Advanced Software Testing
Solving Test Puzzles with Policies, Strategies, Plans
A series of webinars, this one excerpted from Advanced Software Testing: V2, a book for test managers, and Managing the Testing Process, 3e

Test policies, strategies, and plans are about figuring out how to solve your test puzzle—and indeed what the puzzle is to begin with

In this webinar, we’ll look at these useful tools…
Test Policy

- The “why” of testing
- Gives overall test objectives for organization
- Developed by senior managers (test managers and test stakeholder managers)
- Topics include:
  - Business value of testing
  - Test objectives
  - Effectiveness and efficiency metrics
  - Typical test process
  - Test process improvement
- Short, high-level document
Example: Test Policy

Sample Test Policy

Mission of Testing

To effectively and efficiently provide timely, accurate, and useful quality risk management information and services.

[Company Name Here] Risk-Based Testing Strategy

Depending on project objectives, [company name] selects the degree of quality risk mitigation desired. Quality risks are identified, and risk items are assessed to determine their level of risk. The level of risk determines test effort and test sequencing. Test results are reported in terms of mitigated and unmitigated risks.

Sequential Test Levels Performed by the Best-Qualified Participants

Test levels promote mitigation of quality risk as early as possible and to the highest practical extent.

<table>
<thead>
<tr>
<th>Level</th>
<th>Owner</th>
<th>Objective(s)</th>
<th>Key Areas of Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unit</td>
<td>Development</td>
<td>• Detect defective code in units</td>
<td>Functionality and resource utilization</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Reduce risk of unit failure in Production</td>
<td></td>
</tr>
<tr>
<td>Integration</td>
<td>Development</td>
<td>• Detect defects in unit interfaces</td>
<td>Functionality, data quality, unit interoperability and compatibility, performance</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Reduce risk of dataflow and workflow failures in Production</td>
<td></td>
</tr>
<tr>
<td>System</td>
<td>Risk Mitigation and Quality</td>
<td>• Detect defects in use cases and end-to-end scenarios</td>
<td>Functionality, data quality, performance, reliability, usability, resource utilization, maintainability, installability, portability and interoperability</td>
</tr>
<tr>
<td></td>
<td>Assurance</td>
<td>• Assist in mitigating risk of unmet business requirements in Production</td>
<td></td>
</tr>
<tr>
<td>Acceptance</td>
<td>Business</td>
<td>• Demonstrate readiness for deployment</td>
<td>Functionality</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Detect defects in user workflows *</td>
<td></td>
</tr>
</tbody>
</table>
The Test Process for Each Test Level

**Plan**
- Perform risk analysis
- Select risk mitigation level
- Estimate testing effort
- Plan testing

**Prepare**
- Develop test scripts
- Acquire test data
- Configure test environment
- Determine test sequence

**Perform**
- Acquire test releases
- Install releases in environment
- Execute test scripts
- Track test results

**Perfect**
- Report defects
- Report test results
- Perfect process and product

---

The degree of formality for the various activities will depend on the level of risk associated with the project as a whole.

**Key Process Indicators (KPIs)**

Each group responsible for one or more test levels shall establish KPIs for test activities that address the following areas:

- Defect detection effectiveness
- Risk coverage and sequencing

In consultation with IT management, each group manager shall develop and execute plans for project-by-project alignment of KPIs across groups and long-term plans for test process improvement at each level and across levels.
Test Strategy

The “how” of testing, at a general level
- Managing product and project risks
- Test levels, entry/exit criteria and coverage
- High-level test activities

Types of strategies:
- Analytical
- Model-based
- Methodical
- Process- or standard-compliant
- Reactive
- Consultative
- Regression-averse

Strategies vary by lifecycle, type of application, type of project, software lifecycle model, level of risk, regulatory requirements, short-term vs. long-term, etc.

The test strategy should align with the test policy
Other Contents of Test Strategy Documents

- Integration procedures
- Test specification techniques
- Independence of testing
- Mandatory and optional standards
- Confirmation testing and regression testing
- Test automation
- Reusability of work products
- Test environments
- Test tools
- Test control and reporting
- Test measurements and metrics
- Defect management
- Testware configuration management
- Roles and responsibilities
**Example: Test Strategy and Activities**

<table>
<thead>
<tr>
<th>Initial Risk</th>
<th>Approach</th>
<th>Entry Criteria</th>
<th>Coverage/ Risk Mitigation</th>
<th>Exit Criteria</th>
<th>Residual Risk</th>
</tr>
</thead>
</table>
| Very High    | Pl       | No process, unit test, review variance | MC: Extensive  
Non-MC: Broad  
Regr: Automated | High coverage, confidence, quality | Low to Very Low |
| High         | Au       | Little process, unit test, review variance | MC: Broad  
Non-MC: Cursory  
Regr: Manual | Good coverage, confidence, quality | Low to Very Low |
| Medium       | Ag       | Some process, unit test, review variance | MC: Cursory  
Non-MC: Opportunity  
Regr: None | Fair coverage, confidence, quality | Low to Very Low |
| Low          | Pb       | Discretionary process, unit test, review variance | MC: Opportunity  
Non-MC: None  
Regr: None | Low coverage, confidence, quality | Low to Very Low |
| Very Low     | Hg       | -              | No testing                | -             | Very Low      |
## Example: Test Strategy and Activities 2

<table>
<thead>
<tr>
<th>Initial Risk</th>
<th>Selected Approach and Consequent Residual Risk</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Pl</td>
</tr>
<tr>
<td>Very High</td>
<td>Low to Very Low</td>
</tr>
<tr>
<td>High</td>
<td>Very Low</td>
</tr>
<tr>
<td>Medium</td>
<td>Very Low</td>
</tr>
<tr>
<td>Low</td>
<td>Very Low</td>
</tr>
<tr>
<td>Very Low</td>
<td>Very Low</td>
</tr>
</tbody>
</table>

This table shows the residual risk consequences of selecting particular approaches which vary from the suggested initial risk/approach combinations shown on the previous slide. Selecting a weaker approach will leave a higher level of risk in exchange for a quicker release.
Developing Test Plans

Why write test plans?
- Confront challenges and crystallize thinking
- Communicate plan to testers, peers, managers

Consider multiple test plans when tests have…
- Different time periods
- Different methodologies and tools
- Different objectives
- Different audiences

…but then you may want a master test plan

Circulate one or two drafts
- Promotes early feedback and discussion
- Prevents wasted time if you’re on the wrong track
Contents of Test Plans

- Items to be tested and not to be tested
- Quality characteristics to be tested and not to be tested
- Relationships among the levels
- Scope of testing at each level, including items and quality characteristics
- Testing schedule and budget
- Test execution cycles (cross-referencing the software release plan)
- Relationships and deliverables between testing and stakeholders/participants
- Entry, continuation, and exit criteria for each level
- Ownership for each test level
- Deliverables to/from each test level
- Test-related project risks (including actions to be taken)
- Test governance
Organizational Setting Diagram
Transitions: Entry Criteria

- Entry criteria measure whether the system is ready for a particular test level
  - Necessary documentation, design, and requirements information provided?
  - System under test ready for delivery to test?
  - Supporting utilities, accessories, and prerequisites available?
  - System at appropriate level of quality?
  - Test environment ready?
- These tend to become increasingly rigorous at the higher levels
Sample Entry Criteria

System Test can begin when:
1. Bug tracking and test tracking systems are in place.
2. All components are under formal, automated configuration and release management control.
3. The Operations team has configured the System Test server environment, including all target hardware components and subsystems. The Test Team has been provided with appropriate access to these systems.
4. The Development Teams have completed all features and bug fixes scheduled for release.
5. The Development Teams have unit-tested all features and bug fixes scheduled for release.
6. Less than 50 must-fix bugs (per Sales, Marketing, and Customer Service) are open against the release.
7. The Development Teams provide software to the Test Team 3 business days prior to starting System Test.
8. The Test Team completes a 3 day “smoke test” and reports on the results to the System Test Phase Entry meeting.
9. The Project Management Team agrees in a System Test Phase Entry Meeting to proceed. The following topics will be resolved in the meeting:
   ✓ Whether code is complete.
   ✓ Whether unit-testing is complete.
   ✓ Assign a target fix date for any known “must-fix” bugs (no later than 1 week after System Test Phase Entry).
Transitions: Exit Criteria

Exit criteria measure whether the test level has achieved its objectives

- Assess quality, completeness of testing
- Facilitate discussion about business impact of shipping now with current quality/risk, delaying for reduced quality risk
- Present alternative plans for moving forward if shipping is delayed

Exit criteria should be clearly aligned with the objectives set out in the test policy
**Sample Exit Criteria**

System Test will end when:
1. No changes (design/code/features), except to address System Test defects, occurred in the prior 3 weeks.
2. No panic, crash, halt, wedge, unexpected process termination, or other stoppage of processing has occurred on any server software or hardware for the previous 3 weeks.
3. No client systems have become inoperable due to a failed update during System Test.
4. The Test Team has executed all the planned tests against the GA-candidate software.
5. The Development Teams have resolved all “must-fix” bugs per Sales, Marketing, and Customer Service.
6. The Test Team has checked that all issues in the bug tracking system are either closed or deferred, and, where appropriate, verified by regression and confirmation testing.
7. The test metrics indicate: product stability and reliability; completion of all planned tests; adequate coverage of the critical quality risks.
8. The Project Management Team agrees that the product, as defined during the final cycle of System Test, will satisfy the customer’s reasonable expectations of quality.
9. The Project Management Team holds a System Test Phase Exit Meeting and agrees that we have completed System Test.
Test Release Management

- Release schedule (i.e., weekly? daily? hourly?)
- Update apply (process to install new build)
- Update unapply (process to remove bad build)
- Build naming (revision level); e.g., X.01.017
- Interrogation (process to determine rev. level)
- Synchronizing with databases, other systems, etc.
- Roles and responsibilities for each step
Managing Test-related Project Risks

The test manager should address test-related project risks in the test plan.

For any risk, you have four options:

- Mitigate via preventive actions
- Create contingency plans (with a trigger and owner)
- Transfer the risk
- Ignore or accept

Any option can provide benefits and opportunities, along with costs and additional risks.
Lifecycle Considerations

- Test policy should consider lifecycle in terms of test process, when metrics are measured
- Test strategy must take lifecycle into account, and provide for integration of testing into the lifecycle
- In sequential lifecycles:
  - Test plan usually written about the same time as the project plan
  - Plan defines exit criteria, aligns test activities with project lifecycle
- In iterative lifecycles (e.g., RUP):
  - Test plan written during project inception
  - Revised or appended at the start of each iteration
- In Agile lifecycles:
  - Test plan written at the beginning of a sequence of iterations
  - Revised or appended at the start of each iteration
Conclusion

In this webinar, we’ve seen how test policies, strategies, and plans can guide our testing.

By working with project participants and stakeholders, we can define what we want to accomplish with testing and how best to accomplish it.

Alignment across these documents is key.

Relevance, conciseness, and focus are also essential.
Contact RBCS

For almost 20 years, RBCS has delivered consulting, outsourcing and training services to clients, helping them with software and hardware testing. Employing the industry’s most experienced and recognized consultants, RBCS advises its clients, trains their employees, conducts product testing, builds and improves testing groups, and hires testing staff for hundreds of clients worldwide. Ranging from Fortune 20 companies to start-ups, RBCS clients save time and money through improved product development, decreased tech support calls, improved corporate reputation and more. To learn more about RBCS, visit www.rbcs-us.com.

Address: RBCS, Inc.
31520 Beck Road
Bulverde, TX 78163-3911
USA

Phone: +1 (830) 438-4830
Fax: +1 (830) 438-4831
E-mail: info@rbcs-us.com
Web: www.rbcs-us.com