Information and Software Technology xxx (xxxx) xxx–xxx

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

Fragment retrieval on models for model maintenance: Applying a multi-objective perspective to an industrial case study

Francisca Pérez[*], Raúl Lapeña, Jaime Font, Carlos Cetina

Universidad San Jorge. SVIT Research Group, Autovía, A-23 Zaragoza-Huesca Km.299, Zaragoza 50830, Spain

ARTICLE INFO

Keywords:
Conceptual models
Traceability links recovery
Bug localization
Feature location
Evolutionary algorithms

ABSTRACT

Context: Traceability Links Recovery (TLR), Bug Localization (BL), and Feature Location (FL) are amongst the most relevant tasks performed during software maintenance. However, most research in the field targets code, while models have not received enough attention yet.
Objective: This paper presents our approach (FROM, Fragment Retrieval on Models) that uses an Evolutionary Algorithm to retrieve the most relevant model fragments for three different types of input queries: natural language requirements for TLR, bug descriptions for BL, and feature descriptions for FL.
Method: FROM uses an Evolutionary Algorithm that generates model fragments through genetic operations, and assesses the relevance of each model fragment with regard to the provided query through a fitness configuration. We analyze the influence that four fitness configurations have over the results of FROM, combining three objectives: Similitude, Understandability, and Timing. To analyze this, we use a real-world case study from our industrial partner, which is a worldwide leader in train manufacturing. We record the results in terms of recall, precision, and F-measure. Moreover, results are compared against those obtained by a baseline, and a statistical analysis is performed to provide evidences of the significance of the results.
Results: The results show that FROM can be applied in our industrial case study. Also, the results show that the configurations and the baseline have significant differences in performance for TLR, BL, and FL tasks. Moreover, our results show that there is no single configuration that is powerful enough to obtain the best results in all tasks.
Conclusions: The type of task performed (TLR, BL, and FL) during the retrieval of model fragments has an actual impact on the results of the configurations of the Evolutionary Algorithm. Our findings suggest which configuration offers better results as well as the objectives that do not contribute to improve the results.

1. Introduction

Amongst the most common and relevant tasks in the Software Engineering field, especially when maintaining software products, are Traceability Links Recovery, Bug Localization, and Feature Location [1–4]. To tackle these tasks, Information Retrieval (IR) techniques, such as Latent Semantic Indexing (LSI) [5,6], have been used successfully [7,8]. However, most research targets code [3,4,9], neglecting other software artifacts such as models. Models raise the abstraction level using concepts that are much less bound to the underlying implementation and technology and are much closer to the problem domain [10]. The practice of Model Driven Engineering has proved to increase efficiency and effectiveness in software development [10].

To increase the automation level when Traceability Links Recovery, Bug Localization and Feature Location are performed over models, we propose an approach named Fragment Retrieval on Models (*FROM*). Our approach uses a Multi-Objective Evolutionary Algorithm to retrieve the most relevant model fragments for different types of queries (natural language requirements for Traceability Links Recovery, bug descriptions for Bug Localization, and feature descriptions for Feature Location). To guide the Evolutionary Algorithm, we use three fitness objectives: Model Similitude through Latent Semantic Indexing (LSI) [5,6], Model Understandability through Model Size [11,12], and Model Timing through the Defect Principle [13,14].

Moreover, we combine the three objectives into a total of four configurations: (1) Similitude, (2) Similitude + Understandability, (3) Similitude + Timing, and (4) Similitude + Understandability + Timing. We analyze the impact of each configuration on the results of the Evolutionary Algorithm for Traceability Links Recovery, Bug

[*] Corresponding author.
E-mail addresses: mfperez@usj.es (F. Pérez), rlapena@usj.es (R. Lapeña), jfont@usj.es (J. Font), ccetina@usj.es (C. Cetina).

https://doi.org/10.1016/j.infsof.2018.06.017
Received 5 September 2017; Received in revised form 23 June 2018; Accepted 23 June 2018
0950-5849/ © 2018 Elsevier B.V. All rights reserved.

Localization, and Feature Location. In order to carry out this analysis, we use the models, natural language requirements, bug descriptions, and feature descriptions, all of them from a real-world case study provided by our industrial partner, Construcciones y Auxiliar de Ferrocarriles (CAF),[1] which is a worldwide leader in train manufacturing.

We record the results of the Evolutionary Algorithm for each configuration and the baseline for each type of query in terms of recall, precision, and F-measure. Also, results are compared against those obtained by a baseline in order to put FROM in perspective of previous works. The baseline retrieves model fragments using model comparisons among models instead of using an evolutionary algorithm or LSI. Our findings reveal that there is not a unique configuration of objectives that retrieves the best results for all of queries. In other words, the usage of different fitness objectives configurations is required to optimize the results of the Evolutionary Algorithm for either Traceability Links Recovery, Bug Localization, or Feature Location. In addition, we provide evidences of the significance of the results by means of statistical analysis.

The rest of the paper is structured as follows: Section 2 presents a motivating example. Section 3 presents our approach. Section 4 describes the evaluation, the results, and the statistical analysis. Section 5 discusses the results. Section 6 presents the threats to validity. Section 7 reviews the related work. Finally, Section 8 concludes the paper.

2. Motivating example

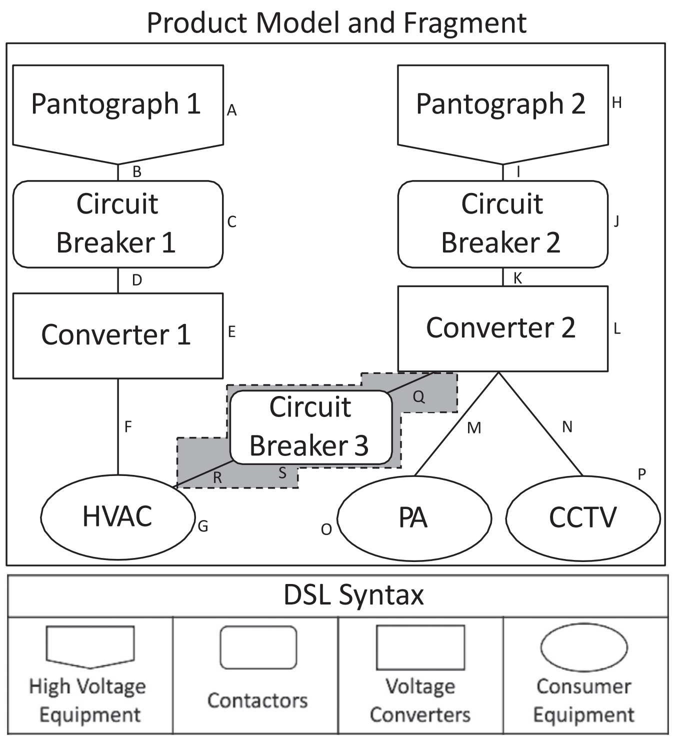
Despite Model-Driven Development has not had the expected widespread success so far, major players in the software engineering field (i.e., tool vendors, researchers, and enterprise software developers) foresee a broad adoption of model-driven techniques because of scenarios that demand more abstract approaches than mere coding [10]. Fostering modeling efforts brings benefits in industrial contexts in order to improve productivity, while ensuring quality and performance [10].

In a model-driven industrial context, companies tend to have a myriad of products with large and complex models behind. The models are created and maintained over long periods of time by different software engineers, and the engineers in charge of the maintenance tasks (Traceability Links Recovery, Bug Localization, and Feature Location) often lack knowledge over the entirety of the product details. Under these conditions, maintenance tasks consume high amounts of time and effort, without guaranteeing good results. Our industrial partner reported performing the maintenance tasks manually at least 25 times per week, costing them a total monthly amount of working time ranging from 43.3 to 66.7 h.

Fig. 1 depicts a model example, taken from a real-world train, specified using the Domain Specific Language (DSL) that formalizes the train control and management of the products manufactured by our industrial partner. The DSL has the expressiveness required to describe both the interaction between the main pieces of installed equipment, and the non-functional aspects related to regulation. It will be used through the rest of the paper to present a running example. For the sake of understandability and legibility, and due to intellectual property rights concerns, we present an equipment-focused simplified subset of the DSL.

Specifically, the example of the figure presents a converter assistance scenario where two pantographs (High Voltage Equipment) collect energy from the overhead wires, and send it to their respective circuit breakers (Contactors), which in turn send it to their independent Voltage Converters. The converters then power their assigned Consumer Equipment: the HVAC on the left (air conditioning system), and the PA (public address system) and CCTV (television system) on the right.

[1] www.caf.net/en.



Model Fragment Encoding:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

Fig. 1. Example of model and model fragment.

The elements of Fig. 1 highlighted in gray conform an example model fragment, including one circuit breaker that connects Converter 2 to a Consumer Equipment assigned to Converter 1. This model fragment is the realization of the 'converter assistance' feature, which allows the passing of current from one converter to equipment assigned to its peer for coverage in case of overload or failure of the first converter.

A model fragment (which always belongs to a parent model) is encoded as a string of binary values that contains as many positions as elements in the parent model, where each position in the string has two possible values: 0 in case the element does not appear in the fragment, or 1 in case the element does appear in the fragment. In Fig. 1, elements Q, R, and S conform the model fragment, so the corresponding values are set to '1' in its binary string representation.

Although it may appear easy to locate the 'converter assistance' feature in the model, it becomes very complex in the models of our industrial partner where each train unit is specified through several thousand elements. According to our industrial partner, software engineers who belonged to the original team of modelers and who work on a monthly basis with the product involved in the example, are able to locate the feature in around 26 min. Another engineer, not related to the project but with knowledge of the products in the company, spent 34 min on the same task. Finally, two newcomer modelers spent around 40 min of combined work until they fulfilled the task, but they did so in a non-accurate manner. Considering these numbers, an approach that automatically retrieves model fragments is strongly needed.

3. Approach

The goal of the presented approach, *FROM* (Fragment Retrieval On Models), is to use an Evolutionary Algorithm to retrieve model fragments for Traceability Links Recovery, Bug Localization, and Feature Location. In addition, we use different combinations of fitness