## List of figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Typical hardware and software failure rates over lifetime</td>
<td>3</td>
</tr>
<tr>
<td>1.2</td>
<td>The minefield analogy of software reliability (the mines represent faults in software, and the path represent a single execution flow of the software)</td>
<td>4</td>
</tr>
<tr>
<td>1.3</td>
<td>Typical software architecture of safety applications in nuclear reactors</td>
<td>10</td>
</tr>
<tr>
<td>1.4</td>
<td>Focus of the present study: failures caused due to software faults (indicated by the shaded portion of the venn-diagram)</td>
<td>10</td>
</tr>
<tr>
<td>2.1</td>
<td>An example of two functionally same programs having difference in MC/DC (calculated through code instrumentation), due to short-circuit evaluation by the compiler. For a given set of test cases: function (a) is likely to have lower MC/DC than function (b).</td>
<td>24</td>
</tr>
<tr>
<td>2.2</td>
<td>An example of two functionally same programs having difference in MC/DC by manipulating the way conditions are written. For a given set of test cases: function (a) is likely to have a lower MC/DC than function (b).</td>
<td>24</td>
</tr>
<tr>
<td>2.3</td>
<td>An example where MC/DC and LCSAJ coverage (50%) is greater than the statement coverage (≈ 1%).</td>
<td>25</td>
</tr>
<tr>
<td>2.4</td>
<td>An example of mutant program: (a) the original program, (b) the mutant program (the induced fault is indicated by the red color).</td>
<td>25</td>
</tr>
<tr>
<td>3.1</td>
<td>A fission reaction</td>
<td>32</td>
</tr>
<tr>
<td>3.2</td>
<td>A typical sodium-cooled, pool-type fast reactor</td>
<td>33</td>
</tr>
<tr>
<td>3.3</td>
<td>The flow of fresh fuel subassembly</td>
<td>34</td>
</tr>
</tbody>
</table>
3.4 Logic diagram of the reactor startup system ($c_i$ is one of the condition to be satisfied for the reactor startup, and $s_{ij}$ is the $j^{\text{th}}$ sub-condition of $c_i$). 35

3.5 Steam generator in a sodium cooled fast reactor ............................. 36

3.6 Schematic of the software based Core Temperature Monitoring System (CTMS) .......................................................... 37

3.7 The schematic of Radioactive Gaseous Effluent System (GES) (Here the symbol $\bigtriangledown$ indicates a pneumatic valve, NRV indicates a non-return valve, FM indicates the flow meter, $C_1$ and $C_2$ are the compressors, and the items controlled by the software are indicated by the blue color) .... 38

3.8 Schematic of one of the four independent and identical loops of safety grade decay heat removal system ........................................... 39

4.1 Various states of a nuclear reactor ................................................... 42

5.1 Execution flow in safety-critical software .......................................... 46

5.2 Test case generation using coverage information and genetic algorithms. (The unique execution path test case selection - genetic algorithm cycle is repeated till the required code coverage is achieved) .............. 48

5.3 Technique to select unique execution path test cases using gcc, gcov and md5sum. (The -abcfu arguments to gcc implies to display coverage information of: all blocks, branch probabilities, branch counts, function summaries, and unconditional branches. The .gcov file consist of the coverage information in text format, where as the .gcda file consist of the arc transition counts and other information in binary format) 49

5.4 Genetic algorithms - inspired by the genetic evolution: crossovers and mutations ........................................................................... 50

5.5 Concatenated LCSAJs for the FSHS. (The green colored nodes indicate the LCSAJ points where faults have been induced and caught; the red colored nodes indicate otherwise) ................................. 53
5.6 Concatenated LCSAJs for the RSU. (The green colored nodes indicate the LCSAJ points where faults have been induced and caught; the red colored nodes indicate otherwise) .................................................. 54

5.7 Concatenated LCSAJs for the SGTLD. (The green colored nodes indicate the LCSAJ points where faults have been induced and caught; the red colored nodes indicate otherwise) .................................................. 55

5.8 Concatenated LCSAJs for the CTMS. (The green colored nodes indicate the LCSAJ points where faults have been induced and caught; the red colored nodes indicate otherwise) .................................................. 56

5.9 Concatenated LCSAJs for the GES. (The green colored nodes indicate the LCSAJ points where faults have been induced and caught; the red colored nodes indicate otherwise) .................................................. 57

5.10 Concatenated LCSAJs for the SGDHR. (The green colored nodes indicate the LCSAJ points where faults have been induced and caught; the red colored nodes indicate otherwise) .................................................. 58

5.11 Dynamic analysis of CTMS mutants using: Valgrind and Change in coverage 60

5.12 Static analysis of CTMS mutants using: Splint, Clang, and Cppcheck . . . 61

5.13 Principal component analysis (PCA) of static, dynamic, and coverage analysis of mutants for: FSHS and RSU .................................................. 62

5.14 Principal component analysis (PCA) of static, dynamic, and coverage analysis of mutants for: SGTLD and CTMS .................................................. 63

5.15 Principal component analysis (PCA) of static, dynamic, and coverage analysis of mutants for: GES and SGDHR .................................................. 64

5.16 Example of mutants with similar static, dynamic, and coverage properties: (a) The original program, (b) Mutant-1, and (c) Mutant-2 (the induced faults are indicated by red color). Both Mutant-1 and Mutant-2 share the same coordinates on the Principal Component Analysis (PCA) plot. .................................................. 65

5.17 Algorithm for detecting equivalent mutants .................................................. 66
5.18 Example of equivalent mutant detection: (a) $P$, the original program; (b) $M$, the equivalent mutant of $P$; (c) $P'$, the mutant of $P$; and, (d) $M'$, the mutant of $M$. (The induced faults are indicated by red color) and keywords by blue color) ............................................. 67

6.1 A 3-version system, where each system runs the same software, but compiled by three different compilers ................................................................. 74

6.2 Monte-Carlo method of determining the value of $\pi$ ................................. 75

6.3 Example of paths in a program, where reliability of the path $p_2$ is known (indicated by $\Rightarrow$) ................................................................. 75

6.4 Faults induced in path $p_3$. (The symbol $\Rightarrow$ indicates a path whose reliability is known, and $\star$ indicates an induced fault.) ....................... 77

6.5 Contour graph showing the combination of $x$ and $y$ for various reliability values (0.05-0.99), when test adequacy is 0.99. ................................. 81

6.6 Wilks criteria (Here $\gamma$ indicates the probability that a single experiment output will not fall in the failure domain ($\Omega$)) .............................. 84

7.1 Estimated reliability vs. the defect density (in Kilo Lines of Code (KLOC)) for all the case studies. As the software under test is $\approx$ 1 KLOC, the results for defect density < 1 Defects/KLOC cannot be plotted. (The upper and lower bounds indicate the $\pm$ 1$\sigma$ limit, and the software reliability is in the range $[0,1]$) ................................................ 88

7.2 Estimated reliability vs. the number of induced faults — for FSH and RSU (The upper and lower bounds indicate the $\pm$ 1$\sigma$ limit, and the software reliability is in the range $[0,1]$) ........................................... 89

7.3 Estimated reliability vs. the number of induced faults — for SGTLD and CTMS (The upper and lower bounds indicate the $\pm$ 1$\sigma$ limit, and the software reliability is in the range $[0,1]$) ........................................... 90

7.4 Estimated reliability vs. the number of induced faults — for GES and SGDHR (The upper and lower bounds indicate the $\pm$ 1$\sigma$ limit, and the software reliability is in the range $[0,1]$) ........................................... 91
7.5 Estimated reliability vs. the number of warnings found during static analysis for the all case studies .......................... 92
7.6 Estimated reliability vs. the number of errors found at dynamic analysis for the all case studies ........................................... 92
7.7 Estimated reliability vs. the safety indicator, for all the case studies . . 95
A.1 An example of a function "add", returning the sum of its parameters: A and B ................................................................. 101
A.2 A function call .................................................................................. 101
A.3 Inline and standalone comments .................................................. 101
A.4 Control flow: (a) decision box, (b) switch case ................................. 102
A.5 An example of branches. The order of execution is Task 1,2,3 .......... 102
A.6 Map and filter using list comprehension: (a) Map and (b) Filter ......... 103