FeatRacer: Locating Features Through Assisted Traceability (Summary)

Mukelabai Mukelabai[1], Kevin Hermann[2], Thorsten Berger[3], and Jan-Philipp Steghöfer[4]

Abstract: This extended abstract summarizes our paper with the same title published in the journal IEEE Transactions on Software Engineering (TSE) 2023 [Mu23].

Keywords: features, feature location, traceability, recommender system, machine learning, AI4SE

Feature location (a.k.a., concern location or concept assignment) [BMW94, KBL18, RC13] is one of the most common tasks of developers. During maintenance and evolution of software systems, developers typically need to identify the exact locations in a codebase where specific features are implemented. Unfortunately, feature location is a laborious and error-prone task, since developers' knowledge of features and their location fades over time, development teams change, and features are often scattered across the codebase. To this end, many *automated feature location techniques* have been proposed, which aim to retroactively recover features. Unfortunately, these techniques have not found any relevant adoption in practice, mainly since they are difficult to set up, recover only coarse-grained locations, produce too many false positives, and require large training datasets. Furthermore, they typically require developers to write complex search queries for features.

We follow a different strategy in addressing the *feature location problem*. We advocate *recording features during development*, when their location is still fresh in a developer's mind. We use embedded code annotations [SMB20] which developers can use to proactively record features and their locations. Then, when the need arises, developers can quickly lookup the feature locations—as opposed to performing the laborious and error-prone manual or automated feature location. Still, developers easily forget to annotate code, and annotations can become outdated during maintenance and evolution, for instance, when code is refactored.

We present FeatRacer [Mu23, Ab18], which foster proactive recording of feature locations by recommending features to developers. FeatRacer learns the developer's feature recording practices within the project, thereby training machine-learning-based classifiers, and reminds the developer about potentially missing features or when they forget to annotate code. Code can be annotated and recommended at three levels of granularity: folder, file and code

1 University of Zambia, Zambia, and Chalmers | University of Gothenburg, Sweden, mukelabai.mukelabai@unza.zm, https://orcid.org/0000-0002-3868-4319

2 Ruhr University Bochum, Germany, kevin.hermann@rub.de, https://orcid.org/0009-0004-6207-4045

3 Ruhr University Bochum, Germany, and Chalmers | University of Gothenburg, Sweden, thorsten.berger@rub.de, https://orcid.org/0000-0002-3870-5167

4 XITASO Gmbh IT & Software Solutions, Germany, and Chalmers | University of Gothenburg, Sweden, jan-philipp.steghoefer@xitaso.com, https://orcid.org/0000-0003-1694-0972

fragment level. FeatRacer's machine-learning classifiers then rely on metrics we designed to measure code-level and process characteristics of specific features. While engineering these metrics, we found that data considering process-related metrics, such as contributions made by specific developers show a higher correlation with the prediction performance of FeatRacer than commit-related metrics, such as number of lines added. When FeatRacer finds non-annotated code during a commit, its machine-learning classifiers can predict which feature it implements based on the its metrics.

We show that FeatRacer outperforms traditional automated feature location based on Latent Semantic Indexing (LSI) and Linear Discriminant Analysis (LDA)—which are two of the most common techniques used for feature location recovery—when predicting features for 4,650 commit changesets from the histories of 16 open-source projects. On 16 open-source projects FeatRacer showed a 3x higher precision and a 4.5x higher recall than LSI and LDA, with an average precision and recall of 89.6 %. Furthermore, FeatRacer is already effective in predicting feature locations when only as few as five features are introduced in a project.

Data Availability

An online appendix with FeatRacer, our used datasets, implementation of LSI and LDA, and evaluation data can be found online [On23].

Bibliography

[Ab18] Abukwaik, Hadil; Burger, Andreas; Andam, Berima Kweku; Berger, Thorsten: Semi-Automated Feature Traceability with Embedded Annotations. 2018.

[BMW94] Biggerstaff, Ted J; Mitbander, Bharat G; Webster, Dallas E: Program understanding and the concept assignment problem. Communications of the ACM, 37(5):72–82, 1994.

[KBL18] Krüger, Jacob; Berger, Thorsten; Leich, Thomas: Features and How to Find Them: A Survey of Manual Feature Location. In (Mistrik, Ivan; Galster, Matthias; Maxim, Bruce, eds): Software Engineering for Variability Intensive Systems: Foundations and Applications. Taylor & Francis Group, LLC/CRC Press, 2018.

[Mu23] Mukelabai, Mukelabai; Hermann, Kevin; Berger, Thorsten; Steghöfer, Jan-Philipp: FeatRacer: Locating Features Through Assisted Traceability. IEEE Transactions on Software Engineering, pp. 1–23, 2023.

[On23] Online Appendix. `https://bitbucket.org/easelab/featracer/`, 2023.

[RC13] Rubin, Julia; Chechik, Marsha: A Survey of Feature Location Techniques. In: Domain Engineering, pp. 29–58. Springer, 2013.

[SMB20] Schwarz, Tobias; Mahmood, Wardah; Berger, Thorsten: A Common Notation and Tool Support for Embedded Feature Annotations. 2020.