The new approach for assessing the dependability of mobile applications

Shmatko Oleksandr¹,
Fedorchenko Volodymyr²,
Senik Volodymyr³

¹ PhD, Associate Professor, Associate Professor of the Department of software engineering and intelligent control systems; National Technical University «Kharkiv Polytechnic Institute»; Ukraine
² PhD, Associate Professor, Associate Professor of the Department of Electronic Computers; Kharkiv National University of Radio Electronics; Ukraine
³ Master's degree II Course; National Technical University «Kharkiv Polytechnic Institute»; Ukraine

Abstract.
With the growing reliance on mobile applications, ensuring their dependability has become a crucial aspect of software development. This article introduces a new approach to evaluating the dependability of mobile applications based on the Corcoran Model, a comprehensive framework that considers various aspects of dependability, including performance, reliability, availability, scalability, security, usability, maintainability, and testability. The Corcoran Model provides a systematic way to assess mobile applications by analyzing key metrics and indicators associated with each of these aspects. By utilizing this model, developers and organizations can gain a holistic understanding of an application’s dependability, leading to better decision-making and targeted improvements. Furthermore, this approach promotes increased end-user satisfaction and trust in mobile applications, ultimately contributing to their widespread adoption and success.

Keywords:
Mobile Application software development evaluating the dependability of mobile applications Corcoran Model
Introduction. The term "software reliability" refers to the degree to which the software development life cycle (SDLC) procedures may be managed and controlled to produce trustworthy programmes. Until the exit conditions for the testing procedure are fulfilled, this metric will be utilized. Moreover, software dependability aids in maintaining and forecasting the software's correctness [1]. With the purpose of assessing and quantifying software quality, software reliability engineering was developed. It demonstrates the fault-free operation of the programme [2] [3]. Software dependability models have been steadily enhanced by researchers and practitioners alike.

The following factors make it difficult to assess and foresee the stability of a mobile app. To begin, mobile environments are distinct from traditional desktop and server setups. Second, new forms of flaws are generated by the introduction of functionality and characteristics specific to mobile contexts, such as energy, network, incompatibility, changed and limited Graphical User Interface (GUI), interruption, and notification [4]. Thirdly, there is a wide variety of mobile platforms and hardware options. Fourth, due to consumer demand, mobile app development has sped up and mobile app functionality has become more complicated [5]. And, of course, mobile devices break down when an app is published. Software engineers rely primarily on problem reports supplied by users, in addition to testing.

Researchers must put in more hours analysing software stability to determine its value in mobile apps. More accurate findings and analyses may be obtained by taking mobile app features into account when testing software dependability.

Software engineers, businesses, and academic institutions all have a vested interest in being able to foresee the failure of mobile applications. Thus, we suggest evaluating mobile app dependability based on faults gleaned from bug reports and generating more precise findings.

Literature review. We identified several surveys and systematic literature reviews (SLRs) related to software reliability [4,5–8].

Several studies and SLRs [9,10–13] were found that focused on software dependability.
None of these studies, however, has specifically reviewed the state of the art of dependability in mobile apps; rather, they have all focused on traditional software. In order to determine what constitutes state-of-the-art research in the field of software dependability, Singhal [14] conducted an SLR analysis that included material released up to 2011. A decade ago, when widespread use of mobile applications was just beginning, this was the case. The study organises 141 publications into categories based on their research question, methodology (e.g., survey, theory), and setting (e.g., academic, industrial). Since the information available at the time was insufficient to prove industrial validity, their results recommended that additional industrial research was needed. Due to the lack of a standardised usage of words referring to software reliability, the authors stressed the significance of manual search to locate relevant material in the subject.

In their analysis of the literature from 1990 to 2010, Shahroknii and Feldt [15] focused on software robustness, which is described as a reliability feature in various standards such as IEEE-STD 610.12-1990. For this study, the authors analysed and categorised 144 papers according to the following criteria: research type (e.g., evaluation, experience report), contribution (e.g., tool, metrics), evaluation type (e.g., academic, industrial), and stage of development (e.g., requirements, design and implementation). Lack of research on the elicitation and definition of software resilience requirements was the biggest gap uncovered by the study.

Most research mainly focused on one component of robustness (invalid inputs), whereas others like unexpected events, timeouts, interruptions, and stressful execution settings were completely ignored. Febrero et al. [16] examined 503 papers from 2003 through 2014 as part of a software reliability modelling SMS. Static and Architectural Reliability models, as well as Software Reliability Growth models, Bayesian approaches, test-based methods, AI-based techniques, and other types of reliability models were categorised into five groups.

Finding that many studies did not adhere to the set quality requirements, the investigation uncovered a knowledge
gap. To address this knowledge gap, the same authors undertook a systematic literature review (SLR) of software reliability evaluation using the quality standard ISO/IEC 25000 SQuaRE from 1991-2014.

According to the results of the latter research, adjusting quality standards and reliability to accommodate the interests of many stakeholders has been given scant attention. They also noted that the existing dependability models' complexity prevented them from being used in routine situations. A lack of agreement and different definitions of dependability also hampered the development of useful models. The authors noted, for example, that 'reliability' and 'dependability' were frequently employed as synonyms, despite their differences.

They were more concerned with how reliability models applied reliability requirements (such ISO/IEC-25000 SQuaRE), while our work investigates the current status of reliability in mobile apps. In addition, our research reviews studies conducted during the past six years or so.

Alhazzaa and Andrews [17] performed an up-to-date SMS that looked at reliability growth models that accounted for the development of software systems. They summed up tendencies in terms of publication year, venue, and the nature of the research (academic, industrial). The studies were categorised based on the suggested approach (analytical and curve-fit) and the study style (empirical or non-empirical) as well as solution extent (the kind and quantity of change: single change-point, multiple change-points). In addition, they used Ali et al.'s [18] criteria to assess the reliability of empirical investigations. They suggested that scholars seek for higher quality empirical investigations with more cooperation from industry. Furthermore, these authors recommended further research into the following questions: how far into the future can these models see? When exactly do professionals need to switch models or adjust their settings? These prior studies (including the one by Alhazzaa and Andrews) all agree that solutions lacked industry level validation since they were largely examined in academic contexts without involving or cooperating with practitioners throughout research.
Proposed Model.

Thus, in order to successfully complete one of the main tasks of this work—creating an integral model for evaluating software reliability—we need to develop an idea of which model for analysing software reliability is most appropriate for our project and how statistical data for this model will be obtained.

Of all the models considered for evaluating software reliability, the Corcoran model was chosen as the most appropriate for use in this paper. There are several reasons for this, but the biggest one is the lack of need for additional work (for example, the introduction of artificial errors) and the focus of this model on the use of quantitative statistics about the project.

The Corcoran model is an example of an analytical statistical model of software reliability since it does not use test time parameters and only takes into account the result of N tests in which Ni errors of the i-th type are detected. The model uses variable failure probabilities for different types of errors.

Unlike other methods of this type, the Corcoran model estimates the probability of failure-free programme execution at the time of evaluation [19].

Applying the model involves knowing the following indicators:

- The model contains a non-static probability of failures for different sources of errors and, accordingly, a different probability of their correction;
- The model uses only such parameters as the result of N tests in which Ni errors of the i-th type are observed;
- The detection of the i-th type errors during N tests occurs with probability a_i.

The reliability level indicator R is calculated using the following formula:

\[ R = \frac{N_0}{N} + \sum_{i=1}^{K} \frac{Y_i \times (N_i - 1)}{N}, \]

where \( N_i \) - is the number of failed (or unsuccessful) tests performed in a series of N tests;
$K$ - is a known number of error types;

$Y_i$ - probability of errors

$$Y_i = \begin{cases} a_i, & \text{if } N_i > 0, \\ 0, & \text{if } N_i = 0; \end{cases}$$

$a_i$ - probability of detection of the $i$-th type errors during testing.

In this model, the probability of a given event should be estimated based on a priori information or statistics from the previous period of software operation.

The number of $N_i$ tests for the Corcoran formula for an incomplete set of test reports is defined as:

$$N_i = \frac{R_i \times N_t \times 0.6}{R_t},$$

where $R_i$ - the number of reports imported into the system;

$R_t$ - total number of reports on the Socorro server;

$N_t$ - the total number of product installations.

**Case study.** Below is an example of using this method.

The 100 tests of the programme were conducted. The 20 out of 100 tests were unsuccessful (without failures), and in other cases, the data shown in Table 1 was obtained.

When all the necessary data is calculated, it is necessary to apply the Corcoran formula to find the probability of failure of the programme at the time of evaluation.

<table>
<thead>
<tr>
<th>Error type</th>
<th>Probability of $a_i$ error</th>
<th>Number of $N_i$ errors occurring during testing</th>
<th>$Y_i$</th>
<th>$(Y_i \times (N_i - 1))/N$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. calculation errors</td>
<td>0.09</td>
<td>5</td>
<td>0.09</td>
<td>0.0036</td>
</tr>
<tr>
<td>2. logic errors</td>
<td>0.26</td>
<td>25</td>
<td>0.26</td>
<td>0.0624</td>
</tr>
<tr>
<td>3. I/O errors</td>
<td>0.16</td>
<td>3</td>
<td>0.16</td>
<td>0.0032</td>
</tr>
<tr>
<td>4. Data Manipulation errors</td>
<td>0.18</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5. pairing errors</td>
<td>0.17</td>
<td>11</td>
<td>0.17</td>
<td>0.017</td>
</tr>
<tr>
<td>6. data definition errors</td>
<td>0.08</td>
<td>3</td>
<td>0.08</td>
<td>0.0016</td>
</tr>
<tr>
<td>7. database errors</td>
<td>0.06</td>
<td>4</td>
<td>0.06</td>
<td>0.0018</td>
</tr>
</tbody>
</table>
Table 2

Example of using the Corcoran model (Part 2)

<table>
<thead>
<tr>
<th>Initial data</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>N =</td>
<td>100</td>
</tr>
<tr>
<td>N₀ =</td>
<td>20</td>
</tr>
<tr>
<td>R =</td>
<td>0,2896</td>
</tr>
</tbody>
</table>

Thus, this algorithm requires a tool to analyze data from similar projects or analyze existing statistics of the current project to set the aᵢ parameter. This tool will be data mining methods using sequential template mining. Information about the total number of installations, the number of worlds and error groups, as well as the probabilities of occurrence of each of the error groups will be used to calculate the software reliability indicator.

**Conclusion.**

In conclusion, the Corcoran Model offers a valuable and comprehensive approach to assess the dependability of mobile applications by considering multiple dimensions, such as performance, reliability, availability, scalability, security, usability, maintainability, and testability. By implementing this model, developers and organizations can gain valuable insights into the strengths and weaknesses of their applications, allowing them to make informed decisions and prioritize improvements.

The adoption of the Corcoran Model in the software development process can lead to higher-quality mobile applications, increased end-user satisfaction, and improved trust in the mobile app ecosystem. Additionally, this model can serve as a benchmark for developers and organizations to compare their applications against industry standards and competitors, fostering innovation and the continuous improvement of mobile applications.

In summary, the Corcoran Model for Assessing Mobile Application Dependability represents a significant advancement in the evaluation of mobile apps, enabling organizations to better meet the evolving needs and expectations of users while ensuring a high level of dependability in the increasingly competitive mobile application landscape.
References:


