Client Input Interface - The Recommender System

Waleed Abdeen and Michael Unterkalmsteiner
Department of Software Engineering
Blekinge Institute of Technology
Karlskrona, Sweden
waleed.abdeen@bth.se, michael.unterkalmsteiner@bth.se

***Abstract*—Requirements traceability to downstream software artifacts is beneficial in producing software products that align with customer requirements. In practice, traceability may not be implemented due to the challenges associated with creating and maintaining trace links. In previous work package (WP), we proposed Taxonomic Trace Links that create trace links between two elements by introducing indirection. This report for WP4, presents a semi-automated way of implementing taxonomic trace links.**

***Index Terms*—Taxonomic Trace Links, Traceability, Classification**

I. Introduction

Requirements traceability to downstream system artifacts, e.g., design models, is beneficial in producing software products that align with customer requirements. Current requirements traceability techniques in general [1] and tools [3] in particular suffer from challenges that hinder the adoption of requirements traceability. In this report, we show how we built the recommender system to classify requirements using a domain-specific taxonomy in order to enable requirements traceability.

II. Taxonomic Trace Links Approach

We introduced in Work Package 2 (WP2) the Taxonomic Trace Link (TTL) approach that creates trace links between two elements using a taxonomy [6]. TTL can connect elements of the same (e.g., requirements with requirements) or different artifacts (e.g., requirements with design models). The TTL approach aims to overcome the barriers that are faced when introducing traceability between different project artifacts, mainly the misalignment between the documents' abstraction level, structure, and time of creation.

Applying TTL requires the association of requirements and design models with classes from a domain-specific taxonomy. The association of requirements with classes from the taxonomy (requirements classification), needs to be done only once per requirement. This association can be used later for tracing the same requirement to multiple design models (from the same or different project). We use two taxonomies from the construction domain, also referred to as classification systems, to implement our approach, SB11 and CoClass.

In WP2 we validated the TTL approach on a set of 27 requirements, 10 generic from TDOK and 17 from the Eastlink project. The purpose of the validation study was to investigate the applicability and practical challenges associated with TTL when applied in practice. The results of the validation study show that it is possible to implement TTL to trace design models to generic and project-specific requirements. However, this depends on the quality of requirements and the taxonomy, and the proper association of codes with both requirements and design models.

III. Methodology

In this WP we build a recommender system to semi-automate the process of requirements classification. The recommender analyses the requirement and proposes classes from SB11 or CoClass. Then, the domain experts or requirements engineers decide on the correct classification. According to Mısırlı et al. [4], a recommendation system has two dimensions, 1) *input* fed by the user or imported from other systems, 2) *output* is the recommendation provided by the recommendation system. In our case, the inputs are a natural language requirement and a classification system, while the outputs are the suggested classes from the classification system.

A. Building the recommender system

Building the recommender system is usually achieved by training a model using machine learning algorithms on labeled data. A difficult part is finding the labeled data. In our case, we would need a set of requirements from the construction domain classified with SB11 or CoClass. This classification needs to be done with the help of experts with domain knowledge. This is challenging, especially if we need to have a large enough dataset to train the model properly.

Song et al. [5] have proposed a dataless approach for hierarchical text classification. In their study, they classify emails from newsgroups using a topic taxonomy with two levels. They use natural language processing to analyze the text, extract info, then find similarities between two documents. Their results show that it is possible to build a text classifier without the need for labeled data. In this WP, we reuse their approach with some modifications to fit our goal.

B. Building the Ground Truth

Although unsupervised learning algorithms do not require training data set to build a model, a test set, which is usually smaller than the training data set, is required to evaluate the model's performance. Walker et al. [8] advise against building

the training data set, which is referred to as the ground truth, automatically and list three options to determine it: 1) ask humans to build it; 2) use a combination of automated approaches; 3) use data collected previously from other projects.

SB11 and CoClass are mainly used in Sweden, and to the best of our knowledge, there are no publicly available requirements annotated with either of the classification systems, thus ruling out the third option. Although it can be argued that manually establishing the ground truth (the first option) is a large investment of resources, however, combining automated approaches (the second option) may not give accurate results. The effort put into establishing the ground truth can be justified by considering the application of the recommender, which is to classify thousands of requirements that are usually used in multiple projects. Thus, we went with the first option.

TABLE I
PARTICIPANTS IN BUILDING THE GROUND TRUTH WITH YEARS OF EXPERIENCE IN THE DOMAIN AND IN RESEARCH

Team	Affiliation	Role	Domain	Research
A	BTH	Researcher	none	13
A	HOCHTIEF	Civil engineer	$\prec 1$	$\prec 1$
B	BTH	Researcher (curator)	none	3
B	HOCHTIEF	BIM manager and consultant	5	4
C	BTH	Researcher	none	14
C	HOCHTIEF	BIM and information manager	7	1

a) Dataset: We sampled 129 requirements, 92 generic requirements from TRVInfra, and 37 requirements from the Eastlink project. The sample was selected by the second author, a senior researcher with knowledge in the infrastructure domain, to be representative of the population (infrastructure projects requirements).

b) Participants: Six people were involved in building the ground truth, three domain experts and three researchers. In Table I, we list the participants' roles and years of domain and research experience.

c) Process: Figure 1 depicts the steps that we followed to build the ground truth. We split the group into three teams of two (one researcher and one practitioner) and followed the steps below:

1) *Step 1 - Annotation.* Each team received a unique subset of the sampled requirements, and each team member was asked to individually annotate the same subset with classes from SB11 and CoClass.
2) *Step 2 - Curation.* The first author (curator) curated the annotations from each team. When team members disagreed on a requirement's annotation, the requirement is discussed in a team meeting; otherwise, the requirement is added to the review list.
3) *Step 3 - Team meeting.* The curator compiled annotation disagreements per team and ran teams meetings. In a team meeting, the curator displayed the requirements with the suggested annotations from each team member and asked the team members to discuss and dissolve the disagreement. When consensus couldn't be reached, the requirement was brought to a group meeting to discuss it. Otherwise, the requirement was added to the review list.
4) *Step 4 - Group meeting.* All participants attended the group meeting. The curator prepared a list of requirements with disagreements in annotations from all teams and grouped them into themes. The requirements were presented, and the group was asked for their opinion.
5) *Step 5 - Review process.* Each team was assigned a requirements subset to review that was annotated by another team. The review was done individually by each

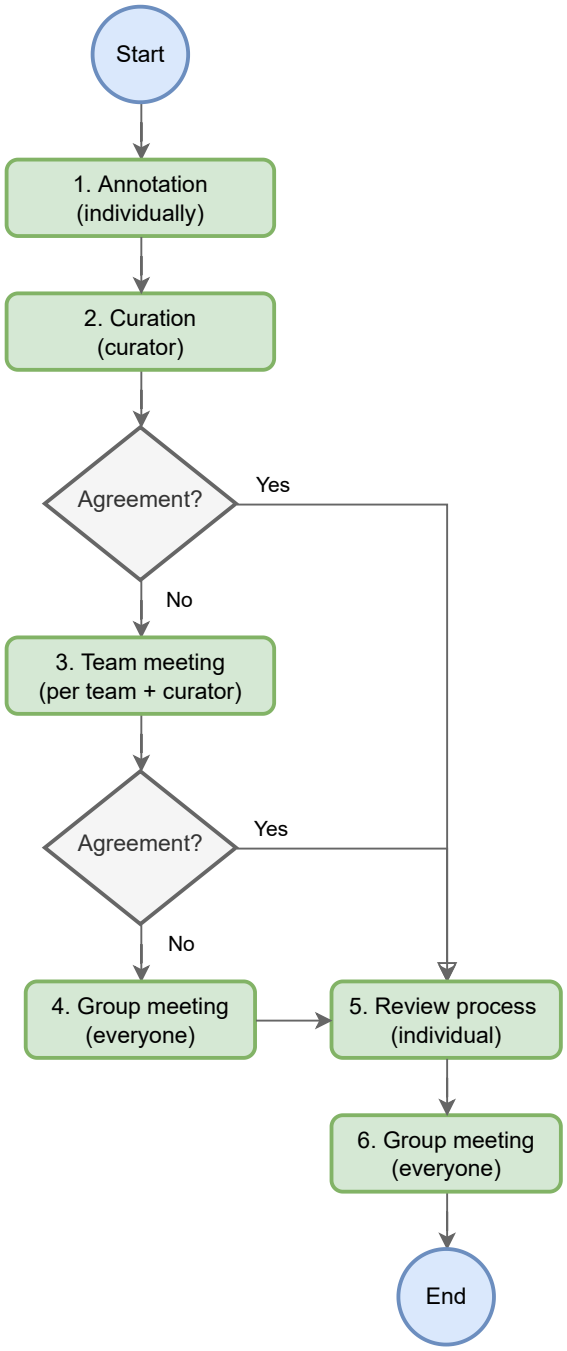

Fig. 1. The process of building the ground truth

team member.

6) *Step 6 - Group meeting.* The outcomes of the reviews were compiled and discussed in the group meeting. The reviewers and annotators of a requirement were asked to discuss their annotations. The other group members participated in the discussion when needed until consensus was reached.

We conducted the process of building the ground truth iteratively. A total of four rounds were conducted.

IV. The Recommender System

The purpose of the recommender system is to help domain experts to classify requirements quicker and more accurately. The inputs to the recommender are the requirement's information, document title, section titles, and requirement's text, and a classification system. The recommender calculates the similarities between the requirement and the classes from the classification system and select the top classes. The outputs of the recommender are top five classes for this requirement.

We developed the recommender based on the study by Song et al. [5]. The source code of the recommender is publicly available on Github[1]. Furthermore, we have prepared a demonstrator[2] to show how the recommender works. The source code of the demonstrator/tool is also available on Github[3].

V. Comparison between CoClass and SB11

A. Classification system quality

In previous work [7], we have developed a set of quality attributes that can be used to objectively compare classification systems. The seven quality attributes are:

QA1 *Comprehensiveness* is the ability to classify all known objects for the domain it was developed for.

QA2 *Robustness* is the ability to differentiate objects of interest.

QA3 *Conciseness* is the ability to classify objects of interest with the least possible amount of dimensions, categories and characteristics.

QA4 *Extensibility* is the ability to allow for changes in its structure, i.e. adding, modifying or deleting dimensions, categories or characteristics.

QA5 *Explanatory* is the quality to enable the user to locate an object in the classification system based on its characteristics, or to deduce from the location of an object what characteristics it has.

QA6 *Mutual exclusiveness* is the ability to identify an object uniquely, i.e. that no object exists in the same dimension under different categories.

QA7 *Reliability* is the ability to support consistent classification decisions.

Table II summarizes the evaluation of CoClass and SB11 with respect to these seven quality attributes. In addition, we also show, for comparison, the evaluation results of two more, widely used classification systems in the construction industry, UniClass and OmniClass. For the task of requirements classification, extensibility is less important. Hence, we discuss only the results of the remaining six quality attributes.

a) Comprehensiveness: CoClass seems to have a slight advantage over SB11 as it originates from a diverse set of sources (standards ISO 12006-2, IEC 81346-1, IEC 81346-2 ISO 81346-12, as explained in [2]) and the developers have a diverse background (illustrated by the partners that have contributed to CoClass[4]). We could not find information on whether CoClass and SB11 has been evaluated by external experts, in contrast to UniClass and OmniClass.

b) Robustness: For the task of classification (whether manual, semi-automated or fully automated), the ability to differentiate objects of interest is very important. Both SB11 and CoClass have scored a significantly higher result than UniClass and OmniClass in this quality attribute. We were suspicious of these rather high differences, and found that it can be explained by a weakness how we calculate the score, in particular the language model we use as a basis for the analysis. In short, we suspect that both SB11 and CoClass contain many domain specific terms that do not appear in the language model that stems from a general purpose document corpus. Hence, the metric is overly optimistic, compared to the results from UniClass and OmniClass (for which use the same language model, but we observed that the terminology uses less compound, domain-specific terms). To conclude, further work is needed to increase the confidence in the achieved evaluation results for robustness.

c) Conciseness: SB11 and CoClass are very similar with respect to the number of dimensions and categories. However, SB11 has more characteristics (i.e., leaf nodes in the hierarchy of the classification tree) and a higher tree depth. This means that SB11 is likely more specific, fine-grained, while CoClass has a higher abstraction level. Both SB11 and CoClass are significantly more concise than UniClass and OmniClass, which makes them more amenable for manual and semi-automatic classification tasks.

d) Explanatory: We did not find any information regarding structuring elements besides dimensions that would help to understand the classification systems. CoClass is unique among the studied classification systems in that it provides definitions for all characteristics and commonly used synonyms.

e) Mutual exclusiveness: Only CoClass mentions explicitly [2] that an object can be member in one class only.

f) Reliability: We have evaluated intra-rater reliability based on the data we collected when developing the ground truth. Table III shows the frequency of agreements and disagreements between the annotators. The reliability of SB11 is slightly higher than CoClass, despite CoClass being more concise. The annotators have observed that it was easier to find an exact match in SB11 than in CoClass because the

[1]https://github.com/munterkalmsteiner/DatalessClassification/tree/SB11Classifier

[2]http://dcat.diptsrv003.bth.se/labeled-data

[3]https://github.com/waleedabdeen/dcat-demonstrator

[4]https://coclass.byggtjanst.se/about#owners

TABLE II
COMPARISON OF CONSTRUCTION CLASSIFICATION SYSTEMS

Quality attribute	Measurement	SB11	CoClass	Uniclass	Omniclass
Comprehensiveness	Does the classification system originate from a diverse set of data sources?	No	Yes	Yes	Yes
	Do the classification system creators have a diverse background?	No data	Yes	Yes	Yes
	Were diverse data analysis methods used to create the classification system?	No data	No data	No data	No data
	Has the classification system been evaluated by external experts?	No data	No data	Yes	Yes
Robustness	$R(T)$	0.98	0.97	0.63	0.79
Conciseness	Dimensions	3	3	12	11
	Categories	352	340	1,827	4,084
	Characteristics	1725	1089	12,501	14,569
	Maximum depth	6	3	5	8
	$C(T)$	0.13	0.14	0.11	0.11
Extensibility	Change process covers the following constructs[a]	No data	No data	No data	ca, ch
	Change process covers the following change types[b]	No data	No data	No data	a, m
	Mandate to change the classification system	No data	Owners/ext	Owners/ext	Owners/ext
	Customization of the classification system	Yes	Yes	No data	No data
	$RoC_{2021-05-04}$	0.00032	0.02001	0.01079	0.00018
Explanatory	Structuring elements besides dimensions?	No data	No data	No data	No data
	Support for choosing dimensions or categories?[a]	di ca	di ca ch	di	di ca
Mutual exclusiveness	Classification constraint	No data	Yes	No data	No data
Reliability	Inter-rater reliability	0.23157	0.22437	Not Implemented	Not Implemented

[a] di = dimension, ca = category, ch = characteristic
[b] a = addition, m = modification, d = deletion

terminology used in the requirements text was similar to the class names in SB11.

B. Classifier performance

We used the ground truth that we prepared (as explained in Section III-B), to evaluate the performance of the hierarchical classifier (HC), that we developed in this project. Furthermore, we compare the HC with the flat classifier (FC), from our previous work [6]. The main difference between the classifiers is how the nodes of the classification system are analyzed. In the HC, the hierarchy of the classification system is considered during the analysis of the nodes. On the contrary, in the FC each node is analyzed alone regardless of its place in the hierarchy.

Figure 2 illustrates the results of comparing the HC with the FC using SB11 and CoClass. The x-axis represents the metrics used for evaluations, while the y-axis represents the value of the metric with maximum value of 1. The ground truth contained 129 requirements. However, not all requirements were considered when calculating the evaluation metrics. We used the requirements labeled using the Byggdelar table from SB11 (123 requirements) and from CoClass we used those labeled using Tillgångssystem (81 requirements).

Overall, the higher recall compared to the precision indicates that the classifier produces fewer false negative than false positive results. That means that the number of true labels that are not retrieved is lower than the number of wrong labels retrieved. In practice, an expert would therefore have to review and exclude wrong suggestions but can, on the other hand, rely on that the suggestions are complete.

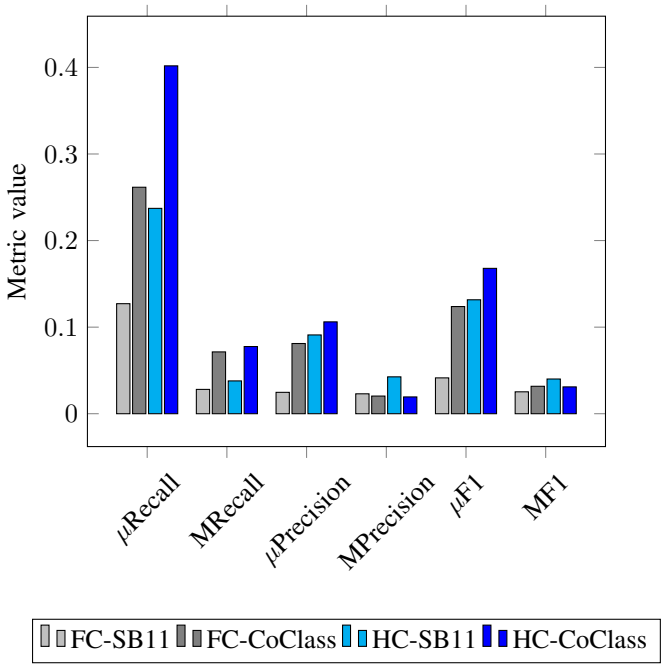

Fig. 2. Comparison of the Classifiers Performance

In all metrics except the MPrecision and MF1, the use of CoClass showed a significant improvement in performance of the flat and hierarchical classifiers compared to the use of SB11. Therefore, CoClass is a better choice as a classification system to classify requirements.

TABLE III
AGREEMENTS AND DISAGREEMENTS BETWEEN ANNOTATORS IN THE GROUND TRUTH CREATION

C.S	Agreements	Disagreements (class)	Disagreement (span)	No Available Code
SB11	88	143	149	48
CoClass	81	132	148	43

The low value of MPrecision, averaging of precision per-class, means that the recommender is better at retrieving some labels but worse at retrieving others. Here, further analysis needs to be done to understand if the attributes of the nodes (e.g., level of depth and number of words in the description) has any correlation with the per-class precision.

C. Discussion

Based on the classifier performance evaluation results in Section V-B, CoClass yields better classification results than SB11. These results can be attributed to CoClass being more concise, has less categories and characteristics, than SB11. Moreover, using the hierarchical classification in the recommender yielded higher performance than the flat classifier.

VI. CONCLUSION

We have implemented a recommender system that suggests labels from two classification systems, SB11 and CoClass, to classify requirements from TRV-Infra and EastLink. In order to evaluate the performance of the recommender, we have built a ground truth that consists of 129 requirements. Furthermore, we have compared the quality of the classification systems, SB11 and CoClass. The results show that the developed recommender using the hierarchical classifier performs better than the flat classifier. However, more enhancements could be made. Possible improvements are adding different pre-processing steps (e.g., word stemming) and trying different methods to calculate similarities between the classification system and the requirement.

REFERENCES

[1] E. Bouillon, P. Mäder, and I. Philippow. A survey on usage scenarios for requirements traceability in practice. In *International Working Conference on Requirements Engineering: Foundation for Software Quality*, pages 158–173. Springer, 2013.

[2] K. Eckerberg. *CoClass - Informationshantering För Byggd Miljö*. Svensk Byggtjänst, 2019.

[3] O. C. Z. Gotel and C. W. Finkelstein. An analysis of the requirements traceability problem. In *Proceedings of IEEE International Conference on Requirements Engineering*, pages 94–101.

[4] A. T. Mısırlı, A. Bener, B. Çağlayan, G. Çalıklı, and B. Turhan. Field studies. In M. P. Robillard, W. Maalej, R. J. Walker, and T. Zimmermann, editors, *Recommendation Systems in Software Engineering*, pages 329–355. Springer.

[5] Y. Song and D. Roth. On dataless hierarchical text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28, 2014.

[6] M. Unterkalmsteiner. TT-RecS: The taxonomic trace recommender system. In *2020 IEEE Seventh International Workshop on Artificial Intelligence for Requirements Engineering (AIRE)*, pages 18–21.

[7] M. Unterkalmsteiner and W. Abdeen. A compendium and evaluation of taxonomy quality attributes. *Expert Systems*, 2022.

[8] R. J. Walker and R. Holmes. Simulation. In M. P. Robillard, W. Maalej, R. J. Walker, and T. Zimmermann, editors, *Recommendation Systems in Software Engineering*, pages 301–327. Springer.