1. Introduction

1.1. Definition

Testing is the act of determining the correctness of developed software. Testing is a more interactive and thorough examination of the software in question than inspection or review. Testing can only find defects in the given software and can never prove the absence of defects. Software testing is more of an investigation and an art form than a rote script that is to be followed. This is because most of the time we are trying to find things of which we are unaware. (Armour 2005) Simply sitting down and putting software through predefined paces will not find errors of which we have not thought, but merely confirm that we have programmed out all the problems of which we have already thought.

1.2. History

1.3. Need for Testing

Good testers have an intuitive feel for what, where, and how much to test. Testing is not a replacement for good engineering.

With the increasing size and complexity of software and a demand for decreasing development cycle times companies are forced into finding new and more effective methods for software testing.

2.1. Units, Systems, and Users

2.1.1. Testing Building Blocks

2.1.2. Alphas, Betas and Release Candidates

2.2. Methodology Descriptions and Matrix

2.2.1. Traditional Structured SDLC Testing

2.2.2. Fault Based Testing

2.2.3. Specification Based Testing

2.2.4. Risk Based Testing

2.2.5. Adaptive Testing

2.2.6. Agile Development

2.2.7. Integrated Testing

2.2.8. Orthogonal Defect Classification

ODC is a methodology that provides a classification scheme for software errors and a set of concepts for guiding the tester through the classified aggregate defect data. (Butcher et. al. 2005)
2.2.9. Collaborative Development Testing

2.2.10. Data Mining Approaches
3. Testing Tools

3.1.1. Testing Automation

3.1.2. Testing Knowledge Management.
4. Best Practices

4.1. Application of Methodologies Matrix

<table>
<thead>
<tr>
<th>Method</th>
<th>Scenario 1</th>
<th>Scenario 2</th>
<th>Scenario 3</th>
<th>Scenario 4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4.2. Organizational Examples of Best Practices

4.2.1. Case Studies

4.2.2. Lessons Learned

4.2.3. Successes
5. Future Research

5.1. Emerging Trends

5.1.1. Scientific and Numerical Analysis

5.1.2. Human Factor Research

5.1.3. Programmer Accountability

5.1.4. Developer Liability

5.2. Research Opportunities Identified