Supplemental Material on the Details of HCRs

In this study, we design a series of Heuristic Construction Rules (HCRs) to guide the action of new attribute construction under the inspiration of the reasons provided by the volunteers. HCRs consist of three phases, including candidate attribute extraction, meaningless attribute removal, and attribute calculation. This document details each phase, including the explanation of a phase, the rules applied in this phase, an example, and some suggestions on automating each phase.

- The *explanation* explains the concept and general process of a phase.
- *Rules* describe the methods to manually apply HCRs by requesters.
- The *example* gives a simple example in a phase. Besides this example, in the paper, we also use examples to explain the reasons to construct every attribute in Section V-B and use a set of examples in Table II to explain HCRs.
- The *suggestion* provides tools and methods to completely or partially automate HCRs. Since the automation tools may accumulate errors, requesters manually infer attributes from the volunteers' reasons by HCRs in this study. In the future, we plan to investigate how to automate CA for large numbers of reasons.

HCR1: Candidate attribute extraction

Explanation. In this step (Phase 1), some candidate attributes are extracted from every reason. HCR1 limits requesters to select only one candidate attribute from each reason. If no candidate attribute is identified, we remove the reason. We then make the candidate attribute and its corresponding reason as a candidate pair, i.e., <candidate attribute, reason>. In this phase, a set of candidate pairs are collected.

Rules. *Rules to extract candidate attributes*. We analyze the part-of-speech and the structure of each reason, and then

1. The subject and object of the reason are identified.
2. We check whether the subject or object of each reason is a noun/adjective term or adjective+noun phrase.
3. We select the candidate attribute according to the frequency of terms and phrases counted by all the reasons.

We take the high frequency term or phrase as a candidate attribute. Hence, a sentence is removed, if no term or phrase is selected, e.g. it is not a complete sentence with subject or object.

Example. For the sentence "the length of the sentence is long", the subject of the sentence is "length" and the object is "long". They are noun and adjective terms which can be taken as candidate attributes. We can count the times that "length" and "long" occur in all the sentences to calculate their frequency.

Suggestions. Several tools can be used to automate this step:

1. Requesters need to split a sentence into terms or phrases. The technique of tokenization can be used for this purpose. A popular tool for tokenization is Stanford Tokenizer [1]. In addition, recent studies also propose some software-specific named entity recognition for software engineering social content [2] to identify some key terms or phrases.
2. Requesters need to identify the noun and adjective terms of a sentence. To detect the part-of-speech of English, Stanford Log-linear Part-Of-Speech Tagger [3] is widely used.
3. Requesters need to identify the subject and object of a sentence. The sentence structure can be analyzed by

tools for sentence dependency analysis, e.g., MINIPAR [4].

HCR2: Meaningless attribute removal.

Explanation. In HCR2 (Phase 2), some meaningless candidate attributes are detected and removed. First, we group the candidate pairs according to the synonyms and morphology of candidate attributes. Second, for each group, we analyze the meaning of the candidate attributes. Some groups with meaningless candidate attributes are removed. At the last of this phase, we select one of the candidate attributes in each reserved group as a representative, since all candidate attributes in the same group are synonyms of the same meaning.

Rules. HCR2 needs rules to group candidate attributes and delete attribute groups.

Rules to group candidate attributes. We add a candidate pair to a group if the candidate attribute is a synonyms or morphology of any candidate attribute in a group. Otherwise, the candidate pair is regarded as a new group. Synonyms mean that the meaning of a term or phrase is exactly or nearly the same as another term or phrase in the same language. Morphology means two terms are in similar forms or structures [5]. Especially, they have the same "root word", or one term is the prefixes or suffixes of another.

Rules to delete attribute groups. We delete a group, if its candidate attributes do not satisfy any of the following rules:

1. The candidate attributes in the group appear in some predefined items of SE data.
   For a bug report, the typical predefined items include status, product, component, version, hardware, importance (priority), target milestone, assigned to, QA contact, URL, whiteboard, keyword, depends on, block, reported time, modified time, CC list, title, attachments, description, comment, reporter name and commenter name.
2. The candidate attributes can be transformed into some measurements. We list some terms from the volunteers' reasons that belong to this category, i.e., we can transform these terms into measurements.
   - terms related to the similarity: similar with, similarity, related to, duplicate;
   - terms express the relationships between two parts: solution to the problem, solving the problem, suggestion to the bug, a description of the problem;
   - terms aim to detect keywords in bug reports: special words, "createWidget", "readAndDispatch", code, related website;
   - terms related to the complexity/importance of a sentence: rich in content, (sentence is) simple, concrete advice;
   - terms related to the length: length, long;
   - terms related to the location: the beginning, first sentence.

Examples. The attribute pairs of "<the length, ...>" and "<lengths, ...>" are put in the same group.

Suggestions. In this part, we need to analyze the synonyms of terms, tools like WordNet [6] and software-specific databases for word similarity [7] can be used for this purpose.

To analyze the morphology of terms, Porter stemming [8] and techniques of morphological forms inference for software-specific terms [9] can be used.

To delete the attribute groups, we can first match each term with the contents of the SE data to automatically check whether this term is a keyword or appears in some predefined items of SE data. Then, requesters can check the remaining candidate attributes to decide whether it belongs to a measurement.

HCR3: Attribute calculation.

Explanation. HCR3 (Phase 3) calculates the values of the representative candidate attributes and merges candidate attributes with the same calculation metrics to achieve the final attributes. In the calculation step, all candidate attributes in the same group are calculated by the same criterion.

Rules. HCR3 needs rules to calculate an attribute and merge candidate attributes with the same calculation metrics.

Rules to calculate an attribute. If a candidate attribute satisfied one of the following rules, we can measure this attribute. Correspondingly, if it can not satisfy any of the rules, we delete the attribute.

1. For candidate attributes that are the same with the attributes in the related work, the same methods are followed to calculate the attributes.
2. If the candidate attribute is a predefined item or a keyword of SE data, we enumerate the status of the item or keyword as attribute values, e.g., using 0 and 1 to denote whether a sentence related to an item, using 0, 1, 2 to denote the bug report Priority P1, P2, and P3, etc.
3. For attributes related to some measurements, requesters can follow the mathematic definitions of the measurements to calculate the corresponding information in SE data, e.g., calculating the length or similarity of sentences.
4. For the remaining attributes, we try to transform them into some measurements, e.g., transforming a candidate attribute into a type of length or similarity. Otherwise, the candidate attribute is deleted.

Rules to merge candidate attributes with the same calculation metrics. The calculation metrics are used to transform an input into a value. For example, if the calculation metric is used to calculate the length of a sentence, it usually inputs a sentence and outputs a value to represent the sentence length. Hence, two calculation metrics are the same,

1. if the inputs of the calculation metrics are the same
2. and the calculation metrics output the same values for all the inputs.

We merge the calculation metrics that satisfy both of the two rules.

Examples. For example, “<the length, ...>” and “<long, ...>” may be merged, because both attributes can be transformed into a value to represent the length of a sentence.

Suggestions. It is a non-trivial work to deriving adequate measures for the attributes. To the best of our knowledge, no previous study investigates the way to automatically derive adequate measures for attributes. However, we can define a serials of mapping rules between the terms and measurements.

- For the similarity between terms, we can use cosine similarity [10], BM25 [11], Edit Distance [12] for calculation.
- The relationships between two parts can be calculated by the similarity or the entropy of two parts.
- The keywords can be detected by regular expressions.
- For the complexity/importance of a sentence, we can measure it by Term Frequency - Inverted Document Frequency, readability [13], the number of low-frequency terms, etc.
- We can calculate the location of a sentence and the number of words or characteristics of the sentence to measure the location and length of a sentence respectively.

Reference

[1] The Stanford NLP Group. Stanford Tokenizer. https://nlp.stanford.edu/software/tokenizer.html.

[2] Ye, D., Xing, Z., Foo, C. Y., Zi, Q. A., Li, J., & Kapre, N. (2016). Software-Specific Named Entity Recognition in

Software Engineering Social Content. IEEE, International Conference on Software Analysis, Evolution, and Reengineering (pp.90-101). IEEE.
[3] The Stanford NLP Group. Stanford Log-linear Part-Of-Speech Tagger. https://nlp.stanford.edu/software/tagger.shtml.
[4] Lin, D. (2003). Dependency-based evaluation of MINIPAR. In Treebanks (pp. 317-329). Springer, Dordrecht.
[5] Aronoff, M., & Fudeman, K. (2011). What is morphology? (Vol. 8). John Wiley & Sons.
[6] Fellbaum, C. (1998). WordNet. John Wiley & Sons, Inc..
[7] Tian, Y., Lo, D., & Lawall, J. (2014, February). Automated construction of a software-specific word similarity database. In Software Maintenance, Reengineering and Reverse Engineering (CSMR-WCRE), 2014 Software Evolution Week-IEEE Conference on (pp. 44-53). IEEE.
[8] Porter, M. (2008). The Porter stemming algorithm, 2005. See http://www. tartarus. org/~ martin/PorterStemmer.
[9] Chen, C., Xing, Z., & Wang, X. (2017, May). Unsupervised software-specific morphological forms inference from informal discussions. In Proceedings of the 39th International Conference on Software Engineering (pp. 450-461). IEEE Press.
[10] Dehak, N., Dehak, R., Glass, J. R., Reynolds, D. A., & Kenny, P. (2010, June). Cosine similarity scoring without score normalization techniques. In Odyssey (p. 15).
[11] Robertson, S., & Zaragoza, H. (2009). The probabilistic relevance framework: BM25 and beyond. Foundations and Trends® in Information Retrieval, 3(4), 333-389.
[12] Ristad, E. S., & Yianilos, P. N. (1998). Learning string-edit distance. IEEE Transactions on Pattern Analysis and Machine Intelligence, 20(5), 522-532.
[13] Dale, E., & Chall, J. S. (1948). A formula for predicting readability: Instructions. Educational research bulletin, 37-54.