Categorization Refactoring Techniques based on their Effect on Software Quality Attributes

Abdullah Almogahed, Mazni Omar, Nur Haryani Zakaria

Abstract: refactoring techniques don’t always improve all aspects of software quality attributes. Different types of refactoring techniques have different types of effect on different software quality attributes, consequently, software practitioners encounter challenges in selecting appropriate refactoring techniques to enhance the quality of software design in support of particular design goals. Therefore, categorization refactoring techniques depending on their influence on quality attributes is significant to enable software practitioners in improving software quality by selecting suitable refactoring techniques. A systematic review has been accomplished to determine and analyze studies which tightly related to categorize the refactoring techniques depending on their influence on quality attributes. 14 primary studies have been found and selected for analysis. The obtained results showed that there is a lack of studies regarding the categorization of the refactoring techniques and the current works are insufficient to solve the challenges facing software practitioners. Several recommendations have been suggested to address these gaps.

Keywords: categorization refactoring techniques, refactoring techniques, software quality attributes.

I. INTRODUCTION

Software refactoring is an approach that aims to change the internal design of a software system devoid of altering its exterior functionalities to strengthen the design quality of the software system (Fowler et al., 2002; Fowler & Beck, 2019). The software refactoring has four key advantages which include improving quality of software design, helping to program faster, making software simpler to understand, and helping to locate defects (Fowler et al., 2002; Fowler & Beck, 2019; Ammerlaan, Venina, & Zaidman, 2015; Alves, Massoni, Duarte, & Machado, 2016). There are 68 refactoring techniques categorized into six categorizations (composing methods, dealing with generalization, 2015). Regardless of these categorizations, each refactoring technique has a specific aim and motivation to use it. Nowadays, refactoring is a significant task involved with software development activities since it is to be caused by a variety of factors which can include new requirements, unsatisfying quality, and adaptation to multiple situations (Arcelli, Cortellessa, & Pompeo, 2018).

Consequently, studying its feasibility and effect on software quality is significant. In this regard, previous empirical studies have investigated whether applying refactoring techniques improves software quality attributes or not in order to confirm or reject Fowler’s declaration that says software refactoring improves software quality. Based on analysis of the related literature, these empirical studies reported mixed results which included: 1) the refactoring techniques positively affect on the software quality (e.g., Alshayeb, 2009; Rachatasumrit & Kim, 2012; Tsantalis, Guana, Strouliya, & Hindle, 2013; Nayya, Counsell, & Hall, 2015; Palomba, Zaidman, Oliveto, & Lucia, 2017), 2) the refactoring techniques negatively affect on the software quality e.g., Kannangara & Wijayanake, 2014; Stroggylos & Spinellis, 2007), 3) the refactoring techniques do not effect on the software quality (e.g., Bavota, Lucia, Di Penta, Oliveto, & Palomba, 2015; Soetens & Demeyer, 2010; Wilking, Khan, & Kowalewski, 2007), and 4) effect of the refactoring techniques on the software quality is unclear (e.g., Elish & Alshayeb, 2009; Alshayeb, 2009; Halim & Mursanto, 2013).

The obtained results are mixed (i.e., positive, negative, no effect, not clear). In other words, the outcomes of the impact of applying refactoring techniques on quality attributes are conflicting with each other and inconsistent. As a result, software practitioners are encountering huge challenges in choosing the suitable refactoring techniques that can be used to greatly enhance the design quality of a software system regarding certain design goals. Therefore, categorization refactoring techniques depending on their influence on quality attributes can enable software practitioners to decide which suitable refactoring techniques can be selected.

This study conducted a systematic literature review (SLR) concerning the effect of refactoring techniques on quality attributes in order to extract studies that categorized refactoring techniques depending on their influence on quality attributes. The two main objectives of the study are 1) identifying the refactoring techniques that have been categorized and software quality attributes that have been included in the categorizations, 2) analyzing the relation between refactoring and software quality in accordance with these categorizations. In this paper, Section II describes the related work. The SLR approach is detailed in Section III. Section IV reports the results and discussion. The conclusion is summarized in Section V.
Categorization Refactoring Techniques based on their Effect on Software Quality Attributes

II. RELATED WORKS

A number of systematic literature reviews have already been conducted highly relevant to software refactoring field where different facets of refactoring were discussed. Abebe and Yoo (2014) performed an SLR to identify the challenges, trends, and opportunities related to software refactoring. They concluded that researches on the software refactoring need more attention, due to there are still a lot of unresolved issues that need more research to address them in the future. Overall, the influence of refactoring on the quality attributes was discussed generally in this review. Misbhauddin and Alshayeb (2015) carried out an SLR of available studies for UML model refactoring. They analyzed the primary studies based on various criteria including UML diagrams deemed for the refactoring, the formalisms utilized to assist the refactoring of UML diagrams, recommendations should consider when developing support refactoring tools, and the influence of the software refactoring on UML models quality. Regarding effect the refactoring on model quality, five studies only evaluated the impact on the model quality. Al Dallal and Abdin (2018) presented comprehensive SLR concerning effect object-oriented code refactoring on quality attributes. They analyzed the primary studies based on different factors including the quality attributes and measurements, refactoring techniques, datasets used, evaluation methods, and methods for analysis impact of the refactoring. Almogahed, Omar, and Zakaria (2018) conducted a review of empirical studies relevant to the effect of refactoring on quality attributes in the industrial setting. They identified the refactoring techniques and software quality attributes that have been investigated in the industrial environment. They also analyzed the relationships between refactoring and quality attributes.

However, none of the existing reviews has discussed the studies concerned with the categorization of refactoring techniques depending on their influence on quality techniques. Therefore, this study aimed to bridge this gap by identifying and analysing the relevant studies regarding categorization refactoring techniques depending on their influence on quality attributes.

III. METHODOLOGY

The SLR is a well-described methodology to recognizes, chooses and critically analyzes research to answer obviously formulated questions in a manner that is impartial and repeatable. A SLR recommendation given by Kitchenham and Charters (2007) was followed in this study to perform the SLR. Six phases we have taken into consideration for this systematic review procedure as presented in Figure 1. Each phase is explained in the following subsections.

![Fig. 1 SLR Phases (Kitchenham& Charters, 2007)](image)


b) Identifying Inclusion and Exclusion Criteria

Table 1 presents the inclusion and exclusion criteria that have been used to select the primary studies from the retrieved related studies.
Table 1: Inclusion and exclusion criteria

<table>
<thead>
<tr>
<th>Inclusion Criteria</th>
<th>Exclusion Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paper must be on investigating the influence of refactoring techniques on quality attributes. Paper must report empirical results. Paper must be categorized (directly or indirectly) refactoring techniques depending on their influence on quality attributes.</td>
<td>Duplicated papers. Books, thesis, workshop and concept papers. Paper is not written in English. Paper is not available in full text. Paper is not relevant to refactoring techniques. Paper does not a primary study. Paper was published written before the year 2000. Paper that not categorize refactoring techniques, it is not selected as a primary study.</td>
</tr>
</tbody>
</table>

(d) Selecting the Primary Studies

The total number of studies that were retrieved from the five databases was 4430 based on the research keywords. The selection process of studies passed through two stages based on recommendations supplied by Kitchenham and Charters (2007) which are Pre-selection (identification and screening) and selection (eligibility and included). In the pre-selection stage, studies were sieved depending on the titles and abstracts. Consequently, 227 studies were the result of the filtering process in the Pre-selection stage. In the second stage, inclusion and exclusion criteria for the rest of the papers were utilized to discover those papers that were still qualified to the study. Due to this filtering process, we ruled out 207 papers and only 71 papers were eligible and 14 papers among them were selected as primary studies that met the inclusion criteria. Figure 2 demonstrates study selection process.

![Fig. 2 Study selection process](image)

(e) Data Extraction

All papers selected as primary studies were analyzed in depth. Data were extracted from the primary studies based on the research questions mentioned above. Data extraction included the following items: publisher (paper’s indexing source), title, summary of the paper, refactoring techniques used in the categorization, internal software quality metrics /properties /attributes were studied under the effect of the refactoring, and external quality attributes were studied under the influence of the refactoring. The extracted data enabled us to analyze in depth of the 14 studies that were selected in this systematic review to address the research questions.

Figure 3 demonstrates the mapping of the number of published papers for a specific year combined with their publishers.
Categorization Refactoring Techniques based on their Effect on Software Quality Attributes

Fig. 3 Number of Papers Published per Year

f) Data Synthesis

In this section, the primary studies obtained resulting from the systematic review process were summarized. These results were analyzed based on different criteria including refactoring techniques categorized, internal quality attributes/ metrics included, external quality attributes covered, the influence of refactoring on the quality attributes (internal and external), limitations of the reviewed studies, and recommendations for future studies.

IV. RESULTS AND DISCUSSION

The 71 eligibility studies were carefully analyzed to extract the studies that categorized refactoring techniques (directly or indirectly) depending on their influence on quality attributes. As a result, 14 studies were selected. Seven studies (Bois&Mens, 2003; Bois, Demeyer, &Verelst, 2004; Elish&Alshayeb, 2009; Elish&Alshayeb, 2011; Elish&Alshayeb, 2012; Alshayeb, Jamimi, &Elish, 2013; Malhotra & Chug, 2016) have aimed to categorized the refactoring techniques by applying them individually using software projects experiments while the other seven studies (Alshammari, Fidge&Corney, 2010; Fontana & Spinelli,2011; Alshammari, Fidge&Corney, 2012; Halim &Mursanto, 2013; Kannangara&Wijayanayake, 2013; Kumari&Saha, 2014; Chavez, Ferreira, Fernandes, Cedrim, & Garcia, 2017) reported impact the refactoring techniques using one and small software project. Table 1 presents the extracted data from the 14 selected studies which are: the refactoring techniques categorized, quality attributes (internal and external) included, the influence each refactoring technique on these attributes, and the overall impact of refactoring on the quality.

These categorizations suffer from several deficiencies. These deficiencies are: 1) limited to set of refactoring techniques 2) limited to few internal and measured external quality attributes 3) generalizing of the results is difficult due to small program examined by Bois and Mens (2003), only one open source program tested by Bois et al. (2004), limited classes investigated by Elish and Alshayeb (2011), and empirical validation was not conducted, and 4) the estimated external quality attributes were completely ignored.

One of the most obvious finding to emerge from this SLR analysis is that contradictory opinions are still existing about the influence of refactoring on quality. This is because there are 68 different refactoring techniques each one has a specific purpose and impact. When performing refactoring techniques, some software quality attributes improve, and some others impair. As shown in Table 2, it was found three issues under this problem which are: 1) Different refactoring techniques may have a significantly different influence on quality attributes( trade-off), 2) A refactoring technique has a different influence on quality attributes (trade-off), 3) Same refactoring techniques have a different influence on the same quality attributes (e.g., Extract Method in Malhotra & Chug (2016) and Chavez et al. (2017) ). Therefore, categorizing the refactoring techniques depending on their influence on quality attributes (internal and external) needs further research theoretically and practically (Abebe and Yoo, 2014). To benefit from refactoring techniques and using them consistently, there is a need to establish models such as reference and prediction models that can serve as guidelines to help software practitioners to overcome the trade-off challenge. It has been noted that these models relevant to software refactoring domain are missing.
Table 2: The existing studies on categorization refactoring techniques based on their effect on software quality attributes

<table>
<thead>
<tr>
<th>Author and year</th>
<th>Refactoring Techniques</th>
<th>Software Quality Attributes</th>
<th>Overall Impact</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>External Attributes</td>
<td>Internal Attributes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>Bos &amp; Mens (2003)</td>
<td>Extract Method</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bos et al. (2004)</td>
<td>1) Extract Method 2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Full Up Method</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Elsh &amp; Elshourb (2009)</td>
<td>Extract Method</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Consolidate Method</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Extend Method</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Hide Method</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Extract Method</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Consolidate Method</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Hide Method</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Extract Method</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Consolidate Method</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Hide Method</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Categorization Refactoring Techniques based on their Effect on Software Quality Attributes

V. CONCLUSION

We performed a systematic review relevant to the relation between refactoring techniques and quality attributes in the purpose of extracting the studies that categorized the refactoring techniques depending on their influence on the quality attributes. The objectives of SLR are to identify the refactoring techniques categorized, software quality attributes included and to analyze the relationship among them. 14 studies were selected as primary studies and undergone for depth analysis. The obtained results showed there is a lack of studies concerned with the categorization of refactoring techniques depending on their influence on quality attributes. Researchers are recommended to fill up these gaps by proposing reference and prediction models that can be used as guidelines by the software practitioners in selecting suitable refactoring techniques to improve software quality to certain design goals.

REFERENCES


