A Novel Model for Software Risk Mitigation Plan to Improve the Fault Tolerance Process

1Ahdieh Khatavhotan (IEEE / ACM Member)  
2Navid Hashemitaba (IEEE Member)  
3Siew Hock Ow

Department of Software Engineering,  
Faculty of Computer Science and Information Technology,  
University of Malaya  
Kuala Lumpur, Malaysia

1khotan@siswa.um.edu.my  
2nhtaba@siswa.um.edu.my  
3show@um.edu.my

Abstract:

Mitigating the risks might increase the fault tolerance of a system; On the other hand, increasing the fault tolerance might raise the risks occurrence. This study considers the dynamic bidirectional link between the fault tolerance process and risk mitigation of software systems. This research proposes a model for increasing the fault tolerance process of software. This process starts with fault injection, which entails the identification of corresponding risks to potential faults while the core of the model is designing risk mitigation plans to reduce the risks consequences and their occurrence probabilities. A conducted case study showed a sound performance for the model to support increasing the fault tolerance for the software system.

Keywords-component: fault tolerance; risk mitigation; risk consequences; fault injection; fault detection; Failure Occurrence; Contigency Plan

I. INTRODUCTION

Any fault occurrence affects and modifies the performance of the system. If an unperceived fault occurs and the system still resumes its function with normal performance, the error must be identified and the system must compensate for this unfamiliar activity. The most common method for fault tolerance in software systems are the forward and backward recovery fault mechanisms [1].

The methods of reducing risk consequences or decreasing the risk occurrence probability are called risk mitigation techniques. This research determines the mutual impacts of the fault tolerance process and risk mitigation activities and presents a model for increasing the dependability of software fault tolerance with the implementation of a risk mitigation plan. In the following sections, concise descriptions of the fault tolerance process, fault injection and a presented model for IT systems mitigation plan are discussed. Finally a case study has been done to verify the performance of the model.

II. FAULT TOLERANCE AND RISK MITIGATION

There is a bidirectional relation between risk reduction and fault tolerance. An increase in fault tolerance may cause several risks. On the contrary, designing and implementing risk mitigation plans may also increase the tolerance of the faults [2].
Finally, Fault tolerance process defines as some interdependent activities to remove potential errors before the occurrence of a failure [3]. Fault tolerance process has four main steps including error detection, error diagnosis, error isolation, and error recovery which are shown in Fig 1.

![Figure 1. Fault Tolerance Process](image)

The first step, error detection, is identifying the potential error cases. While in the second step which is error diagnosis the assessment of the probable damage will be determined. In error isolation step, the errors are prevented to be propagated. The last step that is error recovery the potential error state will be substituted with an error-free state [4].

III. RISK MITIGATION AND CONTINGENCY PLAN

After The Fault Tolerance Risk Mitigation (FTRM) model is based on the verified and extracted data from information systems. This model splits into four steps as follows:

- **Step 1: Creating risk mitigation plan.** According to obtained information from the previous phases of II project and historical data, a mitigation plan should be designed. This plan reduces the likelihood of risks occurrence and lessens the intensity of the adverse consequences of each risk [5].
- **Step 2: Defining triggers.** Some criteria are defined collectively with assessment routines during continuous monitoring in order to clarify the exact time when a risk occurred [6].
- **Step 3: Designing a contingency plan.** If risks occur, risks characteristics determine what measures should be taken to compensate for their outcome [7].
- **Step 4: Driving the actual risks.** In case of any risk occurrence, a contingency plan must be executed. Concurrently, checklists and reports should be dispatched to risk managers to take upon the immediate necessary modifications.

Fig 2 shows a propped model for risk mitigation in IT and software systems and highlights the four steps mentioned above.
IV. FAULT INJECTION AND SOFTWARE FAULTS

Fault injection is the core of the model for risks diagnosis. By injecting the potential faults in different dimensions, the hidden risks are identified and the software fault tolerance is tested as well. Fig 3 shows probable faults of software [8].

![Figure 2. A Proposed FTRM Model for Risk Mitigation in Software Systems](image)

![Figure 3. Types of Software Faults](image)
V. PERFORMANCE FORMULAE FOR FTRM MODEL

The proposed performance formulae for the model are shown in equations (1) and (2) as follows [9]. However, five used variables of the formulae are defined respectively:

- \(\text{EFJ}(i)\) : Effort for Fault Injection for each Potential Fault
- \(\text{EFA}(i)\) : Effort for Fault Analysis for each Fault Injection
- \(\text{EFR}(i)\) : Effort for Risk Mitigation Activity
- \(\text{EFI}\) : Effort for Integration of the Mitigation Plan
- \(\text{SV}(i)\) : Saving Time by Increasing the Fault Tolerance for each Potential Fault

\[
\text{Total Effort (TE)} = \sum_{i=1}^{\text{all Potentional Fault}} (\text{EFJ}_i + \text{EFA}_i + \text{EFR}_i) + \text{EFI} \tag{1}
\]

\[
\text{Total performance} = \sum_{i=1}^{\text{all Potentional Fault}} \text{SV}_i - (\text{EFJ}_i + \text{EFA}_i + \text{EFR}_i) - \text{EFI} \tag{2}
\]

VI. CASE STUDY

In the case study conducted, the users’ interaction with the system was researched and the plausible errors were identified [10]. When the user chose incorrectly and disrupted the command of a crucial activity and the process was halted in the midst of execution, potential faults were injected into the system.

The first column of Table I illustrates the two injected faults while the second column shows the corresponding risks to the aforementioned faults. Third column respectively indicates the activities of a relevant interanated mitigation plan.

<table>
<thead>
<tr>
<th>Injected Fault</th>
<th>Corresponding Risks</th>
<th>Mitigation Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inaccurate Instructions</td>
<td>(\rightarrow) Wrong Output</td>
<td>(\rightarrow) Informing Users of Selecting Reliable Commands</td>
</tr>
<tr>
<td></td>
<td>(\rightarrow) Wasted Resources</td>
<td>(\rightarrow) Controlling Resources to Prevent Waste</td>
</tr>
<tr>
<td></td>
<td>(\rightarrow) Making a Wrong decision</td>
<td>(\rightarrow) Providing Knowledge-Based Similar Decisions and Presenting the Difference Table</td>
</tr>
<tr>
<td></td>
<td>Based on the Result</td>
<td></td>
</tr>
<tr>
<td>Aborting a Necessary</td>
<td>(\rightarrow) Wasted Resources</td>
<td>(\rightarrow) Creating Back-Up and Revealing Susceptible Activities</td>
</tr>
<tr>
<td>Activity</td>
<td>(\rightarrow) Redundant Work</td>
<td>Caution and Warning</td>
</tr>
<tr>
<td></td>
<td>(\rightarrow) Repair</td>
<td></td>
</tr>
<tr>
<td>Halt the Run-Time Program</td>
<td>(\rightarrow) System Crash</td>
<td>(\rightarrow) Server Back-Up for Recovery Upon Errors</td>
</tr>
<tr>
<td></td>
<td>(\rightarrow) Unreliable Circumstances</td>
<td></td>
</tr>
</tbody>
</table>
VII. PERFORMANCE CALCULATION AND EFFECTIVENESS EVALUATION OF FTRM MODEL

Implementing the model in a real environment and gathering the real data has been the most common fault-tolerance performance benchmark method [11]. To understand the performance of a model, using quantitative formula with clarified interpretation is essential [12]. The performance formulae have been applied in the case study and the results are shown in Table II. As is indicated in the table, there is a 28% improvement in the one time performance of the model.

<table>
<thead>
<tr>
<th>Wrong instructions</th>
<th>EFJ</th>
<th>EFA</th>
<th>EFR</th>
<th>SV</th>
<th>Gross Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aborting a Necessary Activity</td>
<td>230</td>
<td>180</td>
<td>300</td>
<td>1100</td>
<td>55%</td>
</tr>
<tr>
<td>Halt the Run-Time Program</td>
<td>170</td>
<td>90</td>
<td>520</td>
<td>950</td>
<td>22%</td>
</tr>
</tbody>
</table>

EFI = 300
Total Performance = %28

VIII. CONCLUSION

Mitigating the risks might increase the fault tolerance of a system; On the other hand, increasing the fault tolerance might raise the risks occurrence. The proposed model in this research is based on identifying effective factors in fault tolerance, the risks consequences, and presenting solutions to reduce the risks. Accordingly, the conducted case study confirms that by identifying risks corresponding to the fault tolerance, a 20-80% increase in fault tolerance can be achieved.

REFERENCES
