Automatic program repair: A systematic literature review

A. T. M. Fazlay Rabbi*, Mahbubul Alam Joarder2

1Institute of Information Technology, University of Dhaka, Dhaka, Bangladesh
*Corresponding Author email: bsse0926@iit.du.ac.bd

ABSTRACT

Automatic Program Repair works by generating patches automatically to fix software bugs. This domain is very popular among the software engineering researchers as it helps the developers to mitigate tiresome debugging effort and increase productivity. This domain mostly focuses on how to develop better program repair techniques. However, a systematic literature review is needed to help the researchers for getting specific and scientific overview of this domain. This paper conducts a systematic literature review on 324 papers for analyzing the trendiness and associated factors of this domain. After that, this paper considers 16 papers to thoroughly review and states the evolution of automatic program repair. Hopefully, this study will be a valuable resource for the researchers of this domain.

Keywords: Automatic Program Repair; Bug Fixing; Patch Generation; Systematic Literature Review

1. INTRODUCTION

Automatic Program Repair (APR) is a process where source code level patches are produced for fixing bugs without human intervention (Monperrus, 2018). Generally, debugging and fixing software bugs require huge amount of time and human effort which increase software development cost. That is why, APR can play significant role to reduce the software development time and deliver a successful software within the estimated budget. Moreover, the idea of automatically repairing bugs is also important since it helps to improve software reliability and engineering productivity. In general, APR works by generating patches for the bugs or faults found during development phase. Patches are the transformed code statements from the buggy codes which can potentially fix and remove the bugs. After that the generated patches are validated by checking whether they successfully pass the test cases or not. If they pass all the test cases in the test suite, they are considered as a valid solution for repairing bugs. Fig. 1 shows the work processes of APR in brief.
The research in the domain of APR is popular since it helps the developer to find bugs and getting probable solution to eradicate those bugs. This motivation has driven lots of researchers to improve the automation result over the years. Many have contributed in the optimization problems of different proposed APR techniques. However, the new researchers who are trying to contribute in this section, pass a hard time to analyze what researches have been done and what are yet to explore. This article which provides a systematic literature review on APR will help them to have a head-start in this domain.

In the next sections, we have discussed that how we have operated the literature review and what can be interpreted from this process.

2. METHODOLOGY

Since our topic is related to the software engineering domain, we have followed the guidelines suggested in literatures (Kitchenham et al., 2009; B. A. Kitchenham & Charters, 2007) for software engineering to do systematic literature review. We have gone through the below five basic steps according to the guideline.

2.1. IDENTIFICATION OF GOALS AND RESEARCH QUESTIONS

From the start our goal was to explore the existing literatures and find out the trend, association of other software engineering topics and evolution of APR. For that purpose, we have set the following research questions:

RQ1: What is the research trend in the domain of Automatic Program Repair?

RQ2: How the external and internal factors of APR is related/associated among themselves?

RQ3: To what extent has the APR evolved?
2.2. **Determination of Selection Criteria**

We need to set up some clear inclusion and exclusion conditions so that we can extract relevant articles on the basis of them. We have only included those articles which are

- Related to automatic program repair domain
- Published in conferences and journals
- Published between 2010 to 2021

For the exclusion criteria, we have not considered the studies that are

- Not written in English
- Published as tutorials, books or blogs.
- Available online but not published yet.
- Not in the scope of APR.
- Duplicate in manner.
- Not available as full text via DOI.

This is to be noted that we have intentionally excluded articles published in the current year (2022).

2.3. **Literature Searching**

The popular tool called ‘Harzing’s Publish or Perish’ was used to search relevant literatures automatically from available articles online. For the searching database, Google Scholar and Scopus were considered. In the search options, year 2010 to 2021 was selected and maximum 1000 number of results were allowed. The following keywords were given as input in an ‘OR’ condition manner:

- Automatic program repair
- Bug repair
- Patch generation
The PRISMA diagram (Fig. 2) demonstrates the steps for article selection process. After searching 1000 articles we have systematically narrowed down the number of literatures to review. It should be noted that, though 324 papers are included for meta-analysis, 16 papers were selected to be read entirely for reviewing.

### 2.4. QUALITY ASSESSMENT

To identify the relevance and determine the quality of selected articles the following factors were considered:

1. The articles reflect the goals of this literature review discussed in Section 2.1.
2. The articles describe the study's context namely Automatic Program Repair. For example: Hardware or network related bug fix is not included in this context.
3. The research approach and methodology are explicitly explained.
4. The data collection (if performed) or dataset organization is well explained.
5. Data analysis method is properly explained.
2.5. Data Extraction

Finally, 324 articles have been selected from which relevant data have been extracted to know the trend of the APR research such as number of publications per year, top authors, top venues and internal-external factors. However, considering personal choice through inspection, only 16 papers (Asad et al., 2019; Chen et al., 2021; Goues et al., 2011; Jiang et al., 2021; Kim et al., 2013; Koyuncu et al., 2020; Li et al., 2020; Liu et al., 2019a; Lutellier et al., 2019, 2020; Saha et al., 2017; Svensson & Vrabac, 2019; Vasic et al., 2019; Wang et al., 2020; Wen et al., 2018; Ye et al., 2021) have been reviewed to explain how the APR research evolve over the years.

3. RESULTS AND DISCUSSION

In this section, we have discussed our findings by comping the extracted data. These findings are discussed through answering the research questions.

3.1. What is the Research Trend in the Domain of Automatic Program Repair?

To answer this question, we have analyzed that how many of the 324 papers have published per year (from 2010 to 2021). It is clear that every year its popularity growing which can be implicated from Fig. 3. However, in 2020 there was a declination of published research papers compared to 2019 and 2021. The probable reason could be the COVID pandemic.

![NUMBER OF PUBLICATIONS PER YEAR](image)

Fig. 3. Publications Per Year

To find more about this domain, we have analyzed the top contributing authors and venues for publication. Claire and Martin are doing tremendous job by continuously working on different research problem for APR. Apart from this, most of the publication are from ICSE (holding highest number 50) which is the most prestigious conference in the Software Engineering Domain. Fig. 4 and Table 1 demonstrate these findings.

![TOP 5 CONTRIBUTING AUTHORS](image)

Fig. 4. Top Contributing Authors
Table 1. Top 5 Venues for Automatic Program Repair Research

<table>
<thead>
<tr>
<th>Venue</th>
<th>Number of publications</th>
</tr>
</thead>
<tbody>
<tr>
<td>International Conference on Software Engineering (ICSE)</td>
<td>50</td>
</tr>
<tr>
<td>ACM Joint European Software Engineering Conference and Symposium on the</td>
<td>31</td>
</tr>
<tr>
<td>Foundations of Software Engineering (ESEC/FSE)</td>
<td></td>
</tr>
<tr>
<td>International Symposium on Software Testing and Analysis (ISSTA)</td>
<td>28</td>
</tr>
<tr>
<td>International Conference on Automated Software Engineering (ASE)</td>
<td>26</td>
</tr>
<tr>
<td>IEEE Transactions on Software Engineering (TSE)</td>
<td>14</td>
</tr>
</tbody>
</table>

3.2. **How the External and Internal Factors of APR is Related/Associated Among Themselves?**

We have analyzed all the terms from Title and Abstract of the extracted papers to know which internal and external factors are associated. Fig. 5 shows that there are 3 clusters where mainly the 2nd and 3rd cluster represents the APR territory. From the analysis, we found that repair techniques, patch, data, software repair etc. are the most commonly used words and they are linked also. This indicates that researcher should be wary about how repair techniques work, how dataset is prepared and how the patches are generated. Again, there are some relevant words like impact, processor represents that the implementation should be optimized and consume less processor power. So, the optimization is also a research problem in this domain. The words such as test case or fault indicates there are some related external topics associated with APR which can be Fault localization or Test Automation. So, the researchers need to have knowledge about these domains as well.

![Fig. 5. Overlay Visualization of Terms](image)

3.3. **To what extent has the APR evolved?**

By reviewing the selected 16 papers thoroughly, we have concluded the following findings:

1. Most of the automatic program repair techniques follow Generate and Validate (G&V) approach. In this approach, first patches are generated and then they are validated by checking test case pass or fail scenario. Fig. 1 shows the steps for G&V approach.
2. According to the patch generation technique, APR can be divided broadly into 3 categories. They are –

- Mutation based approach: This approach (Le Goues et al., 2011) generates patches by mutating buggy codes with some pre-defined conditions. The mutation process is very random which is the biggest limitation of this approach. It can be compared to genetic programming for better understanding.
- Template based approach: In this (Liu et al., 2019b; Saha et al., 2017) initially some templates or patterns are defined based on how human developers fix bugs. While generating patches, these templates are applied to buggy code according to context condition of the templates matched with the buggy program. Then the generated candidate patches are validated. From a recent study (Liu et al., 2019b), 15 categories (total 35 patterns) of templates are known till date. For example: Insert cast checking, insert null pointer checking, mutate operators, mutate conditional expression etc.
- Data driven approach: This approach (Chen et al., 2021; Jiang et al., 2021; Lutellier et al., 2020) uses deep learning methods for generating patches. It trains a model with existing dataset of buggy program and corresponding patches. Then it applies the neural model for the new buggy codes to generate patches. Mostly seq2seq model or NMT (Neural Machine Translation) is used for this kind of approach.

3. The patch validation process is the most important and time-consuming process. It runs every patch through the test suite to know that if a patch passes all the test cases in the test suite or not. If it does, then it is considered to have potential to fix the corresponding bugs.

4. Assessment of patches is another field of research in APR domain. Typically, APR techniques suffer from ‘Overfitting Problem’. After passing all the test cases, a patch is known as plausible patch. It is because, though that patch may pass all the test cases, the developers reject that patch for the real-life scenario. The plausible patches which are accepted by the developers are called correct patch, otherwise known as overfitting patches. So, the overfitting patches are basically the false positive results. This low precision (more overfitting patches generated than correct patches) of solution space causes overfitting problem. For the solution researchers have proposed APCA (Automatic Patch Correctness Assessment) techniques (Wang et al., 2020).

- Static APCA: Static features of code are considered for evaluating correct patches. Patch prioritization (ranking correct patch over the overfitting patches) (Asad et al, 2019) is an example for this.
- Dynamic APCA: In this technique, apart from the given test suite, another test suite is generated using automatic test case generation tools. Then the plausible patches are evaluated by analyzing if they are showing same behavior for newly generated test suite as the given one.

5. Mining fix patterns has become a popular research area for APR. The general intuition is the fix patterns or templates are mined dynamically rather than they are predefined.
6. Since the whole process of APR (generating patch, then validating, then correctness assessment) is very complex and time consuming, sometimes a fixed timer is set by the researchers considering the CPU power. The accuracy of the APR technique is evaluated by the patches generated within the given time frame.
7. Most of the current techniques work for JAVA based project. The popular and benchmark datasets for JAVA are:
   - Defects4j: https://github.com/rjust/defects4j
   - Bears: https://github.com/bears-bugs/bears-benchmark

4. LIMITATIONS

This study covers a few aspects of automatic program repair through literature review. As a result, this article fails to provide information about the actual trendiness of APR. Thorough bibliometric analysis may be solution for this limitation. Moreover, papers published in 2022 are not included in this study which potentially hides the up-to-date active topics of APR from the fresh researchers. Mostly JAVA based APR researches have been reviewed which snatches the spotlight from the other programming language. As this domain is very popular and people are actively contributing, it is possible that some important knowledge or review might be missing in this article. Only findings are stated from the review whereas no gap analysis were done. In future, we are targeting gap analysis in this domain so that we can explore the potential research problems of APR.

5. CONCLUSION

Automatic program repair is one of the challenging, yet important domain in the field of Software Engineering. We have focused on providing a systematic literature review for this field in this study. PRISMA framework is used to do the review process and literature selection. From the 1000 paper, we have narrowed down to 324 papers for the meta-analysis purpose. To state the findings 16 papers have been read entirely and thoroughly and an overview has been included by answering the RQ3. In summary, we discussed the review findings with the help of our three research questions. We are hopeful that this article will help the researchers who are interested in APR.

Author Contributions:

Conceptualization, A. T. M. Fazlay Rabbi and Mahbubul Alam Joarder; methodology, Mahbubul Alam Joarder and A. T. M. Fazlay Rabbi; software, A. T. M. Fazlay Rabbi; formal analysis, A. T. M. Fazlay Rabbi; investigation, Mahbubul Alam Joarder; resources, Mahbubul Alam Joarder; writing—original draft preparation, A. T. M. Fazlay Rabbi; writing—review and editing, Mahbubul Alam Joarder; visualization, A. T. M. Fazlay Rabbi; supervision, Mahbubul Alam Joarder; project administration, A. T. M. Fazlay Rabbi. All authors have read and agreed to the published version of the manuscript.

Funding:

This research received no external funding.
Institutional Review Board Statement:
Not applicable.

Informed Consent Statement:
Not applicable.

Data Availability Statement:
Not applicable.

Acknowledgments:
The authors acknowledge the support of Institute of Information Technology, University of Dhaka for providing the facilities used in the preparation of the manuscript.

Conflicts of Interest:
The authors declare no conflict of interest.

Reference:


Symposium on Software Testing and Analysis, 43-54. https://doi.org/10.1145/3293882.3330577


