The Proceedings of the Seventh International Research Workshop on Advances and Innovations in Software Testing

The University of Memphis

Systems Testing Excellence Program
FedEx Institute of Technology

The workshop is organized by the:

Systems Testing Excellence Program - STEP (step.memphis.edu) at the University of Memphis. STEP is an interdisciplinary program under the auspices of the Departments of Computer Science, Management Information Systems, Computer Engineering, and FedEx Institute of Technology.

step.memphis.edu
October 28-29, 2013
Memphis, Tennessee
Determining the Unit Cost of Test Cases

Mark L. Gillenson, Ph.D., CCP
Fogelman College of Business and Economics
University of Memphis
Mark.Gillenson@memphis.edu

Xihui “Paul” Zhang, Ph.D.
College of Business
University of North Alabama
xzhang6@una.edu

Abstract – As important as testing is to a software development project, it comes at a significant cost. One aspect of the cost is the number of test cases created to test the software and the number of times each test case is executed. This study is the beginning of a research project to gauge the unit cost of functional test cases.

Keywords: Test cases, User Acceptance Testing, White box, Black box.

I. Introduction

Software testing is a critical aspect of software development. With the vagaries of requirements, the complexity of designing and writing code, and the translation issues in moving from one stage of software development to the next, testing software to make sure that it accurately does what it was intended to do is a necessary part of the process. The artifacts produced in each development stage must be tested. In the earlier stages of development, such as the requirements and systems analysis stages, testing generally means conducting reviews and inspections in which the authors of the artifacts present them for comment by a knowledgeable team. While the code itself can (and often should) be reviewed in this manner, historically most software testing effort has gone into executing the code with well-planned test cases.

All types of testing represent a significant cost to the software development organization. With the pressures that these organizations face to produce, maintain, and modify large numbers of applications, there is a natural tendency to invest their resources in code generation rather than in code testing. However, since the importance of testing cannot and should not be underestimated, the next step is to be able to manage it effectively. One aspect managing testing is being able to gauge its cost. This is not an easy task as there are several cost components to testing. These include the costs of personnel, equipment, and the time delay of moving the application into production. Of course, we understand that these are necessary costs as the later cost of
putting an application with defects into production can be much higher.

Code can be tested in order to check whether several aspects of its execution meet expectations. Functionality testing is designed to validate that the code accurately performs the functions for which it was intended. Performance testing determines whether the code's response time meets its goal given a specified level of throughput. Security or vulnerability testing checks whether the code can withstand attacks from hackers. Usability testing is a matter of the user-friendliness of the code. Globalization testing is a matter of verifying that the code will work well in other countries and with other languages. All of these types of testing require the development of test cases consisting of sets of input variable values with associated expected output values.

This paper focuses on one aspect of the cost of testing: the unit cost associated with a test case used in functionality testing. There are compelling reasons for an interest in this. The number of test cases used to test a piece of code can vary widely and is a function the manner in which the test cases were developed, which in turn is often a function of the risk associated with the code when it is put into production. Test cases for functionality testing can be created in several ways with varying costs. They can be created manually as part of the requirements writing process. They can be created by any of several "black box" heuristic techniques. They can be created by "white box" methodologies. They can also be introduced from actual production inputs of an existing system that is being modified or completely rewritten. And they can be created in an ad hoc, manual way by end users during user acceptance testing (UAT). As to the risk of the code being used as a guide to the creation of test cases, it stands to reason that the higher the perceived risk of the code, the larger the number of test cases that will be desired and the larger the number of different methods used for developing the test cases.

II. Test Case Cost Components

A test case can be designed to test a small module of code, an entire application, or any piece of code in between. What is necessary and is in common among these is a well-defined set of input variables to the code and, for a given set of input variable values, an expected output based on the applicable business rules governing the application or part of the application being implemented by that code. Thus, a test case consists of a set of input variable values expected by the piece of code being tested and an expected output value based on how the code is supposed to function.

When considering the cost of a test case, one must take into account the cost of creating it, the cost of running it, the cost of determining its success or failure, and, possibly the cost of using it to fix a defect in the code that it discovered. Thus, as an initial way of evaluating the cost of a test case, we have the following:

The unit cost of functional testing is based on the following factors:
• Cost to create the test case input values (CIV)
• Cost to determine the expected output of the test case (CEO)
• Cost to run the test case (CRT)
• Cost to record the test results (CRR)
• Cost to evaluate the test results (CER)
• Cost to fix a defect if found (CFD)

Unit Cost = CIV + CEO + \sum_{i} (CRT + CRR + CER + CFD)

where \( n \) is the number of times the test case is run until the defect is fixed.

This function recognizes that the cost to create the test case and to determine its expected output is a one-time cost, while the cost to run it and work with its results is repetitive with each time it is executed. Each of the factors in this function can be further analyzed and ultimately defined in more detail. The following is a high-level view of the directions in which the further analysis can go.

The cost to create the test case input values (CIV) varies with the methodology used. Clearly, if more than one method of input case creation is used then the costs become additive. The cost to create a test case from a use case at the requirements stage is directly a function of the personnel cost based on time. Technically, a portion of the cost of creating the use case itself could be factored in, but this should not be the entire cost of the use case as use cases have other functions, as well. Each test case created from a use case can be priced individually. The cost of test cases created from black box heuristic techniques may be priced individually or as a group, depending on the technique being used. For code requiring a small number of input values, creating test cases from an equivalence class and boundary value analysis or a domain analysis is a matter of personnel time with the test cases basically being created on an individual basis. This is as opposed to creating test cases using the pairwise analysis technique, in which the cost is a function of the time it takes to prepare the pairwise analysis software to run for the particular problem at hand plus the time to analyze the results of the output of the pairwise analysis algorithm. While the cost to actually run the algorithm is negligible, the cost to set up the run and to analyze the results may not be. For example, in the case of a large number of input variables, it may take some time to analyze the situation and decide to freeze the value of some of the variables and run the pairwise algorithm on only a subset of them. Further, it may take several runs of the algorithm, comparing the results of each, to decide which to use. Another instance of time needed in using pairwise analysis is studying the output of the algorithm to eliminate an excessive number of negative test cases. The cost of creating test cases using a white box algorithm is, again, the cost in personnel time of setting up and running the algorithm. If the test case comes from an existing production system that is being reprogrammed, then the cost is merely a matter of the time it takes to extract the test case from the production run.
The cost to determine the expected output of the test case (CEO) varies with the complexity of the business function being programmed. In some instances, the expected output is determined manually and the cost is a direct function of personnel time needed to do so. In some situations the tester may decide to write a small program to generate expected outputs based on the sets of input values (although this begs the question of whether this program needs to be tested itself!) Of course, if the test case comes from an existing production system, then effectively, CEO=0 or the small amount of cost involved in looking at the production system's output.

The cost to run a test case (CRT) begins with the time it takes to set the test case up for an execution run along with other test cases. Of course, if test cases are run one-at-a-time, as might be the case in unit testing or exploratory testing, then there is no set-up time. Similarly, the cost of the computing resources needed to run the test cases may or may not have to be factored in depending on the complexity of the software involved. As a summary point, CRT will vary with the degree of automation being used in running the test cases.

The cost to record the test results (CRR) will depend on whether and which defect management system is being used. Once again, the cost is a function of the time to record the actual output of running the test case, unless software is in place to automatically record the results.

The cost to evaluate the result of running the test case (CER) begins with the time cost of comparing the actual result with the expected result. If they differ, then a record noting this must be entered in the defect management system. Typically, a number of fields must be entered including the test case identifier, a description of the anomaly, the name of the person who entered the defect, and so on.

Arguably, the cost to fix a defect based on the particular test case may or may not be included in the equation as company management sees fit. As suggested earlier, the cost of not finding the defect or of ignoring it if found may well be greater, perhaps much greater, than the cost of finding and fixing it. The cost of fixing the defect can vary widely, depending on, among other things, the stage of application development to which it can be traced, the complexity of the code involved, and the skill of the developers.

III. Future Work

The foregoing is just the beginning of what would be needed to have a comprehensive view of the unit cost of test cases. Future work could proceed along several paths:

1. Each of the terms in the cost equation given above could be developed in more detail, including factoring in actual personnel costs, estimated times, and estimated risk.
2. The same kind of analysis could be made for the unit cost of test cases in performance testing, security testing, usability testing, and globalization testing.

3. An analysis could be made of the unit cost of test cases in the agile development environment and compared with the costs in the traditional development environment.
Requirements and Testability (Recoverability) notions for IT Disaster Recovery

Workshop on Advances and Innovations in Software Testing
October 28, 2013

Vikram Dias

What is a Disaster? [Latin root meaning “bad star”]

Comprised of predominantly two types

Natural: Oklahoma EF5, Hurricane Sandy, Pakistan Quakes, Colorado Forest Fires

Man-made: BotNets, Hack Attacks

<table>
<thead>
<tr>
<th>Event</th>
<th>Year</th>
<th>FEMA funding (billion dollars)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Northridge Earthquake</td>
<td>1994</td>
<td>6.961</td>
</tr>
<tr>
<td>Hurricane George</td>
<td>1968</td>
<td>2.251</td>
</tr>
<tr>
<td>Hurricane Ivan</td>
<td>2004</td>
<td>1.947</td>
</tr>
<tr>
<td>Hurricane Andrew</td>
<td>1992</td>
<td>1.813</td>
</tr>
<tr>
<td>Hurricane Charley</td>
<td>2004</td>
<td>1.255</td>
</tr>
<tr>
<td>Hurricane Frances</td>
<td>2004</td>
<td>1.425</td>
</tr>
<tr>
<td>Hurricane Jeanne</td>
<td>2004</td>
<td>1.467</td>
</tr>
<tr>
<td>Tropical Storm Allison</td>
<td>2001</td>
<td>1.387</td>
</tr>
<tr>
<td>Hurricane Hugo</td>
<td>1989</td>
<td>1.307</td>
</tr>
<tr>
<td>Midwest Floods</td>
<td>1993</td>
<td>1.140</td>
</tr>
</tbody>
</table>

Source: FEMA (available at: www.fema.gov)

Financial impacts run into Millions if not Billions $\$$. 
Genetics of man-made disasters -- The Opposite of ‘Hope’ is ‘Ixxxxxxxn’??

- Because Ixxxxxxxn can kill you!
- UK’s National Health Service study
  - 6500 men and women polled
  - Study created a ‘Social Isolation Index’
- UCLA developed ‘loneliness scale’
- Yale psychiatrist studies on ‘Depression Risk Gene’
- Studies concluded:
  - Social Isolation results in higher risk of death in ages 52 years and up
  - Although both isolation and loneliness impair quality of life and well-being, efforts to reduce isolation are likely to be more relevant to mortality
  - Focus on meaningful extra-familial social connections

Disaster Taxonomy models stretch the vision!

The University of Delaware’s Disaster Research Center (DDRC, available at: www.udel.edu/DRC/preliminary/0p304.pdf) differentiates disasters from emergencies and catastrophes as follows:

1. Emergency: An event that may be managed locally without the need of added response measures or changes to procedure.
2. Disaster: An event that:
   - involves more groups who normally do not need to interact in order to manage emergencies;
   - requires involved parties to relinquish the usual autonomy and freedom to special response measures and organizations;
   - changes the usual performance measures and;
   - requires closer operations between public and private organizations.
3. Catastrophe: An event that:
   - destroys most of a community;
   - prevents local officials performing their duties;
   - causes most community functions cease and;
   - prevents adjacent communities from providing aid.
Disaster Hierarchies can be complex!

The Canadian Disaster Database (CDD, available at: www.ocipep.gc.ca/disaster/search.asp) categorized disasters in five different types as follows:

1. biological, such as epidemic;
2. geological, such as earthquake;
3. meteorological and hydrological, such as drought;
4. human conflict, such as terrorism; and
5. technological hazardous, such as chemicals materials.

The Disaster Database Project (DDP, available at: learning.richmond.edu/disaster/index.cfm) conducted by University of Richmond, categorized the disasters in three major classes as follows:

1. conflict based disaster, such as bombing and massacre;
2. human systems failure, such as dam collapse and mine accident; and
3. natural disaster, such as earthquake.

It should be mentioned that sometimes natural disasters are man-made such as damage to ozone layer. There is also a long list of some hypothetical natural disasters such as asteroid impact or mass extinction on earth by hyper nova disaster. It is also important to know that, many times, one natural disaster is accompanied by another. For example, earthquakes and volcanoes sometimes occur together because they are both caused by geologic movements. Epidemics may also be the consequence of disasters of another kind, such as earthquakes, tropical storms, floods, etc.

IT Disaster Recovery defined

Recovery of an organizations critical business process based on a defined recovery time objective (RTO) and recovery point objective (RPO)
IT DR testing process must be structured, repeatable and project manageable

IT DR Test Process Components

**DR Test Planning**
- Test Plan
- Application Architecture Diagram
- Defined User Base
- Environment
- Project Plan

**DR Test Execution**
- Disaster Date/Time
- Network Segmentation
- Recoverability
- Functional Testing
- Sanity Testing
- Actual vs. Expected

**Management Report-Out**
- Test Summary Report
- IT Exec. Presentation
- Lessons Learned

An Enterprise DR topology is embedded into a resilient, scalable, location agnostic Enterprise Architecture (EA)

**Business Architecture**
- Business Process Models
  - L0 – L1 – L2 – L3 – L4
- Business Impact Analysis (BIA)
  - Take – Make – Ship – Invoice - Pay

**Application Architecture**
- Application Architecture Diagram
- Recovery List (Tier 0…n)

**Infrastructure Architecture**
- Network Segmentation
- Gateway Emulator
- Environment Stack
- User Authentication
Recoverability notions can be coupled with concepts of Testability to drive a solid DR test strategy

- **Recoverability (R)**
  - Resumption of key critical business applications based on a defined set of business based RTOs/RPOs

- **Stability (S)**
  - Fewer the changes to the DR environment, the fewer the disruptions to user and/or technical testing

- **Observability (O)**
  - What you see is what can be tested (user or technical)

---

Let's expand R-S-O

- **R**
  - Identify key critical business processes using the Business Impact Analysis (BIA)
  - Interview business and application SMEs (Test Plan)
  - Architect and massage current/future state business process models (BPM)

- **S**
  - Isolate production versus DR at the network layer (Network Segmentation)
  - Set-up DR topology based on approved application diagram and business process model (Recovery List)
  - Changes to the DR architecture are communicated, controlled and orchestrated (CM)

- **O**
  - Past system states and variables are visible or queriable (Web-Methods)
  - Internal errors are automatically detected and reported through self-testing mechanisms (Electric Commander)
A repeatable IT DR testing exercise must lend itself to functional testing involving the business!

- Work with application development to QA test plans
- Execute functional test scenarios
- Validate actual versus expected

All of this can only be achieved with a business resilient project management approach

Project Plan – IT Project Manager

Tiger Team – IT DR Manager, Data Center Ops., Business/Process Stewards, Application Architects, Test Users, Infrastructure Specialists

It's about... LIFE: Life-Changing Injury and Fatality Elimination
What’s next in the DR Ontology?

Integrated Global Supply Chain Testing
Data Loss Recovery
Crisis Management Testing

Appendix
DR Test Project Plan

DR Application Arch. Diagram
DR Test Execution Sample

<table>
<thead>
<tr>
<th>Job</th>
<th>Job Description</th>
<th>Executable (file, and time)</th>
<th>Accepted/Outbound</th>
<th>Internal/External</th>
<th>System</th>
</tr>
</thead>
<tbody>
<tr>
<td>CDR</td>
<td>Authorization to Pay Batch</td>
<td>cda_cdr_befint.bat</td>
<td>Out</td>
<td>Ext</td>
<td>ESY SAP/HR</td>
</tr>
<tr>
<td>CDR</td>
<td>Pay AP Distribution: AP reconciliations, all matching checks for the inter states which have been paid by SAP</td>
<td>cda_cdr_conc.bat</td>
<td>Int</td>
<td>Internal</td>
<td>ESY SAP/HR</td>
</tr>
<tr>
<td>CDR</td>
<td>Processing check payment information from SAP</td>
<td>cda_cdr_conc.bat</td>
<td>N</td>
<td>Ext</td>
<td>ESY SAP/HR</td>
</tr>
<tr>
<td>EIS</td>
<td>Pay Job</td>
<td>The job updates the shipment table for all the approved shipments</td>
<td>na</td>
<td>Int</td>
<td>Internal</td>
</tr>
<tr>
<td>EIS</td>
<td>EIS HR Send to SAP</td>
<td>For EIS Send to SAP</td>
<td>Out</td>
<td>Ext</td>
<td>ESY SAP/HR</td>
</tr>
</tbody>
</table>

DR Exit-Entry Testing Checklist

<table>
<thead>
<tr>
<th>Application</th>
<th>Test Type</th>
<th>KTO (Docs)</th>
<th>IT DR Lead</th>
<th>Architecture</th>
<th>Diagram</th>
<th>Test/Plan</th>
<th>Jump Box Test</th>
<th>Jump Box Test</th>
<th>Jump Box Test</th>
<th>Go/NoGo System for April 22</th>
<th>DR Testing Complete</th>
<th>Test Results Submitted</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Page 17
Testing for Tautology based SQL Injection Attack using Runtime Monitors

Ramya Dharam and Sajjan G. Shiva
Department of Computer Science
University of Memphis
Memphis, TN-38152, USA
{rdharam, sshiva}@memphis.edu

Abstract - Today, all commercial and business applications (e-commerce, banking, blogs, web mail, etc.), are built as web-based database applications. Increasing prominence and usage of these applications has made them more susceptible to attacks because they store huge amount of sensitive user information. Traditional security mechanisms like network firewalls, intrusion detection systems, and use of encryption can protect the network, but cannot mitigate attacks targeted towards web applications. Hence hackers are moving their focus from network to web applications. SQL Injection Attacks (SQLIAs) are one of the most popular and widely performed attacks on web applications. Various security testing techniques exist to detect vulnerabilities in web applications. Vulnerability Scanning performed using vulnerability scanners is one of the widely used security testing techniques, but the rate of false positives and false negatives obtained limit their usage to efficiently detect all the vulnerabilities present in web applications. In this paper, we present and evaluate a security testing technique called Runtime Monitoring to detect and prevent tautology based SQL Injection Attack. For evaluation, we targeted a web application with large number of both legitimate inputs and illegitimate tautology based attack inputs, and measured the performance of the proposed technique.

Keywords - Security Testing; Runtime Monitoring; Data-flow Testing; Basis-path Testing; Web Applications; SQL Injection Attacks (SQLIAs); Tautology.

I. INTRODUCTION

Web applications have become popular means of modern information retrieval and interaction. This increase in demand and usage of web applications has made them vulnerable to attacks, as they contain confidential user data. In June 2013, the Open Web Application Security Project (OWASP) officially released the Top 10 attacks performed on web applications [1]. Once again, SQLIAs have been ranked as the most widely performed attack on web applications. SC Magazine [2] and ZDNet [3] discuss the recent increase in SQLIAs during the year 2013 and 2012 respectively. Popular websites like Sony [4], LinkedIn [5], Nvidia [6] and Gamigo [7] were hacked using SQLIAs, in June and July 2012, and sensitive information like passwords, account details, etc., about millions of users were leaked online. In spite of various methodologies and testing techniques proposed and implemented by researchers and organizations to handle SQLIAs, they still allow breaches, and hence the security of web applications is not completely accomplished.

A web application is structured as a three-tiered architecture consisting of a web browser, an application server and a back-end database server. Such an application will accept input from external users via forms, dynamically construct the database queries using the inputs provided by the user, dispatch them to the underlying database for execution, and finally retrieve and present the data to the user.

SQLIAs are a class of code injection attacks which are directed towards the database residing at the back-end of the web applications and the attacker tries to gain unauthorized access to the confidential user information residing in the database. SQLIAs occur when the input provided by a malicious user consisting of SQL keywords or operators is not properly validated and is included directly as part of the query. This causes the web application to generate and send a query that in turn results in unintended behavior of the web application causing the loss of confidential user information [8]. Different kinds of SQLIAs known to date are discussed in [9, 10] which include the use of SQL tautologies, illegal queries, union queries, piggy-backed queries, etc.

In this paper, we discuss the Tautology based SQL Injection Attack which is the most simple and popular type of SQLIAs. This attack is performed on web applications by injecting code that consists of SQL tokens into one or more conditional statements so that they always evaluate to true. Bypassing the authentication page in web applications, and extracting data is the most common usage of tautology based SQLIA. In this type of attack, a vulnerable input field that is used in the query’s WHERE condition is exploited by the attacker. As the database scans each record in the table, the conditional logic is evaluated and the database returns all the records in the table, if the evaluated conditional logic represents a tautology, and returns true. The attack is successful when the web application either displays all of the returned records or performs some action if at least one record is returned.

For example, if a database contains usernames and passwords, the application may contain code such as the following:
Query = "SELECT * FROM employeeinfo WHERE name = ""+ request.getParameter("name") +"' OR 1 = 1 --" AND password = '"+ request.getParameter("password") +"'";

This code generates a query intended to be used to authenticate a user who tries to login to a web site. If a malicious user enters "‘ OR 1 = 1 -- " AND ‘’‘ ‘‘ ‘’‘ ‘’‘ ‘’‘ ‘’‘ ‘’‘ ‘’‘ ‘’‘ ‘’‘ ‘’‘ ‘’‘ ‘’‘ ‘’‘ ‘’‘ ‘’‘ ‘’‘ ‘’‘ ‘’‘ ‘’‘ ‘’‘ ‘’‘ ‘’‘ ‘’‘ ‘’‘ ‘’‘ ‘’‘ ‘’‘ ‘’‘ ‘’‘ ‘’‘ ‘’‘ ‘’‘ ‘’ ‘’‘ ‘’‘ ‘’‘ ‘’ ‘’‘ ‘’ ‘’‘ ‘’ ‘’‘ ‘’ ‘’‘ ‘’ ‘’ ‘’‘ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘’ ‘”;

Any website that uses this code would be vulnerable to tautology based SQLIA. The character "--" indicates the beginning of a comment, and everything following the comment is ignored. The database interprets everything after the WHERE token as a conditional statement, and the inclusion of "OR 1=1" clause turns this conditional into a tautology whose condition always evaluates to true. Thus, when the above query is executed, the user will bypass the authentication logic, and more than one record is returned by the database. As a result, the information about all the users will be displayed by the application and the attack succeeds.

To detect such attacks and other types of security vulnerabilities, security testing is the most widely used testing technique. The overall goal of security testing is to reduce vulnerabilities within software systems. There are two major aspects of security testing: testing security functionality to ensure that it works, and testing the subsystem in light of malicious attack [11]. Different types of security testing techniques and tools are discussed in [12]. The two main approaches to perform security testing on web applications include White Box Testing and Black Box Testing [13]. White Box Testing consists of the source code analysis of web applications to detect vulnerabilities in them, and is performed manually or by using Source Code Analysis Tools. Due to the complexity of the code, exhaustive source code analysis may be difficult and might not be able to find all security vulnerabilities. Black Box Testing consists of analysis of the execution of application to detect vulnerabilities, and this approach is also known as Penetration Testing. It is performed by Black Box Testing Tools also called as Penetration Testing Tools or Vulnerability Scanners. The usage of vulnerability scanners is regarded as an easy way to test the security of web applications, and to detect critical vulnerabilities like SQL Injection Attacks. The rate of false positives (vulnerabilities detected that did not exist) and false negatives (vulnerabilities existed and were not detected) obtained limit their usage to efficiently detect all the vulnerabilities present in web applications.

Due to the limitations of the techniques mentioned above, web applications are still vulnerable to attacks. In this paper, we present and evaluate a security testing technique called Runtime Monitoring to handle tautology based SQL Injection Attack. For evaluation, we targeted an application with a large number of both legitimate inputs and illegitimate tautology based inputs, and measured the performance of the proposed technique. The results obtained were promising and our technique was successfully able to detect and prevent the tautology based SQLIA on web application, without causing any false positive or false negative. Our proposed technique also imposed a low runtime overhead on the subject application.

The paper is organized as follows. In Section 2, we discuss in detail the Runtime Monitoring Technique to handle Tautology based SQLIA. Evaluation and results obtained are discussed in Section 3. Finally, we conclude in Section 4.

II. Runtime Monitoring Technique

Runtime Monitoring involves examining the behavior of the program under test, determining whether this observed behavior is symptomatic of vulnerability in the software [14], and is usually performed by runtime monitors. In this section, we discuss our framework called Runtime Monitoring Framework to develop and integrate runtime monitors into the web application. The instrumented application obtained performs runtime monitoring of the application to detect and prevent tautology based SQLIA. Fig. 1 shows the high-level view of our framework consisting of the following modules: i) Critical Variables Identification Module ii) Critical Paths Identification Module iii) Runtime Monitor Development and Instrumentation Module.

![Fig. 1 High Level View of Runtime Monitoring Framework.](image_url)

**Critical Variables Identification Module**: identifies all the critical variables, i.e. variables that are initialized with the input provided by external user and those that become a part of SQL query. Input to this module is a Java web application, and it outputs the critical variables.
Critical Paths Identification Module: identifies the critical paths generated by data-flow and basis-path testing techniques. This module takes the identified critical variables as input and returns the paths that need to be monitored. Data-flow testing [15] of the critical variables helps in identification of all the legal sub-paths that can be taken by critical variables during execution. Basis-path testing [16] is performed to identify the minimum number of legal execution paths of the application. The path identification function as discussed in [17] builds the set of critical paths to be monitored in the application to detect and prevent tautology based SQLIA.

Runtime Monitor Development and Instrumentation Module: develops the runtime monitor for the identified critical paths and instruments it into the respective module of the application. Runtime Monitor is developed using AspectJ [18] which is an aspect-oriented extension to the Java language. A special compiler provided by AspectJ called the AspectJ Compiler (ajc is the command to be used to invoke this compiler) is used to integrate the runtime monitor into the respective module of the application. Henceforth, on every query execution, the runtime monitor tracks the identified critical variables by monitoring their execution path. When a critical variable follows an invalid path, the runtime monitor immediately detects the abnormal behavior of the application as a possible tautology based SQLIA and halts the execution of the application.

III. Evaluation

To evaluate our proposed approach, we performed experiments to assess the false positive, false negative, and runtime overhead imposed by our technique.

We used a web application named Employee Directory from the SQL Injection Application Testbed [19] which has been used to evaluate other techniques [20, 21, 22, and 23] to detect and prevent SQLIAs. The subject web application that we used for our experiment is the Employee Directory web application which is an online employee directory consisting of about 5000 lines of code developed using JSPs.

By surveying various sources which included government security websites such as NVD (http://www.nvd.nist.gov/), OWASP (https://owasp.org/), Build Security In (https://buildsecurityin.us-cert.gov/), US-CERT (http://www.us-cert.gov/), security related mailing lists, and research papers, etc., we generated both legitimate and attack set inputs to the Employee Directory web application. In total, we used 10 tautology attack inputs and 30 legitimate inputs.

We first test the instrumented Employee Directory web application with the collected attack inputs. The runtime monitor detects the abnormal behavior of the application displaying all the records present in the table as tautology based SQLIA, and halts its execution. Then, we ran the legitimate inputs on the same instrumented Employee Directory web application, and all the legitimate inputs were executed successfully.

The experimentation performed clearly demonstrates the success of our developed runtime monitor to handle tautology based SQLIA on the Employee Directory application. The monitor successfully allowed the legitimate queries to be executed on the application and detected the tautology based SQLIA performed on the application i.e. both false positive and false negative were handled effectively. Table I and Table II below show the false negative and false positive results obtained respectively.

<table>
<thead>
<tr>
<th>Subject</th>
<th>Total # of Tautology based Attack Inputs</th>
<th>Total # of attacks detected on instrumented web application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Employee Directory</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Subject</th>
<th>Total # of Legitimate Inputs</th>
<th>Total # of legitimate inputs successful on instrumented web application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Employee Directory</td>
<td>30</td>
<td>30</td>
</tr>
</tbody>
</table>

To determine the runtime overhead imposed by our proposed approach, we ran the legitimate inputs on the Employee Directory application which is not instrumented, and measured the response time of the application. We then ran the same legitimate inputs on the instrumented version of the Employee Directory application and recorded the response time. The difference in the response time obtained from the two versions of the application is determined as the overhead imposed by our proposed technique.

Only the legitimate test set is used for the overhead calculation, because the attack set would cause different paths of execution between the two versions, where the attacks would be successful in the original application, but be prevented in the instrumented application, leading to incorrect timing comparisons [23]. We ran our experiments five times and recorded the average time to ensure accuracy. We found that the runtime overhead imposed by our proposed technique on the Employee Directory application is no more than 4% which is comparatively less than the average overhead of WASP [74] listed as 6%. Table III below shows the comparison results obtained.

<table>
<thead>
<tr>
<th>Technique</th>
<th>% Overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td>Runtime Monitoring</td>
<td>4%</td>
</tr>
<tr>
<td>WASP</td>
<td>6%</td>
</tr>
</tbody>
</table>
IV. CONCLUSION

We presented a novel security testing technique called runtime monitoring to detect and prevent tautology based SQLIA on web applications, based on the development of runtime monitors using our runtime monitoring framework. The results of our evaluation showed that the runtime monitor was successfully able to detect and prevent the tautology based SQLIA performed on the target web application, and allowed legitimate inputs to access the database. Also, our technique imposed low overhead on the application. Thus, our technique ensures that the security of the application is accomplished during its post-deployment.

V. REFERENCES

[12] https://buildsecurityin.us-cert.gov/articles/tools
Testing Practices
Iterative Software Development Projects

Robin Poston, Associate Professor
Department of Management Information Systems
309 Fogelman Administration Building
Memphis, TN 38152-3120
rposton@memphis.edu

Jignya Patel, Ph.D. Candidate
Department of Management Information Systems
365 Fogelman College Class Bldg
Memphis, TN 38152-3120
jmpatel@memphis.edu

Jasbir Dhaliwal, Associate Dean
Department of Management Information Systems
426B Fogelman Administration Building
Memphis, TN 38152-3120
jdhaliwal@memphis.edu

Abstract—Focusing on quality software testing practices has become central to the success of iterative development projects. Much of the extant research has not examined the testing practices observed in thriving iterative (also called agile herein) development teams. Our research identifies testing practices found in case study input from members of large multinationals and small consulting companies providing services to large companies. From our interviews with multiple development teams following iterative development processing, we identify testing practices in light of existing literature to highlight how testing practices deliver long-term value. Through this discussion, we delineate and discuss testing techniques. Implications for theory and practice are also discussed.

Keywords — Iterative software development, testing practices, team software development processes

I. INTRODUCTION

Agile software development is characterized as a highly-iterative, incrementally-evolving approach involving strong collaboration among project stakeholders, including testers. Following an agile methodology can be a radical departure from following traditional structured waterfall approaches. Agile approaches emphasize continuously-evolving software over extensively-planned specifications, collaboration over command and control, informal over formal communications, flexibility over bureaucratic formalization [7]. Agile software development refers to a disciplined methodology using iterations to deliver prioritized subsets of functionality, frequent user input, small and frequent releases, rigorously tested code, and an achievable work pace [3], [9]. While iterative practices are rather general and could encapsulate all roles of the software development process, this study will focus specifically on testing practices in iterative software development projects.

Benefits of following iterations agile approaches involve taking advantage of lessons learned earlier in the project in later stages, focusing on keeping the project work simple, implementing only a subset of the software requirements each time, and iteratively enhancing the product until the full system is built. An iterative approach stands in stark contrast to widely-used non-iterative processes of software development. Non-iterative software development teams follow structured waterfall-like approaches to achieve standardized role-oriented teams or organizations. Traditional software development processes call for customers to establish pre-defined work plans that identify detailed functionality desired by pre-specified dates, then implementation teams to execute the plan to develop the software solution without much customer involvement (Ambler, 2006). Both customers and software development team members have years of experience with this approach which is in direct conflict with agile practices. Given the oftentimes radical departure from wide-followed structured development processes and focus on continuous testing in iterative methods, this study explores the best practices of how embedded testing practices into iterative software development projects.

Considerable research exists on examining factors related to agile concepts, organizational settings, and team members in iterative agile development settings; see Table 1 for select summary of studies. Our study contributes to this research stream by examining specifically the testing practices found in our case study companies, then comparing and discussing these practices in the light of existing literature to highlight how the testing practices support iterative development process to deliver long-term value. Through this comparison, this study explores testing issues and how case study companies succeeded in addressing these issues by adopting techniques that fit the iterative culture and approach.
To frame the testing practices proposed by our research, we draw on Mesu and Jain’s [15] adaptive systems principles and best practices (summarized in Table 2). Their agile practices framework offers a comprehensive set of themes related to agile software development which are well suited and strongly apply to our case study contexts. Thus, we organize our findings based on their themes to offer additional insights about the practices found in our research.

This article is organized as follows. First, we discuss our research methodology, followed by the discussion of our findings from the case study interviews from members of a set of large multinational and small local companies with multiple development teams following iterative development processes. Based on our findings, we discuss testing issues and techniques. Finally, implications for theory and practice are discussed.

II. RESEARCH METHODOLOGY

We use a qualitative approach in gathering investigative case study data on testing practices in iterative software development environments. The research methodology follows a qualitative approach in gathering case study data in order to provide descriptive and explanatory insights into the management activities in iterative development work. This approach has been used successfully in prior research [17], [19] and allows us to induce a theoretical account of the activities found in empirical observations and analysis of team member’s viewpoints. This approach is also known to lead to accurate and useful results by including an understanding of the contextual complexities of the environment in the research analysis and outcomes. Finally, this approach encourages an understanding of the holistic systemic view of the issues and circumstances of the situation being addressed, in this case the issues of managing iterative development projects from team member perspectives about their testing practices [6], [22].

**TABLE I. SELECT STUDIES ABOUT ITERATIVE AGILE DEVELOPMENT SETTINGS**

<table>
<thead>
<tr>
<th>Reference</th>
<th>Theory/Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agile Concepts</td>
<td></td>
</tr>
<tr>
<td>What agile teams think of agile principles, [21]</td>
<td>Survey of online agile user groups</td>
</tr>
<tr>
<td>Agile requirements engineering practices and challenges: An empirical study, [18]</td>
<td>16 US software development organizations case studies</td>
</tr>
<tr>
<td>Field experiences with XP: Developing an emergency response system, [10]</td>
<td>Development project of web-based, distributed info system</td>
</tr>
</tbody>
</table>

**Organizational Setting**

| Integrating software product line engineering and agile development, [16] | Complex Adaptive Systems /Case study of an enterprise software development organizations |
| Collective agility, paradox and organizational improvisation: The development of a particle physics grid, [23] | Collective agility /Agile systems development project of the Large Hadron Collider Computing Grid (LCO) case study |
| Control of flexible software development under uncertainty, [12]          | Dynamic capabilities theory and control theory/Case study |

**Team Members**

| Extending agile principles to larger, dynamic software projects: A theoretical assessment, [2] | TCE, Social exchange theory, expectancy theory |
| The role of project management in ineffective decision making within Agile software development projects, [14] | Groupthink/Exploratory longitudinal study of software development teams |
| Are two heads better than one for software development? The productivity paradox of pair programming, [11] | Group performance studies/Lab experiment with students |
| User acceptance of agile information systems: A model and empirical test, [20] | Status quo, Omission bias theories, availability heuristic, UTAM, Continuance, Habit/477 users of agile IS for one company’s new web portal |

**TABLE II. MAPPING ITERATIVE TESTING PRACTICES TO COMPLEX ADAPTIVE SYSTEM PRINCIPLE AND AGILE PRACTICES (ADAPTED FROM MESU AND JAIN, 2006)**

<table>
<thead>
<tr>
<th>Complex Adaptive System Principle</th>
<th>Agile Practice Theme</th>
<th>Iterative Testing Practice</th>
</tr>
</thead>
<tbody>
<tr>
<td>Principle of growth and evolution and continuous integration</td>
<td>Frequent releases and continuous feedback</td>
<td>Practice 1 - Test early and continuously  Practice 2 - Use Testing tools</td>
</tr>
<tr>
<td>Principle of transformative feedback loops</td>
<td>Need for frequent feedback</td>
<td>Practice 3 - Shift testing workload  Practice 4 - Test more and produce better quality output</td>
</tr>
<tr>
<td>Principle of emergent order</td>
<td>Proactive handling of changes to the project requirements</td>
<td>Practice 5 - Maintaining a testing discipline</td>
</tr>
<tr>
<td>Principle of distributed control</td>
<td>Loosely controlled development environment</td>
<td>Practice 6 - Developers doing testing  Practice 7 - Testers on the team</td>
</tr>
<tr>
<td>Principle of growth and evolution, and Principle of emergent order</td>
<td>Planning kept to a minimum</td>
<td>Practice 8 - Iterative through to final testing which is linear</td>
</tr>
<tr>
<td>Principle of growth and evolution, and Principle of interactions and relationships</td>
<td>Enhancing continuous learning and continuous improvement</td>
<td>Practice 9 - Skills needed for agile testing</td>
</tr>
<tr>
<td>Principle of path of least effort</td>
<td>Emphasis on working software product</td>
<td>Practice 10 - Testing governance compliance</td>
</tr>
</tbody>
</table>
To identify the practices, we selected four companies: three large multinationals and two smaller consulting companies known to have successful iterative development projects. The focus of our study is specific to the testing practices of project teams developing software following iterative practices.

A. Data Collection

The results reported in the present study are based on interviews with software development and testing team members. Our data gathering began with the creation of semi-structured interview protocols which comprised both closed and open-ended questions. To inform our interview question development, we reviewed documentation about each company, and held background discussions with company personnel. The data collection methods employed focused on interviewees’ perspectives on testing issues, roles played by various stakeholders involved, and the iterative challenges of incorporating testing activities in an iterative development process. Face-to-face interviews of approximately 1 to 1.5 hours were conducted with various project stakeholders. The goal of these interviews was to identify and better understand the issues related to iterative development software testing. These stakeholders included project managers, test leads, developers, scrum masters, business analysts and architects. In total, we interviewed 19 stakeholders. Interviews were conducted between December 2012 and June 2013, with additional follow-up clarification Q&A sessions conducted over e-mail. Job descriptions of those interviewed are shown in Table 3.

<table>
<thead>
<tr>
<th>TABLE III. JOB DESCRIPTION OF INTERVIEWEES</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Responsibility</strong></td>
</tr>
<tr>
<td><strong># of times Interviewed</strong></td>
</tr>
<tr>
<td>-------------------------------------------</td>
</tr>
<tr>
<td><strong>Multinationals</strong></td>
</tr>
<tr>
<td>Project Manager at logistics company</td>
</tr>
<tr>
<td>IT Lead/SCRAM master at logistics company</td>
</tr>
<tr>
<td>Director of Automation Services at logistics company</td>
</tr>
<tr>
<td>Senior Developer at logistics company</td>
</tr>
<tr>
<td>Developer at software company</td>
</tr>
<tr>
<td>SCRAM Master at retail company</td>
</tr>
<tr>
<td><strong>Consulting Companies</strong></td>
</tr>
<tr>
<td>Project Manager at consulting company 1</td>
</tr>
<tr>
<td>Senior Manager at consulting company 1</td>
</tr>
<tr>
<td>Senior Manager at consulting company 2</td>
</tr>
<tr>
<td><strong>Total Interviews</strong></td>
</tr>
</tbody>
</table>

By collecting and triangulating data across a variety of methods and from a variety of stakeholders and companies, we were able to develop robust results because of the multiple perspectives we gained about testing issues in iterative software development. This approach provides in-depth information on emerging concepts, and allows cross-checking the information to substantiate the findings [8], [11], [17].

III. RESULTS: TESTING PRACTICES USED IN ITERATIVE DEVELOPMENT

The purpose of this study is to develop a set of testing practices based on case study input. To achieve our goal, we categorized practices identified through our data analysis in 7 main themes drawn from the agile practices framework [15] to allow for a comparison of the experiences from our case study companies to a current framework provided in the literature. See Table 4 for a summary of these practices. Selected direct quotes for each practice are provided in the Appendix.

A. Theme 1: Frequent Releases and Continuous Integration

The first theme in the practices framework is Frequent Releases and Continuous Integration [15]. This theme involves developing the software artifact iteratively, allowing it to acquire increasing complexity in each iteration,
reevaluating development methods used frequently, starting simple and modifying as needed, reevaluating team configurations, and scaling and transforming team composition as work progresses [15, p. 23]. Organizations that succeed in agile development examine the product, process, and people dimensions frequently making incremental changes and adaptations as circumstances arise. Within this theme, we offer the first two testing practices from our case study interviews. Explanations of those practices can be found in Appendix A.

Iterative Testing Practice 1: Test Early and Continuously
Iterative Testing Practice 2: Use Testing Tools

B. Theme 2: Need for Frequent Feedback

The second theme in the practices framework [15] is Need for Frequent Feedback. This theme involves iteratively testing and validating software and making improvement, time-boxing development efforts with measurable progress milestones, and pragmatically involving stakeholders to carefully seek and listen to feedback [15, p. 23]. Organizations that succeed in agile development can skillfully identify feedback opportunities and implement the mechanisms to effectively coordinate changes to processes and to the product. We discuss two testing practices that agile teams employed to achieve this practice. Explanations of those practices can be found in Appendix A.

Iterative Testing Practice 3: Shift testing in workload
Iterative Testing Practice 4: Test more and Produce Better Quality Output

C. Theme 3: Proactive Handling of Changes to the Project Requirements

The first theme in the practices framework is Frequent Releases and Continuous Integration [15]. This theme involves tailoring the agile development process to take into account the changes in the requirements and the feedback gained from exercising frequent releases and integration [15, p. 23]. Organizations that succeed in agile development can invite users during the development to allow the team to produce software that is more current and reflective of the changing business environment. Within this theme, we offer the next testing practice from our case study interviews. Explanations of this practice can be found in Appendix A.

Iterative Testing Practice 5: Maintaining a Testing Discipline

D. Theme 4: Loosely Controlled Development Environment

The first theme in the practices framework is Frequent Releases and Continuous Integration [15]. This theme involves using a modular development approach to develop loosely coupled software artifacts [15, p. 23]. Organizations that succeed in agile development keep successive iterations fairly independent of each other. Within this theme, we offer the two additional testing practices from our case study interviews. Explanations of those practices can be found in Appendix A.

Iterative Testing Practice 6: Developers Doing Testing

Iterative Testing Practice 7: Testers on the Team

E. Theme 5: Planning Kept to a Minimum

The first theme in the practices framework is Frequent Releases and Continuous Integration [15]. This theme involves keeping documentation, product planning and stakeholders involvement to a minimum, doing it only when and where necessary [15, p. 23]. Organizations that succeed in agile development plan to a certain extent both the software and the process used to develop the software. Within this theme, we offer the next testing practice from our case study interviews. Explanations of this practice can be found in Appendix A.

Iterative Testing Practice 8: Iterative Through to Final Testing Which is Linear

F. Theme 6: Enhancing Continuous Learning and Continuous Improvement

The first theme in the practices framework is Frequent Releases and Continuous Integration [15]. This theme involves experimenting with the design of the product to allow learning leading to a better quality product [15, p. 23]. Organizations that succeed in agile development allow development teams to form collaborative partnerships to cultivate productivity, creativity and quality. Within this theme, we offer next testing practice from our case study interviews. Explanations of this practice can be found in Appendix A.

Iterative Testing Practice 9: Skills Needed For Agile Testing

G. Theme 7: Emphasize on Working Software Product

The first theme in the practices framework is Frequent Releases and Continuous Integration [15]. This theme involves building an effective working product with least effort [15, p. 23]. Organizations that succeed in agile development foster an environment to produce an error-free product by allowing natural team configurations to emerge. Within this theme, we offer the last testing practice from our case study interviews. Explanations of this last practice can be found in Appendix A.

Iterative Testing Practice 10: Testing Governance Compliance

IV. CONCLUSIONS AND IMPLICATIONS

In this research, we gathered and analyzed interview data from large multinational and small consulting companies with multiple development teams following iterative development processes. This article used the themes of the agile practices framework [15] to organize our discussions about the various testing practices that emerged from this analysis. The findings offer insights about systemic thinking of iterative testing practices (Frequent Releases and Continuous Integration) Test Early and Continuously and Use Testing Tools; (Need for Frequent Feedback) Shift Testing Workload and Test More and Produce Better Quality Output; (Proactive Handling of Changes to the Project Requirements) Maintaining a Testing Discipline; (Loosely Controlled Development Environment) Developers Doing Testing and Testers on the Team; (Planning
Kept to a Minimum) Iterative Through to Final Testing Which is Linear; (Enhancing Continuous Learning and Continuous Improvement) Skills Needed For Agile Testing and (Emphasize on Working Software Product) Testing Governance Compliance. As a result of the comparison of the main topics/themes from the literature to the findings from our case study, we explored the testing issues and how agile teams succeeded in solving these issues by adopting techniques that fit the work dynamics. Each practice deployed by agile teams involves steps to improve their testing activities and overall high-quality product outcomes. Testing issues as a focus is not widely addressed in the literature and deserves more research attention. An initial list of iterative testing practices that emerged from our study is summarized in Table 4.

TABLE IV. ITERATIVE TESTING FRAMEWORK FOR AGILE SOFTWARE DEVELOPMENT PROJECTS

<table>
<thead>
<tr>
<th>Theme 1: Frequent Releases and Continuous Integration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iterative Testing Practice 1: Test Early and Continuously</td>
</tr>
<tr>
<td>Iterative Testing Practice 2: Use Testing Tools</td>
</tr>
<tr>
<td>Theme 2: Need for Frequent Feedback</td>
</tr>
<tr>
<td>Iterative Testing Practice 3: Shift in Testing Workload</td>
</tr>
<tr>
<td>Iterative Testing Practice 4: Test More and Produce Better Quality Output</td>
</tr>
<tr>
<td>Theme 3: Proactive Handling of Changes to the Project Requirements</td>
</tr>
<tr>
<td>Iterative Testing Practice 5: Maintaining a Testing Discipline</td>
</tr>
<tr>
<td>Theme 4: Loosely Controlled Development Environment</td>
</tr>
<tr>
<td>Iterative Testing Practice 6: Developers Doing Testing</td>
</tr>
<tr>
<td>Iterative Testing Practice 7: Testers on the Team</td>
</tr>
<tr>
<td>Theme 5: Planning Kept to a Minimum</td>
</tr>
<tr>
<td>Iterative Testing Practice 8: Iterative Through to Final Testing Which is Linear</td>
</tr>
<tr>
<td>Theme 6: Enhancing Continuous Learning and Continuous Improvement</td>
</tr>
<tr>
<td>Iterative Testing Practice 9: Skills Needed For Agile Testing</td>
</tr>
<tr>
<td>Theme 7: Emphasize on Working Software Product</td>
</tr>
<tr>
<td>Iterative Testing Practice 10: Testing Governance Compliance</td>
</tr>
</tbody>
</table>

V. IMPLICATIONS FOR RESEARCH AND PRACTICE

The findings of this study offer several important theoretical implications. A major contribution of this study is the identification of eleven iterative testing practices which highlight activities to improve managing agile software development projects. These practices primarily involve testing issues and how agile teams succeeded in solving these issues through adopting certain techniques that fit the practices of software development teams. While prior research has established numerous generally accepted best practices, guidelines, and frameworks, few studies focus on highlighting the importance of establishing adequate testing practices across critical capabilities in order to mitigate agile development problems. Future research is needed to identify where proposed testing techniques work best and the complementary activities needed to promote their successful adoption and how these techniques change in structured versus iterative software development environments.

In summary, iterative development continues to expand. Within the context of software development, this study focused on testing paying special attention to the nature of what is being practiced and whether it has any impact agile team practices. This study has also paid attention to the challenges brought by iterative development practices and found that many of the practices used by teams involved techniques to improve the testing and quality assurance of product outcomes. Although we have understood the importance of software testing for many years, this study identified specific examples of implementations of better testing techniques in iterative development, offering several implications for theory and practice.

REFERENCES

Appendix

### Agile Practice Theory 1: Frequent Releases and Continuous Integration

**Iterative Testing Practice 1: Test Early and Continuously**

Project Manager at consulting company 1 - “Yeah do everything from a high level theoretical and then dive down deep from the core and evolve the software out and start testing early get it in front of the customer early, learn early, and be able to make very nimble, quick changes...[for customers]. One, you're showing them something quickly, and two, you're showing them that your quality assurance testing the data so early in the process...this helps build trust.”

Business Analyst at retail company - “Continuously building the system, continuously testing the system. So to me continuous integration is all those pieces coming together and constantly merging your code checking in your code constantly testing it. Everything is constantly changing and moving but it's a lot easier to make little adjustments as you go than big adjustments at the end.”

IT Lead scrum master at logistics company - “Because you find the problem upfront that’s what I love about the Agile methodology, you’re reviewing as you go and you're seeing what your output is as you go. And so you can figure out that a requirement wasn’t met early on, which saves a ton of rework and reccoding and which translates into dollars, right?...We, you know, one thing that we did end up having to do sometimes was when we would put the next sprint, we might need to put some defect fixes in the next sprint from the testing that went on in the previous sprint.”

IT Manager at logistics company: “Number two advantage that I am seeing is we are having the ability to improve the testing from the earlier phases of the development we are not typically. I mean in typical waterfall projects you are waiting till the last minute to start your testing. I mean once the project starts or enters the launch phase. Whereas in agile you are starting your testing from sprint one and continuing the testing through all the sprints so that’s allowing us to find issues and also helping development team resolve those issues early than the normal process. So from that point of view my preference is for testers or testing organization agile is proving to be more valuable than waterfall.”

**Iterative Testing Practice 2: Use Testing Tools**

Project Manager at consulting company 1 - “We do have some testing tools but we're not using any one tool in particular.”

Fig. 1. **Developer at testing retail company** - “Unfortunately we still have to rely on manual testing. And then I would say that has requested for years to start doing more automated testing, but there's a lot of challenges because we're using a certain product...So it is very difficult, it doesn't really work great. We've gotta try several times, but because I guess partially because of the architecture but these people are difficult to get any meaningful automation testing really fully working very good.”

Fig. 2.

Fig. 3. Developer at testing retail company - “I mean there are a lot of good tools, but again, because especially the technical product leads have to do so much firefighting that we really have enough time to focus on quality and improve quality. That's very, very bad and it affects everything, but it's very difficult to get out of that rut. But I'm hopeful we'll be able to get to that one with the shorter releases and the fact that we have a new manager. And I think he may be open to trying new things because he sees what's going on, right. So you know, sometimes it takes fresh eyes to see something.”

Fig. 4.

Fig. 5. **IT Lead at retail company**: “We implemented test-driven development, we have continuous integration, automated tests, automated testing. Automated testing has made a huge difference for us because it's freed up the developers. They had to run these test cases manually, it was taking forever. Once we started automating them then it kind of freed us up their time a little bit and we were able to pick up speed a little bit. It's been a long road to try to figure some of this stuff out.”

### Agile Practice Theme 2: Need for Frequent Feedback

**Iterative Testing Practice 3: Testing Shift Workload**

Business Analyst at retail company - “However, with agile you have a lot of burden put on project managers and testing. Right? Because project managers then have to deal with anything that didn’t make it into sprint I that fell into sprint 2, right? Documentation, your SOX compliance, all of that. And then, testing is constantly doing current sprint tests plus any defect fixes from the last sprint that made it into the next sprint are going into the third sprint. Right? So, it is faster paced and it works well, but I think those are some of the drawbacks that I see from time to time.”

Business Analyst at retail company - “The workload increases for the PM and the testers because they're constantly having to either shift back and revisit as well as the same time they're doing new functionality tests. So like with waterfall, you kind of have a deal where you go through and you do the whole testing cycle and then everything kind of just settles and then you go into production. And then you test any defects that come out of UAT that's already in production. Where with waterfall it's just constant, constant, constant, retests, new tests, new testing functionality.
**Iterative Testing Practice 4: Testing More and Produce Better Quality Output**

**Project Manager at logistics company** - "I think the better quality is frankly you are testing more often at more levels; it is much easier to put in a quality assurance program in an agile environment... when working in a linear fashion, people assume quality assurance is nothing but more than testing, I say No, it really isn’t and that gets down into Why. Service oriented architecture, modular infrastructure, all these pieces, like you say, that are loosely coupled, it works so much better; there aren’t that many dominions that fall off when you touch something."

**Director of Automation Services at logistics company** - "You know, [with agile] I can’t say that more work gets done. I will say that the quality of work tends to be higher... That seems to be a consistent pattern regardless for projects that fit the agile kind of mode correctly. We just finished one... That’s a brand new application that a small team wrote using agile but following our restrictions. There were a handful of defects that were done. It’s a very complex app, it’s a new app. So I think the quality is better. I think developers are happier, I think marketing is happier, I think all of those things. But I don’t know if the actual amount of work getting done has increased."

**Project Manager at retail company** - "The testing. I think the testing, the iterative testing throughout the project, is really a key to the success in the quality that you can come out with... We have testers on the team. We had two teams on the project that I worked on, one developed the GUI, you know, the front-end piece, and the other was doing the backend piece that talked to all the different systems. And we had testers on both sides. So they worked together. We tried different ways of doing testing. We did testing during the sprint, you know, as things were designed and coded and then tested in the sprint. And that didn't work as well because we found out we were trying to rush our coding, so we pulled testing out and did it in the following sprint for what was coded in the previous sprint, and that worked much better for us and then we made defects a priority so if they did come up with defects during the testing, those went to the top of our backlog and not just, you know, pushed off till later. And at one point, because we had so many defects, we brought in an individual just to handle defects on the team, and she was just part-time, you know, for a short period of time on the project."

**Project Manager at retail company** - "I think with Agile you know if the quality's there because you're doing your design and development right there at that time. Maybe you have a really well documented design document, but it's not what the customer wants, you get into the actual testing and production and the customer's like, "Well, that's not what I asked for... So you can have really good documentation and terrible code, or you can have really good code and terrible documentation. I mean, it can go either way."

**Agile Practice Theme 3 - Proactive Handling of Changes to the Project Requirements**

**Iterative Testing Practice 5: Maintaining a Testing Discipline**

**IT Lead Scrum Master at logistics company** - "[need to be] disciplined in their testing approach, because it is very easy to say we don't have time; we're not going to test that. And we don't do that. We make sure that we plan for testing first. We do use-driven development. Test cases are written from the requirements, and then they fail until they get it coded correctly and it passes. And I think within this agile thing, you have to be very disciplined about your testing approach because if not it'll all fall apart. You can put stuff out all day long. If it's not tested, that's the use? And we just made sure that quality... we affect the enterprise, and the quality has to be there or people won't trust us and they won't use us. So I think the team members recognize this and they're very much sticklers about testing."

**Agile Practice Theme 4: Loosely Controlled Development Environment**

**Iterative Testing Practice 6: Developers Doing Testing**

**Developer at software company** - "It's a little fragile. I mean, the whole team does the sprint. We do the, you know, sprint planning and retrospectives and demos and all those things. And we try to pick the right things off the backlog and do the right things in order. Um, so, all those bits of agile happen. Developers do all the unit tests, actually, our developers do, um, much more than unit tests. Our development teams are actually doing a lot of functional testing as well."

**Developer at software company** - "I thought of another team over at [another division in the company] who, um, they're agile but they don't do sprints at all. They work purely from a kanban board. ... that team actually did not have any testers. The development team did all the testing as well."

**IT Lead Scrum Master at logistics company** - "Mm, I'm thinking. No, no it wasn't because unit testing results have to be posted to [tracking tool], so I would say the developer, the developers would do their unit testing as they went through their sprints, and then their unit test results were submitted as you know, one ultimate artifact... Versus an end user you know, using an external facing product of [our company] or something. So really we used testers to do the testing I would say the integration test, the UAT was kind of combine."

**IT Lead Scrum Master at logistics company** - "Yes [developers fix defects]. Although in theory with the Agile methodology you should be able to hand it to anybody on the team to make those lines, right?"

**Iterative Testing Practice 7: Testers on the Team**

**IT Lead Scrum Master at logistics company** - "... we decided that we would try to go with agile. And it's worked really well for this team. We have integrated testers on our team."

**IT Lead Scrum Master at logistics company** - "It is hard to understand, but just know that requirements are started three iterations back or four based on how complicated the requirement might be or the story. And then we build those requirements... we build those requirements, and at the beginning of the iteration we discuss them, and then we finalize them during the iteration. And at the same time, while we're finalizing these results, coding's going on, unit test cases, then testing is testing. At the same time, it's all in a circular pattern... the tester assists with us with requirements; he is a member of our team. And so every planning session we have, every meeting we have, he's in there. Every stand up. And then we have some offshore model too, so he coordinates with the offshore people."

**IT Lead Scrum Master at logistics company** - "We started out with two weeks and then we went to three. And then we went to four, and the reason we went to four is we had a new feature that came in that had maybe 8 or 900 test cases that had to be run, and while we were trying to automate those -- we still haven't got them all automated yet -- we lengthened the length of the iteration to accommodate the testing. We do a
hardening iteration right before we release and we do maybe 10,000 test cases, some of them automated, some of them have to be run manually. And then our testers run a bunch. In the last two years, we’ve had three production defects."

**Director of Automation Services at logistics company** - "The thing I like about agile is I like marketing, development, testing, requirements writing all sitting together, all working through the project in a single area and rapidly solving problems and moving forward. That I do like."

**Director of Automation Services at logistics company** - "Oh, I think it’s A) developers understand the requirements much better because they’re sitting there with their marketing partners I think testing understand the requirements and so the test cases are relevant and spot-on. I think those are really kind of… everyone understands what we’re trying to do, so the developers know without looking at the requirements, even though you’re kind of writing them, what needs to be done and testing those test cases that need to be run. And so I think you just get better quality. And happier people at the end of the day I think are going to write better code."

**Project Manager at logistics company** - "I think agile from a testing point of view is a huge benefit to our organization because you are getting engaged in agile projects. Typical waterfall projects, because you have these business & development community typically coming to the testing teams towards the end or typically late. With agile as you are collocated with the business & development teams as you are getting engaged in the daily calls their conversations and you are more engaged than the typical waterfall projects. So from that point of view you are more aware and you can plan your activities more appropriately, that’s the number one advantage."

**IT Lead SCRUM Master at retail company** - "We had testers on our team. We had two teams. One was the client team and one was the backend team. So we had testers doing both testing functions. But having them actually on the team is critical, I think, to the success of the testing aspect of the project. We have an independent testing group usually what you do is you do all your code and design. Testing is kind of involved in the project, but they really don’t get heavy into it until we turn it over to them to do the testing. So this was a little different to have testing integrated in the Agile team, and I think that that was important. But they were contract resources, you know, we off-shored them, and we actually brought them on site to work with us in Dallas because it’s a little more difficult to do if they’re in another country or another site or something."

**IT Lead Scrum Master at logistics company** - "We had testers involved as part of the team, as part of the Scrum team, they did testing you know, they did testing as we went and they built regression tests, and they automated some of that. We were trying to get into a continuous test model, but we didn’t, we didn’t quite make it."

**Agile Practice 5: Planning Kept to a Minimum**

**Iterative Testing Practice 8: Iterative Through to Final Testing Which Is Linear**

**Project Management at retail company** - "When we have a lot of impacted systems that have to interface with each other, that works better with Waterfall. And that was one of the other things we had to force fit on my project. We have what’s called an integration, testing date, and everybody comes together, all the systems come together, and you start doing your testing together at that point. So with our Agile project, the one I mentioned earlier, you know, we found ways to meet their deliverables for the system requirements where we could get approvals at various times, or whatever, but we really couldn’t get around having to come together on that integration testing date at the same time because the backend systems weren’t available for us to test earlier in our Agile cycle. And, actually, 50 of the 59 defects we went to production with were on one system because they couldn’t come to meet, you know, participate with us until integration testing. And so once we get integration testing that’s where we ended up with a lot of defects. So if they had been able to participate earlier, but we had to have environment set up and we have to have resources and, you know, everything has to kind of come together at one point."

**Business Analyst at retail company** - "Developers [do the unit testing] So, developers are responsible for their own unit piece. And then it gets passed down to testing unit....not traditional as waterfall, but the role of testing kind of stays the same, right? No matter what, they’re always going to receive, they’re always going to say okay, we have code out there ready for you to test. You’re going to get to a point where you have to take your code that the developer has and push that into your test environment. No matter if it’s waterfall, Agile, whatever methodology you’re using, there’s still that handoff that has to happen. So once they get it, it’s business as usual for them. They’re going to go through and test all their scripts. Now, with an Agile, right, this is where you get into, it’s a quicker process."

**Developer at software company** - "Whereas, individually, all those teams are, you know, on their own product backlogs and doing, um, um, you know, a lot of the agile practices. Um, but we have some, uh, some stabilization time kind of built into the progress process because we can’t, we have a hard time keeping those things all working together all the time....we do a little bit of; yeah, a little bit of stable, yeah, a little bit of testing quality in at the end sometimes....looking for the integration issues that happen when we start plugging all those things together. But when you’re starting to put together, you know, basically, you know, a million or two lines of code worked on by, you know, hundreds of engineers, um, making sure all the gliches that, uh, happened due to misconmunication or to mis-implementation, all those things that are, uh, actually try to plug all those pieces together, it all gets straightened out."

**Project Manager at logistics company** - "...about my agile projects, I am primarily doing you know functional testing & performance testing so when it comes to end to end testing even these agile projects that are getting developed following their own timelines and things like that ultimately have to come together during a [final test step] so that we can do end to end testing. ...We cannot do end to end testing as part of agile at least right now we are not equipped to test end to end testing during agile development phases....you know we have several applications that need to come together, right, and all the applications should integrate at one point of time, one period of time in order for me to perform end to end testing. Following agile you can have your own schedule which differs from the application that you need to integrate with right you can just scrub some data have some mock responses or whatever and make progress with your development effort but the actual integration may not happen during your agile process."

**IT Lead Scrum Master at logistics company** - "I think it’s because of the division of duties here. Because we have an organization that all they do is test, and they are not Agile. I mean, I think they would like, I think it would be, they would like to. And they do provide for Agile teams, like we have a issue that’s assigned to us from that team and then the interfaces with the offshore developers, you know, we’ll line up to a point and then when we get this done, we’ll churn it out and send it over to the testing team."
<table>
<thead>
<tr>
<th>Agile Practice Theme 6: Enhancing Continuous Learning and Continuous Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iterative Testing Practice 9: Skills Needed For Agile Testing</td>
</tr>
<tr>
<td>IT Lead at logistics company - Well, I think as long as they could develop — they had some skills in development, they had some skills in writing requirements, they had some skills in testing, that those soft skills would be the more important ones that would come in to a team member. The ones that are able to work with others well, that are... it's just, I think those soft skills that teams have, those are more important in an agile team. You don't have to necessarily be the strongest developer and you don't have to necessarily be the strongest tester, but you have to be able to work with your team members well.</td>
</tr>
<tr>
<td>Agile Practice Theme 7: Emphasize on Working Software Product</td>
</tr>
<tr>
<td>Iterative Testing Practice 10: Testing Governance Compliance</td>
</tr>
<tr>
<td>IT Lead at logistics company - &quot;We don't report back to governance group. They may come audit us later. But we have a tool here that monitors where we are in the steps. So we just have to make sure that that tool, we sign off on one before we sign off on the next one. But there can be a one second difference on when we sign off on one to the next one. At the end of the iteration we sign off on requirements then we sign off on unit test cases and unit test results then we sign off on functional testing then we sign off on the release. They can be one second between those as long as they're done in order.&quot;</td>
</tr>
<tr>
<td>Director of Automation Services at logistics company - &quot;And within my company there are a lot of — and you know some people to take the agile approach in that they'll keep going and they'll be making changes into integration testing and that just causes complete chaos. So what we really have is we have a lot folks who are doing waterfall and then you have some people who are doing agile. Well depending on what that agile group is delivering, what that agile group is delivering. And if they're changing it up at the last minute that makes life hell. It is like a change? We changed the data definition of this interface. In the middle of testing. Well why? Well, we needed to change it because we wanted to do this... we’re just like that’s something that should’ve been defined six months ago. You can’t change that now. So I think throughout my company people like to talk about we’re doing agile, but I think what we do is really like I said: it’s waterfall or iterative with agile-like features.&quot;</td>
</tr>
<tr>
<td>IT Manager at logistics company - &quot;It was more of a iterative mode than pure agile in theory, every time you go through an iteration, you can release something to production and [my last company] was not anywhere near mature enough and frankly, [my current company] is a leader except in some unique cases. So you do your iteration and the development, the requirement generation and the development space and when you are ready to go to production, and then you go back in the linear mode then transition production stuff... for us, the final stage is final transition production stuff where you do your final testing and signoff that everything works, final user acceptance testing because you are doing bits of UAT all over through it, from a governance perspective. Again, one of the things that will start talking when you say what are the hardest things about making the transition is most companies are, their governance processes are linear and that’s the biggest issue we have at [my current company]. How do you let agile be agile and still interface with linear governance systems, I mean if everybody in the company is agile then that’s fine but it’s not how it works [at my current company] when maybe 8% of the projects were Agile at the current time and maybe [my old company] it was maybe about the same and so it’s more of a governance perspective and compliance perspective and make sure you touched all the basics from a governance perspective to release something into production.&quot;</td>
</tr>
<tr>
<td>Director of Automation Services at logistics company - &quot;We’ve adjusted our process to support agile requirement definitions. At the end of the agile process, the way that kind of overcome it is basically hey, look, you can be the wild west for as long as you want but at the end of the day before you go into testing you need to provide us all of those documents I mentioned all signed off. So everything needs to be captured; you need to take your 80 billion sticky notes off the wall and actually write a document that can be captured in our requirement system so we know what you built... I don’t think compliance is really an issue. I think what’s really an issue is we don’t have that many applications that are isolated enough to be able to do agile safely and like I said budget and testing and delivery timelines kind of screw a lot of the things that make agile... when you deliver twice a year, what are you being agile about?&quot;</td>
</tr>
</tbody>
</table>
Developing Project Development Sequence

Michael Racer  
University of Memphis  
Marketing and Supply Chain Management  
Department  
Memphis  
mracer@memphis.edu

Mark Gillenson  
University of Memphis  
Management Information Systems  
Department  
Memphis  
mgilnnsn@memphis.edu

Abstract - One of the challenges of creating a successful project in agile development is the identification of the sequence for developing modules within the program. The challenge here is that the difficulty of finding the best solution is a function of the number of modules to be created, giving rise to a NP complete problem.

Keywords: agile, optimization

I. Introduction

We begin by thinking about the structure of the testing environment. The team begins with a set of modules that need to be programmed and a specified time interval for completing the project. The agile environment takes a different approach to this scenario by emphasizing an iterative approach to the process. This is applied across the environment and puts an emphasis on the possibility for revisits to modules that have been developed and need to be redeveloped.

In this study, we focus on the initial development of an application. Our goal is to assist the team in developing a story sequence for creating the software package. The issue here is to identify a starting structure for the project and identify an initial plan for when each module is to be started.

The concept of telling a “story” as the development continues is a challenge in that we are seeking to develop a sequence that ensures that all members of the team - from developers, management, all the way to the customer - are well-informed of the progression of the product.

II. Support material

We can visualize this problem as a network. Each node represents a task to be completed. And associated with each task is a completion time. And then an arc between two tasks indicates that there is an influence from one task to another.

The properties of networks are well studied in the operations research literature. Two particular problems are important with respect to the agile problem that we are studying. These two factors are that there are values associated with each node, and each project has to be completed. This gives rise to two sets of literature in operations research, studies -
the idea of “covering” and what is known as the
generalized assignment problem (GAP).

This background was used to develop a
procedure for creating a quality solution to the
recognition of a satisfying agile problem.

III. Method development - literature
background

The basic methodology of the agile system is
well detailed in the text by Crispin and Gregory.
This text was used as a reference with respect
to the major agile issues.

Recall the 10 principles in the agile
environment:

- provided continual feedback
- deliver value to the customer
- enable face-to-face communication
- have courage
- keep it simple
- practice continuing improvement
- respond to change
- self-organize
- focus on people
- enjoy

In order to generate a sequence for telling the
stories we modified a technique developed by
Racer and Amini for the GAP problem. This
technique relies on several operations research
techniques in order to develop a final solution
that is a quality solution and identified in a very
timely manner.

IV. Example

Agile Development Optimization Problem

The following example shows how the user can
represent his environment for the agile problem
and use the methodology to identify a low-cost
solution framework environment. We begin
with identifying the structure of the
environment. We have a set of relationships
between components of the project, and we
recognize the costs that are involved in
scheduling these.

Consider the following specifications for an
agile software development problem.

1. There will be four iterations, numbered
   1-4.

2. Each iteration is capable of producing
eight man-weeks of software
development work. (Each iteration
lasts a fixed three actual weeks but this
will not enter into the calculations.
What is important is the number of
man-weeks of work each iteration can
produce.)

3. There are ten requirements for the
   application.
   a. Each requirement has a “cost”
      measured in the number of
      man-weeks to develop it.
   b. Each has a priority or criticality
      factor measured on a scale of 1-
      5, where 1 is the most critical
      and 5 is the least critical.
   c. Certain requirements must be
developed before certain other
requirements can begin to be
developed. This will be indicated by a
mandatory ordering factor. Other
than these mandatory ordering
factors, the requirements are all
considered to be independent of each
other (i.e. the requirement letters are
just labels and are not meant to imply
an ordering.)
The requirements and their numeric factors are shown in the following table:

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Cost</th>
<th>Priority</th>
<th>Mandatory Ordering</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>3</td>
<td>1</td>
<td>A must be in iteration 1.</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td></td>
<td>Priorities 4</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>4</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>2</td>
<td>1</td>
<td>F must be done before G and J</td>
</tr>
<tr>
<td>G</td>
<td>2</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>3</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>4</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>J</td>
<td>2</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

Cost equal to

Priority sum 76

Solution after first modification
Cost = 74

Solution after second modification
Cost = 68

The goal is to decide which requirements should be developed in which iterations while:

1. Respecting any mandatory orderings stated.
2. Making sure that the amount of work assigned to each iteration does not exceed its capacity.
3. Trying to maximize the priority factor. This means trying to maximize the sum of the products of the iteration numbers times the priority factors of the requirements developed in them.

Starting with an initial solution that was just random

A1
B1
C1
D2
E2
F1
V. The Next Step

The basic goal in this step was to develop an algorithm for solving the baseline problem. The next step will be to modify this approach to take into account such issues as:

- Quality (testing)
- Developer feedback
- Customer feedback

In addition, because of the diversity of the individuals involved in the process, from developers to management to customers - this also gives rise to the likely existence of competing objectives. The sensitivity of this environment will also be addressed and tested as well.

The format of the algorithm currently in place lends itself to being readily adaptable to a feedback environment.

We are currently in the process of developing a matrix-based tool that will accommodate the interests of each of the various parties into the development of the storyline and the sequence development.

This tool is being developed in a way that recognizes the various users and the inter-relationships between them. Because these different contributors are necessarily equal, the technique will be developed with some sensitivity capabilities to allow for an assessment of the trade-offs between accomplishing the requests of the various members in the system.

VI. References


AN EVALUATION OF AN OPEN SOURCE LOAD TESTING TOOL BASED ON THE JAVA LOAD TESTING FRAMEWORK

Manish Shrivastava
Ted Lee
Son Ngoc Bui
Jasbir Dhaliwal

Objectives

• Load test a multi tier web application
• Use Open source testing tools to avoid licensing cost
• Automate the testing scenarios so that it can be reused
• Repeat the testing with various load conditions
• Observe the performance and CPU utilization of various tiers of application
Web Application Testing Overview

<table>
<thead>
<tr>
<th>Testing Activity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance Testing</td>
<td>Verify system performance such as response time, speed, scalability and stability</td>
</tr>
<tr>
<td>Load Testing</td>
<td>Verify web application behavior under normal and peak load level</td>
</tr>
<tr>
<td>Stress Testing</td>
<td>Evaluate web application behavior when it is pushed beyond normal and peak load conditions, and reveal defects under heavy load conditions</td>
</tr>
<tr>
<td>Compatibility Testing</td>
<td>Uncover failures due to different web browsers or configurations</td>
</tr>
<tr>
<td>Usability Testing</td>
<td>Identify issues centered on user interfaces</td>
</tr>
<tr>
<td>Accessibility Testing</td>
<td>Verify the accessibility to the content of web application</td>
</tr>
<tr>
<td>Security Testing</td>
<td>Verify the effectiveness of web application against attacks outside its environment or unauthorized access resources of web application</td>
</tr>
</tbody>
</table>


Web Application Testing Overview

- Two types of web application testing:
  - The functional requirement
  - The non-functional requirement
  - Unique to web application testing
Web Application Background

Transportation of Products:

Delivering Products when you need it where you need it.
Factory to Warehouse
Warehouse to Market
Supplier to Factory etc.

3 PL Business Process Related To Systems

A typical 3PL company provides multiple logistics services for use by customers. These services may include transportation, warehousing, cross-docking, inventory management, packaging, and freight forwarding etc.
Core Functions of Application

- **Transportation Management System (TMS):**
  - Delivering transportation planning and execution solution
  - Shipments visibility, tracking and tracing features
  - Carrier visibility
  - Freight payment
  - Used by shippers or third party logistics providers
  - Can be used Software-As-A-Service model
  - Reduce transportation cost and asset utilization
  - Web enabled application
Why Grinder

- Open source software that is scripted by Jython
- Software customization
- Code reusability
- Fast deployment
- Simplicity in running test cases
- Load test independency for programmer

Functional Test Cases Classifications

- Type 1: test fundamental 17 functional test cases.
- Type 2: test Load Planning Center user
- Type 3: test Visibility User
- Type 4: test both Load Planning Center user and Visibility User
  - 6 test cases were used with varying percentage for type 2, type 3 and type 4
Type 1 Test

- Each of 17 functional test cases performed one at a time with three different user levels, 1 user, 50 users and 100 users, by doing the same steps simultaneously
Type 2 and Type 3

- 50 users used in order to simulate users doing different tasks.
- 6 test cases were used.

<table>
<thead>
<tr>
<th>Test Identifier</th>
<th>Test Case Description</th>
<th>User Proportion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test 5</td>
<td>Users performing a manual shipment creation</td>
<td>10% 5%</td>
</tr>
<tr>
<td>Test 7</td>
<td>Users performing a shipment tender</td>
<td>40% 5%</td>
</tr>
<tr>
<td>Test 21</td>
<td>Users performing a manual order base creation</td>
<td>10% 5%</td>
</tr>
<tr>
<td>Test 35</td>
<td>Users performing a shipment inquiry</td>
<td>20% 30%</td>
</tr>
<tr>
<td>Test 38</td>
<td>Users performing shipment document creation</td>
<td>6% 5%</td>
</tr>
<tr>
<td>Test 39</td>
<td>Users performing operational report generation</td>
<td>4% 50%</td>
</tr>
</tbody>
</table>
Type 4

- Five complete runs of 50, 100, 200, 300, and 400 users.
- A test with mixed scenarios that include loads from both previous two tests, Type 2 and Type 3,
Comparison of Test Processing Times for Four Different User Levels
Conclusion

- Two important and critical performance issues found
  - The connection issue between the Web Servers and Application Servers
  - No activation of the second Database Server
- Open source testing tool can test complex web-based application system.
  - Very cost-effective and useful to identify the potential problems before actual production deployment
Thank you

Q & A
INTRODUCTION
This paper presents a body of work that identifies and validates the critical dimensions required in developing a QA strategy for mobile applications. The content that makes up the sections are drawn from Mphasis' extensive experience in developing and testing mobile applications.

NEED FOR MOBILE TESTING
Mobile usage is rapidly growing and is a key channel to further economic growth around the globe for various industry types—noteable ones include Retail, Media, Logistics, Insurance, Education, Healthcare, Finance, Games and Social. This means that mobile devices are rapidly becoming the primary method of interaction for consumers and businesses worldwide. According to Forrester Research, 95% of enterprises have already deployed or are in the process of evaluating wireless e-mail, and approximately 33% of North American and European businesses have implemented sales force automation mobile applications.

TESTING GOALS
Our testing goal shall not be limited to finding errors but to provide a high quality product meeting the needs of the end user. There are unique challenges when it comes to testing and many a times specific testing types may not satisfy all of the user expectations. A combination of any of the below test types forms the basis of our mobile testing strategy.

TEST TYPES
A. Usability and User Experience Testing covers functional verification, screen navigation, landscape-portrait view and toggle feature. GUI including the overlay verification, text visibility, pinching, zoom in/out, scroll and tap features are also covered here.

B. Platform Compatibility Testing includes different OS-Device fragmentation verification across various screen sizes and pixel resolution.

C. Field-Network Testing covers localization testing (content, dates, chars, direction, postal codes, phone numbers)—Support features that are unique to specific markets and carrier specific testing.

D. Services Testing acts as a checkpoint to verify that mobile app do not act as a server.

E. Performance Testing provides the response times, battery consumption, memory leaks information and code optimization.

F. Security Testing to protect sensitive data communication, block unintended user and prevent malicious content from entering the device.

F. Other Exploratory Testing includes:
- Accelerometer response
- Retrial / Signal Loss Testing
- Sleep mode response
- UI Response in case of lighting conditions
- Recovery testing in case if battery goes down and data lost while app upgrading
TEST MODELS

A. Native Apps
Native app is an application program that has been developed for use on a particular platform or device. It is directly installed on a mobile device or can be downloaded from a public or private app store and then installed on the mobile device. These apps can take advantage of OS features and other software that is typically installed on that platform.

B. Mobile Web/ Web Apps
Web applications are developed for multiple platforms and not installed locally but made available over the internet through a browser.

C. Hybrid Apps
Hybrid app combines both native app and web applications. It makes use of an embedded browser to improve access to the online web content. It integrates with device file system and web-based services.

TEST SOLUTIONS

A. Real Time Devices
- Realistic result capture
- Complete test coverage achievable
- Perform all test types possible
- Huge cost involved

B. Emulators
- Cost-Effective solution with high return on investments
- Leverage existing infrastructure
- Complete test coverage not achievable
- Usability experience cannot be tested including device interruptions

C. Cloud Based Approach
- Can be implemented via Physical device or Emulators using web interface
- Cost effective to an extend
- Parallel tests can be run
- Device logs directly available for troubleshooting
- Portability across different platforms

D. Automation Tools
There are several automation tools available-Open source and paid ones. Some are listed below:
- Device Anywhere
- Monkey Talk
- Perfecto Mobile
- Selenium
- Silk Mobile
- Experitest
- Egg Plant
- SeeTest

Listed below are the essential automation tool requirements:
- True object identification
- Script portability across different platforms
- Support both device and emulator types
- Support native, hybrid and web applications
- Scheduled execution preferences
- Avoid Rooting/ Jail breaking devices

QA CHALLENGES

A. Diversity of the device environment includes
- Cross Browser Platform compatibility
- Operating Systems and its supported versions
- Device manufacturers

B. Hardware configuration and network related challenges
- Multiple network types
- Multiple carriers
- Connectivity speed variation
- Differences in communication protocols

C. Rapid Application Development(RAD)
- Impacting test execution across different devices and browsers

D. Soaring Expectations of Users
- Performance
- Usability
- Security
- Data Usage, Installation, Launch
### MOBILE TEST MATRIX SHEET

<table>
<thead>
<tr>
<th>Device Manufactures</th>
<th>Apple, Samsung, Nokia, HTC, Sony, Motorola, Blackberry, Windows</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS Versions</td>
<td>iOS: iOS 4.x, iOS 5.x, iOS 6.x, iOS 7.x</td>
</tr>
<tr>
<td></td>
<td>Android: 2.x, 3.x, 4.x</td>
</tr>
<tr>
<td></td>
<td>BlackBerry: BB 4.x, BB 5.x, BB 6.x</td>
</tr>
<tr>
<td>Pixel Resolution</td>
<td>640 x 1136, 1080 x 1920, 768 x 1204, 540 x 960</td>
</tr>
</tbody>
</table>

### Android Tools
- Android SDK Kit
- DDMS for screen capture
- Checksum tool for file size comparison
- System Panel Lite Tool to pull logs
- Appgrader tool for app comparison
- Testflight App

### iOS Tools
- Xcode 4.7
- Mac OS 10.7 or later
- Testflight App
- iTunes

### BB Tools
- OTA Downloader
- BB Launcher Kit with MDS Services

### Language Testing
- Locale Select App
- MoreLocale2

### Build Installation Techniques
- .APK installation via ASTRO File Manager/Download
- .APK installation via command prompt
- .JAD installation via OTA downloader
- .IPA installation via ITUNES

### Simulator n/w settings
- Command Prompt: cd <Android Tools Path> emulator – avd <emu name> - dns server <I.P> - <Proxy url.getPort>

### REAL TIME TESTING TIPS/ OBSERVATION
- Categorizing DUT within pre-determined pixel density ranges.
- Adopting major market share and min/max strategy
- Duplicating the bug in the desktop browser just to ensure it being a device specific bug rather a bug in the basic application
- Credential passing while integrating with multiple test models
- More occurrences of device bug being noticed while scrolling (embedded PDF pages within browser), text selection, list box selections, calendar pop-up, embedded back button navigation.
- Ensuring keyboard/touch-screen functionality is clear and consistent.
- Checking for data caching issues
- Checking for all links within responsive pages to ensure it does not open to full site with the native app
- Checking whether the native app stores or delete saved data properly
- Test like a customer approach
- Ensuring backward compatibility of each older generation of the device/OS for native apps.
- Screen minimize/maximize inconsistency for flex pages within browser
- Occurrences of static header sub-menu items within Safari browser for iOS

### MOVING TRENDS
- Mobile web expansion over native apps
- Launch of more hybrid apps to the stores
- Agile methodology in software lifecycle

### STATISTICS REFERENCE:
http://www.smartinsights.com/mobile-marketing/mobile-marketing-analytics/mobile-marketing-statistics/