NET.OBJECTDAYS 2004

Measuring the Effectiveness of a Test

(Converting Software Testing from an Art to a Science)

Harry M. Sneed
Software Test Engineer
ANECON GmbH., Vienna, Austria

Abstract: The proposed paper presents a set of metrics developed by the author while working as a test consultant for a Viennese software house from 1998 until 2003. They were intended to be used to measure the performance of the test department there, but they are equally valid for measuring test operations anywhere. In fact, with these metrics it should be possible to convert software testing from an art as perceived by Glenford Meyers in 1975 to a science as defined by Lord Kelvin in 1875. The metrics were obtained using the Goal/Question/Metric Method of Basili and Rombach and were refined through many years of practical application. They are supported by a set of tools designed for both static and dynamic analysis as well as for evaluating the results of both.

Keywords: *Test Management, Test Objectives, Defect Analysis, Test Coverage, Software Metrics, Test Metrics.*

PRESENTATION

1) Lord Kelvin on Measurement
2) Tom DeMarco on Measurement
3) Test Metric Categories
4) Testability at the Unit Test Level
5) Testability at the Integration Test Level
6) Testability at the System Test Level
7) Measuring the Complexity of Test Cases
8) Measuring the Quality of Test Cases
9) Test Case Analysis Report for FIVS
10) FIVS Test Case Quantity
11) FIVS Test Case Complexity & Quality
12) Calculating Test Costs
13) Estimating the Number of Test Cases
14) Calculating Test Effort with COCOMO-II
15) Metrics for Measuring Test Coverage
16) Metrics for Evaluating Test Effectiveness
17) Ratio of Tester to User Error Reports
18) Test Metrics from the GEOS Project
19) Defect Analysis in the GEOS Project
20) Test Metric Conclusion
21) More Research on Test Metrics required

Lord Kelvin on Measurement

„When you can measure what you are speaking about, and express it in numbers, you know something about it, but when you cannot measure it, when you can not express it in numbers, then your knowledge is of a meagure and unsatisfactory kind.“

from Lord Kelvin

British physicist, 1882

Tom DeMarco on Measurement

„You can not control what you can not measure. Mesurement is the prerequisite to management control. “

from Tom DeMarco

American Consultant, 1982

Test Metric Categories

- Metrics for assessing the testability of the software
- Metrics for evaluating test cases
- Metrics for calculating test costs
- Metrics for measuring test coverage
- Metrics for assessing test effectiveness

Testability at the Unit Test Level

- Unit Complexity = Size * Cohesion * Coupling
- Control path complexity = Control flow branches / Statements
- Interface complexity = Interfaces + Parameters / Statements
- Data complexity = Conditional variables / Variables used
- Unit Testability = 1 - Average (Unit-Complexity, Control-Flow-Complexity, Interface-Complexity, Data-Complexity)

Testability at the Integration Test Level

- Interface volume = Interfaces / Interfaces + Components
- Interface complexity = Parameters / Parameters + Interfaces
- Database access frequency = Components without Database accesses / Components
- Interface Visibility = invisible Interfaces / Interfaces
- Integration Testability = 1 - Average (Interface-Volume, Interface-Complexity, Database-Access Frequency, Interface-Visibility)

Testability at the System Test Level

- User Interface Volume = User Interfaces + Controls / System Variables
- System Interface Volume = System Interfaces + Data Elements / System Variables
- Database Volume = Tables + Attributes / System Variables
- UseCase Volume = UseCases / System Functions
- System Testability = 1 - Average (User-Interface-Volume, System-Interface-Volume, Database-Volume, UseCase-Volume)

Measuring the Complexity of Test Cases

- Test data complexity = test data types / test data instances
- Test data density = test control variables / test data
- Test case volume = 1 – (test cases / test data instances)
- Test case intensity = 1 – (use cases / test cases)
- Test case complexity = Average (Test data complexity, Test data density, Test case volume, Test case intensity)

Measuring the Quality of Test Cases

- Test case impact = 1 – (test cases / impacted functions)
- Test case reusability = (automated test cases / test cases)
- Test case conformity = (formally correct test case attributes / total test case attributes)
- Test case affectivity = (weighted errors detected / test cases executed)
- Test case quality = Average (Test case impact, test case reusability, test case conformity, test case affectivity)

Test Case Analysis Report for FIVS

```
+-----------------------------------------------------+
| Module:   GWMBRACO   Number of Test Cases =   106   |
| Module:   GWMDERIV   Number of Test Cases =   761   |
| Module:   GWMEMIBE   Number of Test Cases =   128   |
| Module:   GWMEMISS   Number of Test Cases =   325   |
| Module:   GWMEXDAT   Number of Test Cases =   167   |
| Module:   GWMFETAG   Number of Test Cases =   139   |
| Module:   GWMFI      Number of Test Cases =  3070   |
| Module:   GWMFIBEZ   Number of Test Cases =   880   |
| Module:   GWMFIKAT   Number of Test Cases =   597   |
| Module:   GWMFIKNU   Number of Test Cases =   341   |
| Module:   GWMIDENT   Number of Test Cases =   886   |
| Module:   GWMINDX    Number of Test Cases =   838   |
| Module:   GWMINSKA   Number of Test Cases =   168   |
| Module:   GWMINVRL   Number of Test Cases =    40   |
| Module:   GWMKURS    Number of Test Cases =   133   |
| Module:   GWMRAFWZ   Number of Test Cases =   240   |
+-----------------------------------------------------+
| Modules =    167     Number of Test Cases = 58931   |
+-----------------------------------------------------+
```

FIVS Test Case Quantity

```
Test Case Quantity
+-----------------------------------------------------+
| FIVS     Total Number of Functions tested   =   217 |
| FIVS     Total Number of Modules  tested    =   167 |
| FIVS     Total Number of Projects tested    =    10 |
| FIVS     Total Number of System   TestProcs =   259 |
| FIVS     Total Number of System   TestCases = 13634 |
| FIVS     Total Number of Online   TestCases =  5689 |
| FIVS     Total Number of Batch    TestCases =    49 |
| FIVS     Total Number of Interfac TestCases =  7896 |
| FIVS     Total Number of Testcase Types     =     7 |
| FIVS     Total Number of Test  Deficiencies = 36309 |
| FIVS     Total Number of Major Deficiencies =  7645 |
| FIVS     Total Number of Media Deficiencies =   276 |
| FIVS     Total Number of Minor Deficiencies = 28388 |
+-----------------------------------------------------+
```

FIVS Test Case Complexity & Qualtity

```
Test Case Complexity
+-----------------------------------------------------------+
| FIVS     Testcase  Data Complexity Ratio    =     0.765   |
| FIVS     Testcase  Test Density    Ratio    =     0.554   |
| FIVS     Testcase  Test Intensity  Ratio    =     0.810   |
| FIVS     Testcase  Test Volumne    Ratio    =     0.231   |
| FIVS     Overall   Test Complexity Rating   =     0.590   |
+-----------------------------------------------------------+
Test Case Quality
+-----------------------------------------------------------+
| FIVS     Testcase  Impact          Ratio    =     0.769   |
| FIVS     TestCase  Reusability     Ratio    =     0.432   |
| FIVS     TestCase  Conformity      Ratio    =     0.560   |
| FIVS     TestCase  Coverage        Ratio    =     0.984   |
| FIVS     Overall   Test Quality    Rating   =     0.686   |
+-----------------------------------------------------------+
```

Calculating Test Costs

Primary factors for estimating Test costs

- Number of Test Cases required
- Testability of the Software
- Test Productivity = Test Cases/Tester Days

Estimating the Number of Test Cases

- **Blackbox-Test Cases** = {UseCases x Steps x Rules }
+ {GUI‘s x Objects x States }
+ {DB-Tables x Tuples x Instances }

- **Greybox-Test Cases** = {Interfaces x Parameters x Values }

- **Whitebox-Test Cases** = {Methods x Method Invocations }
+ {Objects x Object states }
| Control paths

Calculating Test Effort with COCOMO-II

$$\text{Test Effort} = \text{ST} \left\{ \frac{\text{Number Test Cases}}{\text{Test Productivity}} \right\} ^{**\ \text{SE}} \times \text{Testability Factor}$$

Where ST = System Type (0,5:4)

and SE = Scaling Exponent (0,91:1,23)

Testability Factor = 0,5 / Testability Ratio

If the standard test productivity = 20 test cases per day and there are 1000 test cases to be tested and the testability ratio is 0.4 with a scaling exponent of 1,10 for a distributed system the testing effort will be:

(((1000/20 = 50) ** 1.10 = 74) x 1.25 = 92.5) x 2 = 185 Days

Metrics for Measuring Test Coverage

$$\textbf{Requirements Coverage} = \frac{\text{Tested Requirements}}{\text{Specified Requirements}}$$

$$\textbf{Architectural Coverage} = \frac{\text{Tested Architectural Features}}{\text{Architectural Features}}$$

$$\textbf{Code Coverage} = \frac{\text{Tested Statements, Branches, Paths}}{\text{Statements, Branches, Paths, States}}$$

$$\textbf{Test Case Coverage} = \frac{\text{Executed Test Cases}}{\text{Specified Test Cases}}$$

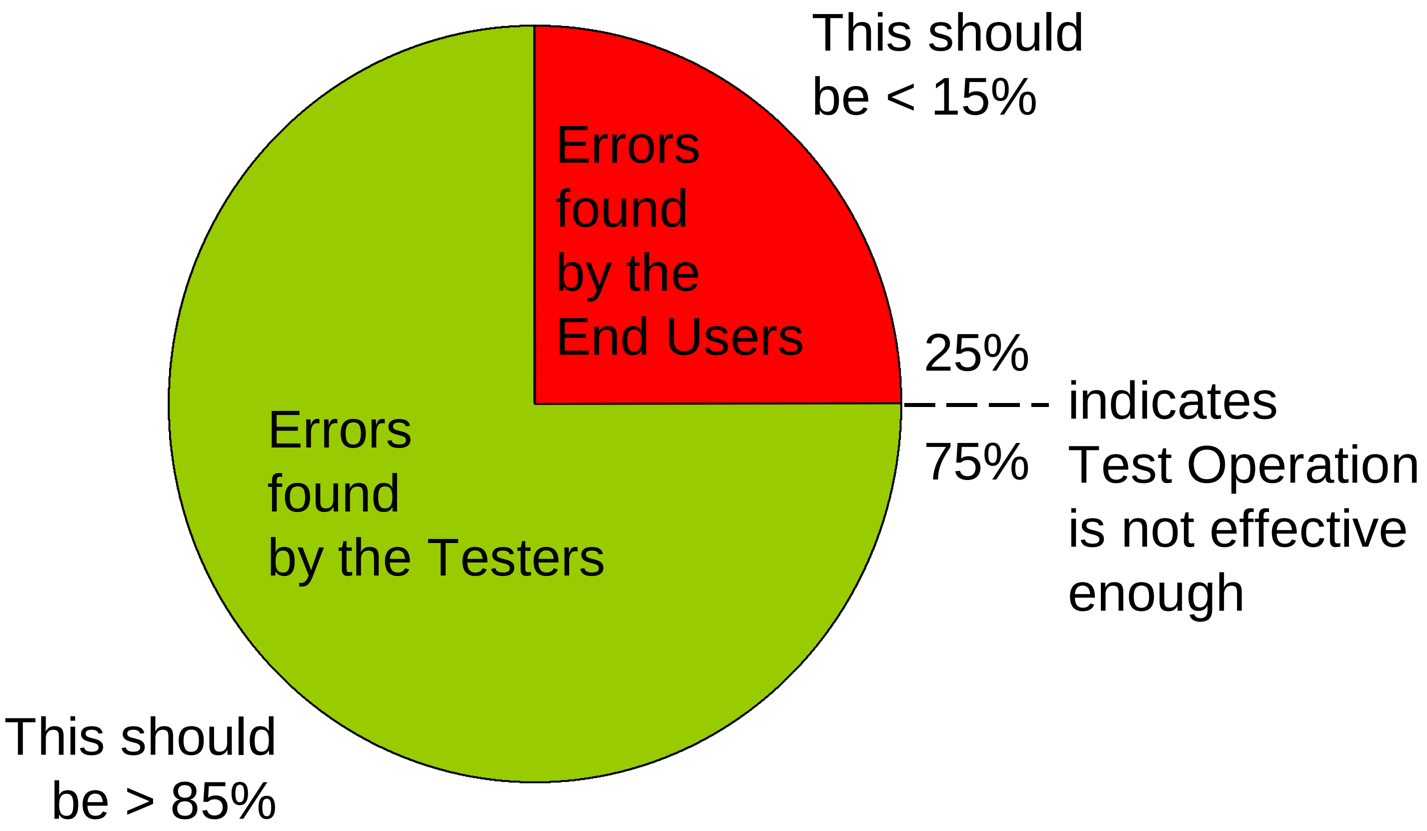
Metrics for evaluating Test Effectiveness

$$\textbf{Test Effectivness} = \frac{\text{Weighted Errors reported by Testers}}{\text{Total weighted Errors reported}}$$

Total weighted Errors = Tester reported + User reported Errors

$$\textbf{Test Confidence} = 1 - \left\{ \frac{\text{weighted Errors}}{\text{executed Test Cases}} \right\} \times \text{Test Coverage Rate}$$

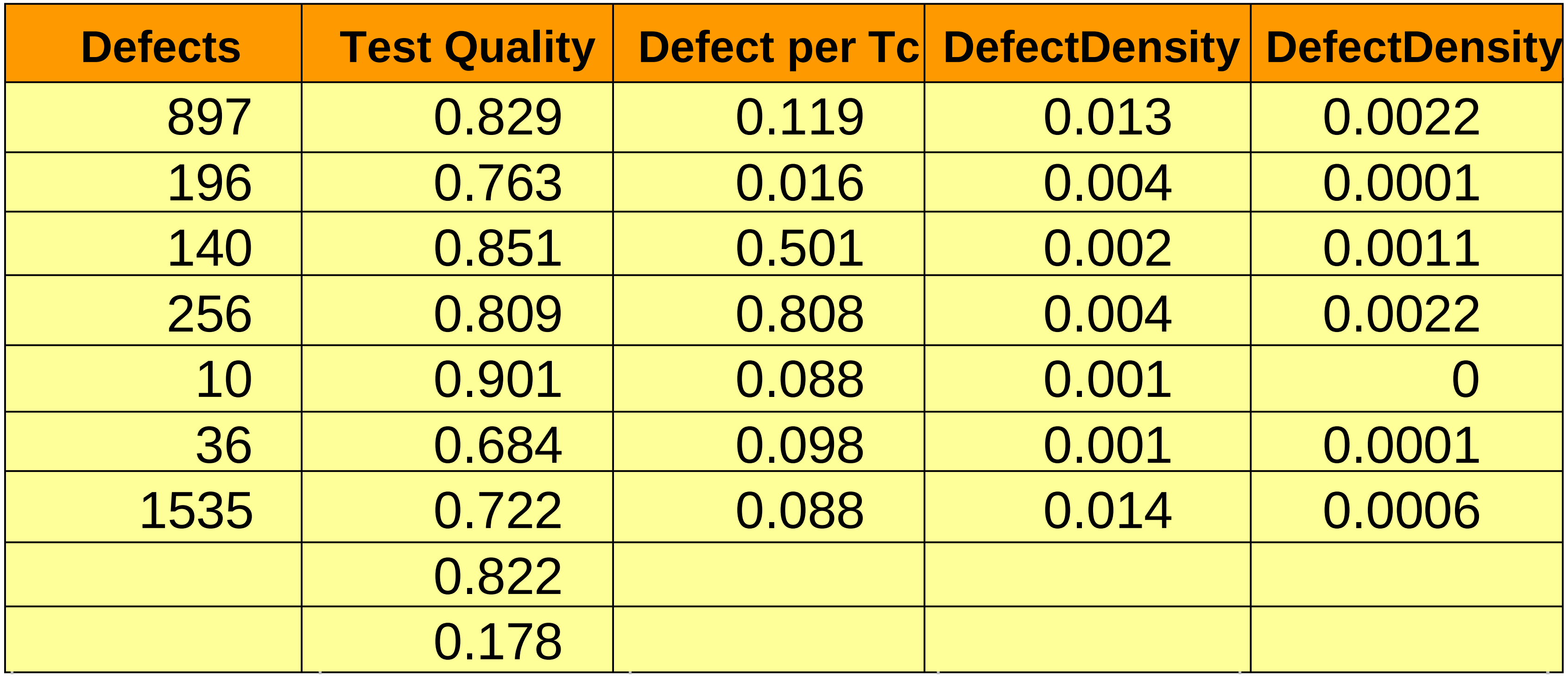
Ratio of Tester to User Error Reports



Test Metrics from the GEOS Project

GUI Panels	Reports	Parameters	Test Data	TestCases (soll)	TestCases
225	94	674882	788384	39232	36735
363	400	307879	761315	38942	37521
67	28	37118	16862	2845	1411
35	46	3719	598	1703	1343
104	37	148796	78300	2177	2145
111	10	129220	95579	925	793
905	615	1301614	1741038	85824	79948

Defect Analysis in the GEOS Project



Defects	Test Quality	Defect per Tc	DefectDensity	DefectDensity
897	0.829	0.119	0.013	0.0022
196	0.763	0.016	0.004	0.0001
140	0.851	0.501	0.002	0.0011
256	0.809	0.808	0.004	0.0022
10	0.901	0.088	0.001	0
36	0.684	0.098	0.001	0.0001
1535	0.722	0.088	0.014	0.0006
	0.822			
	0.178			

Comparison of SubSystems

Test Metric Conclusion

Five categories of Test Metrics have been presented here:

- Metrics for measuring Testability
- Metrics for assessing the quantity, quality & complexity of the Test Cases
- Metrics for calculating Test Costs
- Metrics for measuring Test Coverage
- Metrics for assessing the benefits of the Test Operation

More Research on Test Metrics required

The metrics presented here are intended to make testing more transparent. Testing has evolved into a major resource consumer and cost driver. Many managers are beginning to ask what they are getting for the money they are investing in testing. What are the benefits of testing? This study has offered two metrics for helping to answer that question. There are certainly others to be discovered. That is why this study can only be considered as a first step in the process of transforming software testing from an art into a science. With more research coupled with empirical studies it may someday even be possible to understand what we are doing according to the criteria of Lord Kelvin.