Moderator Variable on Outcome Variable

presented in Figure 4 (B), where $W$ controls the relation '$X$ *causes* $Y$', without interacting with $X$.

A *Moderation effect* then occurs when the relationship between the FLTs/their configurations and FLTs performance is, at least, partially dependent upon the feature characteristics. To probe the moderation effect, the statistical interaction between the treatment variable (FLTs and configurations) and the moderator (features' characteristics) is examined [18], [19]. Path diagram Figure 4 (C), graphically illustrates the study of moderators' effects: i.e. the relation between $X$ and $Y$ where the effect of $W$ has also been controlled. The slope of the relation between $X$ and $Y$, between $W$ and $Y$, and between $X$ interacting with $W$ and $Y$ are respectively represented by $\tau'$, $\beta$ and $\lambda$. In regression, such interactive relations between $X$ and $W$ can be represented by adding an interaction term in (2) as follows:

$$Y = \beta_0 + \tau' X + \beta W + \lambda XW + \varepsilon \tag{3}$$

This shows that $X$, in conjunction with (moderated by) $W$, impacts on $Y$ by a product of the slope coefficients used to describe the $X$ to $W$ and $W$ to $Y$ relations. Importantly, this also shows that, when $\lambda$ is non-zero, $\tau$ (in equation 1) will differ from $\tau'$ (in equation 3). That is, in the case of a positive or negative result of $\lambda$, $W$ will enhance or decrease [18], [19] the magnitude of the impact $X$ has on $Y$. By factoring out $X$, (3) can be re-written as:

$$Y = \beta_0 + X(\tau' + \lambda W) + \beta W + \varepsilon \tag{4}$$

TABLE 3
Systems and Associated Feature Set Characteristics

| Systems | Version | LOC | Method | Feature | $\frac{goldsets}{feature}$ | $\frac{terms}{feature}$ | $\frac{lexicon}{feature}$ | $\frac{lexicon}{query}$ |
|---|---|---|---|---|---|---|---|---|
| jEdit | 4.3 | 104K | 6,413 | 150 | 4.8 | 578 | 146 | 31 |
| Rhino | 1.5 | 32K | 2801 | 328 | 663.1 | 36,253 | 10,462 | 379 |
| JabRef | 2.6 | 74K | 4,607 | 39 | 6.7 | 1282 | 304 | 37 |
| ArgoUML | 0.22 | 149K | 11,464 | 91 | 4 | 555 | 178 | 37 |
| ArgoSPL | N/A | 120K | 14,654 | 23 | 5328 | 19,769 | 7,213 | 220 |
| Eclipse | 3 | 1.46M | 121K | 45 | 3.4 | 363 | 87 | 29 |
| iBatis | 2.3 | 14K | 1,859 | 85 | 12.5 | 826 | 271 | 195 |
| Mylyn | 1.0.1 | 13K | 482 | 23 | 7.2 | 598 | 164 | 264 |
| muCom. | 0.8.5 | 77K | 8,187 | 92 | 7 | 865 | 210 | 30 |

The last four columns present the average number of methods per feature, average terms per feature, average unique terms per feature and average unique terms per query, respectively

Which makes it clear that the effect of $X$ on $Y$ is a function of $W$. In observational studies, moderators are, mostly defined as the pre-test characteristics (excluding the actual treatment) of the participants tested in an experiment [44]. In the arena of FL, participants are the features derived from software systems as the feature sets, and the pre-test characteristics are the feature-set characteristics.

The next section presents the empirical design used to probe the impact of the proposed feature metric suite.

## 4 Research Design

As the proposed feature metrics reflect feature set characteristics, the following research questions are targetted in this research:

***RQ1****-Does the proposed feature metric suite explain the performance of FLTs, as measured by currently established evaluation measures? In other words, do the feature characteristics presented in this research have strong evidence of causality over FLTs' performance?*

***RQ2****-Do the feature metrics reflect different degrees of moderation on different FLTs? In other words, do the feature characteristics presented in this research have strong evidence of causality over performance differences between FLTs?*

### 4.1 Dataset and Empirical Practice

#### 4.1.1 Dataset

In FL, controlled experiments tend to focus on breadth-of-analysis, where feature sets are drawn from a set of systems [3]. In selecting our feature sets, we focused on those which are either unambiguous (for example, macros that are used for implemented system configurations), are created using more than one ground truth (triangulation) [32] or ones that are the most commonly used in FL evaluation [3]. Table 3 presents the feature sets in terms of their associated systems, the number of features contained and their characteristics, thus demonstrating some high heterogeneities of the feature sets. For example, the average number of methods that constitute a feature varies from 3.4 to 5328 over the systems. The combined feature sets consist of 878 features.

#### 4.1.2 Common/Best Empirical Practice

**1. Feature Location Techniques Employed in the Study**-Four FLTs were chosen for this study. VSM-Lucene, LSI-MATLAB, and LDA-R were chosen because Vector Space Modelling (VSM), Latent Semantic Indexing (LSI), and Latent Dirichlet Allocation (LDA) are commonly used approaches in FLT, being employed as comparators in 20%, 21% and 10% of FLT studies respectively [3]. The particular implementations were chosen because they are openly available, allowing other researchers to replicate the findings and because they seem to outperform other FLTs based on these three approaches [32] [1]. Additionally, a state-of-the-art technique, where PageRank is combined with LSI, was selected as a fourth technique, again based on its favorable history in comparator studies [14], [46].

**2. Evaluation Measures**-As discussed in Section 2.1, FLTs can be divided into techniques that attempt to find a source-code foothold into the feature or the (near-full) extent of the feature in the code. Razzaq et al. [3] observed that precision, recall, F-measure and Mean Average Precision (MAP) are the most commonly used evaluation measures to assess FLTs for near-full FL whereas Mean Reciprocal Rank (MRR) is a commonly used measure to assess the foothold location goal. The F-measure was not employed here because it obscures the individual effect of precision and recall [47] and the goal here is not to show improved performance but affected performance. In addition, to mitigate the impact of feature size on the measures chosen, we measured the relative values of precision and recall in line with the protocol suggested by Shin et al. [47]: there, precision and recall were calculated on ten intervals, in the ranked list, where each interval corresponds to one-tenth the size of the feature in the feature set. Such relative measures diminish the direct connection of precision to $\mathcal{RFS}$.

**3. Best Identified Configurations**-The configuration of the FLTs, pre-processing steps, and composition of the feature sets (source code elements and feature-queries) are also important considerations in empirical design. We employed best practices in these considerations, as derived by the empirical community [32] (see Table 4).

### 4.2 Managing Extrapolation and Counterfactual Inferences

Experiments without appropriate control provide less direct causation evidence [44]. However, in cause-probing research, the control of experimental effects is always an issue because of two major concerns: 1) avoiding the omission of underlying independent variables which can cause incorrect evaluations of the magnitude of causation and, 2) selecting high-quality counterfactual inferences which are different from the treatment condition [44], [45], [49], [50]. This research aims to address the first issue by looking at an additional moderation variable: the characteristics of the features located. In doing so it employs a multiple regression technique where the experimental design is triangulated with extrapolation and counterfactual inferences [44], [49], to address both issues.

In terms of extrapolation, in a feature set, a feature is represented by a set of source code elements and a feature query. Consequently, source-code product-level metrics [21], [51], [56], and textual metrics [9], [10] associated with the feature-query, could be employed towards measuring the

1. https://www.lero.ie/research/datasets/feature_location/comparison

TABLE 4
Commonly Employed Configuration and Normalization

| Parameter | Total Configuration Tested | Studies Empirically Investigating | Common Best Practice |
|---|---|---|---|
| **Settings Common to all FLTs** | | | |
| **Feature Query** | 1. Bug report title<br>2. Bug report descr[1].<br>3. Title + Descr.<br>4. 10 PBR[2]<br>5. PBR-All<br>6. Title + Descr.+PBR | Thomas et al. [6]<br>Moreno et al. [16]<br>Biggers et al. [12] | Title + Descr. |
| **Source Code Element** | 1. Ident.[3]<br>2. Comm.[4]<br>3. Ident. + Comm.<br>4. Lits[5]<br>5. Ident. + Lit.s<br>6. Comm.+ Lits.<br>7. Ident.+Comm.+Lits. | Thomas et al. [6]<br>Moreno et al. [16]<br>Biggers et al. [12] | Ident. + Comm. |
| **Pre-processing Steps** | 1. None<br>2. Split<br>3. Stop<br>4. Stem<br>5. Digit<br>6. Special chars<br>7. Split + Stop<br>8. Split + Stem<br>9. Stem + Stop<br>10. Split+Stop+Stem<br>11. Setting 10 + PBR | Thomas et al. [6]<br>Moreno et al. [16]<br>Panichella et al. [29] | Split+Stop+Stem |
| **Specific to VSM** | | | |
| **Term weight** | 1. Tf-Idf<br>2. Sub-linear Tf-Idf<br>3. Boolean<br>4. tf<br>5. tf-entropy<br>6. logij=log(tfij+1) | Thomas et al. [6]<br>Moreno et al. [16]<br>Panichella et al. [29]<br>Wang et al. [48] | Tf-Idf |
| **Similarity Function** | 1. Cosine<br>2. Overlap<br>3. Jaccard<br>4. Dice | Thomas et al. [6]<br>Moreno et al. [16]<br>Panichella et al. [29]<br>Wang et al. [48] | Cosine |
| **Specific to LSI** | | | |
| **Term weight** | 1. Tf-Idf<br>2. Sub-linear Tf-Idf<br>3. Boolean<br>4. tf<br>5. tf-entropy | Thomas et al. [6]<br>Moreno et al. [16]<br>Panichella et al. [29] | Tf-Idf |
| **Similarity Function** | 1. Cosine<br>2. Jaccard<br>3. Dice | Thomas et al. [6]<br>Moreno et al. [16]<br>Panichella et al. [29] | Cosine |
| **Number of Topics** | 32-300 | Thomas et al. [6]<br>Moreno et al. [16] | 200-250 |

[1]Description, [2] Past Bug Reports, [3]identifiers, [4]comments, [5] literals

characteristics of such features. While our literature review [3] suggests no specific work in feature set characterization, substantial metric work has been performed in other software engineering activities e.g. defect prediction [54], fault proneness [54], impact analysis [9], [10], [22], and software quality [54]. Table 5 presents a state-of-the-art catalog of product-level [21] metrics which could be employed to measure other feature characteristics. For extrapolation, a range of these metrics will be assessed, regarding their ability to inform on FLT performance.

To select a comprehensive set of characteristics, we employed the frequently used metrics in SE [20], [21], [51], [52], [53], [54]. In brief, we relied on three systematic surveys [21], [56], [57] which explored more than (overall) 400

TABLE 5
State of the Art Categories of Product-level Metrics

| Metrics Category | Description | Example of Studies Proposed Metrics |
|---|---|---|
| Size Metrics | These metrics measure different aspects of code size, for example, Lines of Code | Fenton et al. [51]<br>Mills et al. [15] |
| Complexity Metrics | Complexity usually measures the interaction between programmer and program, defined on the basis of program size and control structures | Radjenovi et al. [52] |
| Coupling Metrics | Coupling measures the inter-module association strength established due to some sort of connection (e.g. dependencies) | Briand et al. [53]<br>Briand et al. [54] |
| Cohesion Metrics | Cohesion measures the degree to which code elements in a software module are structurally bound together | Briand et al. [20]<br>Briand et al. [54] |
| Structural Metrics | Metrics capture dimensions and aspects of program structures (e.g. Depth of Inheritance). Note that these can overlap with other metrics | Briand et al. [20]<br>Chidamber et al. [55] |

metrics used in SE and reported on the frequently used metrics. We restricted the metric set based on coverage of the above categories and the common granularity levels that reflect FL gold-sets [1], [3], [5]. For example, we did not select system-level metrics since such a metric provides only one datum per system and could not be validated in this research. Table 6 presents the set of metrics selected for this research. These code characteristics are obtained using jHawk[2] and SourceMeter[3], at method and class levels, and then summed up to feature level in the spirit of Revelle et al. [10]. It should be noted that, since these characteristics are based on the aggregation of the established individual code characteristics, they too obey the validation properties presented by Briand et al. [39].

The second central task is to establish counterfactual inferences. In a within-participant design, counterfactual inferences could be achieved by more than one investigation of independent variables for plausible different (more than one) dependent variables [44], [50]. In such multiple investigations of participants, one acts as its own control for the second [50]. The details of how this was achieved, in the analysis protocols of this within-participant design [19], are described below, for each research question.

***1. Testing whether the feature characteristic suite controls overall FLTs' performance*** - In this study, feature metrics are repeated for observations of the independent variables and the research builds a performance model based on the multiple impactful feature characteristics which may interact with independent variables [58]. Therefore, empirical designs that mostly assume non-repeated measures or *no interactions* (e.g. ANCOVA) are not an appropriate choice [18], [19], [58]. Instead, to meet the maximum assumptions of the feature set's data [19], [59], [60], we employed multiple regression analysis.

To test the assumptions that our data meets the criteria for multiple regression, we employed the Durbin-Watson test (normal range 1.5-2.5) for the *independence of residuals* and Cook's distance (normal value $< 1$) to verify that data do not have significant outliers. In testing the *multi-collinearity* of independent variables, we found our feature data is collinear (tolerance mostly larger than 0.1). Even-

2. http://www.virtualmachinery.com/jhawkprod.htm
3. https://www.sourcemeter.com/

TABLE 6
The set of Extrapolation Metrics

| Category | Metric | Description |
|---|---|---|
| Size | LOC | Lines of code in a feature* |
| | LLOC | Logical (conditional) lines of code in a feature* |
| | NOS | Number of statements in a feature* |
| | LOCC | Lines of code in a feature** |
| | LLOCC | Logical lines of code in a feature** |
| | NOSC | Number of statements in a feature** |
| | NA | Number of attributes in a feature |
| | NM | Number of methods in a feature |
| | NumPAR | Number of parameters in a feature* |
| | NS | Number of setters in a feature** |
| | NG | Number of getters in a feature** |
| | CD | Comment density in a feature* |
| | CDC | Comment density in a feature** |
| | CLOC | Comment lines of code in a feature* |
| | CLOCC | Comment lines of code in a feature** |
| Complexity | WMC | weighted methods per class in a feature** |
| | McCC | McCabe's cyclomatic complexity in a feature* |
| | HCPL | Halstead's calculated program length in a feature* |
| | HDIF | Halstead's difficulty in a feature* |
| | HEFF | Halstead's effort in a feature* |
| | HNDB | Halstead's number of delivered bugs in a feature* |
| | HPL | Halstead's program length in a feature* |
| | HPV | Halstead program vocabulary in a feature* |
| | NL | Nesting level in a feature* |
| | NLE | Nesting level else-if in a feature* |
| | NLC | Nesting level in a feature** |
| | NLEC | Nesting level else-if in a feature** |
| Coupling | CBO | Coupling between objects in a feature** |
| | CBOI | Coupling between objects in a feature** |
| | NII | Number of incoming invocations in a feature** |
| | NOI | Number of outgoing invocations in a feature** |
| | RFC | Response set for class in a feature** |
| | NIIM | Number of incoming invocations in a feature* |
| | NOIM | Number of outgoing invocations in a feature* |
| | SiggFF | Total incoming and outgoing invocations in a feature* |
| Cohesion | LCOM5 | Lack of cohesion in methods in a feature** |
| Structural | DIT | Depth of inheritance tree in a feature** |
| | NOA | Number of ancestors in a feature** |
| | NOCh | Number of children in a feature** |
| | NOD | Number of descendants in a feature** |
| | NOP | Number of parents in a feature** |

* Summed for all methods belonging to a feature
** Summed for all classes belonging to a feature

though collinearity diagnostics of our regression models show that variance proportions of each selected characteristic are largely along different dimensions, we suspect that collinearity could impact on the causal inference. Hence, we performed two tests: *stepwise regression analysis* and *principal component analysis (PCA)* for the sake of rigor [18], [60].

Stepwise regression analysis builds a model that initially identifies the characteristics that have the single largest correlation with FLTs performance. It then adds other characteristics to the model based on the part of the variance in the performance that they explain and their partial correlation with the characteristics already in the model. In each iteration, the model re-evaluates itself and removes the characteristics that do not significantly contribute towards variance explanation. Finally, the optimal set of the characteristics, that best explain the maximum possible variance, are retained in the model [61].

A stepwise regression model might end up with two highly correlated characteristics. PCA addresses this issue by reducing a large number of characteristics to a smaller number of uncorrelated weighted combinations of characteristics which accounts for the maximum possible variance. Such weighted combinations of characteristics are known as principal components. Here, we employed the *Varimax* method of orthogonal rotation as it maximizes variance among squared values of loadings for each factor [62].

Another important assumption of multiple regression is *Homoscedasticity* of data i.e. standardized residuals have a homogeneous variance from the predicted values across all values of the independent variables [19], [63]. This assumption rarely holds in real data [19]. In testing our data's homoscedasticity, we found it mostly heteroskedastic. Although the estimator of the linear regression is unbiased when this assumption is violated, it can impact the significance of the estimator [19]. Multiple methods to reduce the effects of heteroscedasticity on causal inference in linear regression have been presented [19]. Here, we employed MacKinnon et al.'s method [63] to keep the test size at a nominal level, regardless of heteroscedasticity. Finally, to establish counterfactual inferences different from evaluation measures, different FLTs are employed.

***2. To test whether feature characteristics, as measured by the proposed metric suite, impact different FLTs' performance*** - if the performance of FLTs is moderated by specific feature characteristics, to different degrees.

Moderation analysis is different in within-participant (as opposed to between-participant) design [18]. In a within-participant design, the regression weights of a model, belonging to the outcome variable Y, predicted by *W*, are allowed to vary by treatment conditions (X). Therefore, in the within-participant case, equation 2 in Section 3.3 accommodates the impact of the third variable

$$Y = \beta_0 + \tau' X + \beta W + \varepsilon$$

and can be written as follows [18]:

$$Y_{ij} = \beta_{0j} + \beta_j W_i + \varepsilon_{ij} \tag{5}$$

Where $Y_{ij}$ is the measure of the outcome $Y$ for feature $i$ using FLT $j$. Notice that $W_i$ is not measured-repeatedly, hence it does not have subscript $j$. In the case of two FLTs, the intercepts and slopes are allowed to differ by treatment conditions which result in two models for each outcome variable, and equation 5 can be re-written as follows:

$$Y_{i1} = \beta_{01} + \beta_{11} W_i + \varepsilon_{i1} \tag{6}$$

$$Y_{i2} = \beta_{02} + \beta_{12} W_i + \varepsilon_{i2} \tag{7}$$

This defines the causality relation of *W* for the outcome in each condition (discussed under research question 1). When $\beta_{11} \neq \beta 12$, then the slope that describes the relation between FLTs and $Y$ depends upon *W* [18]. Algebraically, this situation is captured by subtracting the second condition (in equation 7) from the first (in equation 6).

$$Y_{i2} - Y_{i1} = (\beta_{02} - \beta_{01}) + (\beta_{12} - \beta_{11}) W_i + (\varepsilon_{i2} - \varepsilon_{i1})$$

$$Y_{Di} = \hat{\beta}_0 + \hat{\beta}_1 W_i + \hat{\varepsilon} \tag{8}$$

Here $\hat{\beta}_1$ is the coefficient for *W* and reflects the difference between $\hat{\beta}_{11}$ and $\hat{\beta}_{12}$. Hence, to test the moderation effect, the hypothesis that needs to be tested is: *Is $\hat{\beta}_1$ significantly different from zero?*. If the decision is yes, we accept the null hypothesis; otherwise, we reject it.

To answer research question 2, the data assumptions are mostly the same as for research question 1. However, since FLTs are controlled in research question 2, we employed four different evaluation measures, i.e. precision, recall, MAP and MRR, as the counterfacts. In a within-participant design, this will enable the investigation of multiple outcomes (evaluation measures) for each feature to establish the counterfactual inference in testing the moderation effect in this case [50]. To measure the significance of results, this study used traditional thresholds (i.e., $\alpha$ = 0.05, two-tailed) to answer each research question.

## 5 RESULTS

### 5.1 Causal Effect of the Feature Metric Suite

In a regression model, $R^2$ signifies the amount of variance of an outcome variable explained by the set of predictors [61]. Adjusted $R^2$, on the other hand, explains any bias in the $R^2$ values by normalizing $R^2$ values with a standard error of estimate values. This means that, while adjusted $R^2$, is not the true estimator of the regression model, particularly in a control experiment, it is more generalizable than $R^2$. Table 7 presents the results of stepwise regression models in terms of the feature characteristics selected in the final iteration of the model. These results were obtained across all 9 systems: 878 features. As is evident from the table, $R^2$ has almost the same values as adjusted $R^2$, which indicates the absence of bias in the models. Here, the standard error (of the estimator) measures the amount the actual FLTs' performance deviates from the model-predicted performance.

The column *Characteristics Selected* presents the distribution of $R^2$ across feature characteristics, ordered according to the variance they explain from higher to lower. Here, for example, the first four rows should be read, "precision is mostly explained by the $\mathcal{RFS}$, $\mathcal{QS}$, and $\mathcal{ULC}$ metrics across all FLTs employed". It is apparent from the table that much of the variance is explained by our feature metric suite. They explain between 84-94% of the variance in precision down to 28-78% in MAP. However, when our feature characteristics suite explained less variance, (for example in the case of MAP) standard error is also much less, which shows that the proportion of the data that deviated from the actual performance was lower. The sign against each metric represents the direction of the causality. For example, the higher the $\mathcal{RFS}$, the better the precision score of LSI-Matlab, VSM-Lucene, LDA-R, and PageRank. In contrast, the lower the $\mathcal{QS}$ the higher the precision score for the four techniques.

Stepwise regression might have selected highly correlated feature characteristics [60]. In that case, stepwise regression models have likely over-fitted the data in explaining the variance. To overcome this collinearity issue, we generate the principal components of the data [62]. Table 8, on the left-hand side, shows the principal components having factor loadings larger than 0.9 for each component. The right side of Table 8 shows the top 20 characteristics in each component after components having a factor loading of greater than 0.9.

The *Variance* row in the table (on the left side) shows the distribution of the variance explained by each component. A total of 98.5% variance has explained by two components. Results of the PCA are mostly in agreement with stepwise regression in regards to the characteristics which capture most of the variance (92.8%), and the main contributors to that principal component. $\mathcal{QS}$ dropped off, which is surprising given its prevalence in Table 7 and given that it is logically quite orthogonal to the other proposed metrics but its explanation power in Table 7 is quite low.

To mitigate against the overestimation that might have been caused by the stepwise regression model, we build the regression model using the top characteristics of principal components achieved by PCA analysis. To keep all the top characteristics in the model, the *enter* regression method is used this time [60]. Table 9 presents the resultant model's results. It reinforces the high accuracy of the models presented in Table 7 at predicting FLTs performances, as indicated by the decrease in $R^2$ values mostly equaling the variance explained by $\mathcal{QS}$ in the regression models, without principal components. This further indicates the effect of our feature metric suite where all models are statistically significant at the 99 percent confidence level.

*Answer to Research Question 1* – From the stepwise regression analysis and PCA results, we can conclude that our presented feature metric suite highly explains the variance in FLTs performance, thus affirming our hypothesis that *the feature metric suite is a strong predictor of FLT's performances*: $\mathcal{RFS}$ is a strong precision/MAP predictor. $\mathcal{ULC}$ is a strong indicator of recall over all four FLTs. The picture is less clear for MRR but $\mathcal{CiF}$ seems to be a strong predictor for it.

### 5.2 Moderation effect of Feature Metric Suite

Table 11 shows the results of the moderation analysis. Since the independent variable in this analysis is the *difference in performance* we first have to test whether the performance of employed techniques differs significantly (before investigating which characteristics can moderate the FLTs' performance). Considering that the feature set data distribution is mostly normal, but non-normal in a few cases, we employed the Wilcoxon signed-rank and a paired t-test. Table 10 shows the p values for both analyses, for each pair of techniques, performed over 878 features. It shows that the performance of all techniques differs significantly in almost all evaluation measures with p values frequently less than 0.001.

Table 11 presents the results of the top 10 moderators which impact significantly the relative performance of the employed FLTs across each evaluation measure (complete results can be found in the replication package). For example, in the first data-cell of Table 11, the $\beta_{PR} - \beta_{LDA}$ data specifies the coefficient $\hat{\beta}_1$ of feature characteristics which moderates the difference between $\beta_{PR}$ and $\beta_{LDA}$ (Equation 8). Thus, the value of $\beta_{PR} - \beta_{LDA}$ for the $\mathcal{RFS}$ metric under the MRR column indicates that, with each one-unit increase in $\mathcal{RFS}$ value, the performance difference between PageRank and LDA will increase by 2.1717.

Note that components of the presented feature metric suite are always selected in the top 10 moderators. The

TABLE 7
Characteristics Selected by the Stepwise Regression Model in each Evaluation Measure

| Evaluation Measures | Counterfacts | Characteristics Selected | | | | | | $R_2$ | Adjusted $R_2$ | Standard Error | P |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Precision** | LDA | $\mathcal{RFS}$ | $\mathcal{QS}$ | $\mathcal{ULC}$ | $\mathcal{TE}$ | | | 0.939 | 0.939 | 0.028 | 0.000 |
| | | 0.934 | 0.002 | 0.002 | 0.001 | | | | | | |
| | | (+) | (-) | (+) | (-) | | | | | | |
| | PageRank | $\mathcal{RFS}$ | $\mathcal{QS}$ | $\mathcal{ULC}$ | | | | 0.938 | 0.938 | 0.030 | 0.000 |
| | | 0.933 | 0.003 | 0.002 | | | | | | | |
| | | (+) | (-) | (+) | | | | | | | |
| | LSI | $\mathcal{RFS}$ | $\mathcal{QS}$ | $\mathcal{ULC}$ | | | | 0.899 | 0.899 | 0.036 | 0.000 |
| | | 0.89 | 0.005 | 0.004 | | | | | | | |
| | | (+) | (-) | (+) | | | | | | | |
| | VSM | $\mathcal{RFS}$ | $\mathcal{QS}$ | $\mathcal{ULC}$ | | | | 0.841 | 0.841 | 0.051 | 0.000 |
| | | 0.824 | 0.014 | 0.004 | | | | | | | |
| | | (+) | (-) | (+) | | | | | | | |
| **Recall** | LDA | $\mathcal{ULC}$ | $\mathcal{CiF}$ | $\mathcal{LS}$ | $\mathcal{QS}$ | HEFF | $\mathcal{TE}$ | 0.926 | 0.925 | 0.100 | 0.000 |
| | | 0.916 | 0.004 | 0.001 | 0.001 | 0.001 | 0.001 | | | | |
| | | (+) | (+) | (-) | (-) | (+) | (-) | | | | |
| | PageRank | $\mathcal{ULC}$ | $\mathcal{QS}$ | $\mathcal{CiF}$ | CBOI | $\mathcal{TE}$ | WMC | 0.896 | 0.895 | 0.130 | 0.000 |
| | | 0.880 | 0.008 | 0.004 | 0.002 | 0.001 | 0.001 | | | | |
| | | (+) | (+) | (+) | (+) | (-) | (-) | | | | |
| | LSI | $\mathcal{ULC}$ | $\mathcal{QS}$ | $\mathcal{CiF}$ | $\mathcal{LS}$ | $\mathcal{TE}$ | $\mathcal{RFS}$ | 0.817 | 0.815 | 0.156 | 0.000 |
| | | 0.794 | 0.014 | 0.002 | 0.004 | 0.001 | 0.002 | | | | |
| | | (+) | (-) | (+) | (-) | (-) | (+) | | | | |
| | VSM | $\mathcal{ULC}$ | $\mathcal{QS}$ | $\mathcal{LS}$ | | | | 0.652 | 0.651 | 0.204 | 0.000 |
| | | 0.615 | 0.035 | 0.002 | | | | | | | |
| | | (+) | (-) | (-) | | | | | | | |
| **MAP** | LDA | $\mathcal{RFS}$ | HNDB | $\mathcal{QS}$ | DIT | | | 0.711 | 0.710 | 0.071 | 0.000 |
| | | 0.699 | 0.005 | 0.004 | 0.003 | | | | | | |
| | | (+) | (+) | (-) | (-) | | | | | | |
| | PageRank | $\mathcal{RFS}$ | $\mathcal{QS}$ | | | | | 0.776 | 0.776 | 0.073 | 0.000 |
| | | 0.768 | 0.008 | | | | | | | | |
| | | (+) | (-) | | | | | | | | |
| | LSI | $\mathcal{RFS}$ | $\mathcal{QS}$ | McCabe | | | | 0.56 | 0.558 | 0.094 | 0.000 |
| | | 0.542 | 0.014 | 0.004 | | | | | | | |
| | | (+) | (-) | (+) | | | | | | | |
| | VSM | $\mathcal{RFS}$ | $\mathcal{QS}$ | NOS | | | | 0.285 | 0.283 | 0.153 | 0.000 |
| | | 0.239 | 0.038 | 0.008 | | | | | | | |
| | | (+) | (-) | (+) | | | | | | | |
| **MRR** | LDA | $\mathcal{CiF}$ | $\mathcal{QS}$ | $\mathcal{ULC}$ | $\mathcal{TE}$ | $\mathcal{LS}$ | | 0.915 | 0.915 | 0.138 | 0.000 |
| | | 0.906 | 0.004 | 0.002 | 0.002 | 0.001 | | | | | |
| | | (+) | (-) | (+) | (-) | (-) | | | | | |
| | PageRank | $\mathcal{ULC}$ | $\mathcal{QS}$ | $\mathcal{RFS}$ | NA | $\mathcal{LS}$ | | 0.570 | 0.568 | 0.179 | 0.000 |
| | | 0.533 | 0.028 | 0.004 | 0.003 | 0.002 | | | | | |
| | | (+) | (-) | (+) | (-) | (-) | | | | | |
| | LSI | $\mathcal{CiF}$ | $\mathcal{QS}$ | $\mathcal{ULC}$ | $\mathcal{LS}$ | | | 0.835 | 0.835 | 0.191 | 0.000 |
| | | 0.82 | 0.008 | 0.004 | 0.003 | | | | | | |
| | | (+) | (-) | (+) | (-) | | | | | | |
| | VSM | $\mathcal{CiF}$ | $\mathcal{QS}$ | $\mathcal{ULC}$ | $\mathcal{TE}$ | CBOI | $\mathcal{LS}$ | 0.717 | 0.715 | 0.25 | 0.000 |
| | | 0.68 | 0.017 | 0.009 | 0.007 | 0.003 | 0.001 | | | | |
| | | (+) | (-) | (+) | (-) | (+) | (-) | | | | |

TABLE 8
Principal Components Analysis Results*

| | C1 | C2 | C1 | | | | C2 | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Variance | **92.8** | **5.7** | 1 | NA | 11 | WMC | 1 | LCOM5 | 11 | NII |
| | | | 2 | NOS | 12 | HPV | 2 | CD | 12 | NS |
| CiF | 0.972 | 0.216 | 3 | McCC | 13 | LLOCC | 3 | NOP | 13 | RFC |
| $\mathcal{RFS}$ | 0.949 | 0.305 | 4 | HNDB | 14 | NOI | 4 | CDC | 14 | NIIM |
| $\mathcal{LS}$ | 0.946 | 0.313 | 5 | HPL | 15 | NLEC | 5 | CLOC | 15 | CLOCC |
| $\mathcal{ULC}$ | 0.932 | 0.353 | 6 | NUMPAR | 16 | NOIM | 6 | NG | 16 | NM |
| $\mathcal{TE}$ | 0.932 | 0.347 | 7 | LLOC | 17 | NLE | 7 | CBOI | 17 | SigFF |
| HEFF | 0.914 | 0.354 | 8 | HCPL | 18 | QS | 8 | DIT | 18 | NLC |
| NOCh | 0.227 | 0.967 | 9 | NOSC | 19 | LOCC | 9 | NOA | 19 | HDIF |
| NOD | 0.216 | 0.945 | 10 | LOC | 20 | NL | 10 | CBO | 20 | NL |

*C1: component 1, C2: component 2

TABLE 9
Regression Models on Principal Components

| Evaluation Measures | Counterfacts | $R_2$ | Adjusted $R_2$ | Standard Error | P |
|---|---|---|---|---|---|
| **Precision** | LDA | 0.935 | 0.935 | 0.029 | 0.000 |
| | PageRank | 0.934 | 0.934 | 0.031 | 0.000 |
| | LSI | 0.892 | 0.891 | 0.038 | 0.000 |
| | VSM | 0.825 | 0.824 | 0.054 | 0.000 |
| **Recall** | LDA | 0.925 | 0.925 | 0.101 | 0.000 |
| | PageRank | 0.89 | 0.889 | 0.133 | 0.000 |
| | LSI | 0.806 | 0.804 | 0.161 | 0.000 |
| | VSM | 0.623 | 0.62 | 0.213 | 0.000 |
| **MAP** | LDA | 0.702 | 0.699 | 0.073 | 0.000 |
| | PageRank | 0.77 | 0.768 | 0.074 | 0.000 |
| | LSI | 0.544 | 0.539 | 0.096 | 0.000 |
| | VSM | 0.242 | 0.235 | 0.156 | 0.000 |
| **MRR** | LDA | 0.911 | 0.91 | 0.142 | 0.000 |
| | PageRank | 0.553 | 0.549 | 0.183 | 0.000 |
| | LSI | 0.826 | 0.825 | 0.198 | 0.000 |
| | VSM | 0.692 | 0.689 | 0.261 | 0.000 |

TABLE 10
P Values for Wilcoxon Signed-rank Test and Paired t-test

| Techs. | Test | MRR | MAP | Recall | Precision |
|---|---|---|---|---|---|
| **PageRank vs. LDA** | t-test | 0.000 | 0.000 | 0.000 | 0.000 |
| | Wilcoxon | 0.000 | 0.000 | 0.000 | 0.000 |
| **PageRank vs. LSI** | t-test | 0.000 | 0.002 | 0.047 | 0.047 |
| | Wilcoxon | 0.000 | 0.000 | 0.000 | 0.000 |
| **PageRank vs. VSM** | t-test | 0.000 | 0.000 | 0.000 | 0.000 |
| | Wilcoxon | 0.000 | 0.551 | 0.031 | 0.000 |
| **LDA vs. LSI** | t-test | 0.000 | 0.000 | 0.000 | 0.000 |
| | Wilcoxon | 0.000 | 0.000 | 0.000 | 0.000 |
| **LDA vs. VSM** | t-test | 0.000 | 0.000 | 0.01 | 0.000 |
| | Wilcoxon | 0.000 | 0.000 | 0.000 | 0.000 |
| **LSI vs. VSM** | t-test | 0.000 | 0.000 | 0.000 | 0.000 |
| | Wilcoxon | 0.000 | 0.000 | 0.137 | 0.000 |

coefficients highly depend on the absolute scores of FLT evaluation measures on which they are measured [50], [61]. Therefore, to indicate the significance of such coefficients (i.e. $\beta_{PR} - \beta_{LDA}$ like slopes), the rows labeled *AAD* (Absolute Average Difference) present the average of the absolute values of outcome variables (i.e. performance differences in terms of each evaluation measure) for each pair of techniques. In essence, Table 11 shows that the magnitude of treatment effect (performance difference between two FLTs) depends on the value of the moderating metrics [32].

*Answer to Research Question 2* – The results in Table 11 imply that differences in performance between FLTs can be significantly moderated by the feature characteristics measured in the metric suite proposed i.e. $\mathcal{RFS}$, $\mathcal{CiF}$, $\mathcal{TE}$, $\mathcal{ULC}$, $\mathcal{LS}$, and $\mathcal{QS}$. In terms of the moderation effect size, $\mathcal{RFS}$, $\mathcal{ULC}$, and $\mathcal{LS}$ seem to be the largest contributors.

## 6 Discussion and Lesson Learned

### 6.1 Discussion

***Effect of Feature Suite*** – Table 7 highlights two important aspects of the observed FLTs' performance. Firstly, different aspects of FLT performance, captured by the four evaluation measures, seem aligned with feature metrics. For example, $\mathcal{RFS}$ better explains precision and MAP, most likely because the proportion of the feature-related elements in the system directly affects the proportion of the feature-related elements in retrieved results. Unsurprisingly, it seems less able to explain MRR. Instead, $\mathcal{CiF}$ better explains MRR in the baseline IR techniques, reflecting the hypothesis in Section 3 that higher $\mathcal{CiF}$ scores indicate the presence of more basic/core source-code elements that offer a gateway to many features and that the presence of these elements facilitates finding feature footholds.

As expected, $\mathcal{ULC}$ can better explain recall, presumably because a feature that encodes more unique domain knowledge seems more likely to be mapped to a (sensible) feature-query. However, in PageRank which combines structural information with IR, $\mathcal{ULC}$ better explains the foothold location goal. This is surprising because $\mathcal{ULC}$ is largely agnostic to structure and, if anything, would intuitively seem to increase when there are fewer structural connections to a code element. It is possible though, that $\mathcal{ULC}$'s higher mapping to the query may dominate any such effect.

On the contrary, in the vast majority of cases, $\mathcal{QS}$ seems to explain the reducing performance part of FLTs (note the '-' sign). This may because, the larger $\mathcal{QS}$, the less distinctive the query, which makes it difficult for the FLTs to discriminate between the features successfully. At first glance, this implies that the query expansion work done by other researchers in the FLT domain [15], [64], [65] might not provide the expected benefits. However, the difference here is that the study uses larger feature queries, as provided, whereas work on query expansion in FLTs concentrates on smart approaches to expanding the query, and this may explain their positive impact, as suggested by Sisman et al. [64] and Mills et al. [15]. Our finding suggests that the longer sized queries in our feature sets were less succinct. Note though that this finding does not imply an absolute inverse relation where the performance of the FLTs always reduces, with an increase in $\mathcal{QS}$ value. Instead, it shows that,

TABLE 11
Moderation Analysis for Performance Difference in FLTs

| | MRR | | | MAP | | | Recall | | | Precision | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Metric | $\beta_{PR}-\beta_{LDA}$ | P | Metric | $\beta_{PR}-\beta_{LDA}$ | P | Metric | $\beta_{PR}-\beta_{LDA}$ | P | Metric | $\beta_{PR}-\beta_{LDA}$ | P |
| **PageRank vs. LDA** | $\mathcal{RFS}$ | 2.1717 | 0.000 | $\mathcal{RFS}$ | 0.20712 | 0.000 | $\mathcal{RFS}$ | 0.20513 | 0.000 | $\mathcal{RFS}$ | 0.0445 | 0.000 |
| | $\mathcal{ULC}$ | 0.67763 | 0.000 | $\mathcal{ULC}$ | 0.06247 | 0.000 | $\mathcal{ULC}$ | 0.06194 | 0.000 | $\mathcal{ULC}$ | 0.01331 | 0.000 |
| | $\mathcal{LS}$ | 0.3164 | 0.000 | $\mathcal{LS}$ | 0.0298 | 0.000 | $\mathcal{LS}$ | 0.02963 | 0.000 | $\mathcal{LS}$ | 0.00635 | 0.000 |
| | CD | 0.00243 | 0.000 | CDC | 0.00018 | 0.000 | CDC | 0.0002 | 0.001 | $\mathcal{C}extitiF$ | 0.00004 | 0.000 |
| | CLOCC | 0.00235 | 0.011 | CD | 0.00017 | 0.001 | CD | 0.0002 | 0.000 | $\mathcal{QS}$ | 0.00003 | 0.000 |
| | $\mathcal{C}extitiF$ | 0.0016 | 0.000 | $\mathcal{C}extitiF$ | 0.00015 | 0.000 | $\mathcal{C}extitiF$ | 0.00015 | 0.000 | CD | 0.00003 | 0.001 |
| | $\mathcal{QS}$ | 0.00138 | 0.000 | $\mathcal{QS}$ | 0.00012 | 0.000 | $\mathcal{QS}$ | 0.00009 | 0.000 | $\mathcal{TE}$ | 0.00002 | 0.000 |
| | $\mathcal{TE}$ | 0.00076 | 0.000 | $\mathcal{TE}$ | 0.00007 | 0.000 | $\mathcal{TE}$ | 0.00007 | 0.001 | RFC | 0.00002 | 0.000 |
| | NLE | 0.00069 | 0.001 | NLE | 0.00006 | 0.01 | NOP | 0.00006 | 0.000 | NL | 0.00002 | 0.014 |
| | NL | 0.00062 | 0.01 | NOP | 0.00005 | 0.012 | NL | 0.00006 | 0.013 | CLOCC | 0.00001 | 0.01 |
| **AAD** | **0.2528** | | | **0.0466** | | | **0.086** | | | **0.0163** | | |
| | Metric | $\beta_{PR}-\beta_{LSI}$ | P | Metric | $\beta_{PR}-\beta_{LSI}$ | P | Metric | $\beta_{PR}-\beta_{LSI}$ | P | Metric | $\beta_{PR}-\beta_{LSI}$ | P |
| **PageRank vs. LSI** | $\mathcal{RFS}$ | 1.96753 | 0.000 | $\mathcal{RFS}$ | 0.269 | 0.000 | $\mathcal{LS}$ | 0.6594 | 0.000 | $\mathcal{RFS}$ | 0.05897 | 0.000 |
| | $\mathcal{ULC}$ | 0.6142 | 0.000 | $\mathcal{ULC}$ | 0.08263 | 0.000 | $\mathcal{RFS}$ | 0.45053 | 0.000 | $\mathcal{ULC}$ | 0.0179 | 0.000 |
| | $\mathcal{LS}$ | 0.28575 | 0.000 | $\mathcal{LS}$ | 0.0391 | 0.000 | $\mathcal{ULC}$ | 0.13871 | 0.000 | $\mathcal{LS}$ | 0.00858 | 0.000 |
| | SigFF | 0.00226 | 0.001 | CDC | 0.00026 | 0.004 | CD | 0.0005 | 0.001 | $\mathcal{C}extitiF$ | 0.00005 | 0.000 |
| | CDC | 0.00217 | 0.000 | CD | 0.00026 | 0.000 | CLOCC | 0.00048 | 0.02 | $\mathcal{QS}$ | 0.00004 | 0.000 |
| | $\mathcal{C}extitiF$ | 0.00145 | 0.000 | $\mathcal{C}extitiF$ | 0.0002 | 0.000 | $\mathcal{C}extitiF$ | 0.00032 | 0.000 | CD | 0.00004 | 0.004 |
| | $\mathcal{QS}$ | 0.00122 | 0.000 | $\mathcal{QS}$ | 0.00018 | 0.000 | $\mathcal{QS}$ | 0.0003 | 0.000 | $\mathcal{TE}$ | 0.00002 | 0.000 |
| | $\mathcal{TE}$ | 0.0007 | 0.000 | $\mathcal{TE}$ | 0.0001 | 0.000 | $\mathcal{TE}$ | 0.00016 | 0.000 | RFC | 0.00002 | 0.000 |
| | NL | 0.00063 | 0.003 | NIIM | 0.00008 | 0.002 | NOP | 0.00013 | 0.006 | NL | 0.00002 | 0.001 |
| | NOP | 0.00057 | 0.006 | LLOCC | 0.00007 | 0.01 | NLE | 0.00013 | 0.008 | CLOCC | 0.00001 | 0.01 |
| **AAD** | **0.2292** | | | **0.0372** | | | **0.068** | | | **0.0124** | | |
| | Metric | $\beta_{PR}-\beta_{VSM}$ | P | Metric | $\beta_{PR}-\beta_{VSM}$ | P | Metric | $\beta_{PR}-\beta_{VSM}$ | P | Metric | $\beta_{PR}-\beta_{VSM}$ | P |
| **PageRank vs. VSM** | $\mathcal{RFS}$ | 1.62 | 0.000 | $\mathcal{RFS}$ | 0.40334 | 0.000 | $\mathcal{RFS}$ | 0.91532 | 0.000 | $\mathcal{RFS}$ | 0.01614 | 0.000 |
| | $\mathcal{ULC}$ | 0.5072 | 0.000 | $\mathcal{ULC}$ | 0.126 | 0.000 | $\mathcal{ULC}$ | 0.28493 | 0.000 | $\mathcal{ULC}$ | 0.00487 | 0.000 |
| | $\mathcal{LS}$ | 0.2361 | 0.000 | CBO | 0.0661 | 0.000 | $\mathcal{LS}$ | 0.1336 | 0.000 | $\mathcal{LS}$ | 0.00227 | 0.000 |
| | CD | 0.00186 | 0.001 | $\mathcal{LS}$ | 0.059 | 0.000 | CD | 0.00105 | 0.000 | $\mathcal{C}extitiF$ | 0.00074 | 0.000 |
| | CLOCC | 0.00179 | 0.009 | CD | 0.00045 | 0.000 | CDC | 0.001 | 0.001 | CD | 0.00003 | 0.000 |
| | $\mathcal{C}extitiF$ | 0.00119 | 0.000 | CDC | 0.00043 | 0.001 | $\mathcal{C}extitiF$ | 0.00066 | 0.000 | CDC | 0.00002 | 0.003 |
| | $\mathcal{QS}$ | 0.00096 | 0.000 | $\mathcal{QS}$ | 0.00031 | 0.000 | $\mathcal{QS}$ | 0.00065 | 0.000 | NL | 0.00002 | 0.000 |
| | $\mathcal{TE}$ | 0.00057 | 0.000 | $\mathcal{C}extitiF$ | 0.00029 | 0.000 | $\mathcal{TE}$ | 0.00032 | 0.000 | $\mathcal{TE}$ | 0.00002 | 0.000 |
| | NLE | 0.00053 | 0.001 | $\mathcal{TE}$ | 0.00014 | 0.000 | NLE | 0.00029 | 0.006 | $\mathcal{QS}$ | 0.00002 | 0.000 |
| | NOP | 0.00047 | 0.005 | NLE | 0.00013 | 0.000 | NOP | 0.00026 | 0.002 | DIT | 0.00001 | 0.006 |
| **AAD** | **0.29** | | | **0.064** | | | **0.1283** | | | **0.0315** | | |
| | Metric | $\beta_{LDA}-\beta_{LSI}$ | P | Metric | $\beta_{LDA}-\beta_{LSI}$ | P | Metric | $\beta_{LDA}-\beta_{LSI}$ | P | Metric | $\beta_{LDA}-\beta_{LSI}$ | P |
| **LDA vs. LSI** | $\mathcal{RFS}$ | 0.2042 | 0.000 | $\mathcal{LS}$ | 0.0926 | 0.000 | $\mathcal{RFS}$ | 0.2454 | 0.000 | $\mathcal{RFS}$ | 0.01447 | 0.000 |
| | $\mathcal{ULC}$ | 0.06347 | 0.000 | $\mathcal{RFS}$ | 0.0614 | 0.000 | $\mathcal{ULC}$ | 0.07678 | 0.000 | $\mathcal{ULC}$ | 0.00459 | 0.000 |
| | $\mathcal{LS}$ | 0.03065 | 0.000 | $\mathcal{ULC}$ | 0.02017 | 0.000 | $\mathcal{LS}$ | 0.03631 | 0.000 | $\mathcal{LS}$ | 0.00223 | 0.000 |
| | CDC | 0.00018 | 0.001 | $\mathcal{QS}$ | 0.0006 | 0.000 | CD | 0.0003 | 0.000 | $\mathcal{QS}$ | 0.00003 | 0.000 |
| | CLOC | 0.00017 | 0.019 | $\mathcal{C}extitiF$ | 0.0004 | 0.000 | CDC | 0.00028 | 0.001 | $\mathcal{C}extitiF$ | 0.00002 | 0.000 |
| | $\mathcal{QS}$ | 0.00016 | 0.000 | $\mathcal{TE}$ | 0.0002 | 0.000 | $\mathcal{QS}$ | 0.00021 | 0.000 | RFC | 0.00002 | 0.000 |
| | $\mathcal{C}extitiF$ | 0.00015 | 0.000 | CDC | 0.00008 | 0.008 | $\mathcal{C}extitiF$ | 0.00018 | 0.000 | NL | 0.00002 | 0.000 |
| | $\mathcal{TE}$ | 0.00007 | 0.000 | CD | 0.00008 | 0.000 | $\mathcal{TE}$ | 0.00009 | 0.000 | CD | 0.00002 | 0.001 |
| | NIIM | 0.00007 | 0.007 | DIT | 0.00007 | 0.000 | NLE | 0.00008 | 0.01 | $\mathcal{TE}$ | 0.00002 | 0.000 |
| | NLE | 0.00006 | 0.000 | NOP | 0.00002 | 0.001 | NOP | 0.00007 | 0.003 | CLOCC | 0.00001 | 0.006 |
| **AAD** | **0.0722** | | | **0.036** | | | **0.073** | | | **0.0177** | | |
| | Metric | $\beta_{LDA}-\beta_{VSM}$ | P | Metric | $\beta_{LDA}-\beta_{VSM}$ | P | Metric | $\beta_{LDA}-\beta_{VSM}$ | P | Metric | $\beta_{LDA}-\beta_{VSM}$ | P |
| **LDA vs. VSM** | $\mathcal{RFS}$ | 0.554 | 0.000 | $\mathcal{RFS}$ | 0.19621 | 0.000 | $\mathcal{RFS}$ | 0.7102 | 0.000 | $\mathcal{RFS}$ | 0.06064 | 0.000 |
| | $\mathcal{ULC}$ | 0.17031 | 0.000 | $\mathcal{ULC}$ | 0.06344 | 0.000 | $\mathcal{ULC}$ | 0.223 | 0.000 | $\mathcal{ULC}$ | 0.01817 | 0.000 |
| | $\mathcal{LS}$ | 0.08031 | 0.000 | $\mathcal{LS}$ | 0.0292 | 0.000 | $\mathcal{LS}$ | 0.10397 | 0.000 | $\mathcal{LS}$ | 0.00862 | 0.000 |
| | CDC | 0.0006 | 0.000 | CD | 0.00027 | 0.001 | CLOC | 0.00084 | 0.022 | NL | 0.00029 | 0.01 |
| | CD | 0.00057 | 0.02 | CDC | 0.00025 | 0.008 | $\mathcal{QS}$ | 0.00056 | 0.000 | $\mathcal{C}extitiF$ | 0.00005 | 0.000 |
| | $\mathcal{QS}$ | 0.00043 | 0.000 | $\mathcal{QS}$ | 0.00019 | 0.000 | $\mathcal{C}extitiF$ | 0.00052 | 0.000 | CDC | 0.00005 | 0.000 |
| | $\mathcal{C}extitiF$ | 0.00041 | 0.000 | $\mathcal{C}extitiF$ | 0.00014 | 0.000 | $\mathcal{TE}$ | 0.00025 | 0.000 | CD | 0.00005 | 0.000 |
| | $\mathcal{TE}$ | 0.0002 | 0.000 | $\mathcal{TE}$ | 0.00007 | 0.000 | NIIM | 0.00023 | 0.001 | $\mathcal{TE}$ | 0.00002 | 0.000 |
| | NLE | 0.00017 | 0.000 | NLE | 0.00007 | 0.001 | NLE | 0.0002 | 0.000 | $\mathcal{QS}$ | 0.00002 | 0.000 |
| | NL | 0.00015 | 0.001 | NOP | 0.00006 | 0.01 | NOA | 0.00018 | 0.021 | DIT | 0.00001 | 0.01 |
| **AAD** | **0.1231** | | | **0.0756** | | | **0.1335** | | | **0.0393** | | |
| | Metric | $\beta_{VSM}-\beta_{LSI}$ | P | Metric | $\beta_{VSM}-\beta_{LSI}$ | P | Metric | $\beta_{VSM}-\beta_{LSI}$ | P | Metric | $\beta_{VSM}-\beta_{LSI}$ | P |
| **LSI vs. VSM** | $\mathcal{RFS}$ | 0.35 | 0.000 | $\mathcal{RFS}$ | 0.135 | 0.000 | $\mathcal{RFS}$ | 0.465 | 0.000 | $\mathcal{RFS}$ | 0.075 | 0.000 |
| | $\mathcal{ULC}$ | 0.11 | 0.000 | $\mathcal{ULC}$ | 0.043 | 0.000 | $\mathcal{ULC}$ | 0.15 | 0.000 | $\mathcal{ULC}$ | 0.0228 | 0.000 |
| | $\mathcal{LS}$ | 0.05 | 0.000 | $\mathcal{LS}$ | 0.02 | 0.000 | $\mathcal{LS}$ | 0.068 | 0.000 | $\mathcal{LS}$ | 0.011 | 0.000 |
| | $\mathcal{QS}$ | 0.00027 | 0.000 | CDC | 0.00018 | 0.015 | CDC | 0.00052 | 0.029 | RFC | 0.00007 | 0.004 |
| | $\mathcal{C}extitiF$ | 0.00026 | 0.000 | $\mathcal{QS}$ | 0.00013 | 0.000 | $\mathcal{QS}$ | 0.00035 | 0.000 | SigFF | 0.00007 | 0.000 |
| | $\mathcal{TE}$ | 0.00013 | 0.000 | $\mathcal{C}extitiF$ | 0.0001 | 0.000 | $\mathcal{C}extitiF$ | 0.00034 | 0.000 | $\mathcal{C}extitiF$ | 0.00006 | 0.000 |
| | NLE | 0.0001 | 0.022 | $\mathcal{TE}$ | 0.00005 | 0.000 | NLE | 0.00015 | 0.000 | $\mathcal{QS}$ | 0.00003 | 0.000 |
| | NL | 0.00009 | 0.024 | NOP | 0.00005 | 0.039 | NL | 0.00013 | 0.000 | $\mathcal{TE}$ | 0.00003 | 0.000 |
| | NumPAR | 0.00007 | 0.000 | NLE | 0.00005 | 0.000 | DIT | 0.00012 | 0.000 | NL | 0.00002 | 0.000 |
| | NOIM | 0.00004 | 0.035 | NL | 0.00004 | 0.000 | $\mathcal{TE}$ | 0.0001 | 0.000 | McCC | 0.00002 | 0.000 |
| **AAD** | **0.0951** | | | **0.0574** | | | **0.0962** | | | **0.0305** | | |

in combination with the main regressor variable which explains enhancement ($\mathcal{CIF}$ in the case of MRR), $\mathcal{QS}$ explains the antagonistic side [18], [19] regarding the portion of the effect not explained by the main regressor.

Similar to $\mathcal{QS}$, $\mathcal{TE}$, and $\mathcal{LS}$ explain a small portion of the FLTs' reducing performance when they appear. This suggests that, as expected, tangling and vocabulary replication may help explain the antagonistic part of the relationship, in combination with the main regressor(s).

Another important aspect highlighted in Table 7 is that the pattern across techniques (PageRank, LDA, LSI, VSM) is highly consistent, particularly concerning their major sources of variance. This suggests the metrics' efficacy with respect to the measurement methods. Overall, the results show that our feature metric suite controls the performance of FLTs more than the other code metrics employed.

The results in Table 8 also suggest that the information captured by the metrics may not be very unique/has high repetition. To assess this we moved beyond the default implementation of SPSS, where a new component is only selected if its Eigen value is greater than one. Instead, factors were kept fixed, with eight different components. But, in that analysis, the unique variance was still limited, other than in the initial two components, which explained 98.56% variance in total. It seems that the metrics studied are best clustered only based on a textual-IR dimension and on another code-structural dimension, the latter containing characteristics like Number of Children (NOCh).

***Top Moderators*** – Overall, of the evaluation measures presented in Table 11, $\mathcal{RFS}$, $\mathcal{ULC}$, and $\mathcal{LS}$ are found to be the strongest metrics that moderate the relative performance of different FLTs. It is interesting that all of these metrics measure the relative size of features with respect to the system in terms of the number of code elements, unique lexicon, and corpus, respectively. This suggests that the performance difference between FLTs can be significantly impacted simply by altering feature size. It is also interesting to note that while two of these measures are based on lexicons and might be expected to impact on IR-based FLTs, the third is not: it is purely based on the numbers of code elements. Intuitively though, a larger $\mathcal{RFS}$ suggests a larger number of lexicons and so this may explain the impact.

As an example, consider the difference between PageRank and LDA techniques in terms of MRR where the moderation coefficient of $\mathcal{RFS}$ is 2.1717. This means, for every one-unit (i.e. value 1) change in the $\mathcal{RFS}$ value of the feature sets, the performance of PageRank and LDA differ by 2.1717 in terms of MRR [66]. More concretely, if the to-be-located feature is composed of 10% of the methods in a system (i.e. 0.1 $\mathcal{RFS}$), then LDA would be expected to outperform PageRank, in terms of MRR score, by 0.21717 based purely on that $\mathcal{RFS}$ value. That is, if PageRank identifies the top feature-related element at $40^{th}$ position (0.025 MRR), then LDA could have an MRR score of 0.025 + 0.21717 = 0.24217, implying the top feature-related element would be in $4^{th}$ position in its list. Of course, the effect is not always that dramatic: if some other feature comprises of 25% of the system elements (i.e. the $\mathcal{RFS}$ is 0.25), and if the better-performing technique identifies the first feature-related element at the $1^{st}$ position on the list (i.e. MRR score 1), the other technique would be expected to identify the first feature-related element at the $2^{nd}$ position in the list: that is with a MRR score 1 - (2.1717/4) = 0.4570. While we anticipate that these high values of $\mathcal{RFS}$ are not typical, the examples illustrate the potential impact of the metric suite across FLT approaches for different evaluation metrics.

Note also that the complete feature metric suite proposed is selected in the Top 10 moderators, in all cases. This further validates the effect of the metric suite, as a set of characteristics causing performance differences over FLT performance measures. However, characteristics may also have internal interactions (i.e. multiplicative moderation [18], [19]), which causes the characteristics in our metric suite to differently effect the performance across FLTs and the performance differences between FLTs.

### 6.2 Practice Guide and Recommendations

Table 12 summarizes the findings garnered from this study. The left-hand-side of the table, where evaluation measures are mapped to the techniques, summarizes FLTs' performance with respect to feature characteristics, whereas the right-hand-side of the table summarizes the performance difference between pairs of FLTs with respect to the same characteristics. An '↑' symbol in the first part indicates the main regressor: where the characteristic, as quantified by our metrics, causes the maximum performance of a technique. A '+' or '-' symbol shows that the characteristic seems to combine with the main regressor to explain an enhancing or antagonistic effect on the performance of the FLT measure, respectively. An empty cell reflects no impact.

An '↑' symbol in the second part of the table shows the main moderators with respect to different FLTs: The characteristics that cause the major performance difference between pairs of FLTs. Additionally a '|' symbol shows that the respective characteristics may impact FLTs' performance to a small degree. Based on the findings in Table 12, the following guidance for the community is suggested:

1) **Empirical Design Guidance for Researchers (1):** It is clear from the table that precision and MAP are mostly impacted by $\mathcal{RFS}$, recall is impacted by $\mathcal{ULC}$ and MRR, in the IR-based techniques, is impacted by $\mathcal{CIF}$, whereas in the hybrid-technique it is impacted by $\mathcal{ULC}$. This should guide researchers' empirical designs: If they select feature sets that have high $\mathcal{RFS}$, then they can expect higher precision and MAP and should acknowledge that expectation explicitly in their papers. Likewise, if that feature set has low $\mathcal{ULC}$, then they can expect lower recall, and should declare that also. Reversing this, if a researcher is aiming for (for eg) high precision in their study, they should ensure that they employ at least two feature sets: one with a high $\mathcal{RFS}$ and one with a low $\mathcal{RFS}$. More generally, it suggests that researchers should select a 'balanced, fair' feature set or a number of feature sets that have ranges of $\mathcal{RFS}$, $\mathcal{ULC}$, and $\mathcal{CIF}$ values, to report on how well their FLTs work in specific metric-contexts, based on the evaluation measure chosen. (The 'balanced, fair feature set' idea is returned to on the next page).
2) **Empirical Design Guidance for Researchers doing comparison studies (2):** Additionally, if re-

TABLE 12
Summary of the Findings for Practice Guide

| | Precision | | | | Recall | | | | MAP | | | | MRR | | | | Performance Difference in Pairs of FLTs | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PR | LDA | LSI | VSM | PR | LDA | LSI | VSM | PR | LDA | LSI | VSM | PR | LDA | LSI | VSM | PR-LDA | PR-LSI | PR-VSM | LDA-LSI | LDA-VSM | VSM-LSI |
| **RFS** | ↑ | ↑ | ↑ | ↑ | | | + | | ↑ | ↑ | ↑ | ↑ | + | | | | ↑ | ↑ | ↑ | ↑ | ↑ | ↑ |
| **ULC** | + | + | + | + | ↑ | ↑ | ↑ | ↑ | | | | | ↑ | + | + | + | ↑ | ↑ | ↑ | ↑ | ↑ | ↑ |
| **CiF** | | | | | + | + | + | + | | | | | + | ↑ | ↑ | ↑ | \| | \| | \| | \| | \| | \| |
| **LS** | | | | | - | - | - | - | | | | | - | - | - | - | ↑ | ↑ | ↑ | ↑ | ↑ | ↑ |
| **TE** | | | | - | - | - | - | - | | | | | - | | | - | \| | \| | \| | \| | \| | \| |
| **QS** | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | \| | \| | \| | \| | \| | \| |

searchers' empirical studies compare different FLTs, they should be cognisant of the relative impact of the metrics on different FLTs and the impact that may have on their comparison results. Consequently, they should declare their feature sets' associated feature characteristics (especially $\mathcal{RFS}$, $\mathcal{ULC}$, and $\mathcal{LS}$), ideally selecting a number of feature sets that have a range of feature characteristics.

3) **Guidance for Article Reviewers:** When reviewing a paper, reviewers should be aware of the potential impact of feature characteristics on the results obtained. They should require a feature-set profile to be included in the empirical design section and resultant consideration of these feature characteristics in the results-discussion section. This is particularly important in comparison studies where different feature characteristics may result in two, otherwise identical, studies returning quite different results.
4) **FLT Selection for Practitioners:** As new results emerge from the literature, practitioners will be in a better place to select appropriate FLTs for their work contexts: If they know the characteristics of their systems' feature sets (possibly through analysis of past feature enhancement commits), and are aware of studies that evaluate the relative performance of differing FLTs, given different feature characteristics, then they should be better placed to identify the most appropriate FLT for their specific context. For example, Table 12 suggests that, if a system has a large $\mathcal{RFS}$, PR should be considered first.

Figure 5 illustrates the analysis process performed in this study, i.e. evaluation of the FLTs to identify the characteristics either causing the performance across or between FLTs. Each different goal of an FLT evaluation, as measured by precision, recall, MRR, or MAP, may be prone to a different set of such characteristics. As an example, the centre of Figure 5 shows a partial profile of feature characteristics affecting performance across-and-between FLTs for MRR (see Table 7 and Table 11 for a complete profile, respectively). As time goes by, the research community can work to expand the profile adding more characteristics (i.e. new columns in these tables). Likewise, as new FLTs proposed by researchers are assessed cognisant of the feature characteristics, additional rows can be added to these profiles.

Using this framework a mapping between features' characteristics and FLTs/evaluation goals can be leveraged towards recommending FLTs or feature-sets for evaluation:

1) **Guidance towards FLT Recommendation** – The profile of the performance-causing characteristics records two sets of information: 1) which set of characteristics impact the performance of an FLT and, 2) how much a set of these characteristics cause different FLTs' performance. A developed profile of such characteristics can act as a recommendation system for developers who may be interested in determining which FLT is appropriate for their software-system based, perhaps, on historical bug repositories that can act as proxies for feature sets. Figure 5, on the left-hand-side, demonstrates this scenario of FLT recommendation. A given feature-set can be first assessed for its feature characteristics. Then, identified characteristics could be mapped to the FLTs, and an FLT that performs better on such characteristics can be recommended. For example, given MRR as a goal and a feature-set with high $\mathcal{CiF}$ value, the system may recommend LDA as the appropriate technique (given the tables in Figure 5).
2) **Guidance towards Understanding Conflicting Findings** – The profiling of performance differences between feature characteristics keeps a record of how much each pair of techniques differ in their performance, given an impactful characteristic. Hence, a documented profile of such characteristics can be leveraged towards understanding some of the performance difference between a pair of FLTs. The right-hand-side of Figure 5, illustrates how this can happen. For example, given PageRank and LDA as a pair of FLTs compared using the feature set employed in this study, with each unit change in $\mathcal{RFS}$, they will differ in their relative performance by 2.17 for MRR-based evaluation.
3) **Guidance towards Curating a Fair and Balanced Evaluation Benchmark** – An important insight that follows from this paper is that researchers should strive towards a fair, balanced benchmark for FLT evaluation, in terms of feature-set characteristics. By fair and balanced we imply: 'representative of' the feature sets faced by researchers and practitioners in the software systems they study and maintain. But the only way this could reasonably be achieved would be to analyse many (all) features over many (all) systems and use that data to derive a repre-

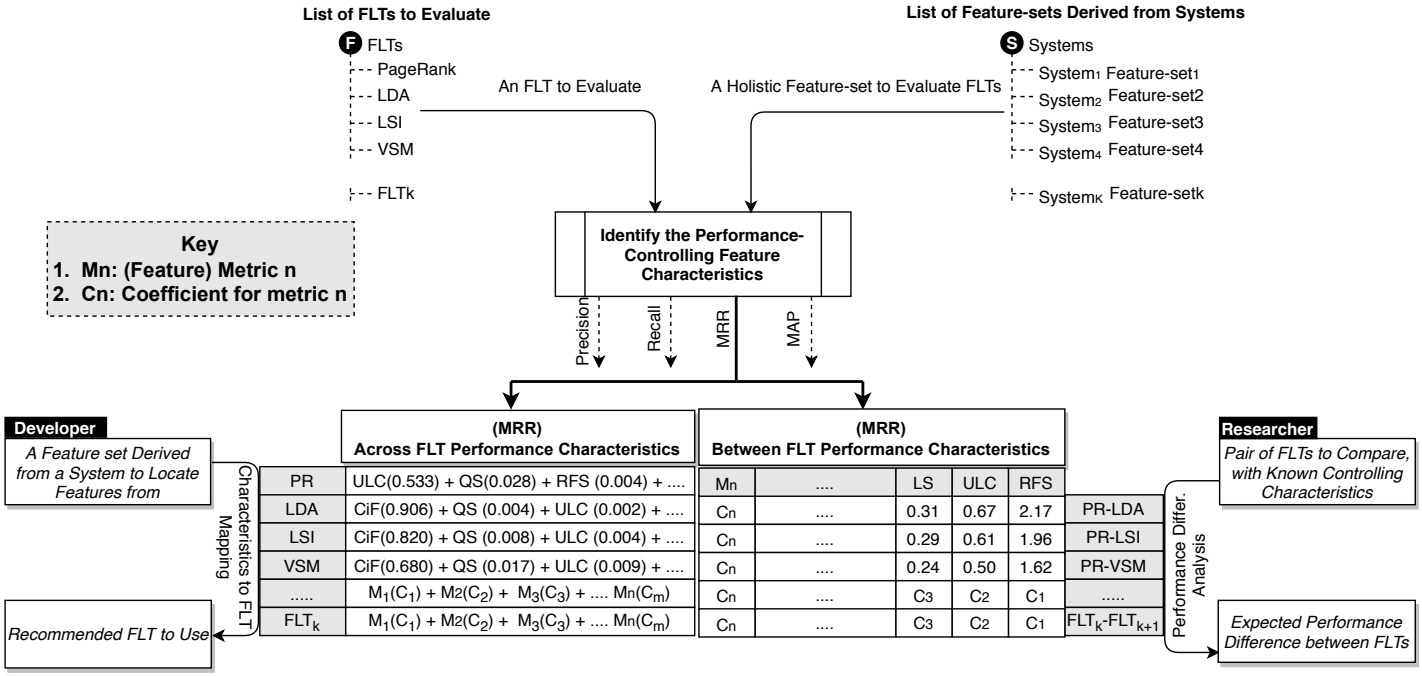


Fig. 5. FLT and Feature Set Recommendation Framework

sentative feature set, in terms of its feature-metric values. Such a 'Holistic Feature set' is presented to the top right-hand-side of Figure 5.

The work reported on here can be contextualized as beginning that process towards a fair and balanced benchmark: By calculating the feature-characteristic metrics for each system employed in the empirical study, a profile for each system (and across systems) can be generated for impact-ful feature metrics. Table 13 presents this profiling of our dataset based on the metric suite presented. Researchers may build on these profiles with their own datasets[4] but should also consider the following factors:

a) Change in individual FLT performance, based on changes in feature metrics' values;
b) Performance differences between pairs of FLTs, caused by an increase or decrease in those metrics' values;
c) That the individual systems in Table 13 vary widely in terms of their feature metrics and so a representative benchmark may be of lesser utility for practitioners who have to deal with specific, individual systems.

## 7 Threats to Validity

As with any empirical study, a number of threats to validity exist and these are discussed in terms of internal, construct and external validity [67].

*Internal Validity* – There could be other potential moderators that would have to be controlled to explain part of the variance in both research questions. For example, the 41 software metrics that made up the second component of our analysis explains less variance, but NL and NLE were found to have a small moderation effect. Likewise, metrics other than code metrics (e.g. process-level metrics [57]) may also impact on FLTs performance. However, the inclusion of such metrics does not nullify the validity of the impact of our metric suite, as determined here.

In a few cases, performance measurement data were non-normally distributed. However, in multiple regression analysis, collinearity and heteroscedasticity are more important than normality of data [55], [58], and these have been addressed in this study. Also, the large feature set, containing 878 features, lessens the limitation caused by non-normality of data: Finch et al. [68] suggested a negligible impact of non-normal data on parameter estimation in structural equation models when a large sample size is employed: As suggested by Green [69], a sample size of 427 or more is required for the 47 metrics.

The strength and significance of the relationships between our metric suite and the performance of FLTs are ultimately dependent on the quality of the feature sets used to determine FLTs performance scores. But absolutely correct and complete feature sets in FL are nearly impossible to come by, apart maybe from macro-based feature-sets. We used these macro-based feature-sets where possible [70] and, for nearly half of the features (468 out of 878) employed in this study, triangulation was used in their initial creation to buttress their quality [7]. The feature sets for the remaining portion of the features are created solely from re-enactment data-sets; however, these feature sets are the most commonly used ones in FLTs evaluation [3].

Finally, we employed feature-level aggregation of the source code metrics (e.g. aggregation of the LOCs) for all feature-related elements, when extrapolating the presented metric suite. It is an open question as to whether this is the

4. Please submit your feature set at: http://metriccalculator.lero.ie/FeatureMetricCalculator/

TABLE 13
Holistic and By-system Profile of the Metric Suite

| | Impactful Metric | | | | | |
|---|---|---|---|---|---|---|
| | **RFS** | **ULC** | **CiF** | **LS** | **TE** | **QS** |
| **Holistic** | 0.091 | 0.321 | 123.5 | 0.638 | 255.8 | 189.2 |
| **Rhino** | 0.237 | 0.785 | 326.2 | 1.639 | 665.1 | 379.1 |
| **ArgoSPL** | 0.031 | 0.228 | 5.826 | 0.242 | 211.1 | 219.6 |
| **jEdit** | 0.001 | 0.028 | 1.667 | 0.013 | 1.633 | 31.21 |
| **JabRef** | 0.001 | 0.049 | 1.231 | 0.032 | 1.154 | 37.31 |
| **ArgoUML** | 0.001 | 0.023 | 0.945 | 0.008 | 2.055 | 37.21 |
| **muCoammnder** | 0.001 | 0.039 | 1.631 | 0.016 | 1.478 | 29.31 |
| **Mylyn** | 0.015 | 0.108 | 0.401 | 0.144 | 0.401 | 263.6 |
| **iBatis** | 0.007 | 0.052 | 9.412 | 0.095 | 11.424 | 195.1 |
| **Eclipse** | 0.001 | 0.005 | 0.089 | 0.001 | 0.133 | 30.31 |

correct approach. For example, using the average LOC of the elements implementing a feature might impact the results.

*Construct Validity* – The four evaluation measures that are used, while considered best of breed, do not capture all facets of FLT-evaluation. For example, FLTs could be evaluated for usefulness, usability, efficiency, and/or performance: The measures employed here focus on performance only. We selected performance because that facet of evaluation is predominant in FL literature, employed in 86% of the studies which undertook some form of evaluation [3].

*External Validity* – The generalization of the experimental effect is always an issue if the experiment is designed without creating strong counterfactual inferences and/or without including the maximum possible relevant extraneous variables (to remove other plausible explanations). We employed 47 metrics in total, combined with counterfactuals including four different FLTs (in research question 1) and four evaluation measures (in research question 2). However, to a certain degree, the causal relationships built in this study are limited by the systems employed in that they are limited in number, older in age, open-source and implemented only in Java. Notwithstanding, the selected systems are the most used in the feature location field and, in the future, we are committed to expanding our experiments to systems written in other programming languages.

Another external-validity limitation is the type of FLTs assessed. That is, only three IR-based and one structural-combined-with-IR techniques are employed. Other types of FLTs may likely be impacted by other feature characteristics. However, this research proactively targeted these best-of-breed [32] and state-of-the-art [14] FLTs, in an effort to drive more encompassing comparability across FLT research.

Finally, of course, this is just the first study on feature characteristics and their effect on FLTs. Further work is needed to identify additional metrics of interest, not just for the FLTs studied here, but for techniques in related fields and more advanced FLTs, performing more sophisticated analyses. For example, several techniques in the related field of bug localization employ sample data to derive heuristics that, in turn, can be used to identify code elements that need to be changed [71]. We contend that that sample data should be selected cognizant of the impactful feature metrics presented here. Similarly, several novel techniques employ machine learning approaches to dynamically build vectors that can be used in locating feature-related code [72]. In such cases, already known metrics values could be used to assess the quality of the vectors dynamically selected.

## 8 Conclusions and Future Work

The vast numbers of FLTs proposed by the research community impose difficulties for practitioners and researchers when trying to decide on the appropriate technique to employ. These difficulties have been aggravated by the conflicting findings across studies that attempt to cross-compare FLTs and the inconsistent empirical designs that prohibit generalizing across the results. In this paper, we argue that only by assessing the FLTs performance, cognisant of features' characteristics, will we allow practitioners to select the appropriate FLTs for given system contexts and researchers to design trusted feature sets which provide coverage over relevant feature characteristics. Towards this agenda, this paper presents a feature-metric suite that contains six new feature-set metrics. These were tested, in combination with 41 existing software metrics, in order to assess the impact of the underlying feature characteristics on FLT performance.

This paper empirically evaluated the potential impact of the presented metric suite, employing a set of 878 features, using a controlled experiment. The control relation in the experiment was established using extensive extrapolation and counterfactual inferences. Results of this study suggest that our metric suite 1) explains major variance in the performance of FLTs using established evaluation measures and 2) strongly impacts the relative performance differences between different FLTs. The following findings have been garnered from this work:

1) It illustrates the effect of features characteristics on FLTs' performance and suggests that, based on these characteristics, the feature sets employed in FLTs evaluation differ in their affinity to different FLTs. Thus, reviewers of the literature need to be aware of this effect when evaluating FLTs compared using different feature sets. To that end, we have provided a website: http://metriccalculator.lero.ie/FeatureMetricCalculator/ to allow researchers to submit a feature set and generate a feature set profile, in terms of the metrics presented here. This site also contains the metric profile for all the feature sets used in this study. Using this, researchers should be better able to critique conflicting results in FLTs' comparisons [6], [8], [14], that may occur due to different feature sets.
2) The presented metric suite allows the cognisant-selection of features for FLT evaluation. Research in this direction would initially allow the creation of features sets, which provide high and low-value coverage for relevant feature characteristics.
3) Subsequent research might focus on the creation of a fair and balanced benchmark. But such a benchmark may be of lesser utility given the wide spread of feature-metric values in different systems. For example, practitioners who are interested in determining the best FLT for their system will want to identify FLTs that work best on systems with a

similar feature-metric profile. If a balanced benchmark differs significantly, then studies using that benchmark are of lesser utility to that practitioner.

4) More generically, this research moves towards standardizing the empirical design for FLTs evaluation by exposing a third variable (feature characteristics) as a moderator which can control FLTs' performance. As Dit et al. note [5], systems-tested-against is a core component in the empirical evaluation of FLTs and studies like this can also be employed to test confounding effects [18], [19] of other characteristics (e.g. process-level metrics [21], [51], [52]).

In the future, we are planning to enhance our feature metric suite with static [28], [56], dynamic [1] and process-level [21], [51], [52] feature metrics for static, dynamic and historical analysis on systems written in different languages. In addition, we did not consider the relationships between metrics here. Hence, in the future, we want to assess the impact of multiple moderator (feature characteristics) interactions through multiplicative moderation [18], [19].

## ACKNOWLEDGMENTS

This work was financially supported by Science Foundation Ireland grant 13/RC/2094 and co-funded under the European Regional Development Fund through the Southern Eastern Regional Operational Programme to Lero - the Irish Software Research Centre (www.lero.ie)

## REFERENCES

[1] B. Cornelissen, A. Zaidman, A. Van Deursen, L. Moonen, and R. Koschke, "A systematic survey of program comprehension through dynamic analysis," *IEEE Transactions on Software Engineering*, vol. 35, no. 5, pp. 684–702, 2009.

[2] M. P. Robillard, W. Coelho, and G. C. Murphy, "How effective developers investigate source code: An exploratory study," *IEEE Transactions on software engineering*, vol. 30, no. 12, pp. 889–903, 2004.

[3] W. A. E. C. Razzaq, Abdul and B. Jim, "The state of empirical evaluation in static feature location," *Transaction on Software Engineering Methodology*, vol. 28, no. 1, p. TO APPEAR, 2019.

[4] T. Eisenbarth, R. Koschke, and D. Simon, "Locating features in source code," *IEEE Transactions on software engineering*, vol. 29, no. 3, pp. 210–224, 2003.

[5] B. Dit, M. Revelle, M. Gethers, and D. Poshyvanyk, "Feature location in source code: a taxonomy and survey," *Journal of software: Evolution and Process*, vol. 25, no. 1, pp. 53–95, 2013.

[6] S. W. Thomas, M. Nagappan, D. Blostein, and A. E. Hassan, "The impact of classifier configuration and classifier combination on bug localization," *IEEE Transactions on Software Engineering*, vol. 39, no. 10, pp. 1427–1443, 2013.

[7] A. Marcus and J. I. Maletic, "Recovering documentation-to-source-code traceability links using latent semantic indexing," in *Proceedings of the 25th international conference on software engineering*. IEEE Computer Society, 2003, pp. 125–135.

[8] D. Binkley, D. Lawrie, C. Uehlinger, and D. Heinz, "Enabling improved ir-based feature location," *Journal of Systems and Software*, vol. 101, pp. 30–42, 2015.

[9] D. Poshyvanyk, A. Marcus, R. Ferenc, and T. Gyimóthy, "Using information retrieval based coupling measures for impact analysis," *Empirical software engineering*, vol. 14, no. 1, pp. 5–32, 2009.

[10] M. Revelle, M. Gethers, and D. Poshyvanyk, "Using structural and textual information to capture feature coupling in object-oriented software," *Empirical software engineering*, vol. 16, no. 6, pp. 773–811, 2011.

[11] X. Ye, R. Bunescu, and C. Liu, "Mapping bug reports to relevant files: A ranking model, a fine-grained benchmark, and feature evaluation," *IEEE Transactions on Software Engineering*, vol. 42, no. 4, pp. 379–402, 2016.

[12] L. R. Biggers, C. Bocovich, R. Capshaw, B. P. Eddy, L. H. Etzkorn, and N. A. Kraft, "Configuring latent dirichlet allocation based feature location," *Empirical Software Engineering*, vol. 19, no. 3, pp. 465–500, 2014.

[13] B. Dit, *Configuring and Assembling Information Retrieval based Solutions for Software Engineering Tasks*. The College of William and Mary, 2015.

[14] B. Dit, M. Revelle, and D. Poshyvanyk, "Integrating information retrieval, execution and link analysis algorithms to improve feature location in software," *Empirical Software Engineering*, vol. 18, no. 2, pp. 277–309, 2013.

[15] C. Mills, G. Bavota, S. Haiduc, R. Oliveto, A. Marcus, and A. D. Lucia, "Predicting query quality for applications of text retrieval to software engineering tasks," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 26, no. 1, p. 3, 2017.

[16] L. Moreno, G. Bavota, S. Haiduc, M. Di Penta, R. Oliveto, B. Russo, and A. Marcus, "Query-based configuration of text retrieval solutions for software engineering tasks," in *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*. ACM, 2015, pp. 567–578.

[17] C. M. Judd, G. H. McClelland, and E. R. Smith, "Testing treatment by covariate interactions when treatment varies within subjects." *Psychological Methods*, vol. 1, no. 4, p. 366, 1996.

[18] C. M. Judd, D. A. Kenny, and G. H. McClelland, "Estimating and testing mediation and moderation in within-subject designs." *Psychological methods*, vol. 6, no. 2, p. 115, 2001.

[19] J. H. Bolin, "Hayes, andrew f.(2013). introduction to mediation, moderation, and conditional process analysis: A regression-based approach. new york, ny: The guilford press," *Journal of Educational Measurement*, vol. 51, no. 3, pp. 335–337, 2014.

[20] L. C. Briand, J. W. Daly, and J. Wüst, "A unified framework for cohesion measurement in object-oriented systems," *Empirical Software Engineering*, vol. 3, no. 1, pp. 65–117, 1998.

[21] B. Kitchenham, "Whats up with software metrics?–a preliminary mapping study," *Journal of systems and software*, vol. 83, no. 1, pp. 37–51, 2010.

[22] H. Kagdi, M. Gethers, and D. Poshyvanyk, "Integrating conceptual and logical couplings for change impact analysis in software," *Empirical Software Engineering*, vol. 18, no. 5, pp. 933–969, 2013.

[23] B. Li, X. Sun, and H. Leung, "Combining concept lattice with call graph for impact analysis," *Advances in Engineering Software*, vol. 53, pp. 1–13, 2012.

[24] B. P. Eddy, N. A. Kraft, and J. Gray, "Impact of structural weighting on a latent dirichlet allocation–based feature location technique," *Journal of Software: Evolution and Process*, vol. 30, no. 1, p. e1892, 2018.

[25] S. Wang and D. Lo, "Version history, similar report, and structure: Putting them together for improved bug localization," in *Proceedings of the 22nd International Conference on Program Comprehension*. ACM, 2014, pp. 53–63.

[26] D. Kim, Y. Tao, S. Kim, and A. Zeller, "Where should we fix this bug? a two-phase recommendation model," *IEEE transactions on software Engineering*, vol. 39, no. 11, pp. 1597–1610, 2013.

[27] M. Gethers, R. Oliveto, D. Poshyvanyk, and A. De Lucia, "On integrating orthogonal information retrieval methods to improve traceability recovery," in *Software Maintenance (ICSM), 2011 27th IEEE International Conference on*. IEEE, 2011, pp. 133–142.

[28] C. Kästner, A. Dreiling, and K. Ostermann, "Variability mining: Consistent semi-automatic detection of product-line features," *IEEE Transactions on Software Engineering*, vol. 40, no. 1, pp. 67–82, 2014.

[29] A. Panichella, B. Dit, R. Oliveto, M. Di Penta, D. Poshyvanyk, and A. De Lucia, "Parameterizing and assembling ir-based solutions for se tasks using genetic algorithms," in *Software Analysis, Evolution, and Reengineering (SANER), 2016 IEEE 23rd International Conference on*, vol. 1. IEEE, 2016, pp. 314–325.

[30] G. Antoniol, G. Canfora, G. Casazza, A. De Lucia, and E. Merlo, "Recovering traceability links between code and documentation," *IEEE transactions on software engineering*, vol. 28, no. 10, pp. 970–983, 2002.

[31] D. Poshyvanyk, M. Gethers, and A. Marcus, "Concept location using formal concept analysis and information retrieval," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 21, no. 4, p. 23, 2012.

[32] A. Razzaq, A. Le Gear, C. Exton, and J. Buckley, "An empirical assessment of baseline feature location techniques," *Empirical Software Engineering*, pp. 1–56, 2019.

[33] J. Lee, D. Kim, T. F. Bissyandé, W. Jung, and Y. Le Traon, "Bench4bl: reproducibility study on the performance of ir-based bug localization," in *Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis*. ACM, 2018, pp. 61–72.

[34] Z. Shi, J. Keung, and Q. Song, "An empirical study of bm25 and bm25f based feature location techniques," in *Proceedings of the International Workshop on Innovative Software Development Methodologies and Practices*. ACM, 2014, pp. 106–114.

[35] C. S. Corley, K. L. Kashuda, and N. A. Kraft, "Modeling changeset topics for feature location," in *Software Maintenance and Evolution (ICSME), 2015 IEEE International Conference on*. IEEE, 2015, pp. 71–80.

[36] C. S. Corley, K. Damevski, and N. A. Kraft, "Exploring the use of deep learning for feature location," in *2015 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2015, pp. 556–560.

[37] A. Mahmoud and G. Bradshaw, "Estimating semantic relatedness in source code," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 25, no. 1, p. 10, 2015.

[38] A. Mahmoud and N. Niu, "On the role of semantics in automated requirements tracing," *Requirements Engineering*, vol. 20, no. 3, pp. 281–300, 2015.

[39] L. C. Briand, S. Morasca, and V. R. Basili, "Property-based software engineering measurement," *IEEE transactions on software Engineering*, vol. 22, no. 1, pp. 68–86, 1996.

[40] A. Razzaq and R. Abbasi, "Automated separation of crosscutting concerns: earlier automated identification and modularization of cross-cutting features at analysis phase," in *2012 15th International Multitopic Conference (INMIC)*. IEEE, 2012, pp. 471–478.

[41] M. Eaddy, T. Zimmermann, K. D. Sherwood, V. Garg, G. C. Murphy, N. Nagappan, and A. V. Aho, "Do crosscutting concerns cause defects?" *IEEE transactions on Software Engineering*, vol. 34, no. 4, pp. 497–515, 2008.

[42] M. P. O'Brien, J. Buckley, and T. M. Shaft, "Expectation-based, inference-based, and bottom-up software comprehension," *Journal of Software Maintenance and Evolution: Research and Practice*, vol. 16, no. 6, pp. 427–447, 2004.

[43] E. Figueiredo, C. Sant'Anna, A. Garcia, T. T. Bartolomei, W. Cazzola, and A. Marchetto, "On the maintainability of aspect-oriented software: A concern-oriented measurement framework," in *2008 12th European Conference on Software Maintenance and Reengineering*. IEEE, 2008, pp. 183–192.

[44] T. D. Cook, D. T. Campbell, and W. Shadish, *Experimental and quasi-experimental designs for generalized causal inference*. Houghton Mifflin Boston, MA, 2002.

[45] S. Easterbrook, J. Singer, M.-A. Storey, and D. Damian, "Selecting empirical methods for software engineering research," in *Guide to advanced empirical software engineering*. Springer, 2008, pp. 285–311.

[46] G. Scanniello, A. Marcus, and D. Pascale, "Link analysis algorithms for static concept location: an empirical assessment," *Empirical Software Engineering*, vol. 20, no. 6, pp. 1666–1720, 2015.

[47] Y. Shin, J. H. Hayes, and J. Cleland-Huang, "A framework for evaluating traceability benchmark metrics," 2012.

[48] S. Wang, D. Lo, and J. Lawall, "Compositional vector space models for improved bug localization," in *Software Maintenance and Evolution (ICSME), 2014 IEEE International Conference on*. IEEE, 2014, pp. 171–180.

[49] M. Mogstad and A. Torgovitsky, "Identification and extrapolation of causal effects with instrumental variables," *Annual Review of Economics*, vol. 10, pp. 577–613, 2018.

[50] H. J. Seltman, "Experimental design and analysis," *Online at: http://www. stat. cmu. edu/, hseltman/309/Book/Book. pdf*, 2012.

[51] N. E. Fenton and M. Neil, "Software metrics: roadmap," in *Proceedings of the Conference on the Future of Software Engineering*. ACM, 2000, pp. 357–370.

[52] D. Radjenović, M. Heričko, R. Torkar, and A. Živkovič, "Software fault prediction metrics: A systematic literature review," *Information and software technology*, vol. 55, no. 8, pp. 1397–1418, 2013.

[53] L. C. Briand, J. W. Daly, and J. K. Wust, "A unified framework for coupling measurement in object-oriented systems," *IEEE Transactions on software Engineering*, vol. 25, no. 1, pp. 91–121, 1999.

[54] L. C. Briand, J. Wüst, J. W. Daly, and D. V. Porter, "Exploring the relationships between design measures and software quality in object-oriented systems," *Journal of systems and software*, vol. 51, no. 3, pp. 245–273, 2000.

[55] S. R. Chidamber and C. F. Kemerer, "A metrics suite for object oriented design," *IEEE Transactions on software engineering*, vol. 20, no. 6, pp. 476–493, 1994.

[56] A. S. Nuñez-Varela, H. G. Perez-Gonzalez, F. E. Martínez-Perez, and C. Soubervielle-Montalvo, "Source code metrics: A systematic mapping study," *Journal of Systems and Software*, vol. 128, pp. 164–197, 2017.

[57] C. Catal and B. Diri, "A systematic review of software fault prediction studies," *Expert systems with applications*, vol. 36, no. 4, pp. 7346–7354, 2009.

[58] J. Leppink, "Analysis of covariance (ancova) vs. moderated regression (modreg): Why the interaction matters," *Health Professions Education*, vol. 4, no. 3, pp. 225–232, 2018.

[59] A. F. Hayes and L. Cai, "Using heteroskedasticity-consistent standard error estimators in ols regression: An introduction and software implementation," *Behavior research methods*, vol. 39, no. 4, pp. 709–722, 2007.

[60] G. Bollinger, "Book review: Regression diagnostics: Identifying influential data and sources of collinearity," 1981.

[61] S. K. Kachigan, *Multivariate statistical analysis: A conceptual introduction*. Radius Press, 1991.

[62] J. E. Jackson, *A user's guide to principal components*. John Wiley & Sons, 2005, vol. 587.

[63] J. G. MacKinnon and H. White, "Some heteroskedasticity-consistent covariance matrix estimators with improved finite sample properties," *Journal of econometrics*, vol. 29, no. 3, pp. 305–325, 1985.

[64] B. Sisman and A. C. Kak, "Assisting code search with automatic query reformulation for bug localization," in *Proceedings of the 10th Working Conference on Mining Software Repositories*. IEEE Press, 2013, pp. 309–318.

[65] M. Chochlov, M. English, and J. Buckley, "A historical, textual analysis approach to feature location," *Information and Software Technology*, vol. 88, pp. 110–126, 2017.

[66] A. K. Montoya, "Moderation analysis in two-instance repeated measures designs: Probing methods and multiple moderator models," *Behavior research methods*, vol. 51, no. 1, pp. 61–82, 2019.

[67] H. K. Wright, M. Kim, and D. E. Perry, "Validity concerns in software engineering research," in *Proceedings of the FSE/SDP workshop on Future of software engineering research*. ACM, 2010, pp. 411–414.

[68] J. F. Finch, S. G. West, and D. P. MacKinnon, "Effects of sample size and nonnormality on the estimation of mediated effects in latent variable models," *Structural Equation Modeling: A Multidisciplinary Journal*, vol. 4, no. 2, pp. 87–107, 1997.

[69] S. B. Green, "How many subjects does it take to do a regression analysis," *Multivariate behavioral research*, vol. 26, no. 3, pp. 499–510, 1991.

[70] J. Martinez, N. Ordoñez, X. Tërnava, T. Ziadi, J. Aponte, E. Figueiredo, and M. T. Valente, "Feature Location Benchmark with ArgoUML SPL," in *Systems and Software Product Line Conference (SPLC)* , Gothenburg, Sweden, Sep. 2018. [Online]. Available: https://hal.sorbonne-universite.fr/hal-01722316

[71] T. Dilshener, M. Wermelinger, and Y. Yu, "Locating bugs without looking back," *Automated Software Engineering*, vol. 25, no. 3, pp. 383–434, 2018.

[72] X. Gu, H. Zhang, and S. Kim, "Deep code search," in *2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE)*. IEEE, 2018, pp. 933–944.

**Abdul Razzaq** Dr.Abdul Razzaq received the BS(CS) and MPhil degrees in computer science and software engineering from PMAS UAAR and Quaid-i-Azam University, Pakistan, in 2010 and 2013, respectively. He received the Ph.D. degree from Lero-Irish Software Research Center, University of Limerick. His research interests include designing replicable empirical designs for evaluating heterogenous software engineering techniques and measuring the impact of the mostly uncontrolled variables in empirical practice. He has extensive experience in static code analysis and feature/concept/bug location to support software maintenance activities. He also did extensive work on human-factors to improve the developer experience in performing software engineering activities.

**Anthony Ventresque** Dr Anthony Ventresque received his Ph.D. degree in Computer Science from the University of Nantes INRIA France in 2008. He is currently a Lecturer in the School of Computer Science at University College Dublin, Ireland, and a Funded Investigator with Lero, the SFI Irish Software Research Centre. Previously he held positions as Research Fellow at NTU, Singapore (20102011), UCD, Ireland (20122014), and IBM Research Dublin, Ireland (20142015). Dr Anthony Ventresque founded and leads the UCD Complex Software Lab (https://csl.ucd.ie/).

**Rainer Koschke** Rainer Koschke is a full professor for software engineering at the University of Bremen in Germany and heading the software engineering group. His research interests are primarily in the fields of software engineering, program analyses, and software visualization. His current research includes program analyses, clone detection, visualization in VR and AR, reverse engineering, architecture recovery, feature location, and security. He is one of the founders of Axivion GmbH (founded in 2006) providing solutions for stopping software erosion. He received a doctoral degree in computer science at the University of Stuttgart, Germany in 2000.

**Andrea De Lucia** Andrea De Lucia received the Laurea degree in computer science from the University of Salerno, Italy, in 1991, the M.Sc. degree in computer science from the University of Durham, U.K., in 1996, and the Ph.D. degree in electronic engineering and computer science from the University of Naples Federico II, Italy, in 1996. He is a Full Professor of software engineering at the Department of Computer Science of the University of Salerno, the coordinator of the PhD program in Computer Science, the Head of the Software Engineering Lab, and the Director of the International Summer School on Software Engineering. Previously he was at the Department of Engineering and the Research Centre on Software Technology of the University of Sannio, Italy. His research interests include software maintenance and testing, reverse engineering and re-engineering, source code analysis, code smell detection and refactoring, mining software repositories, defect prediction, empirical software engineering, search-based software engineering, traceability management, collaborative development, workflow and document management, and visual languages. He has published more than 250 papers on these topics in international journals, books, and conference proceedings and has edited books and journal special issues. Prof. De Lucia is co-editor in chief of Science of Computer Programming (Elsevier) and serves on the editorial board of Empirical Software Engineering (Springer) and Journal of Software Evolution and Process (Wiley). He has also served on the organizing and program committees of several international conferences in the field of software engineering. He is a senior member of the IEEE and was member-at-large of the executive committee of the IEEE Technical Council on Software Engineering.

**Jim Buckley** Jim Buckley was awarded an MSc degree in Computer Science from the University of Limerick and a PhD in Computer Science from the same University in 2002. He currently works as a senior lecturer in the Computer Science and Information Systems Department/Lero at the University of Limerick, Ireland. His main research interests focus on supporting software developers who are tasked with maintaining and evolving software systems. Thus, specific areas of interest include feature location, software comprehension and architectural analysis of such systems.