

## Test Vector - Are You Testing in the Right Direction?

*Un-vectored testing potentially wastes time, money, and other resources*

By Randy Raymond

### Funny Story

A bicyclist pulls along side a jogger running down the road and asks the jogger "how fast are you running?" The jogger looks at the GPS training device on his wrist and replies "7.25 miles per hour." The bicyclist then asks the jogger "where are you going?" The jogger replies "I'm going this way as fast as I can."

The bicyclist rides away saying to herself "I wonder where 'this way' is and when the jogger knows he has reached it."

### Introduction

Seasoned software testing veterans understand validating requirements are a critical success factor for software testing. Too often organizations focus on the volume of test cases executed and/or the number of defects recorded by testing in a given test cycle. Defect slippage to production is used to determine the effectiveness of testing.

Managers focus on test case "volume" metrics to fix problems, erroneously assuming that test case coverage has been established through requirements traceability. The volume of test cases, defects found, and defects not caught in testing are examined. With all of these measures in place there is an assumption that test cases are mapped to requirements. Testing management and project management assume that there are test cases for each requirement and the test cases actually validate those requirements. With such a large volume of test cases we must have covered everything, right?

In all instances the volume of something other than the volume of requirements validated is examined in attempts to mitigate project distress. A test case-centric or defect-centric view of test planning is not focusing on the right success factors.

This whitepaper introduces a new perspective for discussing test planning with traceability to create a requirements-centric view, with accompanying vocabulary of testing and reporting results.

## Concepts from Physics and Software Testing

Engineers use mathematics and physics, among other sciences, to apply established principles in designing practical solutions to technical problems. Software testers can view the process surrounding requirements testing the same way an engineer views mathematics and physics. By applying analogies for their test processes, test planning, and test execution activities to concepts in physics, testers can control the various components in the testing process and create positive outcomes.

### *Concept: Test Direction*

Test Direction describes testing the “right things” for a given test cycle. In software testing these “things” are requirements. Requirements validation should be the goal of all software testing. If you are not testing to validate the requirements of the system then you are testing in the “wrong direction.”

Here are a few examples of statements of Test Direction:

**Test Direction = High priority requirements**

**or**

**Test Direction = Critical severity defects**

**or**

**Test Direction = Critical severity defects against high priority requirements**

There can be many test directions. Test Direction accommodates both new code and defect fixes. The “right direction” in this case is a plan to validate requirements. Test Direction is a requirements-centric statement of what shall be tested during a given test cycle.

Sample Test Direction measures:

- High priority requirements
- Medium priority requirements
- High priority requirements for new code, medium priority requirements for regression testing
- High and medium priority requirements for new code

The “right direction” also applies to testing cycles where defects fixes are involved. Defect severity and requirement priority set the Test Direction.

Sample Test Direction measures for test cycles with defect fixes:

- Critical severity defects against high priority requirements
- Critical severity defects against medium priority requirements
- High severity defects against high priority requirements

- High severity defects against medium priority requirements

Ideally, defects are mapped to the test cases that discovered them and those test cases are in turn mapped to requirements.

### *Concept: Test Magnitude*

In mathematics, magnitude is the size of an object. Test Magnitude describes the number (size) of test cases (object) that have been written to test a software system.

### **Test Magnitude = Number of test cases**

Sample Test Magnitude measures:

- Number of total test cases to test critical severity defects
- Number of manual test cases to test high severity defects
- Number of automated test cases to test critical severity defects
- Number of multi-use test cases to test high severity defects

### *Concept: Test Vector*

A vector is an object with both direction and magnitude. Test Vector describes the magnitude and direction of a testing effort. That is to say, Test Vector describes the number of test cases executed (Test Magnitude) that have direct links to requirements (Test Direction) to be validated.

Typical Test Vector measures:

- Number of total test cases (Test Magnitude) to test [number] of high priority requirements (Test Direction)
- Number of manual test cases (Test Magnitude) to test [number] of high priority requirements (Test Direction)
- Number of automated test cases (Test Magnitude) to test [number] of high priority requirements (Test Direction)
- Number of test cases (Test Magnitude) to test the [specified] severity defect fixes mapped high priority requirements (Test Direction)

Test Vector answers the question “how many test cases do I need to execute in order to validate all of the requirements and defect fixes?” Test Vector is always stated in terms of requirements since Test Direction is part of the definition of Test Vector.

In this paper's perspective, defects equate to requirements since a defect should be mapped to a test case which is in turn mapped to a requirement.

## **Test Sizing and Planning with Test Vector**

Test Vector can assist with sizing and planning a test cycle. Starting with a list of requirements and/or defects to be validated the requirements trace matrix will provide linkage to the test cases that need to be executed. This answers the question "how many test cases do I need to execute in order to validate all of the requirements and defect fixes?"

Risk-based testing can also be described by the Test Vector. By reviewing requirement priorities and defect severity and then deciding the appropriate level of testing for the test cycle, a suitable Test Vector can be determined.

In both planning activities the Test Magnitude component of the Test Vector will provide the number of test cases that needs to be executed in order to validate the requirements test management has selected for a given test cycle.

## **Requirements Centric Discussion**

The term "Test Vector" is modeled after the term in software malware "threat vector." Threat vector is the method malicious code propagates and infects a computer. Test Vector is the method that testing propagates via test cases and validates requirements in a given test cycle. This gives all stakeholders a requirements-centric vocabulary. Test Vector describes the test plan, test execution, and test results in terms of requirements being validated.

Test plans should be created to validate software requirements. Test results should be reported in terms of requirements tested, requirements passed (validated), requirements failed (business risk), and requirements not tested (business risk).

## **Back to the Funny Story**

In the funny story at the beginning of this article the bicyclist is a seasoned veteran test professional. The jogger is a typical test manager in a busy testing department who is under pressure from the PMO to produce results quickly. The GPS training device is test case execution reports detailing the number of test cases and daily rate of test cases executed. "This way" is what the test manager is doing with testing - executing all of the test cases. It's implied that everyone

knows that "this way" is north, south, east, west, or some intermediate direction although nobody actually has a compass to validate direction. 7.5 mph is the speed of execution shown on the test case execution reports.

What is missing is a direction (which test cases) and destination (validating which requirements).

## **Vectoring in the Right Direction**

Using the Test Vector vocabulary to describe testing in terms of requirements validated takes testing in the right direction. Focusing on the volume of test cases executed, number of defects found, and the number of defects in production puts the spotlight on test cases, without regard to the requirements the software is designed to satisfy. Un-vectorized testing, that is to say testing as many test cases as you can as fast as the team can execute them, is potentially a waste of scarce resources, time, and money.

Alternatively, the Test Vector vocabulary establishes that requirements validation should be the primary concern for all stakeholders.

What is the Test Vector for this test cycle? We're executing 473 (magnitude) test cases to validate all 326 high priority business requirements and 37 critical defects that map to high priority business requirements (direction).

## **Conclusion**

Test Vector is a statement of the requirements and/or defect fixes that will be validated for a given test cycle. These requirements and defect fixes map to the test cases that will validate whether a requirement has been met or the defect traced to a requirement has been repaired. Using a requirements trace matrix we can determine the number of test cases needed to accomplish testing for the test cycle.

By using a vocabulary to describe test planning, test execution, and test results in terms of requirements validated, stakeholders focus on software functionality that is important to the project instead of the volume and speed of test cases run and the volume of defects discovered.

Test Vector changes the conversation on test planning and results reporting while creating positive software delivery outcomes.