

2 At a glance

1	PREFACE	3
2	AT A GLANCE.....	5
3	TABLE OF CONTENTS	9
4	INTRODUCTION	17
5	WHY TEST SOFTWARE?	19
5.1	WHY TEST SOFTWARE?	19
5.2	LIMITATIONS OF TESTING.....	20
5.3	ALTERNATIVE TO TESTING – PROVING	38
5.4	THREE OBJECTIVES OF TESTING SOFTWARE.....	39
6	WHAT’S SOFTWARE TESTING ABOUT?	43
6.1	THE DIFFERENCE BETWEEN A TESTER AND A DEVELOPER	43
6.2	THE THREE-HEADED DRAGON OF SOFTWARE DEVELOPMENT	46
6.3	THE LANGUAGE OF SOFTWARE TESTING.....	48
6.4	A BUG’S LIFE	54
6.5	SEVERITY OF BUGS.....	56
6.6	THE KINDS OF BUGS	59
7	STAGES OF SOFTWARE DEVELOPMENT FOR A TESTER	72

7.1	CHALLENGING REQUIREMENTS.....	72
7.2	CHALLENGING SPECIFICATIONS.....	73
7.3	CHALLENGING ARCHITECTURE AND TECHNICAL DESIGN	79
7.4	TESTING CODE.....	81
7.5	CHALLENGING DOCUMENTATION	81
7.6	RELEASE.....	83
8	DIMENSIONS OF SOFTWARE TESTING	85
8.1	WHY “DIMENSIONS”?.....	85
8.2	BY OBJECT	86
8.3	BY SOURCE CODE VISIBILITY.....	86
8.4	BY LEVEL.....	88
8.5	BY PLACE.....	93
8.6	BY PURPOSE.....	95
8.7	BY ATTACK SURFACE	116
8.8	BY FAULT TYPE	123
9	QUALITY OF TESTING	131
9.1	HUH?	131
9.2	CODE COVERAGE.....	131
9.3	SOFTWARE METRICS	132
9.4	TEST PLAN REVIEW	133
9.5	SEEDING BUGS AND CODE MUTATION.....	134
9.6	TESTABILITY	134
9.7	BETA RELEASE/BETA TESTING	136
10	POST-RELEASE FUN.....	138
10.1	DEPLOYMENT TESTING	139
10.2	COMPATIBILITY TESTING	139
10.3	SYSTEM TESTING.....	139
10.4	STAGING AND PRODUCTION ENVIRONMENT	140
11	TESTING SOFTWARE IN A GLOBAL TEAM	141
11.1	TECHNICALLY SPEAKING	141
11.2	HUMAN-ORIENTED	142
12	SURVIVING AS A TESTER.....	145
12.1	FUZZY VALUE IN THE SOFTWARE DEVELOPMENT ..	145

12.2	COST/VALUE BALANCE	146
12.3	RELIGIONS AND CONFESSIONS	147
12.4	IEEE ROLE	151
12.5	OFFICE POLITICS.....	151
12.6	OUTSOURCING AND JOB SECURITY	154
12.7	SEGREGATION	157
12.8	FUTURE OF THE TRADE.....	158
13	APPENDIX A. TEST CASE.....	168
14	APPENDIX B. BIBLIOGRAPHY	170
14.1	STANDARDS AND CLASSICS.....	170
14.2	EXTREME TESTING & RELATED	171
14.3	OTHER	172
15	ENDNOTES	174

3 Table of Contents

1	PREFACE	3
2	AT A GLANCE	5
3	TABLE OF CONTENTS	9
4	INTRODUCTION	17
5	WHY TEST SOFTWARE?	19
5.1	WHY TEST SOFTWARE?	19
5.2	LIMITATIONS OF TESTING	20
5.2.1	<i>The First Law of Software Programming</i>	20
5.2.2	<i>Nothing is ever perfect</i>	21
5.2.2.1	Requirements	22
5.2.2.2	Design	23
5.2.2.2.1	Requirements context, missed or ignored requirements	24
5.2.2.2.2	Overusing formal verification tools without proper qualification	27
5.2.2.2.3	Late design changes and code fixes	29
5.2.2.3	Code	29
5.2.2.4	Platform	30
5.2.2.5	Tests	30
5.2.2.6	Users	31
5.2.2.6.1	Monkey	32

5.2.2.6.2	Farmer	33
5.2.2.6.3	Hacker	34
5.2.2.7	The Product Team	34
5.2.3	<i>Richness of possibilities</i>	34
5.2.3.1	Code paths, state transitions	35
5.2.3.2	Inputs (including input editing and timing)	36
5.2.3.3	Data in the environment	37
5.2.3.4	In essence – you cannot try everything	38
5.3	ALTERNATIVE TO TESTING – PROVING	38
5.4	THREE OBJECTIVES OF TESTING SOFTWARE	39
5.4.1	<i>So, why?</i>	39
5.4.2	<i>Objective #1: Quality Bar</i>	39
5.4.3	<i>Objective #2: Acceptance</i>	41
5.4.4	<i>Objective #3: Baseline</i>	41
6	WHAT'S SOFTWARE TESTING ABOUT?	43
6.1	THE DIFFERENCE BETWEEN A TESTER AND A DEVELOPER	43
6.1.1	<i>A Castle</i>	43
6.1.2	<i>Debugging is not testing</i>	44
6.1.2.1	Goals	44
6.1.2.2	Actor	45
6.1.2.3	Tools	45
6.1.2.4	Results	45
6.1.3	<i>Super-pessimism – Testing that the glass is half empty</i>	45
6.2	THE THREE-HEADED DRAGON OF SOFTWARE DEVELOPMENT	46
6.2.1	<i>Software developer</i>	46
6.2.2	<i>Software architect, analyst, project manager</i>	47
6.2.3	<i>Software tester</i>	48
6.3	THE LANGUAGE OF SOFTWARE TESTING	48
6.3.1	<i>The bug: what is it?</i>	48
6.3.2	<i>Test case</i>	49
6.3.3	<i>Test plan</i>	50
6.3.4	<i>Bug tracking</i>	50
6.3.5	<i>Version control</i>	51
6.3.6	<i>Test automation</i>	53
6.4	A BUG'S LIFE	54

6.4.1	<i>Bug opened</i>	54
6.4.2	<i>Bug approved</i>	55
6.4.3	<i>Bug fixed</i>	55
6.4.4	<i>Bug verified and closed</i>	56
6.5	SEVERITY OF BUGS	56
6.5.1	<i>Know Thine User!</i>	57
6.6	THE KINDS OF BUGS	59
6.6.1	<i>Code defect</i>	59
6.6.2	<i>Crash</i>	60
6.6.3	<i>A special kind: Regression</i>	61
6.6.4	<i>Security</i>	61
6.6.5	<i>Performance-related</i>	62
6.6.5.1	Throughput (performance)	62
6.6.5.2	Stress	63
6.6.5.3	Volume	64
6.6.5.4	Scalability	64
6.6.5.5	Resources consumption (leaks)	65
6.6.6	<i>Usability</i>	67
6.6.7	<i>Documentation</i>	69
6.6.8	<i>Test "issue"</i>	70
6.6.8.1	Bugs in tests and frameworks	70
6.6.8.2	Mixing up a bug and a feature	70

7 STAGES OF SOFTWARE DEVELOPMENT FOR A TESTER _____ 72

7.1	CHALLENGING REQUIREMENTS	72
7.2	CHALLENGING SPECIFICATIONS	73
7.2.1	<i>Challenging assumptions</i>	74
7.2.2	<i>Challenging completeness</i>	75
7.2.3	<i>Challenging behavior</i>	77
7.2.4	<i>Challenging usability</i>	78
7.2.5	<i>Challenging testability</i>	78
7.2.6	<i>That's not the end</i>	79
7.3	CHALLENGING ARCHITECTURE AND TECHNICAL DESIGN	79
7.4	TESTING CODE	81
7.5	CHALLENGING DOCUMENTATION	81
7.6	RELEASE	83

8 DIMENSIONS OF SOFTWARE TESTING _____ 85

8.1	WHY “DIMENSIONS”?	85
8.2	BY OBJECT	86
8.2.1	<i>Positive testing – capabilities</i>	86
8.2.2	<i>Negative testing – environment</i>	86
8.3	BY SOURCE CODE VISIBILITY	86
8.3.1	<i>Black (“opaque”) box testing</i>	86
8.3.2	<i>White (glass, clear) box testing</i>	87
8.3.3	<i>“Gray” (“Translucent”) box testing</i>	88
8.4	BY LEVEL	88
8.4.1	<i>Unit testing</i>	88
8.4.2	<i>Component testing</i>	89
8.4.3	<i>Integration testing</i>	90
8.4.4	<i>System testing</i>	91
8.4.4.1	Scenario-based	92
8.4.4.2	Free-form	92
8.5	BY PLACE	93
8.5.1	<i>Product Team Testing (vendor)</i>	93
8.5.2	<i>Customer (user) Side Testing</i>	93
8.5.3	<i>Third Party Testing (certification)</i>	94
8.6	BY PURPOSE	95
8.6.1	<i>Functional testing</i>	95
8.6.2	<i>Regression testing</i>	96
8.6.3	<i>Performance related testing</i>	97
8.6.3.1	Performance (throughput) testing	97
8.6.3.2	Volume (size) testing	97
8.6.3.3	Load (stress) testing	98
8.6.3.4	Scalability testing	98
8.6.3.5	Resource consumption (including leaks) testing	98
8.6.3.6	Sizing guidance	99
8.6.4	<i>Conformance testing</i>	99
8.6.4.1	Standards	100
8.6.4.2	Legislation	100
8.6.4.3	Coding policy testing	100
8.6.5	<i>Security testing</i>	100
8.6.5.1	Functional security testing	100
8.6.5.2	Security model and STRIDE	101
8.6.5.3	Security penetration testing	103
8.6.6	<i>Globalization testing</i>	103

8.6.6.1	Encoding	103
8.6.6.2	Watch your language	104
8.6.6.3	Formats	106
8.6.6.4	More calendar fun	106
8.6.6.5	Units	107
8.6.6.6	Strings	108
8.6.6.7	Length of text	108
8.6.6.8	Making of an error message	109
8.6.6.9	That's not all	110
8.6.7	<i>Localization testing</i>	110
8.6.7.1	Cultural differences	111
8.6.8	<i>Platform and dependencies testing</i>	112
8.6.9	<i>Pseudo-testing</i>	114
8.6.9.1	Usability	114
8.6.9.2	Reliability	115
8.6.9.3	Robustness	115
8.7	BY ATTACK SURFACE	116
8.7.1	<i>What is an attack surface</i>	116
8.7.2	<i>Input</i>	117
8.7.2.1	Input – GUI	117
8.7.2.2	Input – API	118
8.7.2.3	Input – Files and DB content	119
8.7.3	<i>Environment</i>	119
8.7.3.1	Environment – Resources (storage, memory, etc.)	119
8.7.3.2	Environment – Access rights	120
8.7.3.3	Environment – Network	120
8.7.3.4	Environment – Files and folders	120
8.7.3.5	Environment – Media	121
8.7.3.6	Environment – Output devices	121
8.7.3.7	Environment – Platform	122
8.7.4	<i>User</i>	122
8.7.4.1	User – monkey	122
8.7.4.2	User – farmer	123
8.7.4.3	User – hacker	123
8.8	BY FAULT TYPE	123
8.8.1	<i>A word on the fault model</i>	123
8.8.2	<i>Boundary conditions</i>	124
8.8.3	<i>Arithmetic overflows</i>	125

8.8.4	<i>Buffer overflows</i>	126
8.8.5	<i>Race conditions</i>	127
8.8.6	<i>Uninitialized variables</i>	129
8.8.7	<i>Uncaught exceptions</i>	130
8.8.8	<i>Resource allocation</i>	130
8.8.9	<i>That's not all again</i>	130
9	QUALITY OF TESTING	131
9.1	HUH?	131
9.2	CODE COVERAGE	131
9.3	SOFTWARE METRICS	132
9.4	TEST PLAN REVIEW	133
9.5	SEEDING BUGS AND CODE MUTATION	134
9.6	TESTABILITY	134
9.7	BETA RELEASE/BETA TESTING	136
10	POST-RELEASE FUN	138
10.1	DEPLOYMENT TESTING	139
10.2	COMPATIBILITY TESTING	139
10.3	SYSTEM TESTING	139
10.4	STAGING AND PRODUCTION ENVIRONMENT	140
11	TESTING SOFTWARE IN A GLOBAL TEAM	141
11.1	TECHNICALLY SPEAKING	141
11.2	HUMAN-ORIENTED	142
11.2.1	<i>Time zone differences.</i>	142
11.2.2	<i>Work relations and cultural differences</i>	142
11.2.3	<i>Life in a global org</i>	143
12	SURVIVING AS A TESTER	145
12.1	FUZZY VALUE IN THE SOFTWARE DEVELOPMENT	145
12.2	COST/VALUE BALANCE	146
12.3	RELIGIONS AND CONFESSIONS	147
12.4	IEEE ROLE	151
12.5	OFFICE POLITICS	151
12.6	OUTSOURCING AND JOB SECURITY	154
12.7	SEGREGATION	157
12.8	FUTURE OF THE TRADE	158
12.8.1	<i>From nobility to peasants</i>	158

12.8.2	<i>Crushing craftsmen</i>	161
12.8.3	<i>Assault on software development</i>	162
12.8.4	<i>What's next?</i>	165
13	APPENDIX A. TEST CASE	168
14	APPENDIX B. BIBLIOGRAPHY	170
14.1	STANDARDS AND CLASSICS	170
14.2	EXTREME TESTING & RELATED	171
14.3	OTHER	172
15	ENDNOTES	174