A FRAMEWORK OF THE USE OF INFORMATION IN SOFTWARE TESTING

By Payman Kaveh

B.S., Electrical Engineering, May 1996, University of Maryland

A Dissertation submitted to

The Faculty of
The School of Engineering and Applied Science
of The George Washington University
in Partial Satisfaction of the Requirements
for the Degree of Doctor of Science

May 16, 2010

Dissertation directed by

Abiodun O. Bada
Assistant Professor of Engineering Management and Systems Engineering
The School of Engineering and Applied Science of The George Washington University
certifies that Payman Kaveh has passed the Final Examination for the degree of Doctor of
Science as of May 13, 2009. This is the final and approved form of the dissertation.

A FRAMEWORK OF THE USE OF INFORMATION IN SOFTWARE TESTING

Payman Kaveh

Dissertation Research Committee:
Abiodun O. Bada, Assistant Professor of Engineering Management and Systems
Engineering, Dissertation Director
Edward Lile Murphree, Professor of Engineering Management, Committee Chair
Michael A Stankosky, Professor of Engineering Management and Systems Engineering,
Committee Member
Shahryar Sarkani, Senior Project Manager, Fannie Mae, Committee Member
Dedications

To my parents.
Acknowledgements

I would like to thank my advisor, Dr. Abiodun Bada for his continuous guidance, assistance, and invaluable advice.

I would also like to thank my committee: Dr. Abiodun Bada, Dr. Shayryar Sarkani, Dr. Michael Stankosky, and Dr. Edward Lile Murphree.

In addition, I would like to thank Dr. Shahram Sarkani, Dr. Michael Donnell, Dr. Ahmed Abdalla Nasser, Dr. Farhad Karimi, and Dr. Thomas Mazzuchi for their support and guidance.
Abstract

A Framework of the Use of Information in Software Testing

With the increasing role that software systems play in our daily lives, software quality has become extremely important. Software quality is impacted by the efficiency of the software testing process. There are a growing number of software testing methodologies, models, and initiatives to satisfy the need to improve software quality. The main assumption in these methodologies has been that software testing is principally a technology phenomenon and inadequate attention has been given to understanding of information in relation to the people who are using them in this field.

The purpose of this research is to investigate use of information by people in software testing. This research was done using grounded theory as it provided the best tools for theory generation from the collected field data. Data was gathered by interviewing thirty-four software testers.

This study investigates how software testers conceptualize, seek, and use information. In addition, this study explores the effect of organizational, behavioral, and ethical factors influencing software testing.

The main finding of the research is a framework of the use of information in software testing. The framework of information use has eight main categories: information seeking elements, domain knowledge, individual role context, organizational environment, meta-information, testing strategy, information use behaviors, and decision making process information use behaviors being the core category.

The study also shows that project size and duration play a key role in information seeking, information use behaviors, and testing methodology used. Whereas small
projects tend to be more context driven and ad hoc, the more complex and lengthier the software project gets the more it tends to follow some sort of best practices and a more standard model driven approach.
Table of Contents

Dedications ........................................................................................................................................... iii
Acknowledgements........................................................................................................................ iv
Abstract............................................................................................................................................. v
Table of Contents............................................................................................................................ vii
List of Figures .................................................................................................................................... xiv
List of Tables ..................................................................................................................................... xv
Glossary of Terms ............................................................................................................................. xvi
Chapter I - Introduction .................................................................................................................... 1
  1.1 Background of the Problem ................................................................................................. 2
  1.2 Research Study.................................................................................................................... 3
  1.3 Research Questions ............................................................................................................ 4
  1.3 Objectives .......................................................................................................................... 4
  1.4 Contributions....................................................................................................................... 5
  1.5 Research Limitations .......................................................................................................... 5
  1.6 Organization of the Dissertation ....................................................................................... 6
Chapter II – Literature Review: Software Testing ................................................................. 7
  2.1 Defining Quality .................................................................................................................. 7
  2.2 Quality and Software Testing ............................................................................................ 8
  2.3 Software Quality Measures ............................................................................................... 10
    2.3.1 Functionality .............................................................................................................. 11
    2.3.1.1 Functional Test .................................................................................................... 11
    2.3.1.2 Security Test .................................................................................................... 11

2.3.1.3 Volume Test......................................................................................................... 12
2.3.2 Usability ............................................................................................................. 12
2.3.3 Reliability .......................................................................................................... 12
  2.3.3.1 Integration Test ............................................................................................. 12
  2.3.3.2 Structure Test .............................................................................................. 12
  2.3.3.3 Stress Test ................................................................................................. 13
2.3.4 Performance ...................................................................................................... 13
  2.3.4.1 Benchmark Test ......................................................................................... 13
  2.3.4.2 Contention Test ......................................................................................... 13
  2.3.4.3 Load Test ................................................................................................. 13
  2.3.4.4 Performance Profile .................................................................................. 14
2.3.5 Supportability ................................................................................................... 14
  2.3.5.1 Configuration Test .................................................................................... 14
  2.3.5.2 Installation Test ......................................................................................... 14
2.4 Overview Of Testing Techniques ........................................................................... 14
  2.4.1 Black-Box Testing (Functional)...................................................................... 14
  2.4.2 White-Box Testing (Structural) ...................................................................... 15
  2.4.3 Gray-Box Testing .......................................................................................... 15
  2.4.4 Exploratory Testing ....................................................................................... 16
  2.4.5 Simulation ...................................................................................................... 16
  2.4.6 Prototyping ..................................................................................................... 16
  2.4.7 PIÑATA Approach ......................................................................................... 17
  2.4.8 Capability Maturity Model Integration ........................................................... 17
Chapter III - Literature Review: Information Use ............................................................ 21
3.1 Defining Information .............................................................................................. 22
3.2 Conceptualizing Information .................................................................................. 24
3.3 Information Needs .................................................................................................. 25
3.4 Information Seeking................................................................................................ 26
3.5 Information Use ...................................................................................................... 28
    3.5.1 Cognitive Dimensions ...................................................................................... 28
    3.5.2 Affective Dimensions ...................................................................................... 30
    3.5.3 Situational Dimensions .................................................................................... 32
3.6 Information Use in Software Testing...................................................................... 34
Chapter IV - Methodology................................................................................................ 36
4.1 Qualitative Vs. Quantitative Approach................................................................... 36
4.2 Qualitative Research ............................................................................................... 37
    4.2.1 Sampling .......................................................................................................... 38
    4.2.2 Institutional Review Board .............................................................................. 39
    4.2.3 Selection of Grounded Theory ......................................................................... 39
4.3 Grounded Theory .................................................................................................... 40
4.4 Data Analysis .......................................................................................................... 42
4.5 Data Collection ....................................................................................................... 45
4.6 Coding Methods ...................................................................................................... 46
    4.6.1 Open Coding .................................................................................................... 46
    4.6.2 Axial Coding .................................................................................................... 47
4.6.3 Selective Coding .............................................................................................. 48
4.7 Verification and Validation..................................................................................... 49
Chapter V - Findings ......................................................................................................... 50
5.1 Information Stages .................................................................................................. 51
5.2 Information Conceptualization .............................................................................. 54
  5.2.1 Source Conception ........................................................................................... 55
  5.2.2 Process Conception .......................................................................................... 55
  5.2.3 Knowledge Conception .................................................................................... 56
  5.2.4 Organizational Conception .............................................................................. 56
5.3 Information Seeking Behaviors .............................................................................. 57
5.4 Information Seeking Elements .............................................................................. 59
  5.4.1 Motivations ...................................................................................................... 60
  5.4.2 Negative Elements ........................................................................................... 61
    5.4.2.1 Fear of Rejection and Not Fitting In ......................................................... 61
    5.4.2.2 Fear of Retaliation ..................................................................................... 62
    5.4.2.3 Lack of Motivation ................................................................................... 63
  5.4.3 Sources of Information .................................................................................... 64
  5.4.4 Effort Needed to Find Information Source ...................................................... 64
5.5 Requirements and Associated Risks ....................................................................... 66
  5.5.1 User Requirements ........................................................................................... 66
  5.5.2 System Requirements ....................................................................................... 67
  5.5.3 Project Requirements ....................................................................................... 67
  5.5.4 Requirements Management Risks .................................................................... 68
5.6 Need to Use Information........................................................................................................... 69
  5.6.1 Discussion Among Peers .................................................................................................... 70
    5.6.1.1 Group Meetings ........................................................................................................ 70
    5.6.1.2 Checklists ................................................................................................................ 71
    5.6.1.3 Use of Portals .......................................................................................................... 72
  5.6.2 Cognitive Complexity ...................................................................................................... 73
    5.6.2.1 Testing Financial and Accounting Software .......................................................... 74
    5.6.2.2 Testing Scientific Software .................................................................................... 74
  5.6.3 Focusing on Organization / Project Context ................................................................. 76
  5.6.4 Determine Budgetary Needs ........................................................................................... 77
5.7 Information Use Elements ..................................................................................................... 78
  5.7.1 Domain Knowledge ........................................................................................................ 79
    5.7.1.1 Experience .............................................................................................................. 80
    5.7.1.2 Training / Education .............................................................................................. 81
    5.7.1.3 Recognition of Experts .......................................................................................... 82
  5.7.2 Individual Role Context ................................................................................................ 83
    5.7.2.1 Tester ...................................................................................................................... 83
    5.7.2.2 Developer ............................................................................................................... 85
    5.7.2.3 Technical Writers ................................................................................................... 86
    5.7.2.4 Stakeholders .......................................................................................................... 87
    5.7.2.5 Project Managers ................................................................................................. 87
  5.7.3 Organizational Environment ............................................................................................ 88
    5.7.3.1 Negative Organizational Perception Towards Software Testers ....................... 89
6.2 Context Driven vs. Standard Model Driven .......................................................... 112
6.3 Other Conclusions............................................................................................... 113
6.4 Future Work......................................................................................................... 114
References...................................................................................................................... 116
Appendix A: Interview Consent Form........................................................................ 126
Appendix B: Interview Questions................................................................................ 127
Appendix C: IRB Review Request Form .................................................................... 130
Appendix D: I-35W Mississippi River Bridge Collapse ............................................. 131
Appendix E: IEEE 829 Standard Document Types....................................................... 133
Appendix F: Principles of Context-Driven Software Testing ........................................ 135
List of Figures

Figure 2.1: Relative Cost to Fix Defects ................................................................. 10

Figure 2.2: Software Quality Measures ................................................................. 11

Figure 2.3: Model Driven vs. Context Driven Software testing ............................... 19

Figure 3.1: Organizational Information Processes .................................................. 24

Figure 3.2: Decision Making Cube “Knowing Cube” ............................................... 34

Figure 4.1: Research Methodology Flow Diagram ................................................... 44

Figure 5.1: Analytical Path for the Study ............................................................... 50

Figure 5.2: Software Testing Information Cycle ..................................................... 53

Figure 5.3: Information Use Factors ...................................................................... 78

Figure 5.4: Domain Knowledge Elements ............................................................. 79

Figure 5.5: Individual Role Context Elements ....................................................... 83

Figure 5.6: Organizational Environment Information Use Factors .......................... 88

Figure 5.7: Effects of Team Size and the Method of Communications ...................... 93

Figure 5.8: Meta-Information Elements .................................................................. 94

Figure 5.9: Information Use Cycle .......................................................................... 97

Figure 5.10: Information Use Processes ................................................................. 101

Figure 6.1: Information Use by Software Testers .................................................... 109

Figure E.1: Relationship of Test Documents to Testing Process ............................... 134
List of Tables

Table 2.1: Relative Cost to Fix Defects ................................................................. 9
Table 3.1: Data, Information, and Knowledge ....................................................... 23
Table 4.1: Interviewee Characterization ................................................................. 45
Table 5.1: Information Seeking Factors ............................................................... 65
Table 5.2 Information Need Categories ............................................................... 70
Table 6.1: Effects of Project Size in Methodology Used .................................... 112
Glossary of Terms

Defect: “generally refers to some problem with the software, either with its external behavior or with its internal characteristics” (Tian, 2005, p.20).

Effectiveness: “The degree to which a system’s features and capabilities meet the user’s needs” (DOJ, 2003).

Efficiency: The degree to which a system or component performs its designated functions with minimum consumption of resources (CPU, memory, networks) (IEEE, 1990).

Error: “A human action that produces an incorrect result”. (IEEE, 1990)


Information engineering: “An approach to planning, analyzing, designing, and developing an information system with an enterprise wide perspective and an emphasis on data and architectures” (GAO, 2004).

Information management: “The planning, budgeting, manipulating, and controlling of information throughout its life cycle” (GAO, 2004).

Information resources management (IRM): IRM includes “related resources such as personnel, equipment, funds, and information technology” (GAO, 2004).

Information system: “An organized collection, processing, transmission, and dissemination of information in accordance with defined procedures, whether automated
or manual. Information systems include non-financial, financial, and mixed systems” (GAO, 2004).

Maintainability: “The ease with which a software system or component can be modified to correct faults, improve performance, or adapt to a changed environment” (O’Regan, 2002).

Modifiability: “The degree to which a software system or component facilitates the incorporation of changes once the nature of the desired change has been determined” (Boehm et al, 2000).

Reliability: “The ability of a system or component to perform its required functions under stated conditions” (IEEE, 1990).

Understandability: This characteristic “defines simplicity, readability, and structuredness of the software” (Boehm et al, 2000).

Usability: “The capability of the software product to be understood, learned, used and be of value to the user, when used under specified conditions” (DOJ, 2003).
Chapter I - Introduction

Software is widely used in our lives and almost all facilities and devices include software components (Spillner et al, 2006). “While defects in financial or word processing programs are annoying and possibly costly, nobody is killed or injured. When software-intensive systems fly airplanes, drive automobiles, control air traffic, run factories, or operate power plants, defects can be dangerous” (Humphrey, 2000). Therefore, software quality has become increasingly important. In order to improve software quality an effective software testing process is required.

Over the years many software testing methodologies and techniques have been developed to improve software quality. However, for the most part these methodologies have paid little or no attention to the human factor (Kaner, 2007). Individuals interpret information differently and react differently to the available information (Choo, 2006). The definition of a reliable source of information varies between individuals; and decision making based on available information is another factor that is different among individuals.

This research investigates how software testers conceptualize, seek, and use information while performing their tasks.

This research takes a qualitative approach to study the use of information in the field of software testing. By using grounded theory methodology this research develops a framework of the use of information in software testing.
1.1 Background of the Problem

The quality of software is impacted by the efficiency of the software testing process. The increase in software complexity, importance, and mission criticality; has necessitated the need for a thorough software testing process.

There exists an abundance of testing methodologies (El-Far & Whittaker, 2001), models, and initiatives all claiming to increase the likelihood of success (Conradi et al, 2006). Examples include IEEE Standards (IEEE, 1998), with emphasis on disciplines, tasks, and processes, and suggests there exists a set of key factors and best practices that if followed will lead to project success and Context Driven Software Testing which emphasizes on contextual factors such as software context, testers knowledge, experience, and skill. In Context Driven Software Testing methodology “there are good practices in context, but there are no best practices” (CDT, 2007).

While improving the quality of software is highly desirable, software testing faces a collection of challenges; which is still “largely ad hoc, expensive, and unpredictably effective” (Bertolino, 2007). There is a lack of an integrated theory in the literature as to how software companies are conducting software testing (Bertolino, 2007) and a gap exists between the state of the art and the state of the practice in software testing (Kit, 1995). In addition software defects can be costly. The latest figures published by National Institute of Standards and Technology (NIST) estimates the cost of software defects to U.S. economy at $59.5 billion a year (NIST, 2002).

In addition, software testing has been mainly looked at as a technology phenomenon and hence inadequate attention has been given to human element involved.
1.2 Research Study

This research investigates how software testers conceptualize, seek, and use information. Since individuals interpret and use information differently this research uses a qualitative approach. As the main goal of this research is to develop a framework of the use of information in software testing, grounded theory methodology was selected as it offered the best approach for achieving the research goals.

Conducting semi-structured interviews using the questions in Appendix B as a guideline collected the data for this research. Interviewees’ comments were recorded and transcribed. Open, axial, and selective coding techniques were applied to the interviewees’ comments (Strauss & Corbin, 1998). The data collection included 34 participants that spanned over 15 projects. The selection criteria included individuals that were working, had a role in software testing and quality assurance, or had worked in the field of software testing for the duration of at least one project.

The interviews were stopped when theoretical saturation was reached where additional interviews were not providing any new data (Strauss & Corbin, 1998).

As the research involved human subjects the guidelines provided by The George Washington University institutional review board were followed. The interviewees were informed of the background and goals of the research and were informed that the interviews would remain anonymous.
1.3 Research Questions

This research investigates how software testers use information in software testing; the research develops a grounded theory of information use in software testing. The main research question addressed is:

- How do software testers use information in software testing projects?

In order to answer the main research question the answers to the following sub-questions are investigated:

1. How do software testers conceptualize information used in software testing?
2. How do software testers seek information?
3. How do software testers use information?
4. What factors influence the way software testers use information in practice?

Answering these questions will help develop a theory of information use by software testers.

1.3 Objectives

This research studies information use, associated factors and their influence in software testing. Furthermore, this research develops a framework of the use of information in software testing. In doing so the objectives of the research are:

a. Explore how software testers conceptualize information.
b. Explore how software testers seek information.
c. Explore how software testers use information.
d. Identify critical factors / elements that drive software testers to use information and explore the critical factors effects on how software testers perform their tasks.
e. Provide an understanding of the relationship between how software testers seek and use information with the critical factors/elements identified. (Develop a theory of how information use is related to factors that determine its use).

1.4 Contributions

By developing a framework of the use information in software testing, this research explores how software testers conceptualize, seek, and use information. The framework of information use lays the foundation for future research in this arena.

This research looks at a number of factors including the human elements influencing software testing and brings about a foundation for future research on this topic.

This research identifies a number of ethical and contractual factors influencing software testing projects. Studying these factors would assist contract officers to better structure their contracts to fit the needs of their organizations.

This research proposes that the size and duration of a software testing project is directly correlated with the methodology used. A future study of this relationship would benefit the software testing body of knowledge.

1.5 Research Limitations

This research was conducted using a qualitative approach identifying several factors that influence software testing. As it is the limitation of any qualitative study, this research cannot quantify the degree of importance or influence of these factors.

This research draws conclusions that there exists a relationship between project size and duration and information used in software testing. Although we asked the respondents what type of project they were currently working on and what was the
project size and duration, the interview questions did not limit their responses to the to their current project. Hence, in drawing conclusions about this correlation, we were limited to responses that either mentioned phrases such as “in our current project” or the project size and duration were specifically mentioned.

1.6 Organization of the Dissertation

This dissertation consists of the following six chapters: (I) Introduction; (II) Literature Review – Software Testing; (III) Literature Review – Information Use; (IV) Methodology; (V) Findings; (VI) Conclusions.
Chapter II – Literature Review: Software Testing

The literature review’s focus for this research is twofold: first in Chapter II, we explore literature on quality, how it relates to software testing, and we review standard model based and context driver software testing schools of thoughts. Second, in Chapter III we review the literature on information use and cognitive, affective, and situational factors associated with it.

2.1 Defining Quality

“The definition of quality, taken from the *American Heritage Dictionary of the English Language*, 3rd Edition (1992, 1996), is:

Quality  -1.a. An inherent or distinguishing characteristic; a property. 1.b. A personal trait, especially a character trait. 2. Essential character; nature. 3.a. Superiority of kind. 3.b. Degree or grade of excellence” (as cited in Sarkani, 2004, pp. 86-87).

Merriam-Webster dictionary defines quality as:

"1 a: peculiar and essential character ... b: an inherent feature...2a: degree of excellence ... b: superiority in kind ...4a: a distinguishing attribute ... 5: the character in a logical proposition of being affirmative or negative" (m-w.com, 2009).

Crosby defines quality as “conformance to the requirements” (Crosby, 1979).

Achieving quality “includes identifying the measures and criteria to demonstrate the achievement of quality, and the implementation of a process to ensure that the product created by the process has achieved the desired degree of quality and can be repeated and managed” (RUP, 2009; Kroll & Kruchten, 2003).
2.2 Quality and Software Testing

A measuring tool for achieving project success is the quality and efficiency of the programming task (Mahoney, 1990). Graham et al (2007) defines software quality as “the degree to which a component, system, or process meets specified requirements and/or user/customer needs and expectations” (p.7). “[Software] testing helps us measure the quality of software in terms of the number of defects found, the tests run, and the system covered by the tests” (Graham et al, 2007, p.7). Since people write software and people make mistakes (Kit, 1995); “Software testing process is … used to help identify the correctness, completeness, security, and quality of the developed computer software” (Black, 2004). Software testing also ensures that the developed software meets its requirements (Sinha et al, 2006). Hence, software testing contributes to the “delivery of higher quality software products”, “more satisfied users”, and “lower maintenance costs” (Parrington & Roper, 1989).

“The most effective testing programs start at the beginning of a project, long before any program code has been written” (Dustin, 2003, p.1). Keys “to the correctness aspect of software quality are the concepts of defect, failure, fault, and error” (Tian, 2005, p.20).

“Testing must be cost effective and tied to the costs of production and failure. The budget for testing can be viewed as a form of insurance whose premium reflects its necessity and the cost of failure” (Ould, & Unwin, 1988, p.23).

The software systems people build are increasingly complex and critical, and between 30 to 70 percent of development effort is spent on testing (Lewis, 2005). As Figure 2.1 illustrates, software problems are 100 times more costly to find and repair after deployment.
“Defect prevention is the use of techniques and processes that can help detect and avoid errors before they propagate to later development phases” (Dustin, 2003, p.3).

“Error removal cost multiplies over the system’s development life cycle” (Dustin, 2003, p.4). “Quality management decreases production costs because the sooner the defect is located and corrected, the less costly it will be in the long run” (Lewis, 2005, p.4).

Table 2.1 shows the relative cost to fix a defect in different phases of the software development life cycle.

<table>
<thead>
<tr>
<th>Phase</th>
<th>Relative Cost to Correct</th>
</tr>
</thead>
<tbody>
<tr>
<td>Definition</td>
<td>$1</td>
</tr>
<tr>
<td>High-Level Design</td>
<td>$2</td>
</tr>
<tr>
<td>Low-Level Design</td>
<td>$5</td>
</tr>
<tr>
<td>Code</td>
<td>$10</td>
</tr>
<tr>
<td>Unit Test</td>
<td>$15</td>
</tr>
<tr>
<td>Integration Test</td>
<td>$22</td>
</tr>
<tr>
<td>System Test</td>
<td>$50</td>
</tr>
<tr>
<td>Post-Delivery</td>
<td>$100+</td>
</tr>
</tbody>
</table>

Quality management costs is calculated as the sum of four components costs: prevention, inspection, internal failure, and external failure (Lewis, 2005). “Prevention costs consist of actions taken to prevent defects from occurring in the first place. Inspection costs consist of measuring, evaluating, and auditing products or services for conformance to standards and specifications. Internal failure costs are those incurred by fixing a defective product before it is delivered. External failure costs consist of the costs of defects discovered after the product has been released.” (Lewis, 2005, p.4)
2.3 Software Quality Measures

In order to achieve software quality, “many types of tests are implemented and executed, [with] each test type having a specific objective and support technique” (Grady, 1992; RUP, 2007).

“Each technique focuses on testing one or more characteristics or attributes” (RUP, 2007).

Figure 2.2 shows the software quality measures, all of which are defined, and the associated test types based on the “most obvious quality dimension they address” (Kroll & Kruchten, 2003; RUP, 2007):
2.3.1 Functionality

2.3.1.1 Functional Test

This “test focus[es] on validating …the functions as intended, features, providing the required service or services or methods. [Functional tests are] implemented and executed at [various times] to test different units, integrated units, applications, and systems” (RUP, 2007).

2.3.1.2 Security Test

This test focuses on “ensuring that [the data,] or systems, are accessible to only those users intended” (RUP, 2007).
2.3.1.3 Volume Test

This test verifies the “ability to handle large amounts of data, either as input and output or resident within a database. Volume testing includes test strategies such as creating queries that return the entire content of the database, queries that have so many restrictions no data are returned, or data entry of the maximum amount of data in each field” (RUP, 2007).

2.3.2 Usability

Usability test focuses on human factors. “Usability tests are tests that exercise specific user interface features in order to determine if the software behaves as would be expected by trained/untrained users” (Rakitin, 2001).

2.3.3 Reliability

2.3.3.1 Integration Test

This test focuses on ensuring each module performs correctly within the control structure and that the module interfaces are correct. “During integration testing, the system is slowly built up by adding modules to the core of already integrated modules” (Lewis, 2005, p.93).

2.3.3.2 Structure Test

“This test focuses on … assessing the adherence [of software] to its design and formation. Typically, this test is done for Web-enabled applications, ensuring that all links are valid, appropriate content is displayed, and there is no orphaned content” (RUP, 2007).
2.3.3.3 Stress Test

This is a type of reliability test that “focuses on evaluating how the system responds under abnormal conditions. Stresses on the system might include extreme workloads, insufficient memory, unavailable services and hardware, or limited shared resources. A stress test is often performed to gain a better understanding of how and in what areas the system will break, so that contingency plans and upgrade maintenance can be planned and budgeted for in advance” (RUP, 2007).

2.3.4 Performance

2.3.4.1 Benchmark Test

This is a type of performance test that “compares the performance of a new or unknown target of test to a known referenced workload and system” (RUP, 2007).

2.3.4.2 Contention Test

This test focuses on validating the target of tests that can “acceptably handle multiple [user/operator] demands on the same resource (data records, memory, [etc.])” (RUP, 2007).

2.3.4.3 Load Test

A load test “validate[s] and assess[es] acceptability of the operational limits of a system under varying workloads while the system being tested remains constant. In some variants, the workload remains constant and the configuration of the [tested system] is varied. Measurements are usually taken based on the workload throughput and in-line transaction response times. The variation in workload will usually include emulation of
average and peak workloads that will occur within normal operational tolerances” (RUP, 2007).

2.3.4.4 Performance Profile

This is a test that “monitors the timing profile, including execution flow, data access, function, and system calls” (Lewis, 2005).

2.3.5 Supportability

2.3.5.1 Configuration Test

This test focuses on ensuring the” functions as intended on different hardware or software configurations. This test might also be implemented as a system performance test” (RUP, 2007).

2.3.5.2 Installation Test

An installation test ensures “the target of [the] test installs as intended on different hardware and software configurations under different conditions” (RUP, 2007).

2.4 Overview Of Testing Techniques

In this section some of the well-known testing techniques that have been adopted by the software engineering community are introduced and reviewed.

2.4.1 Black-Box Testing (Functional)

“Black-box or functional testing is one in which test conditions are developed based on the program or system’s functionality [; that is], the tester requires information about the input data and observed output, but does not know how the program or system works.
Just as one does not have to know how a car works internally to drive it, it is not necessary to know the internal structure of a program to execute it. With black-box testing, the tester views the program as a black-box and is completely unconcerned with the internal structure of the program or system.

A major advantage of black-box testing is that it is designed toward what the program or system is supposed to do, and it is natural and understood by everyone” (Lewis, 2005, p.19).

2.4.2 White-Box Testing (Structural)

“In white-box or structural testing test conditions are designed by examining paths of logic. The tester examines the internal structure of the program or system. The data are driven by examining the logic of the program or system, without concern for the program or system requirements” (Lewis, 2005, p.20).

“An advantage of white-box testing is that it is thorough and focuses on the produced code. [Because] there is knowledge of the internal structure or logic, errors or deliberate mischief on the part of a programmer have a higher probability of being detected.

One disadvantage of white-box testing is that it does not verify that the specifications are correct [: that is], it focuses only on the internal logic and does not verify the logic to the specification. Another disadvantage is that there is no way to detect the missing paths and data-sensitive errors” (Lewis, 2005, p.20).

2.4.3 Gray-Box Testing

Gray-box testing is a software testing technique that uses “a combination of black-box testing and white-box testing” (Lewis, 2005, p.30). It is not black-box testing
because the tester does know some of the internal workings of the software being tested. In gray-box testing, the tester applies a limited number of test cases to the internal workings of the software being tested. In the remaining part of gray-box testing, one takes a black-box approach in applying inputs to the tested software and observes the outputs (Lewis, 2005).

2.4.4 Exploratory Testing

Exploratory testing is simultaneous learning, test design, and test execution (Bach, 2003). It is also known as ad hoc testing. “Exploratory testing seeks to find out how the software actually works and asks questions about how it will handle difficult and easy cases. The testing is dependent on the tester’s skill of inventing test cases and finding defects. The more the tester knows about the product and different test methods, the better the testing will be” (Lewis, 2009, p.72).

2.4.5 Simulation

Simulation is used as a technique in the early development process before a fully operational system is available. This can push “the verification of some high-level design ideas or system architecture features to much earlier stages before expensive implementation and rework are involved” (Tian, 2005).

2.4.6 Prototyping

Prototyping is similar to simulation in basic ideas, where it is used as a general approach to gather data from users by building and demonstrating to them some part of a potential application (Lewis, 2005).
2.4.7 PIÑATA Approach

In talking with James Bach at the second annual Conference of the Association for Software Testing (CAST), the “Piñata Approach” was explored under the topic of testing. The main question in this approach, pertaining to software testing, was “When do we know when we are done with testing?” There are several factors that can have an effect on the duration of the testing. Among these factors, release date, resources, and budget were discussed. However, at some point testing reaches a point where either it is not cost effective or the level of bugs found and corrected is acceptable, and a decision can be made to release the software; hence, there is no added value in further testing. At this point the designed software meets the requirements and finding new bugs may be outside its scope.

2.4.8 Capability Maturity Model Integration

The Software Engineering Institute (SEI) at Carnegie-Mellon University was initiated by the U.S. Defense Department to help improve software development processes (Carnegie-Mellon, 2009).

Capability Maturity Model Integration (CMMI) is a model consists of five levels of process 'maturity' that determine effectiveness in delivering quality software. It is geared to large organizations such as large U.S. Defense Department contractors. Additionally, many of the Quality Assurance processes involved, if reasonably applied may be helpful to any organization. The five levels of process maturity (CMM, 2009) are:

“Level 1 – [Which is] characterized by chaos, periodic panics, and heroic efforts required by individuals to successfully complete projects. Few if any processes are in place and successes may not be repeatable.
Level 2 – [At this level] software project tracking, requirements management, realistic planning, and configuration management processes are in place and successful practices may be repeated.

Level 3 – [At level 3,] standard software development and maintenance processes are integrated throughout the organization. Additionally, a software engineering process group is in place to oversee software processes and training programs are used to ensure compliance.

Level 4 – [At this level] metrics are used to track productivity, processes, and products. Project performance is predictable, and quality is consistently high.

Level 5 – [Level 5 is the highest level of CMM, which] focuses on continuous process improvement. The impact of new processes and technologies can be predicted and effectively implemented when required.” (CMM, 2009)

2.5 Testing Methodologies

In order to achieve high levels of software quality, a high-quality testing process needs to be applied. But how does one know if the applied testing strategy is good or not? Are the lack of bugs in a program proof of high-quality software or just low-quality testing? (CAST, 2007)

A variety of testing approaches and methodologies have been developed and presented over the years (Herman, 2006). To show the vast differences between these methodologies, this research focuses on two distinct methodologies / schools of testing: the context-driven school of testing and the traditional model-driven testing standards school. Members of the context-driven school of testing believe that there are no set “best practices” or “critical success factors” of testing, but each testing situation is unique and a
A tester would need to apply his or her experience to suit each unique situation (Kaner et al., 1999; Bach et al., 2002). Traditional model-driven testing standards developed by such organizations as IEEE, that believe there are sets of guidelines and standards that need to be followed to ensure ultimate project success, are reviewed. For example, standard IEEE829 suggests there are certain critical factors and common areas between all software testing areas (IEEE, 1998). Figure 2.3 summarizes the fundamentals of these two schools of thoughts.

<table>
<thead>
<tr>
<th>Model Driven Testing</th>
<th>Context Driven Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Suggests there are certain <strong>critical factors</strong> and <strong>common areas</strong> between all software testing areas (Copland 2004).</td>
<td>There are good practices in context, but there are <strong>no best practices</strong>.</td>
</tr>
<tr>
<td>The generation of test cases from models according to a given coverage criterion (Bertolino 2004).</td>
<td>Projects unfold over time in ways that are often not predictable.</td>
</tr>
<tr>
<td>IEEE Standard 1008</td>
<td>The product is a solution. If the problem isn’t solved, the product doesn’t work” (CDT 2007).</td>
</tr>
<tr>
<td>IEEE Standard 829</td>
<td></td>
</tr>
</tbody>
</table>

Figure 2.3: Model Driven vs. Context Driven Software testing
Appendix E details the standard model driven testing as stated by IEEE standard 829.

Appendix F details the context driven principals.
Chapter III - Literature Review: Information Use

On January 28, 1986, seven crewmembers of the space shuttle Challenger were killed shortly after Challenger was launched from Cape Canaveral, Florida (Choo, 2006).

“The presidential commission investigating the accident subsequently determined that Challenger was destroyed after hot propellant gases flew past the aft joint of the shuttle’s right solid rocket booster, burning through two synthetic rubber seal rings, called O-rings, and vaporizing the seal. The problems with the O-rings were not new. They had been seen as early as 1977 and had been formally classified and documented as a critical life-threatening problem for many years. NASA had in place a multilevel system of rules and routines, checks and balances to assess risk and control launch decisions. On the eve of the last flight of Challenger, concerns were raised repeatedly and vocally about the safety of the mission” (Choo, 2006, p. 251). Why was the Challenger cleared for launch despite availability of information about the technical problems that caused the explosion?

The Presidential Commission (1986) “concluded, ‘The decision to launch the Challenger was flawed.’ Specifically, it identified ‘failures in communication that resulted in a decision to launch Flight STS 51-L were based on incomplete and sometimes misleading information, a conflict between engineering data and management judgment, and a NASA management structure that permitted internal flight safety problems to bypass key shuttle managers’” (p.82) (as cited in Choo, 2006, p.251). The U.S. House of Representatives conducted its own hearing and also concluded, “The fundamental problem was poor technical decision making over a period of several years by top NASA and contractor personnel” (U.S. Congress, 1986; Choo, 2006).
In order to understand this accident and in general how failures occur, we need to analyze how participants make sense of a stream of events (information). Hence, this chapter will define information and explore how people use the available information to make decisions.

3.1 Defining Information

In the literature information has been defined as:

1. “That which reduces uncertainty” (Shannon & Weaver, 1949).
2. “Data endowed with relevance and purpose” (Drucker, 1988)
3. “Observation of states of the world” (Davenport, 1997)
4. “Patterns in data” (O'Dell & Grayson, 1998).
5. “Message, usually in the form of a document or an audible or visible communication, meant to change the way a receiver perceives something and to influence judgment or behavior; data that makes a difference” (Davenport & Prusak, 2000).
6. “The attribute inherent in and communicated by one of two or more alternative sequences or arrangements of something” (m-w.com, 2007).
7. “Communication or reception of knowledge or intelligence” (m-w.com, 2007).

In order to further define information we will first try to make a distinction between data, information, and knowledge.

“Data, information, and knowledge are not easy to separate…; at best [one] can construct a continuum of the three. Still, coming up with working definitions of these terms is a useful starting place. Defining them can show where a company has focused its information technology (IT) energy, whether data it generated have real use, whether the
assumptions for structuring information make sense, and whether any of it has paid off.”
(Davenport, 1997, p9). Davenport (1997) defined and separated data, information, and
knowledge as presented in Table 3.1.

Table 3.1: Data, Information, and Knowledge (Source: Davenport, 1997)

<table>
<thead>
<tr>
<th>Data</th>
<th>Information</th>
<th>Knowledge</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple observations of states of the world</td>
<td>Data endowed with relevance and purpose</td>
<td>Valuable information from the human mind</td>
</tr>
<tr>
<td>• Easily structured</td>
<td>• Requires unit of analysis</td>
<td>Includes reflection, symbols, context</td>
</tr>
<tr>
<td>• Easily captured on machines</td>
<td>• Needs consensus on meaning</td>
<td>• Hard to structure</td>
</tr>
<tr>
<td>• Often quantified</td>
<td>• Human mediation necessary</td>
<td>• Difficult to capture on machines</td>
</tr>
<tr>
<td>• Easily transferred</td>
<td></td>
<td>• Often tacit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Hard to transfer</td>
</tr>
</tbody>
</table>

Knowledge is defined and broken down into three categories (Choo et al, 2000):

*Tacit* – Individual knowledge used to by organization members to perform their tasks.

*Explicit* – “Knowledge that can be expressed formally” (p.34).

*Cultural* – “The cognitive and affective structure that are habitually used to perceive,
explain, evaluate and contract reality” (p.34).

Tacit knowledge is hardest to capture “because it is expressed through action-based
skills and cannot be reduced to rules” (Choo, et al, p.34)

We can visualize sense making, knowledge building, and decision making as
representing three layers of organizational information practices, with each inner layer
building upon the information created in the next outer later (Figure 3.1). According to
Choo (2006) the best way to analyze organization behavior is to analyze the structure and
processes of decision making. As the central theme of this research is that software
testing is an information gathering technique used to make informed decisions, the
research will focus on the inner circle of information processing and decision making, realizing that organizational processes (circles) are highly interconnected.

Figure 3.1: Organizational Information Processes (Source: Choo, 2006)

3.2 Conceptualizing Information

In chapter II we provided an overview of testing practices in general, highlighting both the traditional and more recent schools of thoughts in software testing practices. In this chapter we will review information use by organizations. We use the model of Choo (2006) which classifies information use in three dimensions: the cognitive dimension, the
affective dimension and the situational dimension. The model provides a conceptual model of information seeking and use behaviors.

Conceptually, information seeking behavior which refers to how individuals make choices about where and how to look for information and how they act based on the available information to make decisions consists of (Wilson, 1999; Choo, 2006):

1. Information needs
2. Information seeking
3. Information use

3.3 Information Needs

“Information needs arise when the individual recognizes gaps in his or her state of knowledge and ability to make sense of an experience” (Choo, 2006, p.176). “The lack of clarity and the ability to make sense of an event revolves around two basic questions: In the flood of signals indicating change in the environment, which messages and cues are important and need to be focused on? Given that the information is ambiguous, which interpretation is the most plausible and should be used to understand what the cues mean? The central issue is therefore the management of ambiguity … The lack of information may be addressed by gathering more data that are relevant to an issue, but the lack of clarity has to be met by constructing a plausible interpretation that makes sense of the noticed information. The initial attempt to reduce ambiguity is to try and fit the information with existing assumptions, beliefs, and expectations” (Choo, 2006, pp.105-106).

Information is required during the decision making process to reduce uncertainty in at least three ways (Choo, 2006). “First, information is needed to frame a choice situation.
Boundaries are drawn to delimit a problem space in which solutions are to be searched, stakeholders are to be identified, and influence is to be managed. The framing of a problem to a large extent determines the types and content of information that would be needed in order to be able to make a decision. Second, information is needed to define preferences and to select rules. Multiple goals and interests are clarified, prioritized, and expressed as choice criteria. Third, the information is needed about available courses of action and their projected outcomes” (Choo, 2006. p.230).

3.4 Information Seeking

Information seeking is the process in which the individual searches for information purposefully. “Individual information seeking is shaped by the habits and heuristics that the decision maker has acquired as a result of training, education, or experience” (Choo, 2006, p.230). Allen (1996) postulates that individuals “may suppress their need for information or avoid a problem situation so that no information seeking is necessary” (as cited in choo et al, 2000, p.8). “When information seeking does occur, it is purposive and goal-directed, and resembles a problem-solving or decision-making process” (Choo et al, 2000, p.8).

“The Individual selects [a] source… that is perceived to have the greater probability of providing information that will be relevant [and] useful”. In addition, the individual would be concerned with the accuracy and reliability of the source (Choo et al, 2000, p.23).

Choo et al (2000) suggests that “the selection and use of sources in information seeking is influenced by the amount of time and effort that is required to locate or contact the source and interact with the source to extract information” (p.12). Organizational
rules and routines also influence information seeking base on organizational experiences and goals.

The degree in which an individual is motivated and has an area of interest in a topic influences the level effort individual makes in information seeking (Choo et al, 2000). According to Kuhlthau (1993) “as information search progresses, initial feelings of uncertainty and anxiety fall and confidence rises…. Kuhlthau postulated that information search is composed of six stages: initiation, selection, exploration, formulation, collection, and presentation, each of which is characterized by emotional responses” (as cited in Choo et al, 2000, p.10), which are noted in Table 3.2.

Table 3.2: Six Stages of Information Search (Source: Kuhlthau 1993; Choo, 2000)

<table>
<thead>
<tr>
<th>Stages of Information Seeking</th>
<th>Emotional Responses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initiation</td>
<td>Individual first recognizes a need for more information, and feelings of uncertainty and apprehension are common.</td>
</tr>
<tr>
<td>Selection</td>
<td>Individual identifies the general area or topic to be investigated, and feelings of uncertainty are replaced by optimism and readiness to search.</td>
</tr>
<tr>
<td>Exploration</td>
<td>Individual expands personal understanding of the general area. Feelings of confusion and doubt may increase.</td>
</tr>
<tr>
<td>Formulation</td>
<td>Individual establishes a focus or theme on the problem that can guide searching. Feelings of uncertainty diminish as confidence increases.</td>
</tr>
<tr>
<td>Collection</td>
<td>Individual interacts with the information systems and services to gather information. Confidence increases and interest in the project deepens.</td>
</tr>
<tr>
<td>Presentation</td>
<td>Individual completes the search and resolves the problem. Sense of relief, accompanied by satisfaction if the search is thought to have gone well, or disappointment otherwise.</td>
</tr>
</tbody>
</table>

Information seeking is a function of individual preferences, institutional values, and the decision situation’s attributes (Choo, 2006).
3.5 Information Use

Information use leads to a change in the individual’s ability and capacity to make sense of an event or react based on the new understanding of available information. Choo (2006) suggests that information use is a situated action that is contextual and there is an underlying structure in the ways individuals use information. This underlying structure may be looked at cognitive, effective, and situational dimensions.

3.5.1 Cognitive Dimensions

“At the cognitive level, an individual’s cognitive style and preferences would influence the manner [in which] information is processed and utilized” (Choo et al, 2000, p.24).

“The heuristics and biases school found that people use mental shortcuts to reduce cognitive effort, but these heuristics can lead to errors and biases. Thus, decision makers may over-rely on the use of stereotypes, information that is easy to recall, and initial impressions or estimates (Gilovich et al, 2002). Many of these biases may be amplified in organizational decision situations, as when the subject matter is complex, the information is ambiguous, or time pressure or information overload forces the decision maker to curtail his or her analysis. Another view believes that simple heuristics do in fact work well in enabling individuals to arrive at reasonable decisions (Gigerenzer et al, 1999). These heuristics exploit the ways that information is structured in different environments in order to find and use information cues that seem to work well for the problem at hand” (Choo, 2006, p.241).

A study done by Eisenhardt and Sull (2001) found many companies that made use of a few simple rules that were based on experiences to decide what action to take. For
example, when Cisco first moved to an acquisition strategy to grow, Cisco had a simple rule of acquiring companies with at most 75 employees that three-fourth of their workforce are engineers. In the mid 1980s, Intel followed a simple rule of increasing manufacturing capacity for precuts that had a better gross margin. At Oticon, if a key member left a project, the company would terminate the project.

At the cognitive level, individuals may selectively use information to support a desired outcome (Choo, 2006). O’Reilley (1983) postulates that the “selective processing does not imply that decision makers abbreviate their information search, but rather that they seek more information than is required and use this information to increase their confidence in their choices” (as cited in Choo, 2006, p.242). “Preferences for outcomes may be the least ambiguous component of the decision process, more certain than the definition of the problem, the range of feasible alternatives, or the probabilities associated with various alternatives” (O’Reilly, 1983, p.109).

Feldman (1989) observed “bureaucratic analysts work in a situation characterized most of the time by a lack of attention by decision makers or policy makers. Many reports they write are not read; many contracts they set up are not used; much expertise they acquire is not called upon. Decisions about policies seem to be made on the basis of politics and personal loyalties rather than the information and expertise the analysts have to offer” (p.89). Hence, “decision makers would hire external consulting groups to do evaluation studies not for the purpose of discovering better alternatives but to garner expert support for options already chosen. A difficult situation is when information is not used in policy decision making” (Choo, 2006, p. 242).
3.5.2 Affective Dimensions

When people process information, they assume a certain status within their role in the organization and try to build a good reputation for themselves. “People use information selectively to avoid [negative emotions], embarrassment, conflict, or regret; to maintain self-image; and to reduce personal status or reputation” (Choo et al, 2000, p.18).

Choo (2006) postulates “in some situations, a decision maker becomes increasingly locked into a failing course of action. Decision makers continue to positively evaluate and pursue a course of action even when objective facts indicate withdrawal is necessary to reduce further losses.” (p.244).

Janice (1982) argues that individuals suppress information use that may result in other possible course of action “when they are deeply involved in a cohesive in-group, when the members’ strivings for unanimity override their motivation to realistically appraise alternative courses of action” (p.9). Choo (2006) categorizes this information use behavior into three types: “First, group members share a feeling of invulnerability, which leads to optimism and willingness to take risks. Second, group members are closed minded, collectively rationalizing or discounting aberrant information and maintaining stereotype views of opposing parties as weak and ineffectual. Third, group members press toward uniformity, sustaining a shared impression of unanimity through self-censorship as well as direct pressure against dissenting views. As a result of these affective illusions of invulnerability and solidarity, the group’s seeking and use of information is compromised, and decision making becomes defective” (p.243).

Staw and Ross (1987) identified a number of psychological, social, and organizational factors that can induce escalation and over commitment. Psychological
factors are divided into the need for self-justification and biased information use. For example “in order to protect our own self-esteem, we may hang on or even allocate further resources to “prove” that the project is a success…. If facts challenge a project’s viability, managers try to find reasons to discredit the information sources or the quality of the information. If the information is ambiguous, managers may only make use of favorable facts that support the project” (Choo 2006, p. 244). Staw and Ross (1987) argue that managers respond to social pressures, and at times persist on a course of action even though if appears irrational, because they do not want to appear incompetent, damage their reputation, or lose power. Managers at times will justify the need to continue with a project that does not seem plausible in an attempt to maintain their reputation. In this scenario, the information that managers receive will be filtered so that only information that reinforces the positive effects or reduces the negative effects is processed and used. Organizational factors such as ‘administrative inertia’ and ‘institutional embeddedness’ influence information use. “A decision to cancel a major project can be disruptive and expensive; it may require changing rules and policies, moving or firing people, litigating, or compensating for cancelled contracts, so much so that ‘killing a project is as costly as saving it’…. institutional embeddedness… can happen if a long-standing project or line of business becomes closely identified with the organization. Stakeholder groups develop vested interests in the project. Reputations become tied to the project. In this case, decision makers may believe that ‘killing the project is like killing the very purpose of the organization’ ” (Choo, 2006, p.245).
3.5.3 Situational Dimensions

At the situational level, the degree to which a task has been structured by rules and routines will impact the use of information (Choo et al, 2000).

Choo (2006) characterizes four models of decision making at the situational level in information use: rational mode, process mode, political mode, and anarchic mode. “In the rational mode, information use is relatively controlled, being guided by the principle of selecting an alternative that is good enough to pass acceptable criteria. In the process mode, information use is focused as repeated cycles of information processing converge on a solution which is a specific answer to a specific problem, and which has to be presented to and authorized by upper management. In the political mode, information use is highly controlled and directed as a political tactic to justify preferred outcomes. Information is selectively passed, so that information that contradicts assumptions or expectations is ignored or reinterpreted. In the anarchic mode, information use is uncontrolled, as solutions are attached to problems through happenstance and individual interest, and decisions are made by flight oversight more often that by rational resolution” (Choo, 2006, p.246).

Organizations, using Standard Operating Procedures (SOP) guide the individuals on how to perform their tasks by setting routing and filtering rules (Cyert & March, 1992). According to Cyert & March (1992) there are four basic SOPs: task performance rules, continuing records and reports, information handling rules, and plans and planning rules.

Task performance rules are a set of procedures defined at the organizational level, which specify methods for accomplishing tasks and may apply from regulating the information behavior of managers to controlling the information use of staff.
performance rules use past organizational learning to ensure activities within each subunit of an organization are consistent and coordinated with other subunits (Choo et al, 2000).

Maintaining records by organizations have a control effect since members of an organization assume that the records and reports are maintained for the purpose that someone with check them. As a collection of events, they may help predict future outcomes (Choo et al, 2000).

“An important category of any organization’s standard operation procedures is its information handling rules. These rules direct and constrain the flow and use of information. In particular, information handling rules define the characteristics of the input information taken into the organization; the rules for distributing and condensing the input information; and the characteristics of information leaving the organization” (Cyert & March, 1992, p.127). “As a result of these rules, not everyone in the organization seeks or receives all the information the organization uses. The choices of who is to gather which information can be significant because the individual who encounters the information initially may also be the first to evaluate its relevance, determine its routing, and in general screen, condense, or highlight the information or some aspect of it.” (Choo, 2006, p246).

Organizations use plans and planning rules define how information is used to allocate the resources within the organization (Choo et al, 2000).

The process of information use to make decisions relies on how information needs are experienced, how information is sought and, how information is used or not used. This process is “influenced by cognitive, affective, and situational factors that play out at the
individual, group, and organization levels” (Choo, 2006, p.314). Figure 3.2 represents how these elements are interconnected.

![Decision Making Cube](image)

**Figure 3.2: Decision Making Cube “Knowing Cube” (Source: Choo 2006)**

### 3.6 Information Use in Software Testing

During Conference of the Association for Software Testing (CAST) 2007, Dr. Cem Kaner mentioned his experiences while working at Hewlett-Packard (HP), where he was the lead tester. While testing HP printers, all individual tests were successful on several occasions; however, testing in different sequences would lead to malfunctions. Based on these findings and other test results, the decision was made at HP to test each model in different sequences for 72 hours. So how was the decision made to test in different
sequences for 72 hours? How did HP come up with this number? Was this just an arbitrary number? According to Kaner (2007), this number was selected based on over six months of testing in the test lab. Factors such as cost, duration of warranty offered on each printer, and level of log tracking where the information would still be useful in case there was a malfunction led to this number. As an example, at the time (in the early 1990s) the amount of logs captured over 72 hours would be more than what HP believed to be cost effective and resource intensive to examine.

The reduction of ambiguity lies at the heart of information use in decision making (Choo, 2006). As an example, in a disaster such as the I-35W Mississippi river bridge collapse (Appendix D) when ambiguity is excessively high, organization members lack a clear and stable frame of reference. “Data acquisition, maintenance and delivery are a vital part of organizational life, but problems arise when we fail to recognize the necessary links to information use theory and knowledge” (Mutch, 1997).
Chapter IV - Methodology

In this chapter we discuss the research methodology. The chapter is organized as follow: We first provide a comparison view of the two research methodologies; qualitative and quantitative approaches. Then the selection of qualitative approach and grounded theory for this research is discussed. We conclude the chapter by reviewing the grounded theory techniques and how these techniques were applied to the collected data with the goal of developing a framework for the use of information in software testing.

4.1 Qualitative Vs. Quantitative Approach

Burrell & Morgan (1983) argue “philosophical assumptions underpin the research process, which dispose researchers towards different paradigms and methodologies” (as cited in Coleman & O'Connor, 2007b). According to Reichardt & Cook (1979) “the two research paradigms that have received most attention in the literature can be broadly labeled as positivist and phenomenological” (as cited in Coleman & O'Connor, 2007b). Creswell (2003) stated, “the most commonly used terms to differentiate these paradigms with respect to their associated methods and techniques, are quantitative and qualitative, respectively. Quantitative methods are based on the positivist paradigm while qualitative methods are built on a phenomenological worldview” (as cited in Coleman & O'Connor, 2007b).

Burns (2000) described “scientific [i]nquiry, which employs quantitative research methods, is used to establish general laws and principles and its approach can provide answers that have a provable base” (as cited in Coleman & O'Connor, 2007b). However, according to Strauss & Corbin (1998) “if one wants to study human behavior and the
social and cultural contexts in which it functions then the limitations of quantitative research becomes apparent and directs researchers towards qualitative techniques” (as cited in Coleman & O'Connor, 2007b).

Seaman (1999) stated “advocates of qualitative [research] in software engineering propose that a principal advantage of their usage is that they force researchers to delve into the complexity of the problem rather than abstract away from it, thus making the results richer and more informative” (as cited in Coleman & O'Connor, 2007b).

4.2 Qualitative Research

“By the term ‘qualitative research’, we mean any type of research that produces findings not arrived at by statistical procedures or other means of quantification” (Strauss & Corbin, 1998, p.11). Ritchie & Lewis (2003) argue that “although there is still some debate, the general consensus is that qualitative research is a naturalistic, interpretative approach concerned with understanding the meanings which people attach to actions, decisions, beliefs, and values within their social world, and understanding the mental mapping process that respondents use to make sense of and interpret the world around them” (as cited in ONS, 2007). The purpose of this research is to develop a framework of the use of information by software testers; hence, we want to study how software testers use information to perform their tasks. Since, “qualitative research can describe or provide further understanding of a subject and its contextual setting, provide explanation of reasons and associations, evaluate effectiveness, and aid the development of theories or strategies” (ONS, 2007) and qualitative research allows us to “commence [our] research project from an inductive position, [we] seek to build up a theory which is
adequately grounded in a number of relevant cases” (Saunders et al, 1996) hence, a qualitative approach was selected to complete this research.

4.2.1 Sampling

In qualitative research we look for a sample population that would increase the likelihood of providing the data that explains the research questions and goals. The data collected is the backbone of theory generated; hence, the study sample influences the results significantly (Strauss & Corbin, 1998).

“A qualitative study aims at capturing as many manifestations of the phenomenon as possible. To achieve this, we aim at getting diversity in our respondents to ensure a variety of examples that are related to the phenomenon (Barbour, 2001). We look for a variety of specific cases to maximize variations of cases to analyze. More variations allow the investigator to understand the phenomenon in many different ways. The research seeks to offer a comprehensive description and explanation of the phenomenon.

Lincoln and Guba (1985) suggest that the sampling process in qualitative research should aim at getting a wide range; including negative cases, typical cases, and critical cases relating to the phenomenon under investigation. The participants are chosen because the researcher believes they have the capability to respond to the research question because of their experience or involvement with the phenomenon (Barbour, 2001), or because they can give a different view that will open a window for further data collection and analysis (Cutcliffe, 2000)” (as cited in Mutshewa, 2004, p.97)

The participants of this research were individuals who had worked on a software testing project for duration of at least one project. Research participants and their breakdown by title and project are demonstrated in table 4.1.
4.2.2 Institutional Review Board

This research involved interviewing software testers, in accordance with GWU’s policies that requires an Institutional Review Board (IRB) approval for conducting research that involves human subjects, we completed the needed training and passed the examination in January of 2008 and submitted our application for IRB approval in January 22, 2008. The IRB application (Appendix C) was approved on February 14, 2008. The interviews for data collection and analysis started after the IRB approval.

4.2.3 Selection of Grounded Theory

As the objective of this research is to develop of a framework of the use of information in software testing base on interviews with software testers, a qualitative research (as detailed in section 4.2) was selected to conduct this research. From available methodologies within qualitative research, grounded theory was chosen because:

- Grounded theory enables the researcher to generate “a theory that is inductively derived from the study of phenomenon it represents” (Strauss & Corbin, 1990, p.9).
- “Given the lack of an integrated theory in the literature as to how software companies are conducting” software testing (Coleman & O’Connor, 2007b; Bertolino, 2007), an “inductive approach, which allows theory to emerge based on the experimental accounts of software testers themselves, offers the greatest potential” (Coleman & O’Connor, 2007b).
- Grounded theory is” an established and credible methodology in sociological and health disciplines,… and a burgeoning one in the IT arena” (Coleman & O’Connor, 2007a).
• Grounded theory “is renowned for its application to human behavior” (Coleman & O’Connor, 2007a). Software testing is a labor intensive activity and software testing “process relies on human compliance for its deployment” (Coleman & O’Connor, 2007a).

4.3 Grounded Theory

Glaser and Strauss first introduced grounded theory in 1967. They associated the discovery of grounded theory to Public Health Service Research Grant NU-00047 from the Division of Nursing, Bureau of State Services - Community Health (Glaser & Strauss, 1967). They described the basic theme as “the discovery of theory from data systematically obtained from social research.”

Grounded theory was developed as a “systematic, inductive, and comparative approach for conducting inquiry for the purpose of constructing theory” (Charmaz, 2006; Charmaz & Henwood, 2007), and “its name underscores the generation of theory from data” (Charmaz, 2003). “When the principles of grounded theory are followed, a researcher using this approach will formulate a theory, either substantive (setting specific) or formal, about the phenomena they are studying that can be evaluated” (Busch, 2008, p.72).

“The term ‘grounded theory’ refers both to a method of inquiry and to the product of inquiry. However, researchers commonly use the term to mean a specific mode of analysis” (Charmaz, 2003). Strauss and Corbin (1990) defined grounded theory as “a theory that is inductively derived from the study of the phenomenon it represents.”

“Grounded theory methods are a set of flexible analytic guidelines that enable researchers to focus their data collection and to build inductive middle range theories
through successive levels of data analysis and conceptual development” (Charmaz, 2005, p.507).

“It is impossible to fully comprehend a phenomenon without understanding the context in which it is expressed” (Locke, 2001, p. ix). “A grounded theory is one that is inductively derived from the study of the phenomenon it represents. That is, it is discovered, developed, and provisionally verified through systematic data collection and analysis of data pertaining to that phenomenon. Therefore, data collection, analysis, and theory stand in reciprocal relationship with each other. One begins with an area of study and what is relevant to that area is allowed to emerge” (Strauss & Corbin, 1990, p.23).

“Grounded theory has three characteristics (Orlikowski, 1993): (1) inductive, (2) contextual, and (3) processional” (as cited in Tian, 2006, p.41). Research on information use in software testing fits in these three characteristics. Firstly, new innovations and practices emerge in the field of software testing, and thus an inductive discovery methodology such as grounded theory is appropriate. Secondly, it is important to understand information use in software testing in the specific project and human behavior context. Thirdly, both software testing and information use are processes.

Grounded theory methodology “is described as an analytical process requiring abstract thinking, comparison of data, recognition of values and biases, reflection on experience, and the ability to communicate the written word accurately” (Murphree, 2005, p.11). Creswell (2003) describes grounded theory as “the constant comparison of data with emerging categories and the theoretical sampling of different groups to maximize similarities and differences of information” (p.62).
“All analyses come from particular standpoints, including those emerging in the research process. Grounded theory studies emerge from wrestling with data, making comparisons, developing categories, engaging in theoretical sampling, and integrating an analysis. But how we conduct all these activities does not occur in a social vacuum. Rather, the entire research process is interactive; in this sense, we bring past interactions and current interests into our research, and we interact with our empirical materials and emerging ideas as well as perhaps those of granting agencies, institutional review boards, and community agencies and groups, along with research participants and colleagues” (Chazmaz, 2005, p.510). Holstein & Gubrium (1995) stated “Neither data nor ideas are mere objects that we passively observe and compile” (as cited in Charmaz, 2005).

“Grounded theory is a methodology in which data are collected and analyzed with the end goal of developing a valid, reliable theory that lends reality to the topic being studied. It uses an interactive, often overlapping process applying three levels of coding to data” (Murphree, 2005, p.12).

4.4 Data Analysis

According to Strauss & Corbin (1998); asking questions and constant comparison are two essentials for using grounded theory approach. In order to formulate the areas where we needed to ask questions, we started the research by meeting with a group of four software testers and presented them with the goals of the research. Through an open forum discussion we came up with an initial set of areas to explore in our interviews. These areas are presented in Appendix B in form of questions. As we progressed through the interviews new areas and categories came to light and we updated our questions to collect relevant data accordingly.
“Theoretical sampling refers to the process of collecting, coding and analyzing data [while] simultaneously generating theory” (Coleman & O'Connor, 2007b).

“Next, theoretical sampling was used to select the type of data to sample, and then data was collected through administrating and transcribing interviews. After data collection, the data was analyzed through, open, axial and selective coding. From the results of data analysis, theories emerged and developed. At this point in the process, the data was examined to determine if theoretical saturation had been reached; that is, no new data or relationships were found (Strauss & Corbin, 1998). Until this theory saturation point was met, more samples were collected through theoretical sampling and the process repeated. However, after theory saturation was reached, the next step was to validate theories that were developed, by presenting the theories to a small group of [software testers] for validation.” (Burgos, 2007, pp.27-28). Figure 4.1 is the research methodology flow diagram.
Figure 4.1: Research Methodology Flow Diagram (Adopted from Burgos 2007)
4.5 Data Collection

The data for this research was collected by conducting interviews. A total of 52 interviews were conducted with 34 interviewees that matched the selection criteria for this research. The selection criteria included people that were working or had worked in the field of software testing for the duration of at least one project or lead a software testing project. The interviews spanned over 15 different projects as noted in Table 1.

<table>
<thead>
<tr>
<th>Code</th>
<th>Title</th>
<th>Project</th>
<th>Code</th>
<th>Title</th>
<th>Project</th>
</tr>
</thead>
<tbody>
<tr>
<td>STI-1</td>
<td>Tester (Team Lead)</td>
<td>P4</td>
<td>STI-18</td>
<td>Tester</td>
<td>P15</td>
</tr>
<tr>
<td>STI-2</td>
<td>Tester</td>
<td>P7</td>
<td>STI-19</td>
<td>Consultant</td>
<td>P9</td>
</tr>
<tr>
<td>STI-3</td>
<td>Consultant</td>
<td>P2</td>
<td>STI-20</td>
<td>Tester</td>
<td>P14</td>
</tr>
<tr>
<td>STI-4</td>
<td>Tester</td>
<td>P3</td>
<td>STI-21</td>
<td>Tester</td>
<td>P5</td>
</tr>
<tr>
<td>STI-5</td>
<td>Tester</td>
<td>P3</td>
<td>STI-22</td>
<td>Tester</td>
<td>P1</td>
</tr>
<tr>
<td>STI-6</td>
<td>Tester</td>
<td>P4</td>
<td>STI-23</td>
<td>Tester</td>
<td>P10</td>
</tr>
<tr>
<td>STI-7</td>
<td>Tester (Team Lead)</td>
<td>P7</td>
<td>STI-24</td>
<td>Tester</td>
<td>P10</td>
</tr>
<tr>
<td>STI-8</td>
<td>Consultant</td>
<td>P10</td>
<td>STI-25</td>
<td>Tester</td>
<td>P4</td>
</tr>
<tr>
<td>STI-9</td>
<td>Consultant</td>
<td>P10</td>
<td>STI-26</td>
<td>Tester</td>
<td>P11</td>
</tr>
<tr>
<td>STI-10</td>
<td>Tester</td>
<td>P9</td>
<td>STI-27</td>
<td>Tester/Developer</td>
<td>P4</td>
</tr>
<tr>
<td>STI-11</td>
<td>Tester</td>
<td>P5</td>
<td>STI-28</td>
<td>Tester</td>
<td>P11</td>
</tr>
<tr>
<td>STI-12</td>
<td>Tester</td>
<td>P5</td>
<td>STI-29</td>
<td>Project Manager</td>
<td>P12</td>
</tr>
<tr>
<td>STI-13</td>
<td>Tester</td>
<td>P6</td>
<td>STI-30</td>
<td>Tester</td>
<td>P12</td>
</tr>
<tr>
<td>STI-14</td>
<td>Tester</td>
<td>P6</td>
<td>STI-31</td>
<td>Tester</td>
<td>P4</td>
</tr>
<tr>
<td>STI-15</td>
<td>Tester</td>
<td>P6</td>
<td>STI-32</td>
<td>Tester</td>
<td>P4</td>
</tr>
<tr>
<td>STI-16</td>
<td>Tester</td>
<td>P8</td>
<td>STI-33</td>
<td>Project Manager</td>
<td>P13</td>
</tr>
<tr>
<td>STI-17</td>
<td>Tester</td>
<td>P8</td>
<td>STI-34</td>
<td>Tester</td>
<td>P13</td>
</tr>
</tbody>
</table>

We tried to keep the interviews informal and open-ended to allow for the respondents to speak as freely as possible. At the beginning of each interview we informed the participant of a brief background of the research and that the research is done for academic purposes. We gave the participant the interview consent form to sign (Appendix A) and then we started the interview recording. One of the interviewees did not want to be recorded. In that one interview, we took detailed notes to ensure we capture the data. The interviews were our tool for data collection. Afterwards, we
applied open, axial, and selective coding techniques available within grounded theory to the data collected to develop a framework of the use of information in software testing.

4.6 Coding Methods

4.6.1 Open Coding

Open coding is the first step in data analysis to develop a grounded theory (Strauss & Corbin, 1998). Dey (1999) summarizes open coding as “categorizing the data” (p.98). During open coding we “broke [the collected] data down into discrete parts that are later stitched together again through theoretical connections” (Dey, 1999, p.259) and analyzed the data to identify and name concepts or categories derived from our interviews. The interviews started after the IRB approval on February 14, 2008 and the last interview was held on July 30, 2008. All interviews were recorded digitally and then transcribed to text by using Dragon NaturallySpeaking 9.5© Software. The transcription software was limited to training the software to a particular voice. As a result, the researcher’s voice was used to create voice recognition files and all the interviews upon their completion were listened to and read to the voice recognition software. This process helped us to identify the categories from the interviewee’s comments.

“Open coding requires a brainstorming approach to analysis because, in the beginning, analysts want to open up the data to all potentials and possibilities contained within them. Only after considering all possible meanings and examining the context carefully is the researcher ready to put interpretive conceptual labels on the data” (Corbin & Strauss, 2008, p.160). So, we started with open coding by examining the data carefully and labeled the data based on its context and meaning. At this point we looked through the concepts we had labeled, thoroughly, and reviewed the original comments from the
interviewee for common links and similarities. We wanted to ensure that if two or more labels represent the same concept we would capture that and chose a label or category name that would best represent the finding. “Conceptualizing data not only reduces the amount of data the researcher has to work with, but at the same time provides a language for talking about the data” (Corbin & Strauss, 2008, p.160). This process resulted in identification of 32 labeled categories.

“In this initial phase the objective was to gather as much information as possible so that it could be broken down into descriptive parts. Participants’ words were reviewed by word, by sentence, by paragraph, or by page for what was actually said. From these words concepts or interpretations were identified and assigned labels. Asking questions helped to group similar concepts into categories. Categories are concepts that represent the central ideas of the data being studied. Categories can be broken down into properties (defined as characteristics of the category). Properties can be further broken down into dimensions (locations along a continuum). It is comparison of the dimensions that provide variation” (Murphree, 2005, p.12).

Following this rational process the 32 identified categories were looked at again, and 8 main categories emerged from the data analysis.

4.6.2 Axial Coding

Axial coding is the second step in data analysis to develop a grounded theory (Strauss & Corbin, 1998). Dey (1999) defines axial coding as “connecting categories” (p.98).

According to Strauss & Corbin (1998) during axial coding the theory takes shape by looking at labeled categories and defining relationships between categories. Additionally during axial coding process the relationship between categories and corresponding
subcategories are defined by looking at similarities and ‘constant comparison’ between labeled concepts. This process continued until we reached theoretical saturation. “A category is considered saturated when additional data do not bring forth any new information. Strauss and Corbin indicate … that saturation is a matter of degree. It is more a matter of arriving at a point where collecting additional data seems counterproductive (Strauss and corbin, 1998, p.136)…. When saturation is reached, the researcher moves to selective coding, the last level of coding process” (as cited in Murphree, 2005, p.14).

4.6.3 Selective Coding

Through open and axial coding we identified categories and subcategories and we started to put the pieces of the framework of the use of information in software testing together. The goal of selective coding is to form a theory by identifying the core category and refining and defining the relationships between the core category and other categories (Strauss & Corbin, 1998). Dey (1999) defines selective coding as “focusing on a core category” (p.98). The core category in our research was information use behaviors. Selective coding continued for each category until theoretical saturation was reached (Glaser & Strauss, 1967; Strauss & Corbin, 1990; Locke, 2001). This means until:

(1) “No new or relevant data seem to emerge regarding a category.

(2) The category development is dense, insofar as all of the paradigm elements are accounted for, along with variation and process.

(3) The relationships between categories are well established and validated” (Strauss & Corbin, 1998, p.188).
“Selective coding involves the identification of the core category around which the inquiry then revolves [and this] serves as a way of both delimiting and integrating theory” (Dey, 1999, p.261). Hence, through selective coding we were able to identify the core category which was information use behaviors, and define the relationships between the core category and other categories to form a framework of the use of information in software testing.

The complete findings of this research are detailed in Chapter V; conclusions and the theory from the findings noted in Chapter VI.

4.7 Verification and Validation

Locke (2001) under the topic of “evaluating the ‘goodness’ of the composed theory” (p.59) suggests that “a good theory is one that will be practically useful in the course of daily events…. In a sense, a test of a good grounded theory is whether or not it works ‘on the ground,’ so to speak” (Locke, 2001, p.59). Hence to validate our findings, we presented it to the group of four software testers. The participants’ were asked if the findings were accurate with what they see in their environments. We noted their comments which resulted in a few minor changes in the findings. We also looked through already gathered data for existing data supporting the validation session comments. Data was already existed in the 34 interviews conducted and no new interviews were scheduled.
Chapter V - Findings

In this chapter we document our findings, by applying grounded theory techniques (as described in Chapter IV) to the collected data to develop a framework of information use in software testing.

Figure 5.1 shows the analytical path we have followed. The respondents’ comments are noted in quotations and are used as supporting evidence for the emergent categories.

The respondents’ comments were collected from interviews conducted with 34 participants that matched the selection criteria. The quotations are referenced in the text.
using the codes that were assigned to the interview sessions held during the data collection. In addition the quotations are used as supporting evidence for our findings.

The format for presenting the study is as follows: we first explore what is perceived as information by software testers. Then we start with the first research questions in which the question ‘how do software testers conceptualize information used in software testing?’ is explored which leads to our discovery of two distinct information types identified during data gathering and analysis. Then findings for the second research question ‘how do software testers seek information?’ are presented. Afterwards, the research questions ‘how do software testers use information? And what factors influence the way software testers use information in practice?’ are explored.

5.1 Information Stages

Before delving into answering the research questions, we found it necessary to get an understanding of how participants viewed the term “information” in the context of software testing. We discovered that software testers’ perception of information is directly correlated with their assigned tasks within the software testing project. As an example, one of the respondents that was a project manager, referred to project duration, cost, and the number of bugs remained to be fixed as information. Another respondent that had a role of test designer perceived design documents, requirements documents and any data that would assist him in creating and executing test scenarios as information.

In another word, the role that individuals play within the software testing projects defines ‘relevant information’. Further analysis of relevant information revealed that there exists two distinct information categories:
(1) **Pre Software Testing Information** - Information needed for testing itself that is normally captured and labeled as requirements and design of the software.

(2) **Post Software Testing Information** - Information gathered as a result of conducting a test or series of tests, which is used to perform future tests, create bug reports, and make decisions regarding the quality state of the software.

As one of the respondents states:

“We generally start with design and requirements documents that were gathered during the initial phase. We turn these requirements into test plans and start the testing. We provide our findings to the project manager and developers and collectively based on the test results we decide if further tests are needed and whether the developed software or a particular function behaves as intended and documented in the design and requirements document...and whether the software is ready for implementation” STI-25

STI-25 describes these two distinct information categories sought in order to start the project activities and completing tasks.

Another respondent STI-19 talks about how testing can be designed to produce information that will be used in making an informed decision.

“There are several basic questions that management needs to know and we discuss in our meeting. As an example we need to know if the software we are testing does what we want it to do and if it does not what is the level of effort needed to fix it. Of course, the level of effort cannot be an answer that the result of tests answers directly; however, using and categorizing the bugs would help the developers to better estimate the level of effort, ….” STI-19

Upon further analysis of the two emerging information categories, it was discovered that a logical relationship exists between these information categories (Figure 5.2), which can be illustrated as:

Requirements <-> **Pre Software Testing Information** <-> Developed Software <-> **Post Software Testing Information** <-> Implementation
Where requirements describe the information gathered through meetings with stakeholders and developers during the design and development phase of a software development project. During this phase initial test plans and strategies are developed based on information conceptualization factors and conceptions which are explored in section 5.2.

The Pre Software Testing Information is gathered from the requirements to design development and test plans and strategies. This gathered and conceptualized information is used to ultimately design and develop software.

Testing is done on the developed software and the results (Post Software Testing Information) are compared with the requirements and Pre Software Testing Information to provide a picture of software quality.
The initial information type, requirements, is explored in depth in the Information Seeking section of this chapter. The informational elements are explored in Individual Role Context, Domain Knowledge, Meta-information, Testing Strategy, Organizational Environment, and Information Use Behaviors sections. The information types relating to testing of the intent of the designed software and related categories are explored in the testing strategy and decision making sections.

5.2 Information Conceptualization

The first research question was ‘how do software testers conceptualize information used in software testing?’ as described in section 5.1, relevant information has a direct correlation with the role that individual plays within the project. Similarly, we discovered this pattern to be true for information conceptualization as well. As an example, one of the respondents (STI-2) referred to the software platform and operating system as essential information to conduct testing. Another respondent (STI-33) that was leading a software testing project referred to the project budget, number of resources, and contract structure as essential information. There are also, certain organizational elements that influence information conceptualization.

From the gathered data we identified four ways in which respondents conceptualize information (Conceptions). These conceptions are:

- Source conception
- Process conception
- Knowledge conception
- Organizational conception
5.2.1 Source Conception

Source conception refers to the source of the information and the way information is accessed or presented. This could refer to the physical form of information such as “Paper copy”, “Colored chart”, “Excel [or electronic] file”, and Information retrieved from a system.

The quotations cited below exemplify the expressions used by some respondents when referring to source conception.

“So an up-to-date paper copy of the bug report is handed out in our weekly meeting and everyone in our team will scan through them.” (STI-6)

Another respondent said:

“The number of bugs found and their severity level is stored in our shared folder for everyone to see. It is an Excel file with a colored chart showing percentage of each severity level reported” (STI-8)

The way that information is, presented, stored, accessed, or viewed are the emphasis of the responses in the quotations above.

5.2.2 Process Conception

Process conception refers to the perception of the respondent on the set of procedures that would input or extract data to or from a system. As an example, STI-5, when talking about the process of documenting a bug found during a testing session said:

“So we have a bug tracking application which we call ‘XBUG’… As an example when I find a bug I would login to the XBUG application and put a description of the bug and you see the [XBUG] application has a wizard that will guide you to assign a severity level. As an example a misspelling or a cosmetic error is a sev4, but a submit error is either a sev1 or sev2 [depending on the function it performs and the application guides you through the whole process” STI-5

Another respondent when talking about retrieving information about resource utilization said:
“if I want to know what the resource utilization of a system was for the past month, I would login to our monitoring tool – NetIQ – and run the resource utilization report for the resource in question” STI-8

In the examples above the respondents were emphasizing the data input and data extracting processes more than other attributes used to describe information.

5.2.3 Knowledge Conception

Knowledge conception refers to the previous training, experience, knowledge, and skill of software testers and basically the human involvement in conceptualization of information. In knowledge conception the software tester refers back to a previously acquired knowledge or skill to conceptualize information. As an example one of the respondents said:

“We use LoadRunner to simulate a load for performance testing….. since I have attended the [LoadRunner] training, I have been able to use some of the features that I did not know about in creating a test…. [and] most importantly I understand the test results so much more. Now I am able to look at the test results and let my manager know where the bottlenecks are, which function, page, ….” STI-7

In the quotation above the respondent is using knowledge acquired through a previous training class. This knowledge was found to be essential to comprehend and perform the required tasks.

In section 5.7.1 under the title of “Domain Knowledge” this topic is explored in great depth during information use.

5.2.4 Organizational Conception

In organizational conception the software tester refers to organizational policies during information conceptualization. The quotation sited below exemplifies the expressions used by some respondents when referring to organizational conception:

“When there is a network or an application outage, a mass email is send to the entire organization to let everyone know of the outage and status” STI-2

56
Or another respondent said:

“In our requirements gathering meeting we generally do not commit to any dollar amounts as we would need to check with our contracts department for those matters” STI-9

In responses above respondents referred to and emphasized the organizational policies and rules more than other attributes used to describe information.

During the data gathering respondents pointed out different information seeking behaviors and elements. These behaviors and elements are illustrated in the next two sections.

5.3 Information Seeking Behaviors

The second research question and objective was to determine how software testers seek information. In conducting the field interviews, four information seeking behaviors became apparent. We use the participants’ responses and link their information seeking descriptions to the emerging information seeking behaviors. Based on the data gathering and analysis the information seeking was done in at least one of the following four information seeking modes:

(a) *Software tester knew what he was searching for.* In this method, the software tester generally used some sort of a checklist to gather the information:

“Most of the time, user requirements that users are asking us for a specific thing will normally not be concerned with how the software should be installed, which options need to be installed. They just know which features need to work; those normally would become part of the testing checklist. So we would make sure, based on the checklist, we have the software configured based on the requirements before it is tested. So basically, the testing checklist is a combination of the user requirements in addition to things such as installation procedures, uninstall procedures, and load testing.” STI-2
(b) **Learning mode.** In this method generally, the software tester’s knowledge of the software was outside of his area of expertise. Several participants describe this method when testing specialized software such as scientific software:

“The lack of knowledge with both developers and testers has frustrated our stakeholders. Our developers and testers do not have a strong financial background so it requires a lot of sessions with the stakeholders to the point that one of our stakeholders said in a meeting that if I would have gone and learned how to develop software we would have been done by now….Due to complexity of the software we need to rely heavily on our stakeholders to validate results during testing” STI-10

(c) **Exploratory.** In this method of information seeking; the software tester has an idea of what he is looking for but he is basically fishing for information:

“If there was a previous contract, I would hope the previous contract existence has documented the steps regarding the application, different functions, and what areas need to be tested. That would be the starting point. If it’s an internal project, I would definitely go to previous manager or project manager and other people involved in previous releases and testing and would get as much as information as I can pertaining to the project. And if all of the resources are exhausted I would dig into the system itself. Look in the source code, bug reports, etc.” STI-4

“A lot of times the person requesting the software test, really cannot provide any details or requirements on how users plan to use it. And I really can’t start thorough testing until I talk to someone who knows how to use the system. Otherwise I have to make my own guesses.” STI-2

(d) **Confirmatory.** In this method of information seeking, the software tester has reviewed the design documentation and is seeking for data to confirm the documentation and known information.

Confirmatory information seeking was seen more with software testers working in government organizations where the intended users were the agency or the department employees. The testers were familiar with information such as users, operating system, workstation, and server configurations. During information seeking, generally, a combination of these methods was used.

“After reviewing the requirements document; we generally have a meeting to start our testing. We refer to it as the ‘kick start meeting’. During this meeting we get the developers, our manager and we try to get a semi technical representative from the user side….. In this meeting initially we want to confirm the information in the requirements document. Things like the intended use of software, number of users, the
operating systems, browsers, that the software will be installed on... and then we try to find out as much information as we can by finding out possible workstation configurations, location of the users, any special users such as users that are visually impaired and use screen readers.” STI-25

As interviewer STI-25 points out; during their ‘kick start meeting’ they firstly use confirmatory information seeking techniques, ‘In this meeting initially we want to confirm the information in the requirements document’ and also exploratory information seeking to try to find out any relevant information that may be useful in starting the project ‘and then we try to find out as much information as we can by finding out possible workstation configurations, location of the users, any special users such as users that are visually impaired and use screen readers’. In the case of interviewee STI-10 the tester does not have a good grasp of the software and its concept and exploratory and learning mode information seeking is used.

In mostly all cases exploratory and confirmatory information seeking behavior was used to gather requirements. Requirements are essential in software development life cycle. In the next section we explore elements that influence information seeking behaviors.

5.4 Information Seeking Elements

We analyzed the data for elements influencing information seeking behaviors. Four elements were discovered to influence information seeking behaviors. These elements are:

1. Degree of motivation and self interest (Motivations)
2. Avoiding problem situations (Negative Elements)
3. Sources of information
4. Effort needed to find information source
5.4.1 Motivations

The data showed that degree of motivations and self-interest in the topic or problem contributes to the level of effort and energy the software tester invests in information seeking. For instance, one respondent pointed out that:

“… everyone in our team was looking to find out what the source of the error was. Our project manager was breathing down our necks as we were running late with the release. So as you can imagine I was so excited as a junior member of the team to be able to find out the source of the error which was a database setting that needed to be changed…. It was such a great feeling of accomplishment. You know we have people there that have over 20 plus years of experience” STI-11

In the situation explained above, being able to prove oneself as a contributing member of the team motivated STI-11 to search and seek information.

Also as another respondent stated:

“My supervisor asked me to come up with ideas to have less server outages, and when we do, for us to have a way to respond and find the error quickly. I recommended LiteSpeed monitoring tools that can monitor most of the DBMS available in the market. Our relationship manager bought the LiteSpeed monitoring tools and their client manager had scheduled a training session for the application server. I was really excited to be a part of it as it was a new field and the automated monitoring process would make our life more comfortable. I was also excited to be an administrator of the application server. I finally got the training and started to monitor our databases. I got appreciated by our manager and higher management level for not having any database outages for past six months. I feel good for recommending this monitoring software which made everyone happy.” STI-6

As STI-6 pointed out the positive feeling that was felt by being recognized through making a useful recommendation during information seeking session excited him.

The motivational factors like being recognized, feeling of contributing to the team, learning something new and motivating oneself enhance information seeking by motivating the individuals to seek and provide information more freely and willingly. Cognitive, affective, and situational factors that effect information seeking in negative ways were also discovered. These factors are explored in the next section.
5.4.2 Negative Elements

The data showed that there are three factors that negatively effect information seeking. These factors are:

- Fear of rejection and not fitting in
- Fear of Retaliation
- Lack of motivation

5.4.2.1 Fear of Rejection and Not Fitting In

During the interviews, several respondents pointed out that they have worked on projects where reporting an error has caused anger from developers or managers. Nowadays, in IT organizations, staff have annual reviews that reflects their past performance. These reviews may lead to pay increases and / or promotions. For instance, respondent STI-32 comments in this regard:

“As an example, ehhhh, I worked on a project last year where the developer came to me after the first round of testing and reporting with an angry and upset attitude and told me that he wanted to see the bug report before it is submitted because I was making him look bad by pointing out all these nonsense errors”

When the STI-32 was asked why the reaction of the developer to the bug report mattered STI-32 continued:

“I had just been hired for the project and I was new to the company, on the other hand he [the developer] had been with the company for several years and he was really close with my boss. I really did not want to come out as someone that is not a team player and did not want to rub anyone off the wrong way”

It is apparent that the respondent’s perception of the project environment and wanting to establish a good working relationship with the peers has affected the way work was performed and information was shared and reported as the respondent continued when asked how this affected the job performance:
“I would share my findings with the developer first and he [the developer] would ask for time to fix the bugs found and resubmit the code for a re-test. This made my life a living hell as I had to re-test everything to ensure the new code does not break anything else and on a couple of occasions I missed my deadlines”

Fear of rejection from the more senior member of the project team and being viewed as someone who is not a team player, plays a role in how software testers seek information.

5.4.2.2 Fear of Retaliation

In the previous section we talked about fear of rejection and not fitting in. During the interviews when gathering data several respondents hinted that during the information seeking group meetings they would not participate or voice their opinion if the opinion differed from their superiors in fear of adverse action. For instance one respondent said:

“Few years back while I was on a contract with […] I expressed my opinion in a meeting with the client that the product we are about to purchase to replace our existing training tracking software would not work in our environment. Next thing you know I was called in the meeting with few of my company’s supervisors and they said the client has requested us to let you go!.. make a long story short I was moved to another project and later I found out after a year and a half and about 2 million dollars in expenses the previous client had decided to abandon the project…. Since then I am extremely cautious with voicing my opinion in client meeting” STI-5

As stated above STI-5 has an experience of adverse action taking place for disagreeing with more senior members of the group and that has influenced current and perhaps future information sharing behaviors. Further investigation of this finding revealed that in certain project environments such as projects where a group is subcontracted to research, test, and make recommendations about the best suited product for an environment, the decision about the desired product of use may have already been made before the project start. As an example one of the respondents said:

“we were contracted to assess, test, and make a recommendation for an asset tracking system. In the kick off meeting with the client we soon realized that the client had already chosen and purchased SunFlower software” STI-31
The respondent continued when asked how did this influenced the project:

“our role shifted from evaluators and testers to implementation specialists, as our team started working on deployment of SunFlower asset tracking software”STI-31

As STI-31 elaborates, in these project environments the primary contractor (stakeholders) expects the subcontractor (Consultants) to seek and report data that supports a decision already made. Since the final decision has already been made, information seeking in these environments is biased to support the already made decision. Deviations from the new project tasks and environment could lead to unsatisfactory job performance and adverse actions.

5.4.2.3 Lack of Motivation

Lack of motivation leads to an employee’s lack of desire and commitment and manifests itself in lack of effort to perform tasks. Lack of motivation may be due to individual, team, or project environments. As STI-23 said:

“our manager continuously talks about having an open door policy in our meetings, but every time I try to talk with him it seems like he does not want to give me the time of day and at the end he brushes me off and tells me to just work with the project plan, so I just don’t bother ….”

There is a clear lack of motivation in the STI-23 environment. This lack of motivation has disabled information flow between the tester and his manager. In this case information sources have been limited to meetings and the project plan. STI-23 feels that his findings and suggestions have no value towards the outcome of the project. This project environment has caused lack of motivation to seek and share information. In this environment STI-23 has been eliminated as an information source.

Perception of unequal treatment by supervisors also may lead to an environment that promotes lack of motivation. As one of the respondents states:
“There are other members of the team that their findings or opinions never get questioned but I have to go an extra mile and get bombarded with questions if I voice my opinion…so there are times that I keep my opinion to myself to avoid having to explain” STI-18

As STI-18 states above, the perception of unequal and unfair project environment has negatively effected the contributions that STI-18 would otherwise make, and in turn has reduced STI-18’s motivation to contribute as an information source to the project.

5.4.3 Sources of Information

During information seeking software testers select a source (or sources) of information that is perceived to have the greater probability of providing information that will be relevant and useful. As one of the respondents stated:

“We generally search the error message online. These days you will get a hit for almost anything that you search for. So we look at the sources. If the source is something like Microsoft.com we would dig further to see if their workaround resolves our issue but if it is a blog site we would take a note of that in our bug report and we would let the developers go through the steps and decide whether or not the information is accurate or applies to the bug” STI-34

As another respondent said.

“We have a bug tracking and resolution database where we keep a record of all the bugs and how they were fixed. So when we start a project we look in our bug tracking and resolution database and look for any issues or problems other teams may have experienced with a similar implementation and we include that in our timeline estimations.” STI-22

As stated above during information seeking software testers look in different sources. These sources include online resources (external), internal knowledge base, individuals that may be able to provide the information, and previous projects (internal).

5.4.4 Effort Needed to Find Information Source

Amount of time and effort that is required to locate or contact information source also effect information seeking. As an example one of the respondents stated:
“it is like we ask the […] manager for information and we were expecting to get a response in a few days but it’s like you ask for information and people just disappear… so we end up making our own assumptions” STI-8

Another respondent said:

“A lot of times the person requesting the software test, really cannot provide any details or requirements on how users plan to use it. And I really can’t start thorough testing until I talk to someone who knows how to use the system. Otherwise I have to make my own guesses.” STI-2

As STI-2 and STI-8 stated in quotations referenced above, there is a duration that they would wait to hear back from an information source. Once that time line has lapsed they would look into other sources or make their own assumptions to be able to proceed with their tasks.

To summarize, motivations, negative elements, sources of information, and the availability of information source in the projects emerged as key information seeking elements determining the theoretical entities in software testing process. A listing of these elements is listed in Table 5.1.

<table>
<thead>
<tr>
<th>Positive factors</th>
<th>Negative Factors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Degree of self interest</td>
<td>Fear of rejection</td>
</tr>
<tr>
<td>Learning experience</td>
<td>Fear of retaliation</td>
</tr>
<tr>
<td>Understanding of the assigned tasks</td>
<td>Negative team environment</td>
</tr>
<tr>
<td>Sense of accomplishment</td>
<td>Lack of motivation</td>
</tr>
<tr>
<td>Recognition by peers</td>
<td>Cognitive Complexity</td>
</tr>
<tr>
<td>Positive team environment</td>
<td>High staff turnover</td>
</tr>
<tr>
<td>Known information source</td>
<td>Confusion about assigned tasks</td>
</tr>
<tr>
<td>Availability of information source</td>
<td>Unknown information source</td>
</tr>
<tr>
<td>Sense of credibility of information source</td>
<td>Inability to contact information source</td>
</tr>
<tr>
<td>Ability to contact information source</td>
<td>Perception of unequal treatment</td>
</tr>
</tbody>
</table>

Gathering and managing requirements is one of the initial steps during information seeking. In the next section we will explore this topic and its associated risks.
5.5 Requirements and Associated Risks

The requirements are gathered initially to provide a road map for software developers to code and develop the software or to assist with selection and tailoring a COTS software to satisfy user needs. In both cases, the requirements gathered represent information set for software testers to test the product. The requirements represent the needs of the product’s stakeholders.

“...We tap our user base, to gather requirements and to ensure we have a good idea about what the business need is. So when somebody comes back and says, we expect the software to do, Item 1, Item 2, Item 3, … we collect all those things and then figure out which of these nice to haves are a must have. And then so we, internally prioritize ‘nice to haves’ and ‘must haves’ and we come up with the matrix of the top 5 or the top 10, basic things that any intended software; something that we do in house or something that we go look for outside; like a COTS; those things meet our basic five priorities or requirements that we have in place.” STI-1

Based on interviews in the data gathering Information relevant to software testing at this stage (requirements) generally fell under the following categories:

- User requirements
- System requirements
- Project Requirements

5.5.1 User Requirements

Based on data analysis the respondent perceived that user requirements effectively lay the foundation for developers, testers, and implementers to begin determining the functionality, responsiveness, and interoperability required of that system:

“First thing is always it is what we call the requirements gathering phase or requirements analysis phase. So we try to understand the main requirements from the customer, try to understand their environment, their architecture and network topology and all of that stuff based on data we figure out what the best product is that would meet their requirements.” STI-3

User requirements provide a picture of the ultimate users of the software and their expectations of functions they want the software to perform.
5.5.2 System Requirements

In this phase of information gathering the developers and testers ensure that they understand variables related to the platforms the software would need to support and other important details such as compliance to policies and regulations.

“A lot of times the person requesting the software test, really cannot provide any details or requirements on how users plan to use it. And I really can’t start thorough testing until I talk to someone who knows how to use the system.” STI-2

As user requirements provides a picture of what users expect of the software system; system requirements pinpoints the technical aspects of the software system, such as technology used and supported platforms.

5.5.3 Project Requirements

There are varieties of other factors that affect a software testing project as a whole such as budget, contract terms, and project schedule. In the context of information use by software testers; project requirements provide a baseline for decisions such as resource and budget allocations; time to market; and test selections. The relevant information gathering may vary from project to project but generally this information is available before the project starts in the form of a proposal and will eventually make it into a final document (i.e. contract). As STI-2 states:

“…when we go through our requirements gathering and trying to figure out the user environments we start mapping that to estimate the number of hours or days that would take us to implement that product. So if it is in a government environment our estimates generally make it to our bid which is the response for an RFP and in the private sector we generally provide a forward looking contract on how much things would cost and how long it would take to implement. That by itself could be tricky, as an example we are trying to expand our market share in the defense sector so our pricing tends to be tailored with minimum or no profit to be able to establish ourselves in the defense sector.”

There are different considerations when dealing with gathering project requirements. As STI-2 states, organizational goals affect these project deadlines. Time to market
could be a crucial variable in project requirements. With the competitive nature of the software market, at times release of a product 30 days later than anticipated could make a significant difference in attaining market share. All these variables are taken into consideration when finalizing project requirements.

5.5.4 Requirements Management Risks

Unstable and changing requirements introduce a risk factor in completing the project successfully, on time, and within the allocated budget. Changing requirements during a project may cause re-coding, redesign, and in general re-work of the software development and in turn software testing which can cause delay, and budget shortfalls. As one of the software testers responded when asked about challenges faced during the project:

“Most of the times the issues that we run into are scope creep. During the initial identification phase we have a tough time nailing down things that are good to have, things that are nice to have. And towards the end when we are close to delivering on a product, one of the biggest challenges that we have is scope creep. We need to make sure the user see something and they pick up the logical progression of maybe the future releases. And they want to enroll those features to the base product and that ends up with more testing and more integration issues so those sitting and expectation of users in terms of what is feasible, what can be done, at a later stage; that’s one of the biggest challenges that we have.” STI-1

In this context, setting up a good change management process at the project level with the support of the organization will help manage users and stakeholders’ expectations in turn reducing the risk of changing requirements.

Requirements that are not captured pose a high risk to the success of the developed software. From the field interviews it was apparent that software testers generally are not part of the requirements gathering phase, especially in medium and large size software development projects. Hence, if particular requirements were not documented properly during the initial requirements gathering phase, it would not be part of the software
development documentation and would not make it to the test plan and would never get tested.

“there are times that a particular function has not been captured in the requirements, so the function is not developed and since we don’t know about it we would never test for it” STI-13

This phenomenon may not be directly linked to software testing itself; but it would be beneficial to have a representative from the software testing team in the initial requirements gathering phase, and have the individual design a test plan from the requirements gathering phase and merge the plan with the software developed test plan to reduce the occurrence of lost requirements. Having a formal process of gathering requirements could also reduce the risk of lost requirements, which will provide more accurate information for the software testers, enhancing the accuracy of the test plan and software testing.

5.6 Need to Use Information

In the prior section, the findings for how software testers seek information were explored. In this section we explore why software testers seek information. The analysis of the data showed that the need to use information is the primary reason for software testers to seek information. The need to use information by software testers fall under the following four categories and is summarized in table 5.2:

- Sociological: Discussion Among Peers
- Cognitive: Incomplete information / Knowledge (Cognitive Complexity)
- Economic: Determine budgetary needs
- Organizational: Focusing on organization / Project context.
Table 5.2 Information Need Categories

<table>
<thead>
<tr>
<th>Category</th>
<th>Information Need</th>
<th>Information Use Elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sociological</td>
<td>Discussion Among Peers</td>
<td>• To be able to have informed discussions</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• To be perceived as knowledgeable information source</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Impress peers and managers</td>
</tr>
<tr>
<td>Cognitive</td>
<td>Incomplete Information</td>
<td>• Reduce Ambiguity</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Increased self confidence through acquired knowledge</td>
</tr>
<tr>
<td>Economical</td>
<td>Determine Budgetary Needs</td>
<td>• Cost estimation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Fund allocation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Hiring</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Staff allocation</td>
</tr>
<tr>
<td>Organizational</td>
<td>Focusing on the needs of organization</td>
<td>• Determining project context</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Evaluate Software Context</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Determine Stakeholders</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Prioritize</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Ensure following organizational rules and procedures</td>
</tr>
</tbody>
</table>

5.6.1 Discussion Among Peers

We discovered from the analyzed data that individuals share and discuss ideas regarding performing different tasks within the project. These discussions and sharing of ideas were viewed by respondents as information sharing. These discussions can be in the form of group meetings, use of checklists to get information from other participants in the project, and using of a shared portal.

5.6.1.1 Group Meetings

A common method used for discussions among peers is group meetings. These meetings are done with stakeholders and members that play a role in the project and contribute to information gathering.
“There are several basic questions that management needs to know and we discuss in our meetings.” STI-19

Another respondent talked about meeting with the stakeholders to get a better understanding of their requirements:

“To gather good requirements we should listen to the actual users. Should be able to go and listen what are the problems, what the users want, what new features they want to add to that, it need to listen to the actual users. Probably, that is the most important part in my opinion.” STI-7

Another respondent states:

“In our requirements gathering meeting we generally do not commit to any dollar amounts as we would need to check with our contracts department for those matters” STI-9

In quotations above the respondents points out to gathering information by way of a group meeting with stakeholders, managers, and other information sources that can provide accurate information.

5.6.1.2 Checklists

In gathering requirements, several software testers interviewed discussed the use of a checklist that has been compiled by them or the organization over the years. Not all the items in the checklist may be applicable to a specific project; however, the perception is that using a checklist reduces the possibility of not recording and documenting all the requirements. As the respondent STI-2 describes use of a checklist in gathering and prioritizing requirements:

“They [users] just know which features need to work; those normally would become part of the testing check list. So we would make sure, based on the checklist, we have the software configured based on the requirements. So basically, the testing check list is a combination of the user requirements in addition to such as installation procedures, uninstall procedures, and load testing.......security items on the checklist were always marked as critical. So, if at any point, security defect was found or any other critical item defect, the testing would be stopped and the developers would be contacted for a fix. “ STI-2

And respondent STI-22 compares the use of checklist in software testing with what he has witnessed in a hospital emergency room:
“you go to the emergency room and during the triage the nurse or medical resident will use a checklist to gather some essential information to rate your condition, make a decision on the severity of your condition and pass on information to the emergency room doctor. We use checklist in a similar way. We have a comprehensive checklist that we have put together by going through several projects and lessons learned. We use this checklist to assist with the requirements gathering and rate the importance of different features and functions identified by the client…. In turn we would spend more time and resources on the critical functions identified.” STI-22

Several industries, including medical, law, and financial, have been using checklists as an information gathering tool. A checklist will allow a person who may not be fully knowledgeable of the arena to gather the information and make that information available to the experts.

Although use of standard documentations is a common practice in traditional model-driven testing school of software testing, no evidence of a common standard in the checklists used were found.

5.6.1.3 Use of Portals

With the advancements in network and web technologies the analyzed data showed that more companies are using online technologies to share and use information. As several of the respondents said:

“we have implemented a SharePoint Portal where we post our documents, status and test reports. This is a central location for our documents.” STI-30

“So we have a bug tracking application which we call ‘XBUG’. This is an online application developed by our company used for bug tracking in our projects..... Everyone on the project has access to the software. Us software testers record the bugs, the developers use this to track what bugs have been reported and corrected…. And the system gives our management an overall perspective of project status.” STI-5

“we use a COTS application called ‘eRoom’. The purpose of the application is to store and share documents. It also enables us assign tasks and keep track of it. The application also has the capability of having meetings be recorded and stored online and for participants that were not able to attend the meeting to go online and view the meeting minutes, action items and so on.” STI-6

“we use MS Project to assign tasks to individual team members and keep track of their progress, and the project progress overall” STI-33
All the respondents mentioned above point out the use of some sort of central portal that they use on their respective projects to share, collaborate, and use information.

5.6.2 Cognitive Complexity

Cognitive complexity could arise from incomplete information or knowledge about a particular task or project. This situation could raise a fear of the possibility of failing to successfully complete the assigned tasks or project.

Since most testing requires the tester to understand what the software is supposed to do, high cognitive complexity of some specialized software such as scientific or financial software presents a problem for testing. Complex applications usually involve a higher number of developers and testers and necessitate the presence of domain experts in development and testing phase of the software hence, increasing the need for close collaboration between the domain experts and developers and testers of the software, and increasing the need for discussion among peers. As an example, respondent STI-20 pointed out that in testing nuclear simulation software, generally the software testers do not have a Ph.D. in nuclear physics and the scientists are not fully aware of the technologies used to develop and test the software. In addition, the results may not be verifiable for a variety of reasons including lack of resources, danger of replicating an atomic reactions, and security & sensitivity of the data. Therefore, complexity of an application increases the cognitive complexity. During the interviews, specialized software such as financial, accounting, and scientific applications were among software that testers identified as having a high cognitive complexity.
5.6.2.1 Testing Financial and Accounting Software

The purpose of financial and accounting software is to provide information regarding the companies’ performance. Financial and accounting software are also used to track debts, credits, and for calculating taxes or tax credits among other uses.

“For U.S. companies financial accounting is performed according to Generally Accepted Accounting Principles (GAAP) guidelines.” STI-21

With that in mind the complexity of the tax code is another well-defined problem within Financial and Accounting Software. As U.S. Representative Rob Portman (R-OH) states:

“The income tax code and its associated regulations contain almost 5.6 million words -- seven times as many words as the Bible. Taxpayers now spend about 5.4 billion hours a year trying to comply with 2,500 pages of tax laws....”

Furthermore adding to the complexity, tax code, rates, and tax credits may change on annual basis, necessitating constant changes/upgrades in financial and accounting software. In addition to the Federal tax laws, each state and municipality has its own subset of rules that may need to be taken into consideration. However, the eight software testers that had experience in this field all believed that the quality of Financial and Accounting software has improved significantly over the past decade. Upon further investigation, it appears that, due to the nature of Financial and Accounting software and their prospective uses and the monetary damages that software failure may cause to an organization; information technology and financial and accounting experts have found a way to work together to significantly reduce software defects.

5.6.2.2 Testing Scientific Software

Based on the field interviews it was discovered that testing scientific software presents unique challenges. First, the sciences in all arenas have progressed rapidly and
have become more specialized now. Second, testing scientific modeling and simulation software (results of the black box testing) would need to be verified by a scientist and generally a software tester will not be able to validate the results independently.

During the field interviews, three software testers had prior or present experience with testing scientific software. One of the software testers had experience with testing nuclear reaction simulation software. Due to the top-secret level of the project we could not discuss any details. However, the tester pointed out that in certain cases of nuclear reaction simulations, the results may even be unknown to the scientists and hence validating the test results presents tremendous challenges. In an example such as nuclear reaction simulation software, the expected results are based on certain theories available in that arena and mainly the results would be compared to the theoretical expectations. Another software tester (STI-18) had worked on software that would forecast weather phenomena such as rain, snow, sunny day, or hurricanes based on certain available data. In this case the actual validation testing involved looking at a forecast of a certain region and comparing it with the actual weather on the forecasted day. Even though there was a point of comparison and validation for the software, the degree of accepted deviation from actual results and percentage of accurate verses false forecasting was an open ended problem that ultimately led to STI-18 moving on to a different position. STI-18 stated:

“Accepted degree of accurate forecasting was unknown to me and I believe it was a point of disagreement all throughout the project while I was there. The input data was fed from several monitoring stations. We had no way of verifying that information as National Weather Service operated those stations. Although we had no reason to believe that we were getting wrong inputs but that was a dilemma that we were faced with. Overall there was no agreement between the developers and the scientists, which led to a confusing and uncomfortable work environment for me, and I ultimately left that job.”
As STI-18 pointed out, there is a great degree of ambiguity with testing and evaluating test results and to what degree of accuracy the software would need to perform. Of course, in any case 100% is the goal; however, with weather forecasting and, for that matter, other sciences still evolving, there exists a gray area of accepted deviations in results.

The third software tester that had worked on scientific software had experience with molecular structure research, where the researcher wanted to design software that would model a material in different states and environments. The biggest challenge in this case is that the chemical and chemical byproducts sciences have progressed rapidly over the past few decades. As interviewee STI-10 stated:

“Some of the data models being translated into software are cutting edge research. Developers may implement their science models in software to demonstrate that they work as part of their own research. “

As STI-10 points out, scientific research software is expected to perform more complex tasks today. As the research in this field may have been focused on materials such as water and how the Hydrogen and Oxygen elements bind to form water; now scientists and their research focus on byproducts such as oil and its derivatives like plastic and how to enhance the material to be able to sustain its form in extreme conditions; and scientists more and more every day are turning to information technology to assist them with their tasks and research.

5.6.3 Focusing on Organization / Project Context

Software testers need to know what the intended uses of the software are and what capabilities the software needs to have to be able to use this information to perform functional testing. As one of the software testers described it:
“First thing that we need to find out is what is the context of software functionality that we are testing….so we want to test to find out if the software does what we want it to do, and if not how much work and effort is needed to fix it…. The result of the testing is passed on to the development team and they would take a look at the bug report and see what is the level of effort needed to fix the bugs” STI-19

Different organizations and customers have priorities with what they need to accomplish. Prioritizing the customer needs and taking them into account during the requirements gathering phase is crucial to ensure clients’ satisfaction. One needs to look at the project context within the needs of the organization. As one of the respondents said:

“It is critical to understand where the business value lies with the data warehouse. When testing we look at data accuracy, consistency, maintainability, and usability. There are more than ten million records currently in the data warehouse and it is estimated that the data will grow by five eight percent annually… there is just so much to test and we will not be able to test everything…. so we talked to the stakeholders to see what type of information – which functions – are more critical to them and we concentrated most of efforts in areas identified to be the costliest if there is an error or a failure.” STI-11

In the quotation above STI-11 described how they have selected to focus on certain area of a data warehouse project to perform validity testing based on information gathered from the stakeholders.

5.6.4 Determine Budgetary Needs

Analyzed data showed that the need to determine how much money should be allocated to each phase of a project is a significant factor in information gathering. As one of the respondents said when talking about estimating project costs:

“What we usually do is when you go through our requirements gathering and trying to figure out their environments, we start mapping that to estimate the number of hours or days that would take us to implement that product for the customer….the bottom line is when the going through scoping effort we come up with a number of days and there is a template for each product which would tell us generally the number of days that you would need on average to complete the work, so there are templates and based on customer needs and internal costs like overhead, and other factors we can map the amount of time it would take to complete a project to a dollar amount” STI-3
As stated in the quotation above, when starting a project information is gathered to be able to provide an accurate cost estimate. This estimate is used to provide a price quote to the customer based on the number of hours needed to complete the task and other fixed costs such as overhead and cost of the software. Another respondent said:

“We had gotten a fixed price contract to deliver [the application] and it was determined that would not be financially feasible to hire a software tester, train the person and bring them up to speed so he can test the software.” STI-16

As STI-16 stated, when dealing with staff and hiring issues, one of the key factors looked at is whether or not it is cost feasible to do so.

5.7 Information Use Elements

The fourth research question and objective was to determine what factors influence the way software testers use information in practice. From the interviews, analysis and coding techniques used based on grounded theory, it was discovered that domain knowledge, individual role context, organizational environment, and meta-information influence Information use by software testers. These elements are explored in this section.

![Information Use Factors](image)

Figure 5.3: Information Use Factors
5.7.1 Domain Knowledge

When analyzing the data, we discovered a close conceptual relation among experience and training in the context of the software being tested, and recognition of experts focused codes. This relationship makes up the Domain Knowledge category.

As one of the respondents stated:

“For software testing domain knowledge is a must; to be able to ask the right questions and to be able to make sense of the results of tests….. only a tester that has the knowledge of the system can go through the logical steps…” STI-24

Another respondent stated:

“The level of the test script that you create depends on the knowledge of the software that you are testing and of course the testing software itself. So in essence there is a correlation between the testers’ knowledge and how testing is done.” STI-7
As STI-24 and STI-7 point out, domain knowledge is required to understand what might be sensible questions to ask, what variables might be appropriate to observe, what variables may be insignificant, and what instrumentation or technique can be used to observe these variables.

Knowing about the structure of the software can help identify special cases, delicate features, and important ranges to try, all which will identify and assist with providing the critical information between what software can do and what it will do during testing.

**5.7.1.1 Experience**

The analyzed data showed that prior exposure or experience with a particular software (whether it was software being tested or software used to perform the testing) or lessons learned from a prior task or project may increase the probability of success of a software testing project. Perception of software testers interviewed was that experienced software testers could increase the probability of finding bugs during testing. As an example, respondent STI-4 talks about the importance of having experienced team members.

“The success of the project is very much dependent on the team and their knowledge as well. In that scenario the manager would not be micromanaging that project and will be allowing the testers to use their own experience, and to follow the test plan, and even in certain scenarios where they suspect there may be bugs go beyond the test plan to completely and thoroughly test an application.” STI-4

And as another respondent points out

“As an example if a software tester doesn't have any prior experience with Microsoft Word, he could pass the spell checker, meaning not testing it at all. Or if it's not in the requirements document you will not be able to create the test script. But if someone is knowledgeable about Microsoft Word and knows what the problems are, even if it's not captured in the requirements documents, he can add it to the test plan.” STI-7

As these respondents point out, having experienced software testers enables them to perform exploratory testing, where the software tester suspects flaws or deviations in
application intended functionality. Experienced team members will increase the team’s domain knowledge.

5.7.1.2 Training / Education

Software testers interviewed perceived that education and training could increase their domain knowledge. There were two important aspects of education and training that emerged from the interviews. Firstly, software testers perceived that training in the technologies used for testing automation and reporting will significantly increase their productivity and will assist them with setting up automated tests. Software testers’ perception was that training for the tools that they use in this ever-changing market will assist them to make better sense of the results and information generated by the automated test. As one of the software testers commented on value added from a training course regarding software they used (LoadRunner) to simulate a load for load testing:

“We use LoadRunner to simulate a load for performance testing….. since I have attended the [LoadRunner] training, I have been able to use some of the features that I did not know about in creating a test…. [and] most importantly I understand the test results so much more. Now I am able to look at the test results and let my manager know where the bottlenecks are, which function, page, ….“ STI-7

Secondly, interviewed software testers noted understanding the software they are testing as a high priority. Several of the software testers mentioned that they usually have a session prior to starting the testing with developers and/or stakeholders, and they go through the different features and how the software operates and what the expected results should be. As STI-17 stated:

“we generally have a session in the test lab with the developers and on occasion with the users to get a better understanding of what the software is expected to do and what features are new or more frequently used…” STI-17
Ensuring that the software testers understand and get an overview in either formal or informal training is recognized as critical by software testers interviewed. Education in a related field also contributes to the team’s domain knowledge.

5.7.1.3 Recognition of Experts

During the data gathering phase we talked to several testers who felt that certain areas of testing were out of their level of expertise and they would turn to domain experts in those areas for assistance. As an example, several testers were not comfortable to execute security testing. The liabilities of having the software penetrated by hackers, and any consequences that may have on the software tester evaluation and future employment were among the major concerns perceived by software testers. In these instances, software testers would want an expert in the field to perform the expertise testing. As one of the respondents said:

“I am not familiar with the security aspects so[we have someone in our security team] that runs the security tests. [He] uses couple of known security scanning software, mainly Nexus and AppDetective that checks for known vulnerabilities or possible security risks within the software” STI-28

As STI-28 states, he is not comfortable with performing security testing and he would go to an expert in the field for security testing although he knows what software the expert uses to test for any security issues within the developed software.

Another respondent said:

“Someone with the security knowledge should do security testing, someone who has more functional knowledge of the software to do functional part of the testing. This will help to compensate for the possibility that the requirements are not fully captured” STI-7

As STI-7 states, testers with appropriate experience should perform the unit testing. STI-7 perceives this approach to reduce the risk of having incomplete requirements.
5.7.2 Individual Role Context

The role of an individual in a software testing project determines a set of operations they have to execute and the way they use information. Therefore the role of an individual and the relevant information used is influenced by the individual’s assigned tasks and designation within a project. These individual roles were identified as demonstrated in Figure 5.6.

![Diagram of Individual Role Context](image)

**Figure 5.5: Individual Role Context Elements**

5.7.2.1 Tester

Based on gathered data, software testers use information to perform planning and developing test cases, setting up the test environment, writing and executing test scripts, and writing bug and performance reports with the ultimate goal of testing the state of the software quality. In planning and developing test cases, software testers use information
to write test plans, set up test data, and prioritize testing. As one of the respondents stated:

“Basically we ask for any previous documents that tells us the system design. What are the system requirements. What is the software expected to do. So we require those documents to be submitted to our repository to see what the design looks like, what are the performance requirements, are there any load requirements? And things like that we capture and we test against those requirements.” STI-7

STI-7 continued and talked about the importance of the test environment and how their testing team tries to setup a test lab as close as possible to the production environment.

“…. We try to simulate the client’s production environment in our test lab.”

Executing the designed tests in the test environment is another function that the software tester performs. In this phase of the testing the tester uses available information that has been gathered from developers, stakeholders, requirements documents and other available sources to design and execute tests. As discussed in section 5.1, the results of the tests provide the baseline for the second category of information used in software testing. As the interview with STI-7 continued, he stated:

“the bug report will include the results of a functional testing, a function doesn’t work or for example the drop down list doesn't show so it would be reported that during a certain action the desired result doesn't come. And that is documented clearly.”

The test results or bug reports are a way that a software tester communicates this information with exact steps required to reproduce the unexpected behavior on a particular test or configuration to the project’s software developers. In addition, project managers review this information to gain an understanding of software quality and project status.
5.7.2.2 Developer

The role of a software developer is to code and implement an application to satisfy user requirements. In this role context the developer will interact with the stakeholders, other team members and software testers. Code documentation is one of the ways that a developer communicates with other team members and makes information available for troubleshooting and future releases. Poor code documentation during development of the software was a common concern when talking to software testers. Some interviewees agreed that the code documentation of some projects was so poor that at times even the developers of the code needed ample time to understand the logical flow of a testing software module, complicating the process of bug reporting and bug fixing. As one of the respondents said.

"we were having a problem the [application] submission form where no error was displayed but the information was not getting saved to the database all the time. Since the error was not happening all the time we needed the developers to go through code and find out what was going on. After couple of research the developers realized that one of the if statements did not need to be in the code and it was there because of code reuse from another application. Since the original code did not have sufficient comments the developers did not realize it during the initial development.... This was a severity one bug and needed to be fixed ASAP and the resources were allocated to find the cause of this condition.... Ultimately the software release was pushed back"  STI-15

It is not uncommon for software testers and developers to work together in trying to pin-point the source of a problem. As stated by STI-15, above developers, by following some best practices such as documenting the code and leaving comments where appropriate, can help eliminate bugs during the development phase, hence reducing the number of bugs in the software and improving the software quality. Although rare, there are occasions where the software developer acts as a tester of the software as well. As STI-16 states when asked about playing the role of developer and tester on a project responded:
“well, it’s not like we had a choice. We had gotten a fixed price contract to deliver [the application] and it was determined that would not be financially feasible to hire a software tester, train the person and bring them up to speed so he can test the software…. Though now that I focus mainly on the testing side of this project I certainly can see the value of having a second set of eyes going through the application” STI-16

Several of the interviewees acknowledged that they had worked on projects where they acted as both the developer and tester. The size of the project was the main contributor. On smaller projects, with limited budget where the project has one or two developers, the developers tested their own code and application. All interviewees agreed to some extent that having a dedicated tester or “second set of eyes” would be a more efficient way of increasing software quality.

5.7.2.3 Technical Writers

Technical writers document an application, its functionality, and produce a system and user’s manual. During the interviews, several software testers pointed out that on many occasions technical writers uncover functional errors, and as part of their responsibilities they report these errors. As one of the software testers stated:

“a lot of time I am so involved with different testing aspects that sometimes I don’t see the obvious errors. We have a great team of technical writers and in every project while preparing the documentation they point out some obvious cosmetic errors” STI-26

Technical writers also at times are the closest thing to users of the software (role playing). As one of the respondents said:

“we do our development and testing offsite, so most of the time we don’t have access to our users…. Our technical writers are the closest thing we have to user testing.” STI-4

As technical writers document an application, they go through each function to create a step by step manual and include screen captures. In doing so, technical writers may uncover abnormal behavior with a module within an application.
5.7.2.4 Stakeholders

Stakeholders play a key role through the software development life cycle. Stakeholders at the organization level could be external as shareholders or internal at the project level. For the purposes here stakeholders refers only to internal entities also known as clients or customers.

The stakeholders provide the initial requirements for the system that in turn would be translated into code and then tested. In majority of the projects stakeholders provide the funding needed for the project, which is essential to jumpstart and complete the project.

“…usually when we engage with the customer in the initial meetings, which mostly would be with their technical point of contact and business point of contact. So those two put together would give us all the information that we're looking for. STI-14

The stakeholders would ultimately approve or disapprove a software and its quality. Hence, identifying the stakeholders early to get accurate requirements and ensuring that they are well informed of the progress can help with satisfactory and timely completion of a project.

5.7.2.5 Project Managers

Project management activities entails planning, organizing, directing, and controlling the allocation of human, material, financial, and information resources in pursuit of the organization’s and project’s goals. So a project manager’s role includes resource allocation and hiring, coordination of activities among different groups involved, and most importantly making management decisions.

As respondent STI-19 when talking about a project stated:

“…..our project manager was asking us [software testing team] ‘Is the software ready to ship?’ I had gotten burned before and although I felt comfortable with our results I did not want to make the same mistake twice”
A project manager acts as the liaison between the project teams and the stakeholders. In this role a project manager would gather information regarding the status of the project, quality of the software, and other related information and will communicate that with the team members and stakeholders to ultimately be an enabler for an informed decision making process.

5.7.3 Organizational Environment

The complexity, rate of change within an organization, organizational goals, contract structure, and resource availability all affect the way software testers use information. Our analysis showed that the project, team, and organizational environments could influence these factors. Figure 5.7 is the graphical presentation of this relationship. In this section how these environments and their components affect information use are explored.

---

Figure 5.6: Organizational Environment Information Use Factors
Organization policies influence software processes by specifying required steps, standards, and documents. When there are clear organizational policies, the organizational projects and teams tend to be in more harmony, and communication and information sharing increases.

5.7.3.1 Negative Organizational Perception Towards Software Testers

Tester’s main duty is to test application software and report any bugs within the software. Pending on the number and severity of the bugs found efforts from development teams are needed to fix these bugs and the release date of the software may be jeopardized. In some organizations these factors have contributed to a negative image of software testers. For instance when talking about work environment, one respondent reported that:

“it is difficult to do our jobs……. If you go home to your wife every day and just be fault finding of the things she is doing wrong, without providing a solution, I am sure that relationship will not last long. It is the same thing with software testers. All we do everyday is to find bugs within the software and create work for the developers, documentation staff and project managers….. nobody likes that” (STI-19)

The quotation above shows the difficulty the tester feels when performing their job duties. Another respondent said:

“in stakeholders meeting, we have software developers, documentation specialist, project managers meeting with stakeholders. Generally no one from our team is invited to the meeting……. I feel that no one wants to us in the meeting as they fear someone pointing out possible design errors may not reflect a uniform or professional atmosphere in front of the client.” (STI-7)

In addition the project environment is a crucial subset of the organization that affects the outcome of a testing project. In the next section this topic is explored.
5.7.3.2 Project Environment

As discussed in section 5.7, the type of software being tested and its complexity and the domain knowledge and expertise of the individuals play a direct role on the path a project takes and its success.

As one of the respondents pointed out:

“...I think the strategy depends from customer to customer and project to projects. I work for organizations that at some point there is dedicated project manager and what we realized is that for public sector customers it's pretty critical to have a project manager from our side that was trying to coordinate activities and just because of the fact that there are a lot of different players in the end implementations are not straightforward implementations so they're involve multiple products multiple vendors and contracts multiple organizations so I believe for the public sector it's it is critical that we have a project management present. For the commercial side, what we have seen is that they have clear vision, dedicated staff, and whoever we deal with as a full knowledge of what to deal with; so we found, unless it's a long project, like over a six-month project, what we really don't need a project manager, we can just have worked with our counterparts on the customer side and we get things accomplished.” STI-3

This respondent clearly distinguishes their application testing approach to whether or not the customer is commercial or government. In talking to several other respondents the effect of the contract structure on the project resources, budget, and schedule emerged. Two distinct contract structures were identified as normal practice: Fixed priced contracts, and Time and Material (T&M) contracts. Each of these contracting vehicles plays a role in information use for the purpose of project planning. Fixed-price types of contracts provide for a firm price, or, in appropriate cases, an adjustable price based on fluctuations in material costs or for longer projects, adjustments based on inflation rates or similar indexes. Generally for these types of contracts the contractor is legally bound to complete the tasks specified in the statement of work (SOW). Any requests outside of the SOW will be deemed as not covered and a separate subcontract and fees will be arranged. In a fixed-price contract, having clear and complete requirements provided by the stakeholders was perceived as crucial; otherwise,
the cost of the project completion may greatly exceed the expected cost. The contract provided a clear scope of the work done.

Time and Material (T&M) contracts are based on an initial assessment of how much effort and material are needed and the customer is invoiced based on money spent in some cases up to a ceiling number specified within the contract. The perception of software testers was that in T&M contracts stakeholders play a significant role as information sources.

5.7.3.3 Team Environment

Team environment is characterized by team structure and culture. Team structure refers to information use factors including cohesion and perceived confidence in other team members and its effects on information use. Software testers interviewed perceived having appropriate resources allocated to appropriate tasks as very important. As discussed earlier in section 5.7.1, having staff that are experienced will enhance the information flow and build confidence within the team.

“Unfortunately the current project testing on my project is very weak. In the current situation the tester is not very experienced. So this portion of the project needs more attention from the project manager, and other team members such as developers, to be able to complete their task. This will take away a big portion of the software development lifecycle efforts and the concentration that developers need to do in developing their code and may in turn cause delays in projects.” STI-4

At the same time the perception that some of the staff don’t have enough experience to complete the given tasks, will distract other team members from focusing on their jobs and may deteriorate team cohesion to the point that information provided from certain team members that are perceived as weak would be to some degree or fully ignored, which increases communication risks.
5.7.3.4 Communication Risks and Practices

Clear communication is key to positive work environment. Increasingly, with companies having staff all across the world, organizations are working to create collaborative work environments that foster open communication between employees and keeping stockholders well informed. Software testing is in a way an information gathering technique, hence, communication between team members is extremely important. In order to increase software quality, software testers must be able to communicate well with the developers and project managers. In addition, the stakeholders now need to know the status of the project to ensure they are achieving the organizational goals. One of the respondents said:

“Our client was satisfied with the information we were providing. This was evident when the client requested us to perform new types of testing in different areas under an extended work order.” STI-14

As STI-14 describes open and effective communications with the stakeholders had created a trust relationship. Effective and clear communications is essential to gather and share information during the software project life cycle.

Team size, cohesiveness, rate of turnover, experience, and resource availability all play a significant role in communication risks and practices.

Team size plays a direct role in communication practices in software testing projects. Based on the field interviews, the larger the team was, the more reliance was laid on formal communications. Figure 5.8 shows this relationship. In addition, physical locations of the team members can play a significant role in selection of a method of communications. As an example one of the respondents said:

“Our company has over 40000 employees and it is not uncommon for people in different time zones in US or even Europe (although seldom) to be assigned to a project, so we have implemented a SharePoint Portal where we post our documents, status and test reports. This is a central location for our documents.” STI-30
The use of Internet technologies, such as web portals, was seen as new methods of communications and information sharing. However, several respondents felt that face-to-face meetings were the method of choice for important meetings with stakeholders.

**5.7.4 Meta-Information**

Meta-information refers to information about the quality of information. During the data gathering and analysis it was discovered that the perceptions of software testers on accuracy, comprehensiveness, clarity and validity of information source are contributing factors in information used in software testing. These factors are looked at in this section.
5.7.4.1 Accuracy

Accuracy is often considered to be an indication of “rightness” or “wrongness”. If there are incorrect requirements that may lead to inaccurate software design and in turn will effect software testing. As STI-25 said:

“it’s like that saying ‘garbage in garbage out’ incorrect requirements will cause for the software to be design and tested accordingly and that could lead to disasters”

As this respondent states, incorrect information in the requirements gathering phase (pre software testing information) will lead to a incorrect testing of the software which ultimately will result in inaccurate post software testing information.

5.7.4.2 Comprehensiveness

Information comprehensiveness reduces the risk of project failure by providing a complete set of requirements. Respondents perceived that without a complete set of information, the test design and execution will not be comprehensive and may lead into software bugs going unnoticed. In addition the importance of identifying the stakeholders
was perceived to be “high” as stakeholders are the requirements information source. As one of the respondents said:

“Identifying all the stakeholders is very important…. call them for a meeting, ask them what they need, what are the requirements, and that is still a problem, since, nobody likes to provide requirements, which could be due to lack of training, lack of tools, people are too busy to spend time to provide accurate and detailed requirements. The moment you ask for requirements, the project at times could be dead. And we see that problem a lot.” STI-7

Generally all respondents said they use some sort of a test plan, which is in line with standard model driven software testing best practices. However, there was no indication that there was a uniform model followed in turning requirements to test plans and the context of the project and project environment affected test planning which is more in line with a context driven testing approach.

5.7.4.3 Clarity

Clarity refers to information that would provide testers’ ability to clearly visualize and understand concepts.

Issues that software testers run into include unclear requirements and lack of details about application workflow. As one of the respondents said:

“…. When I was testing one of the input forms, I noticed that the “Home” button was navigating the user to the previous page, not the main menu. These sort of things generally don’t make it to the design or test document. So I reported it as a bug. One of the developers looked at it and made a sarcastic remark and the bug was marked as a ‘feature’ and that was the end of that…. we rolled the application out and it turned out that the previous page did not have a link to the main menu and using the browsers back button on the previous page would redirect the users to the same input form again and that became a big source of complain where we had to roll out an emergency release….. This situation could have been easily avoided ……” STI-13

As stated by STI-13 there are some minor details such as navigational buttons where the developers make assumptions during coding which can lead to undesirable results.
However, if clear requirements are gathered during the initial or coding phase, mishaps similar to this can easily be avoided.

**5.7.4.4 Source Credibility**

Software testers perceive the need for credibility of the information source as “very high”. As an example when talking to STI-34 in regards to how they differentiate between software bugs and operating system related known issues he pointed out:

“We generally search the error message online. These days you will get a hit for almost anything that you search for. So we look at the source. If the source is something like Microsoft.com we would dig further to see if their workaround resolves our issue but if it is a blog site we would take a note of that in our bug report and we would let the developers go through the steps and decide whether or not the information is accurate or applies to the bug” STI-34

As STI-34 describes there is a significant difference on how they react to information posted on a perceived credible website such as Microsoft.com as oppose to a blog website. This pattern of information use behavior was also noted in dealing with stakeholders:

“Although our customer pays us, at times they don’t appear to be very technical or knowledgeable with the latest technology; in that case we would try to find the customer technical point of contact and have them attend our meetings. Of course this varies from project to project.” STI-3

As STI-3 describes the perceived technical knowledge of the stakeholder plays a key role in how the information they provide is used. The more knowledgeable the stakeholder is perceived more significance their information provided would have.

**5.8 Information Use Behaviors**

The information use elements identified in this study influence information use behaviors. From what we observed, individuals seek and use information based on the role they play within the software testing project. Hence, defining clear roles and
Assignments acts as an enabler and will reduce ambiguity among individuals, where each person within the project would seek and use relevant information to perform their tasks. Within their roles, several key elements as identified in Table 5.2 influence information seeking behaviors. These elements contribute to the level of effort and energy the software tester invests in gathering information. Positive elements identified empower and encourage the software testers to perform their assigned tasks proficiently. These elements provide a picture of Information Use Cycle which can be demonstrated as Figure 5.10.

Figure 5.9: Information Use Cycle

In order to understand the information use behaviors of software testers, it was first necessary to have an understanding of the contexts in which software testers seek and use information.

As discussed in earlier sections (Figure 5.4), the analysis of data for this research showed that Individual role context, Domain Knowledge, and Meta-Information define information use elements.
Different factors within meta-information directly affect information use behaviors. The perceived reliability of information source weighs heavily on information use. One of the respondents said:

“When looking at the requirements document, at times there are things there that look like they made it in because someone had used a template from a previous application and did not make the modifications to that particular section” STI-25

As STI-25 describes here, although there may be a clear requirements document available to assist with information gathering but these documents should not be followed blindly. Further investigation during the interview revealed that the STI-25 was new to the environment and the reliability of the information sources had not been established. In this case the software tester reverted to his comfort zone, which in this case was his experience from previous projects and questioned the validity of the requirements document.

In addition the project and organizational environments and established processes effect how software testers use information. When interviewing STI-3 he said:

“one of the requirements for [this] system is that it has to be FIPS 140-1 compliant. So before starting the project I provided the link FIPS 140-1 to everyone on our team; and asked everyone to go read and get familiar with FIPS 140-1 security standards”

The contractual obligations for delivery of a system affects how software testers seek and use relevant information.

In certain cases (Example discussed in section 5.10.4) selective information use is exercised in creating the test plan to avoid legal liabilities.

Contract structure also influences information use behaviors. In T&M contracts there appears to be significant kickoff and information gathering meetings. In these types of contracts, the contracting party generally goes above and beyond to sell an upgrade or additional functions to the stakeholders, as succeeding in convincing the stakeholders for
additional features generally would mean a longer duration and budget allocation for the contract. In this scenario the contracting firm would have one or two oracles or subject matter experts and the rest of the team plays a supporting role. In fixed-price contracts the tendency to have lengthy requirements gathering meetings is perceived to be less in comparison to T&M contracts and the source of information is the actual written contract.

Individual role context and domain knowledge also play a significant role in information use behaviors. One of the respondents said:

“someone with the security knowledge should do security testing, someone who has more functional knowledge of the software to do functional part of the testing. This will help to compensate for the possibility that the requirements are not fully captured. … naturally someone who is performing the security testing will be more concerned with security aspects of the software…” STI-7

As this respondent points out, the area in which a tester is performing the test will effect the type of information the tester uses. Certainly in a well-managed project, domain expertise is a prerequisite for the tester to be assigned to that role.

5.9 Testing Strategy

Similar to a project process, the attempt to optimize the testing process is influenced by the product under test and the environment that the product will operate in. So in order to strategize testing, one must first understand what the product is set out to do.

Testers STI-11 and STI-12 were working on testing a data warehouse. STI-11 said

“It is critical to understand where the business value lies with the data warehouse. When testing we look at data accuracy, consistency, maintainability, and usability. There are more than ten million records currently in the data warehouse and it is estimated that the data will grow by five eight percent annually… there is just so much to test and we will not be able to test everything….. so we talked to the stakeholders to see what type of information – which functions – are more critical to them and we concentrated most of efforts in areas identified to be the costliest if there is an error or a failure” (STI-11)

As it is stated by the tester interviewed, the test was concentrated on the information obtained from the stakeholders to make the determination on how to allocate resources to
testing the data warehouse. As testing all the records would not have been cost effective, in this scenario the cost of failure – or data inaccuracy – was deemed to be various and the costliest modules were tested more thoroughly.

In addition, with advancements in testing simulation software, more companies are moving towards automated testing. Perceived benefits of automated testing including repeatability and its ability to simulate user load (stress testing), were identified. As one of the respondents said:

“Once we have the product ready for testing we create a test script and we let it run to identify any bugs within the software…. The same script is used to run any follow up tests on the software, though there are times that we may need to make minor changes to it.” STI-16

As this respondent stated, test automation has reduced the efforts during, retesting phase of the software. The testing team reviews results of the testing (Post software testing information) for verification of test results. As STI-16 continued:

“We look at the results for any failures or latencies in a module and so on” STI-16

The human element of testing still plays a large role in writing the test scripts for automation and also reviewing results for verification and decision making purposes.

5.10 Decision Making

In interviews with software testers that played a managerial role (team lead, test lead), in which 22 of 34 respondents at some point acknowledged that they have done so, the point was brought up that organizations use the information from the results of the software testing project in conjunction with other information available to make business decisions.

Testing provides the information to make an informed decision. Respondents perceived that there are several questions which need to be answered in a software project
at the organizational level such as: does the software do what we want it to do and if not how much work would be involved in fixing it? In reality, testing by itself cannot answer these questions; but it can supply information that combined with information about the software development process, availability of organizational resources, and the goals of an organization helps identify the answers and reducing the risk in, as an example, estimating what it will take to fix the bug in the software. So in essence, “testing” is an information gathering activity, that helps making an informed decision, and communication through informational elements is extremely important.

The findings showed that there are a variety of contextual factors that effect decision making, such as organizational environment, individual role context, domain knowledge, and the importance of meta-information. These findings can be summarized as:

5.10.1 Risk Management

Testing can provide information that reduces risk. Different individuals and the role they play in the project, different organizational environments, projects and software context, and even different time periods may mean different perceptions of risks thus testing can be designed to provide information to help answer different questions. As STI-19 describes when asked about what information they seek for when testing software.

Figure 5.10: Information Use Processes
“there are several basic questions that management needs to know and we discuss in our meeting. As an example we need to know if the software we are testing does what we want it to do and if it does not what is the level of effort needed to fix it. Of course, the level of effort cannot be an answer that the result of tests answers directly; however, using and categorizing the bugs would help the developers to better estimate the level of effort, …..”

Perceived challenges by software testers in order to make decisions are explored in the next sections.

5.10.2 Challenges

In today’s competitive market there is a continued pressure on software organizations to achieve time-to market timelines in a cost effective manner. Hence, the ability to provide cost estimates at any point during the course of the project is extremely important and challenging. Based on the data gathered, there were three practices within software organizations to deal with this challenge.

5.10.2.1 Method 1: Cost Multiplier

In 8 out of 15 projects surveyed (53%) the cost multiplier method was used. In this method a dollar amount was associated to each bug severity level. At the end of each round of testing the bugs found were multiplied by the associated severity level cost and the sum, and the cost of the retest was used as an estimated cost factor. Variations of the same concept were also seen. Where, for example, for high severity bug category a developer was assigned to the bug and the cost of developer’s labor rate was also factored in the cost estimation.

5.10.2.2 Method 2: Use of Experts

Based on the field data, 27% (4 out of 15) projects were using senior members of the development, testing and project management teams to review the bug reports and
estimate the time and cost associated with rewriting and in essence fixing the bugs. This study did not find any logical flow to this process hence increasing the risk of incorrect assessment. Generally, the sharing of post software testing information was done in a group meeting between testing and development teams with the project manager working as a facilitator.

**5.10.2.3 Method 3: Unorganized Method**

No cost estimation calculation method was used in three projects (20%). In these projects the approach was to start working on the bugs found when a developer is available to do so. The three projects had the following factors in common: (a) they were small projects (less than ten people working on the project) and (b) the contract structure was T&M.

**5.10.3 Defect Tracking and Resolution Life Cycle**

Implementation of a defect tracking and resolution life cycle is essential to capture information gathered during software testing and ensures information flow between software testers and developers of the software. In addition, it will provide an overview to the managers of the status of the software at a given point. During the data gathering phase, several of the respondents talked about the issue of bug reporting. As different people attach different importance to the same information as a result of testing, hence making it difficult to assign severity or significance to a bug. Significance can be described as the importance attached to the bug by the person who gets to decide what to do about the bug and it depends on context.
5.10.3.1 Bug Severity Reporting

Twenty-eight out of thirty-four (over 82%) software testers interviewed had a formal bug reporting process in place. They either used an in-house developed bug reporting tool or had invested in purchasing third party software. The common challenge regardless of the bug reporting tools and cost estimation methods used was identified as bug severity level reporting. Many organizations report their success and failures of their project and assign resources based on how the number of bugs in addition to the severity level assigned to it.

There were three hurdles identified as contributing factors to this challenge. Firstly, resolution of some of the bugs may be unknown to the tester reporting the bug. Secondly, the software tester did not have sufficient knowledge of the application and lastly, lack of well-communicated standards to software testers, which had lead to software testers using their own judgment in bug reporting categorizations. Six projects (40%) had tackled this challenge by implementing a bug reporting software with a Guided User Interface (GUI) that would steer the tester through a series of questions and would ultimately assign a severity level to the bug found. The use of GUI had significantly increased the time it took to report bugs and perception of software testers was that it is taking valuable time away from the actual testing that they are tasked to do. Three projects have setup guidelines for bug severity assignment. As one of the respondents said:

“...Security items on the checklist were always marked as critical. So, if at any point, security defect was found or any other critical item defect, the testing would be stopped and the developers would be contacted for a fix. “ STI-2

The above quotation is an example of guidelines provided where all security related bugs found would be marked as critical.
5.10.3.2 Hidden bugs

Another major challenge with bug reporting was identified as ‘hidden bugs’. These are bugs that cannot be reproduced. One of the interviews when asked said:

“at times when we run a test, we get an error that does not happen all the time, we report the error but the developers are not be able to reproduce it. This is really an annoyance since the developers will come back and want us to reproduce the bug on consistent basis; well this is one of the things that can lead to escalation of problems with the development and testing team” STI-8

when asked how they deal with these sort of errors the tester continued:

“Now when possible we use Snagit to record our testing sessions. So in case a similar situation arises developers can look at the Snagit screen captures. . . .”

5.10.3.3 Information Loss

In software testing projects, one of the biggest challenges is to track information from the beginning to the end of the project. During the requirements gathering phase, it is common not to have representation from the software testing team. In this case, if a system requirement such as number of users is incorrectly captured or missed, the software tester has no way of knowing this information and the information loss may result in an inaccurate load test. This symptom is more commonly seen in large and complex projects.

To avoid losing information during a long project, companies have turned to SharePoint or similar portals to manage the information sharing and flow of information. Information loss introduces a high level of risk to the project’s success.

In smaller companies, the risk of losing information in forms of requirements is less. However, in smaller projects communication between teams is more informal. One of the software testers working on a small project said:
“At times when I find a bug, I walk over the developer and let him know” STI-29

when asked if this has ever caused a problem STI-29 replied,

“Yeah! there has been seldom times that the developer has forgotten about our conversation and in the next round of testing I see the same bug’

Effective communications reduces the risk of information loss.

5.10.4 Contextual Ethical Issues

In a few of the interviews, the testers pointed out the issue of purposefully not testing certain functions of an application. This phenomenon was seen in fixed-price contracts.

As one of the testers pointed out:

“We have a fixed-priced contract….. We are told by management to only use the requirements document that has been signed off by the client for our functional testing…. This is what we have been contracted to do. So we generally don’t do any exploratory testing” STI-30

Another tester pointed out:

“When we were testing the [Software] which is a client/server application I noticed that there are no latency tests in the test plan for our remote users in San Francisco. I have been in [this organization] for a few years and I knew we had users in San Francisco office. When I brought this up to my manager; he said don’t worry about it! Just follow the test plan and soon after I was moved to another project” STI-27

STI-27 continued when I asked what he thought may have been the reason

“It’s like a legal liability. If we test and the software does not perform well we have to report it to the client. But since the remote users were not identified in the official contract documents, it is so much simpler not to test at all…. And if the software fails which in this case it did; my company looks at it as another opportunity to extend the development and in turn testing contract”

In these projects it was known that the software may need to function under certain specifications but since the specifications were not part of the contract and they were not documented in the requirements document; no resources were allocated to develop or test these functions. This behavior had two reasons, first, to avoid legal liabilities. Second, to
necessitate a contract extension for future enhancements and bug fixes as part of the organization’s business strategy.
Chapter 6 - Conclusions

We started this research with the goal of developing a framework of the use of information in software testing. In chapter five, by applying grounded theory open, selective, and axial coding to the gathered data, we defined the emergent categories. In this chapter we finalize our finding by putting the pieces together and show the relationship among emerging categories.

The Grounded Theory of information use in software testing consists of eight main categories. These categories are:

- Information seeking elements
- Information use behaviors
- Domain knowledge
- Individual role context
- Organizational environment
- Meta-information
- Testing strategy
- Decision making

Information use behaviors is the core category. Figure 6.1 is a graphical representation of information use categories that emerged from the gathered and analyzed date and their relationships.

The straight line and arrows shows the direction of information flow and influence. The dotted line represents the possibility of information flow in certain cases.
Software testers perceive information based on the role they play (*Individual role context*)

Positive (Motivations) and negative elements (Fear factor), availability, credibility of the information source, and the efforts involved in finding and getting a response from information source make up the *information seeking elements*.

Software testers' prior experience, education, and relevant training shape the *Domain Knowledge* category.

The perceived level of accuracy, comprehensiveness, source credibility, and clarity of the information make up the *Meta-information* category.

Organizational and project rules, procedures, and goals, team structure, perception of other team members, and team cohesion, shape the *Organizational Environment*.
Information seeking and use behaviors are influenced by information seeking elements, meta-information, individual role context, and organizational environment, and information obtained from the results of software testing (Information use behaviors).

*Information seeking* and *use* are to plan a testing strategy to provide information about the quality of software that is used to improve quality and make informed decisions (decision making).

*Information use behaviors* in software testing to make decisions could several possible outcomes: Stop the software development project, software quality is at a level that is ready for implementation or further development and testing is needed to improve software quality. The cycle of information use by software testers is repeated if further development and testing of the software is needed.
6.1 Effects of Project Size and Duration

Based on data gathered it was discovered that there exists a relationship between project size and duration with information conceptualization, seeking and use. As respondent STI-33 states:

“The larger the project duration, the more formal our approach gets. As an example for a project that runs for over six month we use PM tools to capture project activities and assign resources to these activities….. for a small upgrade testing project, I would assign a team member to run the tests from beginning to end and report the results to me directly.” STI-33

As STI-33 identifies a correlation between project size and methods used to interpret and filter and in essence conceptualize and use information.

Similarly STI-4 states on how frequent group meeting are organized for larger projects:

“….naturally the bigger the size of a project the more complex it will get, from managing staff to budget, schedules and so on. Generally when we have an implementation across physical locations the number of obstacles that we need tackle increases ….. in such scenarios we meet frequently and in these [group] meetings we try to come up with a list of issues that we have or may encounter during the roll out and discuss them to see how we can incorporate the new findings in our project schedule and resource allocations” STI-4

Also, as STI-3 states there exists a difference in the company’s approach between public and commercial sectors, in addition to the size and duration of a project.

“…I think the strategy depends from customer to customer and project to projects. I work for organizations that at some point there is dedicated project manager and what we realized is that for public sector customers it's pretty critical to have a project manager from our side that was trying to coordinate activities and just because of the fact that there are a lot of different players in the end implementations are not straightforward implementations so they're involve multiple products multiple vendors and contracts multiple organizations so I believe for the public sector it's it is critical that we have a project management present. For the commercial side, what we have seen is that they have clear vision, dedicated staff, and whoever we deal with as a full knowledge of what to deal with; so we found, unless it's a long project, like over a six-month project, what we really don't need a project manager, we can just have worked with our counterparts on the customer side and we get things accomplished.” STI-3

So it is the perception of STI-3 that there is a difference between public and commercial sectors unless the duration of the project is over six months.
During the data gathering and analysis phase, we discovered that same people behave differently pending on the scope and size of a project. In small projects the domain knowledge of the testers plays a significant role; and as the size and duration of the project increases more of organizational knowledge and best practices and lessons learned influence software testing.

In conclusion, a correlation between project size and duration and information seeking, conceptualization, and use was seen. It was discovered that there exists a relationship between information conceptualization and project size. The larger the project size and duration the more formal techniques are used. To explore this correlation we are going to break it down into three different project size categories. Small, Medium and Large. Table 6.1 shows this relationship.

<table>
<thead>
<tr>
<th>Project Size</th>
<th>Information Seeking</th>
<th>Information Conceptualization</th>
<th>Information Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;10</td>
<td>Checklist, Informal Communications, Individual Role Context</td>
<td>Known Information Source(s), Individual Role Context, Individual Knowledge, No Formal Version Control</td>
<td>Context-Driven, Personal Experience, Skills, Individual team culture, Ad-hoc</td>
</tr>
<tr>
<td>10-100</td>
<td>Checklist, Team Meetings</td>
<td>Team Context, Team culture, Version Control</td>
<td>Team Environment, Personal Experience, Ad-hoc, Process, Team culture</td>
</tr>
<tr>
<td>&gt;100</td>
<td>Checklist, Frequent group meetings, Large Information Sources, Time to locate Info Sources is long</td>
<td>High complexity, Organizational Environment, Formal Version Control</td>
<td>SharePoint Portals, Formal Communications, Emphasis on Repeatability, Task and process Driven, Organizational Environment</td>
</tr>
</tbody>
</table>

6.2 Context Driven vs. Standard Model Driven

This research started with looking at different schools of thought in software testing, specifically the standard model driven and context driven school of thoughts in software
testing (Chapter II). These two ideologies in software testing fall on two opposite ends of the spectrum. The standard model driven approach introduces a set of standards that can be applied to all software testing projects. On the other hand, the context driven approach is based on the context of the project and although they agree that there may exist some best practices (which is also context based) but in context driven software testing, there are no pre-set definitions for any testing project. The research goal was to explore how software testers use information. During the course of this research and 34 interviews of software testers, elements of both software testing schools of thought were seen but there was no indication that software testing in the real world today is either context driven or follows a set of standards. This research concludes that the state of software testing is widely ad hoc with no set of common standards across projects and industries.

6.3 Other Conclusions

Software is more widely used every day to perform different functions to support a variety of tasks and sciences. Several fields such as medicine, law, and finance, have realized the need to have specialties and concentrate on different parts. As an example, a criminal trial lawyer will know the criminal code and a family lawyer would know the rules of civil unions, divorces, etc… and will not be familiar with criminal codes. Software testing would benefit tremendously if it is divided into sub-specialties. The way that scientific software is tested would differ tremendously from the way that financial software would be tested.

Higher Education system currently does not have a dedicated major for software testing. As important as Software and its quality has become in everyday life the
Software industry would benefit significantly to have a dedicated major in academia for software testing.

Software testing methods within the industry vary substantially. There is not currently an industry wide acceptable terminology for software testing functions. At times even within the same company different terminology is used making it difficult for software testers to communicate across projects and transfer the information to new developers and stakeholders. Terminology is inconsistent throughout different software industries. Test reports include various measurements and have different content and format. It would be beneficial to the software industry to define and implement an industry-wide terminology for software testing.

The type of contract will affect the resource allocation and affects the approach taken by the contracting firm to do the job. Public and private sectors would benefit significantly by educating their contracting officers of variety of contract types.

**6.4 Future Work**

It was this research finding that there are two distinct information types that software testers use to perform software testing: Pre software testing information, and Post software testing information. Future research on concentrating on each individual information type may lead to more detailed findings and information use behaviors in each category.

Future study on factors and their degree of influence in information use in software testing and compare and contrast of these factors in government and private sectors would benefit the body of knowledge significantly.
One of the research findings was that the structure of the software development and testing contract has a direct influence on information use in software testing. Research in this field would benefit the body of knowledge by enabling managers on the stakeholder side on how to educate themselves with the contract structure and selection of a contract type that may benefit them the most.

A study of the level of software tester’s education in related software fields and its relation to information use in software testing may benefit managers in selection and hiring qualified candidates. In addition, it would help the academic body of knowledge in structuring a curriculum for software testing.

It appears that due to the nature of the financial and accounting field and the monetary damages that software failure can cause in this field; information technology experts and financial and accounting experts have found a way to work together to significantly reduce software defects. Future research in the cohesion between the various aspects of developing and testing financial and accounting software may benefit the software testing field.

A quantitative research in factors identified in this research to determine their degree of influence may be beneficial to the software testing body of knowledge.
References


Graham, Dorothy (2002). What part of the system do you test first? *The Proactive Testing Model helps you prioritize, manage a project-level plan and steer clear of risks*, www.doj.com ; Dr. Dobbs Portal.


Mahoney, M. S. (1990), Interview with M.S. Mahoney, 3 December 1990, Princeton University.


Appendix A: Interview Consent Form

Interview Consent Form

I hereby authorize Payman Kaveh to use transcription of my audio recording for his Doctoral Dissertation Research at George Washington University entitled, *A Framework Of The Use Of Information In Software Testing*, in his presentation. I understand that by signing this form, I am releasing all recordings and content of my interview to him for this expressed purpose. I will not receive any compensation for this now or at any time in the future. I further certify that I am over the age of 18 years. I understand that my identity will remain anonymous in this research.

Name (Print): ____________________________

Signature: ______________________________

Date: ________________________________

Please initial: _______

Audio interview only: _________

Investigator’s initials: ___________
Appendix B: Interview Questions

Interview Questions – Version 1

A. Background info

A.1) What is your title and role in your organization?
A.2) How long have you been in your current position?
A.3) To what degree are you involved with your organizations software quality and software testing projects?

1a) When starting a project, what are the initial steps that you take to ensure project's success?
2a) What sources and methods you use to obtain relevant information?
3a) Any issues or challenges that you can tell me about?

2a) Do you have a formal testing process? (Do you follow a known Software Testing methodology? If yes, what methodology? And does it vary from project to project?)
2b) Who would make the final decision? And how are decisions made? (group collaboration, manager making the decision, etc.)
2c/3a) How is the information gathered applied to the project?

3) How often do you interact with your client and do you believe your interaction has any effect on how and in a way you perform your software testing?

4a) Please explain a project that you have worked on that was a success?
4b) Please explain a project that you have worked on that was not a success?
What factors do you believe contributed to the failures.
4c) What are some of the things (information) that you like to have before you kick start a Software testing project?
4d) Are there any factors that you can think of that may change the direction or scope of the project? If yes, explain

5a) What are some of the things (information) that you absolutely feel you need to have before you kick start a Software testing project?
5b) In your opinion what are the criteria's that determines the project success or failure?
5c) how do you measure the status of the Software Testing project with these criteria's at any given time?

127
b. How do you determine if the project is not on course, and how do you make corrections?

· In a perfect world how would you imagine a perfect software testing project would go? And why?

*Interview Questions – Version 2*

Tell me about what type of information you seek to start a Software Testing project
Tell me about the sources you use to obtain relevant information
Tell me about the methods you use to obtain relevant information
Tell me about any issues or challenges you might face during information gathering phase.
Tell me about how you would interact other groups to share info (i.e. Interaction with Development team)
Tell me about the role of a project manager
Tell me about how you feel the information about technology used would be relevant to perform your tasks
Tell me how you apply the information gathered to the software testing project and how does it effect testing.
Tell me in your view how the final decision to stop testing is made

4a) Please explain a project that you have worked on that was a success?
4b) Please explain a project that you have worked on that was not a success?
What factors do you believe contributed to the failures.
4c) What are some of the things (information) that you like to have before you kick start a Software testing project?
4d) Are there any factors that you can think of that may change the direction or scope of the project? If yes, explain

5a) What are some of the things (information) that you absolutely feel you need to have before you kick start a Software testing project?
5b) In your opinion what are the criteria's that determines the project success or failure?
5c) how do you measure the status of the Software Testing project with these criteria's at any given time?

(a. How do you determine that the project is going the right way?
b. How do you determine if the project is not on course, and how do you make corrections?)
In a perfect world how would you imagine a perfect software testing project would go? And why?

*Interview Questions Version 3*

Tell me about what type of information you seek to start a Software Testing project
Tell me about the sources you use to obtain relevant information
Tell me about the methods you use to obtain relevant information
Tell me about any issues or challenges you might face during information gathering phase.
What type of software have you tested?
Tell me about the challenges of testing <type of software>
Tell me about how you would interact other groups to share info (i.e. Interaction with Development team)
Tell how you would gather information for a large scale project.
Tell me if the methods for information gathering for a large scale project varies if the size of the project is smaller.
Tell me about the role of a project manager
Tell me about any contributing factors in Software testing.
Tell me about any project / organizational factors that effect testing.
Tell me about how you feel the information about technology used would be relevant to perform your tasks
Tell me how you apply the information gathered to the software testing project and how does it effect testing.
Tell me what type of process do you use to collect information
Tell me about what type of information do you hope to obtain as a result of Software Testing
Tell me what type of process do you use to interpret gathered information
Tell me in your view how the final decision to stop testing is made
Tell me about the steps you take to ensure software testing success
Tell me if you or your organization follows a set of best practices.
Tell me if you have ever encountered any ethical concerns while planning or conducting software testing.
Tell me about a software testing project / experience that was a success
Tell me about a software testing project / experience that was a failure
Appendix C: IRB Review Request Form

THE GEORGE WASHINGTON UNIVERSITY & MEDICAL CENTER
OFFICE OF HUMAN RESEARCH
INSTITUTIONAL REVIEW BOARD

EXEMPT FROM IRB REVIEW REQUEST FORM

SECTION I: INVESTIGATOR AND TEAM CONTACT INFORMATION

<table>
<thead>
<tr>
<th>IRB#</th>
<th>Version Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>2021</td>
<td>01/22/2019</td>
</tr>
</tbody>
</table>

TYPE OF HIPAA AUTHORIZATION REQUESTED: Not applicable

PROTOCOL TITLE AND SPONSOR:
TITLE: A Framework of the Use of Information in Software Testing
SPONSOR:

PRINCIPAL INVESTIGATOR INFORMATION (MUST BE FACULTY OR STAFF):

<table>
<thead>
<tr>
<th>LAST NAME</th>
<th>FIRST NAME</th>
<th>DEGREE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baba</td>
<td>Abraham</td>
<td>PhD</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DEPARTMENT</th>
<th>SCHOOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENGINEERING</td>
<td>Engineering &amp; Applied Sciences</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CAMPUS ADDRESS</th>
<th>PHONE</th>
<th>EMAIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1716 G Street, NW, Suite 120</td>
<td>(202) 994-1249</td>
<td><a href="mailto:abdulb@gwu.edu">abdulb@gwu.edu</a></td>
</tr>
</tbody>
</table>

PRINCIPAL CONTACT IF OTHER THAN PI (THIS MAY BE THE STUDENT/STAFF):

<table>
<thead>
<tr>
<th>LAST NAME</th>
<th>FIRST NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kasid</td>
<td>Payman</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CAMPUS ADDRESS</th>
<th>PHONE</th>
<th>EMAIL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Recommendations:
☑️ Study Registered as Exempt. Category: __________
☑️ This research does NOT meet the regulatory/institutional requirements for exemption from IRB review. To conduct this research you must complete an IRB submissions package for IRB review. For more information on completing a research submission, contact OHR at 202-994-2518.
☒ The activity is NOT human subject research and does not require exempt registration or IRB approval.

L. B. J. N. S. O. S. K. 1/1/18
IRB Chair/Designee
Signature Date

OHR OFFICE USE ONLY: OHR Track #: __________

Irrelevant

This Exempt Registration does not expire nor does it require renewal.

Reporting Proposed Changes in Research:
This exemption from IRB review determination only applies to the protocol, as currently proposed. Therefore, if there are any proposed changes to this exempted study, e.g., protocol, data gathering instruments, type of information being accessed or disclosed, etc., all changes must be reviewed by the OHR IRB. If changes require further review, such a review will be conducted to determine whether the proposed changes result in the study requiring IRB review and approval, or new exemption determination.
Appendix D: I-35W Mississippi River Bridge Collapse

The I-35W Mississippi River bridge was an eight-lane, 1,907-foot (581-m) steel truss bridge that carried Interstate 35W across the Mississippi River in Minneapolis, Minnesota, United States. The bridge, maintained by the Minnesota Department of Transportation (MnDOT), was Minnesota’s third busiest, carrying 140,000 vehicles daily (DMTV-1).

At 6:05 P.M., during the Wednesday evening rush hour on August 1, 2007, the main spans of the bridge collapsed, falling into the river and onto its banks. Thirteen people died as a result of the collapse (Wright, 2007). The bridge collapse garnered international attention, including visits from President Bush, Minnesota’s two U.S. senators, and swarms of media.

State engineers recommended in 2000 that the Interstate 35W Bridge be replaced or redecked, the *Minneapolis Star Tribune* reported on August 8, 2007. That recommendation followed reports beginning four years earlier that raised concerns about the bridge, *Star Tribune* reporter Pat Doyle said on CNN’s *American Morning*.

Engineers were so concerned, “They ended some reports with exclamation points,” Doyle said. The bridge was undergoing minor resurfacing when it collapsed and was not scheduled to be replaced until 2015 at the earliest (CNN-1, 2007).

Doyle said that in 2005 a consultant hired by MnDOT recommended that steel plates be used to reinforce the bridge superstructure. “MnDOT opted for what they refer to as a more efficient or most cost-efficient alternative to that which is essentially inspecting the bridge,” Doyle said.
However, two reports published since 2001 determined the bridge was safe despite structural problems and deficiencies (CNN-2- 2007).

“The bridge’s deck truss system has not experienced fatigue cracking, but it has many poor fatigue details on the main truss and the floor truss system,” said a report conducted for MnDOT in 2001 (CNN-2- 2007).
Appendix E: IEEE 829 Standard Document Types

“There are eight document types in the IEEE 829 standard which can be used in three distinct phases of software testing:

1. Preparation of Tests
   - Test Plan: Plan how the testing will proceed.
   - Test Design Specification: Decide what needs to be tested.
   - Test Case Specification: Create the tests to be run.
   - Test Procedure: Describe how the tests are run.
   - Test Item Transmittal Report: Specify the items released for testing.

2. Running the Tests
   - Test Log: Record the details of tests in time order.
   - Test Incident Report: Record details of events that need to be investigated.

3. Completion of Testing
   - Test Summary Report: Summarize and evaluate tests.

The preparation for testing is the most important activity in any software testing project... The purpose of this stage is to prepare an effective and efficient set of tests and create the environment for them to run in.”¹ (Coley, 2007)

Figure E.1 shows the relationships of these documents to one another as they are developed and to the testing process they document.

¹ Coley (2007). IEEE 829 Documentation presented by Coley Consulting
http://www.coleyconsulting.co.uk/IEEE829.htm
Figure E.1: Relationship of Test Documents to Testing Process (Source: IEEE)
Appendix F: Principles of Context-Driven Software Testing

“1. The value of any practice depends on its context.
2. There are good practices in context, but there are no best practices.
3. People, working together, are the most important part of any project’s context.
4. Projects unfold over time in ways that are often not predictable.
5. The product is a solution. If the problem isn’t solved, the product doesn’t work.
6. Good software testing is a challenging intellectual process.
7. Only through judgment and skill, exercised cooperatively throughout the entire project, are we able to do the right things at the right times to effectively test our products” (CDT, 2007).