Chains can show, for instance, that a function may fail if called from one place and perform successfully when called from another.

```java
public class C {
    public void a(int i) {
        if (i < 0)
            f(i);
        else
            g(i);
    }
    public void b(int i) {
        if (i < 0)
            a(i);
        else
            g(i);
    }
    public void f(int i) {
    }
    public void g(int i) {
    }
}
```

### Motivation
- Debugging is a very cumbersome process.
- SBFL limits: only hit-based in most cases, which neglects any contextual information.
- Lot of research with various algorithms, but only marginal improvements.

### Spectrum-Based Fault Localization (SBFL)
Code elements are more suspicious to contain a fault that are exercised by comparably more failing test cases than passing ones, while non-suspicious elements are traversed mostly by passing tests.

### Context
Chains can show, for instance, that a function may fail if called from one place and perform successfully when called from another.

### Approach
- An SBFL algorithm computes ranking on all occurring call chains during execution.
- Ranked chains can be used at this point in debugging.
- Or, the most suspicious functions can be computed from the chain ranks (two strategies are available).

### Results
- Effectiveness improvement of Ochiai of 1 to 6 positions on average (relative improvement about 45%).
- 76% (33) of the defects with ranks >10 could be reduced to <10 (25) with average reduction of 84%.

<table>
<thead>
<tr>
<th>Program</th>
<th>Bugs</th>
<th>Ochiai rank</th>
<th>Comb. rank</th>
<th>Diff.</th>
<th>Rel. change</th>
<th>Ochiai rank &gt; 10</th>
<th>Reduced to &lt; 10</th>
<th>Rel. impr.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Commons Lang</td>
<td>46</td>
<td>4.7</td>
<td>3.9</td>
<td>-0.8</td>
<td>-17%</td>
<td>6</td>
<td>4 (9%)</td>
<td>-15.1 (-67%)</td>
</tr>
<tr>
<td>Commons Math</td>
<td>74</td>
<td>8.7</td>
<td>3.8</td>
<td>-4.9</td>
<td>-56%</td>
<td>18</td>
<td>15 (20%)</td>
<td>-24.3 (-87%)</td>
</tr>
<tr>
<td>JFreeChart</td>
<td>18</td>
<td>5.3</td>
<td>3.4</td>
<td>-1.9</td>
<td>-36%</td>
<td>2</td>
<td>2 (11%)</td>
<td>-19.0 (-76%)</td>
</tr>
<tr>
<td>Joda-Time</td>
<td>23</td>
<td>13.4</td>
<td>7.8</td>
<td>-5.6</td>
<td>-42%</td>
<td>4</td>
<td>4 (17%)</td>
<td>-43.1 (-88%)</td>
</tr>
<tr>
<td>Total / Average</td>
<td>161</td>
<td>7.8</td>
<td>4.3</td>
<td>-3.5</td>
<td>-45%</td>
<td>33</td>
<td>25 (16%)</td>
<td>-25.4 (-84%)</td>
</tr>
</tbody>
</table>

### Function call chains
Function call chains are snapshots of the call stack occurring during execution and as such can provide valuable context to the fault being traced. Call chains are artifacts which are well known to programmers who perform debugging.

<table>
<thead>
<tr>
<th>Chain</th>
<th>Function Ranks</th>
<th>Rank merge</th>
<th>Function Ranks</th>
</tr>
</thead>
<tbody>
<tr>
<td>b-g</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>a-g</td>
<td>2</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>a-f</td>
<td>3</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>b-a</td>
<td>4</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>b-g</td>
<td>4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Spectrum-Based Fault Localization (SBFL) code elements are more suspicious to contain a fault that are exercised by comparably more failing test cases than passing ones, while non-suspicious elements are traversed mostly by passing tests. Function call chains are snapshots of the call stack occurring during execution and as such can provide valuable context to the fault being traced. Call chains are artifacts which are well known to programmers who perform debugging.