**Title:** Unit Value of a Test Case

**Authors:** Mark Gillenson, Yao Shi, Xihui Zhang, and Tom Stafford

**Abstract:** Nowadays, software testing is becoming more critical to ensure that software will function properly in the production environment. Consequently, the effort, time, and fund invested on software testing activities increase significantly. As such, managers have to allocate the limited resources of an organization to those critical test cases rather than the exhausted set of test cases. This paper builds a value function for quantifying the unit value of a test case, which uses the value concept, presents as a quantified index, and can be used to single out the key test cases. We take three steps to establish the function. Firstly, we built an initial value function based on our understanding and knowledge as well as the relevant research. Next, we interviewed professionals and managerial staff who are working in the testing department or relevant positions to comment on the initial value function. Finally, based on the feedback of the interviews, we incorporated new factors emerged from the interviews and modified some other factors in the initial function which have different meanings according to the viewpoints of the interviewees. Through these steps, we established the final function, which includes tangible factors such as cost of a test case and amount of use, as well as intangible factors such as internal risk, external risk, and technical risk.

**Title:** Testing and Collaboration on a Mission Critical and Complex System

**Authors:** Melanie Hopwood

**Abstract/Introduction:** At Asynchrony we’ve had to learn to crawl, walk, and run over and over as technologies and our customers have changed throughout the course of the development of the Mobile Field Kit (MFK), one of our government projects. Our project is complex with many moving parts; it’s got multiple technical stacks, changing customers, lots of unusual equipment, lots of developers, and, with this being a software development company, we have had a good number of developers who have cycled in and out of the project team. We feel we have proven ourselves at being really good at changing and adapting. The success of this project has been due in large part to its ability to remain flexible, continuing to meet our customer needs as they evolve, and the creation and achievement of high quality processes. With all these moving pieces, how do we assure we are creating a high quality product and advocating for quality throughout our processes? We are able to do so because of our ability to combine the best traditional testing practices with unconventional quality processes to help our users succeed in their missions.

**Title:** Software Testing and Cyber Security

**Authors:** Robin Poston and Nicole Samson
Abstract: In December 2013, Target Stores’ systems were breached with over 40 million customer credit cards and other personal information reaching the hands of cyber-criminals. If Target had performed more exhaustive testing of their enterprise systems, it is possible they could have caught and fixed some or all of the vulnerabilities that led to the intrusion. What we learn from Target’s loss is that all companies should be testing for security vulnerabilities. As security breaches become more frequent, it becomes increasingly important to protect data systems and ensure they stay continuously protected. Yet, how do we know when our systems are protected? More importantly, how do we effectively test system security? By thoroughly testing the security of the system, we can potentially prevent security breaches from infiltrating our systems. Initially, we conducted a review of the scholarly and best practices literature to address these questions. Through the review, we created an initial draft of a comprehensive matrix, which we call the Security Testing Coverage Matrix, illustrating which test procedures may help with early identification of security vulnerabilities. The goal is to identify potential security weaknesses that may be fixed prior to being breached. The Matrix will then be presented to a panel of experts who will offer feedback for further refinement. The Matrix is useful to security and testing experts in the field who may utilize the Matrix in making testing coverage decisions about new and existing system tests to prevent data breaches. This Matrix is offered to help organizations formulate a comprehensive cyber security test plan.

Title: Formal Methods and Optimization of Software Test Cases

Authors: Rodrigo Silva-Lugo

Abstract: A comprehensive approach to software test case optimization is proposed, including a review of optimization techniques, and their application to three problems: generation of test cases, selection of the test cases, and prioritization of the execution of test cases.

STEP 2014
Journal Articles

Title: The Impact of Software Testing Governance Choices


Authors: Xihui Zhang, Colin Onita, and Jasbir Dhaliwal

Abstract: Software testing is becoming a critical component of software development, especially because of the proliferation of complex, interconnected, and real-time business applications. As a result, information technology (IT) managers are struggling with pragmatic governance mechanisms for integrating testing with development. Governance issues pertaining to how software testing is organized at strategic, tactical, and operational levels, however, have not received adequate attention in the literature. This study explores the impact of three specific governance mechanisms, including the existence of a distinct corporate testing unit, developers and testers reporting to different executives, and the existence of one-to-one matching between developers and testers, on the organizational
integration of testing with development. A national survey of 196 software development and testing professionals was undertaken to investigate the impact of these governance variables on a set of dependent variables comprising organizational, group, and individual outcomes. The results indicate that these governance mechanisms have significant impacts and need to be considered for successful integration of development and testing.

**Title:** Sources of Conflict between Developers and Testers in Software Development


**Authors:** Xihui Zhang, Thomas Stafford, Mark Gillenson, and Gertrude Moeller

**Abstract:** Conflict between software developers and testers is inevitable. Prior work on such conflict has focused on conflict management but not on its prevention. We addressed the research question: What are the sources of interpersonal conflict between developers and testers in software development? To attempt to answer this, we collected and analyzed 50 developer–tester conflict scenarios from professional software developers and testers. Our results indicated that the sources of conflict fell into three major areas: process, people, and communication. Conflict sources are presented in a category–subcategory–description format. Implications for theory and practice in the proactive management of conflict sources are discussed.

**Workshop Articles**

**Title:** Unit Cost of a Functional Test Case

**Authors:** Mark Gillenson, Thomas Stafford, Xihui Zhang, Jasbir Dhaliwal, and Yao Shi

**Abstract/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. Actually, 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.

**Title:** Automated Test Case Generator

**Author:** Chuck Morgan, Son Bui, Mark Gillenson, and Euantae “Ted” Lee

**Abstract/Introduction:** Every newly developed product of any kind has to be tested to ensure that it correctly performs the functions for which it was designed. This is true of an airplane, an oil refinery, or a washing machine. It is also true of software, whether it is systems software or application software.
Software can vary from simple applications to very complex applications and systems, and it all must be tested. The fact that in today’s business environment, software applications have to function on or be accessed from panoply of devices ranging from smart phones to mainframes (each with a variety of operating systems and versions) multiplies the testing problem considerably.

**Title:** W-Model Testing: Software Validation Using Requirements Simulation

**Author:** Jong Seok Lee

**Abstract/Introduction:** Software validation is a critical element of software testing. Validation is often referred to as “are you building the right thing?” [1], as it is centered on evaluating whether a product, service, or system meets the needs of customers or stakeholders [2]. Software development can fail when a software product does not meet the needs of users, even if the product is built according to specifications.

**Title:** A Procedure for Ordering Requirements in Agile Development

**Authors:** Michael Racer, Mark Gillenson, and John Dugan

**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 based on the stated requirements. The challenge here is that the difficulty of finding the best solution is a function of the number of modules to be created plus other factors, giving rise to a NP complete problem. We will start with a simple example.

**Title:** Software in Medical Devices Considerations in Design and Testing

**Authors:** Vaneet Singh and Jashbir Dhaliwal

**Abstract:** Software has evolved as a critical aspect of the medical devices. Apart from being used in making decisions or collecting clinical data, software is now used to control the medical devices. However, malfunctioning software in the medical device can cause life threatening situations. The software development community has been facing complex and ever increasing design challenges to harmonize software in or as medical devices while mitigating any associated risks. This paper highlights the importance of an effective software design along with validation considerations within the framework of IEC 62304:2006. The key concepts (design and testing) along with challenges in the development of software in or as medical devices are explained in this paper.

**Title:** Testing of Software Designs in Health Bio-medicine

**Authors:** Son Bui and Robin Poston

**Abstract:** Testing of software designs in health bio-medicine has been come central to the success of developing bio-medicine devices. The current software testing literature has very limited knowledge on the factors influencing software designs in health bio-medical devices. Draw from socio-technical theory, our literature review suggests that (1) social factors, and (2) technical factors affect the quality of testing
software designs for bio-medical devices. Implication for theory and practices are also discussed. Key word – software design, software testing, socio-technical theory, health biomedicine.

**Title:** Medical Device Systems and 510 (k) Submission Testing  
**Authors:** Robin Poston, Seungho Choi, and Aneta Dziemianczyk  
**Abstract:** The purpose of this paper is to clarify some of the uncertainties and explain how the process of filing and submission of the 510 (k) application works. It provides the reasons of application rejections and delays and gives guidance how to avoid the mistakes that can cause those rejections and/or delays. It also proves that adding testing step into the process could improve it significantly.

---

**Title:** The Impact of Conflict Judgments between Developers and Testers in Software Development  
**Authors:** Xihui Zhang, Jasbir Dhaliwal, Mark Gillenson, and Thomas Stafford  
**Abstract:** The primary role of testers is to verify and validate the software produced by developers to ensure its quality. Testing is designed to catch problems in the software and report them for correction, so it is a conflict-laden, confrontational, and judgmental process. This "audit" role of testing is inherently adversarial, ensuring the development of components of interpersonal conflict judgments between developers and testers. Prior research indicates that such conflict is likely to be negatively associated with software quality and job satisfaction, producing negative judgments about the artifact production process and about the job itself. This study addresses the question: How do judgments of conflict between developers and testers impact the software development process? The authors develop and empirically test a research model which proposes that the conflict judgment targets of both the tasks and the persons who perform them will have direct impact on both software quality and job satisfaction judgments. Results of testing this model indicate that interpersonal judgments arising from conflict, as well as judgments made by testers and developers about the conflict targets of tasks and persons negatively influence subsequent software quality and job satisfaction judgments. Implications for theory and practice are discussed.

---

**Workshop Articles**

**Title:** Determining the Unit Cost of Test Cases  
**Authors:** Mark Gillenson and Xihui Zhang
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 is executed. This study is the beginning of a research project to gauge the unit cost of functional test cases.

Title: Testing for Tautology based SQL Injection Attack using Runtime Monitor

Authors: Ramya Dharam and Sajjan Shiva

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 firewall, 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 numbers of both legitimate inputs and illegitimate tautology based attack inputs, and measured the performance of the proposed technique.

Title: Testing Practices Iterative Software Development Projects

Authors: Robin Poston, Jignya Patel, and Jasbir Dhaliwal

Abstract: Focusing on quality software testing practices has become central to the success of iterative software 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 interview 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.

Title: Developing Project Development Sequence

Authors: Michael Racer and Mark Gillenson

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.
**Title:** Mobile Testing QA Strategy

**Authors:** Arun Bahuleyan Leela and Gary Narvid

**Abstract/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.

**STEP 2012 Journal Articles**

**Title:** The Business Rules Approach and Its Effect on Software Testing


**Authors:** Thomas Meservy, Chen Zhang, Ted Lee, and Jasbir Dhaliwal

**Abstract:** Codification and testing of business rules in application programs has historically been a challenge in software engineering. Many organizations have adopted the business rules approach to formalize and compartmentalize business rules as a separate component from application code. This article investigates and presents the effects of the business rules approach on testing activities in the software development life cycle at a Fortune 500 corporation. The findings suggest that the business rules approach has the potential to engage testing personnel early in the development process and to improve the efficiency and effectiveness of testing activities.

**Title:** Understanding Antecedents of Interpersonal Conflict in Information Systems Development: A Critical Analysis


**Authors:** Gertrude Moeller, Xihui Zhang, and Sandra Richardson

**Abstract:** Interpersonal conflict in information systems development (ISD) projects is common and has been identified as being detrimental to project outcomes. Prior research has primarily focused on assessing the impact of interpersonal conflict on ISD project outcomes. As such, little is known about the antecedents to conflict in ISD contexts. It is proposed here that understanding the conditions that lead to the manifestation and escalation of conflict in ISD contexts is essential to improving ISD project outcomes. The goal of this study is to address the gap in the literature related to the understanding of antecedents of interpersonal conflict in ISD contexts. Specifically in this study, we integrate existing interpersonal conflict theory with current ISD theory to develop a theoretical foundation for an exploratory case study aimed at identifying antecedents of interpersonal conflict in ISD contexts. This study makes a contribution by: (1) extending the existing ISD literature to include the antecedents of
interpersonal conflict in ISD contexts, (2) identifying moderating factors that can mitigate conflict in ISD contexts, and (3) offering a model for identifying both the antecedents and potential mitigation of interpersonal conflict in ISD contexts.

**Workshop Articles**

**Title:** An Integrative Model for Understanding and Managing Software Interfaces for Increased Software Quality  
**Authors:** Thomas Meservy, Jasbir Dhaliwal, Murad Akmanligil, and Son Bui  
**Abstract:** Underlying a business world that is becoming more global and interdependent are complicated networks and systems representing the applications architecture that serves as the central nervous system of the modern, global organization. In such environments that are characterized by complex interdependencies, interfaces between modules and systems are the source of a significant percentage of system faults that are discovered during systems and integration testing. Understanding and managing the interfaces between such networks and systems represents a key challenge to technical managers today. There is a paucity of integrative theoretical ideas and pragmatic guidelines to guide such action today. This paper uses a longitudinal, interpretative approach focusing on a large fortune 500 company to develop an initial, integrative model to guide future theory building for systematically understanding and managing interfaces. This integrative model can be used by technical managers today as a pragmatic guide for understanding and managing interfaces today. The integrative model’s key dimension include: 1) undertaking the root cause analysis of interface errors, 2) understanding the profile of interface defects, 3) determining the complexity related to systems interfaces for focusing resources on risky interfaces, 4) considering a multitude of actions across the development lifecycle to improve interface requirements, design, implementation, and testing, and 5) developing a shared organizational vernacular and culture about interfaces. Another key aspect of the model includes focusing on interfaces earlier in software development. For managers who are struggling with interface issues in complex application architectures, our integrative model can serve as a valuable pragmatic guide for overcoming interface turmoil. Future research may want to build upon our initial integrative model to provide a sounder theoretical base for interface analysis and management.

**Title:** Evaluation of Usability Testing Tools with a Case Study  
**Authors:** Niklesh Katta, Scott Fleming, and Linda Sherrell  
**Abstract:** Usability is the how easily a person can use a web application or software for reaching a particular goal. Usability testing is a technique used to evaluate a design product by testing tools, in the context of a case study. Here, the case study is to build a simulator tool for market research that collects customer experiences when shopping. The case study was developed using the Scrum methodology. Some of the usability testing tools were applied on the simulator. These tools tested the system functionality and the effect of user interface on the user and identified problems with the design. With the test reports, we evaluated the usability testing tools with respect to their features.
Title: A Structured Approach to Test Case Definition with an Exemplary Illustration

Authors: Dani Almog and Tsipi Heart

Abstract: Test cases (TC) are fundamental units in software engineering in general, and particularly in software testing. In addition, TCs are often used as a metric and work unit in monitoring and controlling test efforts. A thorough review of the literature reveals that there is no formal and agreed-upon definition of a TC, hence, we see benefits in formalizing a unified, well-defined and structured TC definition. In this paper, we present a brief literature review of the TC concept, explore the definitions of TCs, propose a classification of the various definitions into four categories, and highlight the conceptualization underlying each category. The focus of the paper is the introduction of an alternative structured definition of a TC, illustrated by a real-world example. The proposed definition presents a TC as a composition of five structures: TC factors, Internal Activities and Flows, Dynamic External Interaction Element, Basic Verification Calls, and the TC Output and Results. We elaborate on the five structures, illustrate their content using a real-world example, and suggest benefits accrued by generating TCs structured accordingly. In conclusion, we discuss implications of this work for theory and practice, limitations, and future research trajectories opened by the proposed TC definition.

Title: Continuous Integration at EDM

Authors: Dean Ptaszynski and Praveen Vasudeva

Abstract: “An important part of any software development process is getting reliable builds of the software. Despite its importance, we are often surprised when this isn’t done” –Martin Fowler

This paper will describe the benefits of Continuous Integration how it makes testing part of the development process. We will describe our specific implementation methodology from build and deploy through functional testing. We will touch on the tools that we used and lastly describe the benefits gained by our team.

Title: Runtime Monitoring- A Post-deployment Security Testing Technique

Authors: Ramya Dharam and Sajjan Shiva

Abstract: Increased usage of software systems in recent years has in turn led to high need for ensuring the (i) confidentiality (ii) integrity, and (iii) availability of these software systems. Pre-deployment security testing techniques do not ensure that all possible behaviors of implementation are analyzed, executed, and tested. This in turn causes the software to often behave differently than what it was designed for during its post-employment, termed as Software Anomaly and is caused mostly due to external attacks such as SQL injection, cross-site scripting, path-traversal attack, etc. To detect such anomalies and to ensure the security and reliability of software systems, a post-deployment security testing technique known as runtime monitoring can be used. This paper reviews various security testing techniques and details runtime monitoring security testing technique. The paper also introduces a framework for the development of runtime monitors to accomplish security testing.
**Title:** Managing Service Level Agreements—Exploring Concepts in the Offshore Software Testing Services Context  

**Authors:** Jignya Patel, Robin Poston, and Jasbir Dhaliwal  

**Abstract:** Service level agreements (SLAs) are the part of the contracts between service providers and service recipients which formally defines delivery times and performance expectations. For standardized activities, e.g., help desk services and internet services, the contracting and delivery processes involved with SLAs depend on relatively objective measures. However, what happens when SLAs are needed for more ambiguous activities, such as those in software testing services delivered by offshore vendors, where an unknown amount and type of software defects may exist? Software testing activities attempt to reveal the quality of the product under test in order to inform the business of the risks of implementation. Utilizing the information systems research literature examining SLAs as the foundation, this research explores how widely accepted SLA concepts fit the offshore software testing services context. As a result, we propose and elaborate on several critical factors that are unique to the contracting and delivery processes in offshore software testing services arena and conclude with potential implications for future research and practice.

**Title:** Re-structuring Software Testing for Greater Efficiency and Effectiveness: Lessons from Financial Auditing  

**Authors:** Colin Onita and Jasbir Dhaliwal  

**Abstract:** Software testing is an important time and resource consuming process in software development. A way to reduce the burden of performing post-facto code testing is to perform an evaluative test of the internal controls and process used in developing software systems. This type of test would evaluate the error prevention, detection and correction capabilities of the software development process. If these capabilities are found to be adequate, testers can focus their substantive tests on the output of those processes and methods that are deemed to have weak error prevention/detection controls.

**Title:** Visualization and Simulation Tool for Software Development: Implications of using iRise for Software Testing  

**Authors:** Insu Park, Euntae “Ted” Lee, and Jasbir Dhaliwal  

**Abstract:** Visualization and simulation are powerful techniques which can be utilized during software development and testing to help realize project benefits and reduce risks in terms of software testing. This paper describes the architecture, available tools, and characteristics of iRise which is a visualization and simulation application tool. Specifically, this study focuses on the functionality of iRise which provides both visualization and simulation tools for the software testing project. The review of software prototyping and simulation is provided. Various aspects of iRise including architecture, features and characteristics, are discussed. What follows is the discussion of impacts of using “iRise” in software testing. Finally, several disadvantages of iRise tools are mentioned.
Title: A Framework and Research Agenda for Crowdsourced Testing

Authors: Thomas Meservy and Son Bui

Abstract: The confluence of several factors (e.g., advancement of technology, globalization, outsourcing of products and services) has created an opportunity to introduce a new testing service model that primarily relies on an unknown population of potential laborers. Crowdsourced testing, though not without its challenges, will likely redefine how certain testing activities may be accomplished. In this paper, we introduce a framework that characterizes crowdsourced testing and differentiates it from other common testing models. We further develop a research agenda to guide investigations related to crowdsourced testing.

Title: Security and Compliance Testing Strategies for Cloud Computing

Authors: Dr. Dipankar Dasgupta and Durdana Naseem

Abstract: Due to rapidly changing Information Technologies, it is becoming more expensive for companies/organizations to regularly update hardware and software, and also maintain a big IT department with highly technical staff. So many organizations are adopting cloud services to reduce the cost and increase the flexibility of their IT infrastructures. While different sectors are adopting the cloud for their IT need, they are also very concern about data security (both in rest and in motion) and various compliance requirements such as PCI DSS, HIPAA, GLBA, SOX, etc. There are several cloud service models, where one model sits on top of another with lowest one as Infrastructure-as-a-Service (IaaS), and above that is Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS). While security requirements are essential for all service models, they vary in the degree of defensive measures. Moreover compliance for each sector requires specific protection for online data such as Accountability, Transparency, Accuracy, Security and Access. Essentially all compliances have the Network and Data Security as the important area of responsibility. The main focus is to find new methods to manage risk factors and compliance violations in the cloud computing security, which the business or companies can utilize to manage processes and decision rights. Hence this provides new testing methods for the companies in reducing risk and compliance efforts. In this paper, we propose a framework that uses a compressive research on different compliance model requirements against each service model, which can be used as a framework by the companies to test risk and compliances against cloud service providers. The developed tool will generate a SLA document for the organization to help in finding correct required services from the vendor in order to certify the compliance in their industry structure.

Title: Dynamic Systems Security Testing using Function Extraction

Authors: Alan Hevner and Richard Linger

Abstract: We describe an approach for applying Function Extraction (FX) technology to the dynamic testing of security in large-scale, operational software systems. FX is used proactively as an intrusion detection and prevention system (IDPS) within a security infrastructure surrounding the operation of a critical software system. An innovative aspect of the FX approach is the concept of computational
security attributes (CSA). The CSA approach to software security analysis provides theory-based foundations for precisely defining and computing security attribute values. The translation of a static security property expressed as an abstraction of external data to the dynamic behavior of a program expressed in terms of its data and functions is a key to the CSA approach for verification of behaviors that meet specific security properties. The paper concludes with a discussion of future research and development directions for applying FX to dynamic security testing.

Title: Data Design and Management Issues in Software Testing: A Conceptual Framework

Authors: Mark Gillenson, Jasbir Dhaliwal, and Son Bui

Abstract: Since data is a central element of every information systems environment, it follows that data and its management is crucial in software testing efforts. In fact, there are a variety of kinds of data that either enter into the software testing process or are created during software testing. These include databases that the software under test (SUT) will access in its eventual production environment, the test cases created to test the software, and the data generated by the defect management system during testing. Taking a broader view of “data”, it includes requirements, systems analysis diagrams, and systems designs diagrams and specifications. If the SUT is an upgrade of an existing application, the data includes the historical production data that has been saved and the regression test case suite that has been saved for future “smoke tests.” This paper presents a conceptual framework for developing data design and management techniques in the software testing environment. It then proposes a new indexing system for tracking and crossreferencing the various kinds of data in the test environment.

Title: Transformational Leadership in Software Testing Teams: A Virtual Team Setting

Authors: Jeewon Cho, Insu Park, Euntae Ted Lee, and Jasbir Dhaliwal

Abstract: This paper examines an important role of a specific leadership style (i.e., transformational leadership) and its attributes and behavioral characteristics that might be positively related to software testing team performance in virtual settings. In this study, we suggest a preliminary research model that could be further developed into a more specific model including mediation and/or moderation once data collection is in process. The theoretical implications call for testing managers’ attention to the importance of transformational leadership development in software testing teams in virtual settings.

STEP 2011

Journal Articles

Title: Alignment within the Software Development Unit: Assessing Structural and Relational Dimensions between Developers and Testers.

Authors: Jasbir Dhaliwal, Colin Onita, Robin Poston, and Xihui Zhang

Abstract: Just as business-IT alignment has received significant focus as a strategic concern in the IS literature, it is also important to consider internal alignment between the diverse subunits within the IT organization. This study investigates alignment between developers and testers in software development to understand alignment within the IT unit. Prior evidence of tension between these subgroups (and others as well) suggests that all is not necessarily well within the IT organization. Misalignment within the IT unit can certainly make it difficult for the IT unit to add strategic value to the organization. This study is an important initial step in investigating IT subunit alignment which can inform future research focusing on the alignment of other IT subunits such as architecture, operations, and customer-support. Using theoretical concepts from strategic business-IT alignment, we test a research model through a survey of professional software developers and testers. Results suggest that relational but not structural dimensions influence IT subunit alignment.

Title: An Empirical Investigation of Client Managers’ Responsibilities in Managing Offshore Outsourcing of Software-Testing Projects


Authors: Radhika Jain, Robin Poston, and Judith Simon

Abstract: Despite an increase in the offshore outsourcing of information technology (IT) projects, little research has been carried out to identify issues that client managers (CMs) encounter when managing projects with offshore vendors. Also research on how CMs’ responsibilities change to effectively manage offshore outsourcing relationships has received little attention. We conducted a case study of an offshore outsourcing relationship between a U.S. client and its Indian vendors engaged in software-testing projects. In this paper, we identify six project management activities that underwent substantial change. We then describe the coping strategies that CMs employed effectively to deal with changing project management activities. We conclude by discussing the interplay of multiple global boundary variables and its effect on the project management activities. We integrate insights from literature on global distributed teams, organizational communications, and offshore outsourcing to theoretically ground the relationship between the boundary variables and the coping strategies.

Title: TESTQUAL: Conceptualizing Software Testing as a Service


Authors: Yang Yang, Colin Onita, Xihui Zhang, and Jasbir Dhaliwal

Abstract: Software testing has emerged as a distinct and critical component in software development. This paper argues that software testing should be conceptualized as a concurrent service throughout the software development process rather than being viewed as a sequential line of responsibility. Testing as
a service has two key aspects: (1) a service to developers, and (2) a service to end users. This paper draws from the service quality and SERVQUAL literature to propose a structured measurement tool for testing as a service. Software quality is the most prominent outcome for TESTQUAL. The potential antecedents and outcomes of TESTQUAL are briefly discussed. The implications to research and practice of TESTQUAL are also discussed.

**Title:** Mitigating Vendor Silence in Offshore Outsourcing: An Empirical Investigation


**Authors:** Radhika Jain, Judith Simon, Robin Poston

**Abstract:** The tendency to remain silent about project-related issues can contribute to suboptimal project performance or project failure. Prior research in offshore outsourcing suggests that client managers should play a critical role to induce offshore vendors’ employees not only to report project problems in a timely fashion but also to brainstorm and contribute ideas to a project. Also, the extant research on cross-cultural teams has emphasized the importance of cultural adaptation in the smooth functioning of these teams, but the role of cultural adaptation in silence mitigation has been largely underdeveloped in the literature. In this research, we bring these concepts of vendor silence and cultural adaptation in cross-cultural teams together and develop a process framework that illustrates how vendor silence may be mitigated in offshore outsourcing through various silence mitigation mechanisms. We then develop three propositions for organizational action toward mitigating vendor silence, which highlight the mediating role of cultural adaptation.

**Title:** Implementing Quality Gates Throughout the Enterprise IT Production Process


**Authors:** Vladimir Ambartsoumian, Jasbir Dhaliwal, Ted Lee, Thomas Meservy, and Chen Zhang

**Abstract:** The concept of quality gates has been successfully applied as a quality assurance mechanism in several industries. The quality gates approach combines aspects related to project management, decision modeling, and work flow management to increase measurability and promote quality. Software development and testing organizations are now investigating the use of this approach for the purposes of promoting software quality and improving software development processes. This paper summarizes prior literature on quality gates and applies the concept of quality gates to the software development context. It reports a case study of implementing quality gates in enterprise IT production process in the context of a large Fortune 500 company. A conceptual framework is also proposed to represent the various levels and disciplines where quality gates may be implemented. This framework suggests that quality gates can be applied at many different levels throughout the organization such as system,
project, and release. 2) the format of quality gates varies by level of implementation, 3) quality gates are useful both as part of an overarching software development methodology as well as for targeted IT projects where quality assurance standards have to be established for procedural success (e.g., transitioning enterprise data centers), and 4) more than the increased measurability promoted by quality gates, greater value may be yielded by a change in developers’ and testers’ mindset of building quality into the software product and development processes.

Title: Engaging Testers Early and Throughout the Software Development Process: Six Models and a Simulation Study


Authors: Mark Gillenson, Michael Racer, Sandra Richardson, and Xihui Zhang

Abstract: Software testing is indispensable in ensuring software quality. Traditionally, testing has been viewed as a separate and distinct stage at the end of the software development process. However, testing activities have evolved from the “code and fix” process of executing a piece of software in an attempt to find coding errors, to a collaborative coordinated effort with testing activities embedded throughout the entire software development life cycle. The benefits of contemporary testing activities include: linking together of perspectives across the entire organization, development of a better software product with fewer errors, and reduced cost by avoiding or finding errors earlier in the development life cycle. In spite of an emerging view that testing activities should be included early and throughout the software development process, there is little research in the area of how this can be accomplished. This paper attempted to address this void by offering six models for engaging testers early and throughout the software development process. It also carried out a simulation study with the in-depth surveyed data from 13 software testing professionals, for the purpose of determining which of the six models would be best under different development environment circumstances.

Title: Alignment within the corporate IT unit: an analysis of software testing and development


Authors: Colin Onita and Jasbir Dhaliwal

Abstract: Strategic alignment between an organization’s business strategy/capabilities and those of its information technology (IT) unit is an extensively researched subject that addresses the issue of fit between business and technology strategies. A key gap in the literature is lack of recognition that underlying this macro level of alignment are other, more granular levels of alignment involving the interdependent subunits within the corporate IT unit. Given the critical interdependencies between development and testing subunits in software engineering, this paper focuses on an alignment model for ensuring that these two functions work together effectively. A development-testing alignment (DTA)
model is described, and a case study investigating its value and application is presented. This DTA is decomposed into distinct components for the purposes of theoretical clarity and pragmatic application. The case study analysis uses the model to understand and interpret development-testing alignment in a Fortune 500 company. We found that the development and testing functions were significantly misaligned, and our model identified close to twenty specific aspects that needed to be considered to enhance alignment. These included changes in specificity of scope, governance, resource availability, competencies, and processes. Our analysis shows that the DTA model can be usefully applied for the purpose of understanding tactical alignment between subunits within a corporate IT unit. It also demonstrates that there is value in considering alignment as a dynamic, context-driven, social phenomenon as well as a useful interpretative lens for exploring organizational interactions and interdependencies.

Workshop Articles

Title: Increasing Software Quality in Large Organizations: A Case Study with a Focus on Software Interfaces

Authors: Murad Akmanligil, Ulrika Nachtigal, Thomas Meservy, and Jasbir Dhaliwal

Abstract: Complexity of software remains a salient antecedent of software quality. While there are many complexity metrics, interdependence of software systems significantly contributes to the overall complexity of developing and maintaining high quality information systems especially in large organizations. Only a paucity of studies in the academic literature have focused on software interface defects, and although a small percentage of overall defects, the cost of these defects is typically enormous often impacting multiple teams across the enterprise. In this article we present academic research that has been conducted related to software interfaces and then present a case study to investigate what one large software organization that has a complex application portfolio has proactively done to address interface related issues in their organization.

Title: Using Operational Business Intelligence to Inform Systems Testing

Authors: Alan Hevner, Balaji Padmanabhan, Jing Cheng, Michael Cuenco, and Crystal Shi

Abstract: Our on-going business intelligence (BI) project is studying the identification and remediation of customer churn based on the data mining of service quality indicators. We propose to extend this research by investigating the root causes of service related problems that relate to software systems defects and failures. The use of operational business intelligence dashboards has the potential to provide early warnings of faults in systems that are not found with traditional software testing methods during systems development. We report on the current status of our research project on customer churn and propose future directions for the use of BI dashboards – or in this case, Software Quality Dashboards - to inform systems testing.

Title: Automated Usability Testing Tool for Web Applications

Authors: Naga Sravana Sunil Saladi and Linda Sherrell
Abstract: Currently, the rate at which Web applications are growing is very high, and organizations developing these applications are pursuing more users by attempting to increase their interest. However, in this competitive world it is very difficult to determine a user’s interest in a company’s applications; therefore companies need more focus on the usability of their applications. This paper discusses an automated tool to test the usability of Web applications. As a case study, a web-based requirements engineering management tool was tested by considering various factors including the time spent on every page-load of every webpage, the time spent on the entire website, the number of page clicks, the number of back presses, and the total number of pages visited. In addition, the usability testing tool recorded suggestions provided by the users.

Title: Evaluating Effectiveness of Test Driven Development

Authors: Shweta Addala and Linda Sherrell

Abstract: Test driven development (TDD), also known as test first development, is not simply a testing technique but a development approach where the unit tests are written before functional code in small and rapid iterations. Research reports on the effectiveness of TDD in different phases of software development such as design, cost estimation, execution-based testing, and maintenance. This paper discusses a case study of a web-based routing application where both a test-first approach and test-last techniques were applied and compared for development time and code quality. Apart from the differences in metrics when using the two approaches, the paper includes a discussion of the positive personal experiences of the developer when using TDD.

Title: Complexity Measures for Implementing Quality Gates

Authors: Rick Prendergast, Chen Zhang, Thomas Meservy, E. Ted Lee, and Jasbir Dhalliwal

Abstract: Developing high quality software is often a complex activity which requires a careful orchestration of numerous technologies, processes and individuals. Assuring quality in such an environment is a non-trivial task. Over the years a number of different approaches have emerged to increase software quality. In our previous research, we suggested that quality gates could be used as a mechanism to better control the quality of production processes throughout the software development lifecycle.

Title: JRUTT: Java Remote Usability Testing Tool

Authors: Swathi Bavanaka and Linda Sherrell

Abstract: There has been tremendous growth in new and exciting Java web applications in the past few years. Remote usability testing allows you to evaluate the most important aspect of these websites, that is, their usability by gathering information from remote users. Usability testing itself is a technique for ensuring that the users of a system can carry out intended tasks efficiently. Several different tools for usability testing have been developed in different technologies, but there is no open source tool available that works for all Java web applications. This paper describes a remote usability testing tool called JRUTT (Java Remote Usability Testing Tool) that offers the capability of usability testing with users.
through user behavior log files and task elicitation. With JRUTT, we perform a case study on the system Security Enhanced Metric Taxonomy for SCRUM (SET4SCRUM) to obtain objective assessments of its design.

Title: Towards Using Abstract Behavior Models to Evaluate Software System Performance Properties

Authors: James H. Hill and Harold Owens

Abstract: Evaluating software system performance properties requires constructing a test driver that exercises a test object (or system) under realistic conditions. In many cases, core intellect defined in the test driver is reinvented across different applications domains. This can cause software testers to expend much time and effort during the performance evaluation phase of the software lifecycle. This paper therefore presents initial results on using abstract behavior models to reduce reinvention of core-intellect in performance tests and facilitate automated performance test generation.

Title: Applying Testability Concepts to Create Testability Guidelines

Authors: Robin Poston, Jignya Patel, Bindu Varriam, and Fatima Qadri

Abstract: This article presents an initial attempt at creating guidelines for software testability. While the guidelines proposed in this study are currently limited to the interface testing domain, with minor modification they should be expandable to the entire testing lifecycle. We start by using an auditing framework and borrow from the concepts of substantive and evaluative procedures found in the financial auditing literature to inform our approach in creating testability guidelines. This approach ensures a more comprehensive and rigorous series if steps is followed. Specific to technology, we also use components of the widely-adopted Control Objectives for Information and related Technology (COBIT) framework. We follow a case study approach to gather data from expert informants. As a result, we create and present an initial set of testability guidelines. Future steps will include cross validating the guideline using actual software projects by measuring the testability attributes of each project per the guidelines proposed.

STEP 2010

Journal Articles

Title: Organizing Software Testing For Improved Quality and Satisfaction


Authors: Xihui Zhang, Jasbir Dhaliwal, and Mark Gillenson

Abstract: The way testing is organized for software development has not been adequately addressed in both the practitioner and academic research literature. In practice, a diverse set of methods is being used to organize testing. Some organizations emphasize one-to-one matching between developers and testers while others do not. Additionally, some organizations have a distinct testing unit for their testing
professionals while others have them in the same unit as developers. Such practices are also influenced by the development methodologies of the organization such as the lifecycle and agile approaches. This paper attempts to shed light on whether these governance choices matter. It considers the influence of the development methods, the existence of one-to-one matching between developers and testers, and the existence of a distinct corporate testing unit on software quality and job satisfaction. The results of this study suggest that development methods do not significantly influence software quality or job satisfaction. However, one-to-one matching of developers and testers has a positive influence on both software quality and job satisfaction. The existence of a dedicated organizational unit for software testing also has a positive influence on the quality of software developed. These results suggest that organizations must emphasize one-to-one matching and a distinct testing unit for improved software quality and job satisfaction.

**Title:** Software Development Methodologies, Trends and Implications: A Testing Centric View


**Authors:** Xihui Zhang, Tao Hu, Hua Dai and Xiang Li

**Abstract:** The practice of software development has evolved steadily over the decades. Numerous methods and models (e.g., life cycle models and agile methods) have been proposed to enhance its efficiency and effectiveness. This study provides a testing centric view of software development practices. Specifically, it reviews software development methodologies (i.e., methods and models), identifies the latest trends in the industry and discusses their implications. The review of methodologies, the identification of these trends and the discussion of their implications will be useful to software development educators, students, practitioners and researchers.

**Title:** Client Communication Practices in Managing Relationships with Offshore Vendors of Software Testing Services


**Authors:** Robin Poston, Judith Simon, and Radhika Jain

**Abstract:** Enabled by the globalization and advances in technology, offshore outsourcing of software development to countries such as India, China, and Russia, continues to increase. Much of the extant research has not focused on the communication practices observed in thriving offshore client–vendor relationships. Our research identifies communication practices found in a case study of a large multinational client’s multi-vendor relationship in offshore outsourcing of software testing projects. We discuss the empirically grounded communication practices in the light of existing literature to highlight how the client–vendor relationships deliver long-term value. Through this discussion, we delineate and discuss communication techniques. Implications for theory and practice are also discussed.
**Workshop Articles**

**Title:** A Model and Simulation of Duplicate Identification Methods’ Cost Effectiveness  
**Authors:** Vasile Rus and Sajjan Shiva  
**Abstract:** We present in this paper a theoretical model and simulation for estimating the cost effectiveness of automated methods that address the task of duplicate bug report identification. The paper argues that precision is an important performance measure for assessing the cost effectiveness of duplicate identification methods and thus their impact on software testing and development processes. This is in response to the incorrect emphasis on recall only, as an assessment measure, in recent work on duplicate identification. We also show that the relation between the precision of a duplicate identification method and the cost necessary to handle false positives, called misses, has a major impact on the cost effectiveness of methods. The simulation shows the necessary precision and false positive handling cost levels for each recall value such that duplicate identification methods become effective. It should be noted that our cost effectiveness model focuses on the worst case scenario.

**Title:** Extending the Development of Test Cases in Conditions of Large Numbers of Variables and Values  
**Authors:** Mark Gillenson, Martin Dinstuhl, Scott Fischer, and Thomas Meservy  
**Abstract:** For software that requires a large number of input variables and/or a large number of values per variable, generating an exhaustive set of test cases is clearly infeasible. A well-known heuristic technique for generating a set of test cases that is both efficient and effective is pairwise testing. However, even pairwise testing has its limits as the number of input variables and/or the number of input variable values is very large. This research describes two efforts to aid in this problem. One is lessons learned in developing an effective test bed for experimenting with different test case generation techniques. The other is the development of a new concept for extending pairwise testing through exploratory testing guided by genetic algorithms.

**Title:** Role of the Testing Group in Selecting an Enterprise Architecture Solution: A Case Study  
**Authors:** Vikram Dias, Robin Poston, Jasbir Dhaliwal, Murad Akmanligil, Dave Miller, and Rashid Koja  
**Abstract:** Software testing groups are playing an increasingly prominent role in both the software development lifecycle (SDLC) and in the long-term planning of technology architectures that support organizational information systems. The advent of integrated enterprise architectures (EA) provides new opportunities for testing groups to play a proactive role in building consistent and testable guidelines for improving enterprise-wide software quality. Given that testing groups historically have not been invited to participate in EA decisions, there is little academic literature or industry best practices on approaches that testing groups might use to guide their participation. This article draws lessons from the experience of a Fortune 100 corporation whose testing group used theoretical notions of “testability” to guide its involvement in an EA acquisition process. It describes how it operationalized testability criteria, incorporating controllability, observability, and simplicity, into various stages of the process and illustrates the benefits and challenges of taking such an approach.
**Title:** Test-Driven Development in the Corporate Workplace  
**Authors:** Ahmed Owian and Linda Sherrell  
**Abstract:** Software engineers in industry use many different processes and techniques in an attempt to create quality software. In today’s market, many software developers are using agile practices, especially test-driven development (TDD). What is TDD, and why is it giving traditional software development practices a run for their money? This work answers these questions, while focusing on a popular agile methodology, Extreme Programming (XP). It places a particular emphasis on the exploratory programming nature of XP and its testing practice, TDD. The paper also summarizes prior research on TDD and includes the results from a research survey conducted to compare TDD with traditional testing practices. The survey shows TDD has many benefits; namely, it helps developers better understand their code and increases their confidence concerning whether the product adheres well to requirements.

**Title:** Implementing Quality Gates throughout the Enterprise IT Production Process  
**Authors:** Bart Dahmer, Rick Prendergast, Vladimir Ambartsoumian, Jasbir Dhaliwal, Ted E. Lee Thomas Meservy, and Chen Zhang  
**Abstract:** The concept of quality gates has been successfully applied as a quality assurance mechanism in several industries. The quality gates approach combines aspects related to project management, decision modeling, and work flow management to increase measurability and promote quality. Software development and testing organizations are now investigating the use of this approach for the purposes of promoting software quality and improving software development processes. This paper summarizes prior literature on quality gates and applies the concept of quality gates to the software development context. It reports a case study of implementing quality gates in enterprise IT production process in the context of a large Fortune 500 company. A conceptual framework is also proposed to represent the various levels and disciplines where quality gates may be implemented. This framework suggests that 1) quality gates can be applied at many different levels throughout the organization such as system, project, and release, 2) the format of quality gates varies by level of implementation, 3) quality gates are useful both as part of an overarching software development methodology as well as for targeted IT projects where quality assurance standards have to be established for procedural success (e.g., transitioning enterprise data centers), and 4) more than the increased measurability promoted by quality gates, greater value may be yielded by a change in developers’ and testers’ mindset of building quality into the software product and development processes.

**Title:** The Impacts of Service Quality Indicators on Customer Churn- A Data Mining Approach  
**Authors:** Alan Hevner, Balaji Padmanabhan, Jing Cheng, Michael Cuenco, and Crystal Shi  
**Abstract:** Customer churn is a major concern for most businesses in a competitive space. Understanding the factors that contribute to churn can help firms design better customer retention strategies. Among key churn factors are customer characteristics and service quality indicators. Examples of customer
characteristics are nature of the business, location, and loyalty; while examples of service quality indicators include missing packages, loss claims, and missed pickup windows. There are, however, a very large number (hundreds) of specific customer and service quality indicators that are known or hypothesized to affect churn. With this large number of indicators identifying the significant ones that matter for churn and quantifying their impact are crucial. In this research, we study the impacts of customer factors and service quality indicators on churn, focusing specifically on knowledge discovery approaches from data mining for this task. Significant contributions will be made to both the theory and practice of identifying and remediating customer churn.

Title: Interface Testing: Understanding and Addressing Software Interface Complexity in Large Organizations

Authors: Thomas Meservy, Jasbir Dhaliwal, and Murad Akmanligil

Abstract: Many of today’s software systems contain complex interdependencies between modules. In many large systems, the interfaces between modules/systems are the source of a significant percentage of system faults that are discovered during systems and integration testing. We assert that changes to interfaces should be tested much earlier in the software development lifecycle. However, due to the complexity of the environment and numerous other factors, many organizations continue to struggle how to categorize and address interface related issues. This paper presents a framework for categorizing interface faults and characterizes salient interface faults derived from a case study at a large organization that has a complex application portfolio. Additionally, insights are shared from onsite interviews on how companies might address interface-related issues. Finally, an interface complexity index is presented that can be used to help companies focus resources on the most risky interface changes.

Title: Governance of Software Testing: Impact of Distinct Testing Unit, Reporting Structure, and One-to-One matching

Authors: Xihui Zhang, Colin Onita, and Jasbir Dhaliwal

Abstract: The proliferation of complex, interconnected and real-time business applications is leading to information technology (IT) managers struggling with pragmatic governance mechanisms for integrating testing with development. However, governance issues pertaining to how software testing is organized at strategic, tactical and operational levels have not been received adequate attention in the literature. This study explores the impact of three specific governance mechanisms, including the existence of a distinct corporate testing unit, developers and testers reporting to different executives, and the existence of one-to-one matching between developers and testers, for the organization integration of testing with development. A national survey of 196 of software development professionals was undertaken to investigate these impacts on a set of dependent variables compromising organizational, group, and individual outcomes. The results indicate that these governance mechanisms have significant impacts that need to be considered for successful integration of development and testing.

Title: Statistical Profile of the Eclipse Bug Database
Authors: Eric Woolley, Vasile Rus, and Sajjan Shiva

Title: Functional Testing in An Agile Development Environment

Speaker: Rajeshwari Bali

STEP 2009

Workshop Articles

Title: On the Need for Careful Definition of and Improved Capabilities in Quality-of-Service Unit Testing

Author: James H. Hill

Abstract: Unit testing traditionally is a process for increasing confidence levels in functional attributes of large-scale distributed systems throughout the software lifecycle. As largescale distributed systems grow larger and more complex, increasing confidence levels in their quality-of-service (QoS) attributes, such as performance, reliability, and security, throughout the software lifecycle is becoming increasingly important. Little work, however, has investigated the challenges associated with unit testing QoS attributes of largescale distributed systems throughout the software lifecycle. This paper provides two contributions to testing QoS attributes of large-scale distributed systems. First, this paper defines the meaning unit testing QoS attributes. Secondly, it discusses challenges associated with unit testing QoS attributes of large-scale distributed. By addressing the challenges presented in this paper, distributed system developers will be able to improve QoS assurance of large-scale distributed systems throughout the software lifecycle.

Title: Detecting Software Errors Using Data Mining on Business Process Logs

Authors: Alan Hevner and Balaji Padmanabhan

Abstract: Business processes may perform poorly or incorrectly due to software system defects that often are not obvious to detect. For example, packages delivered to a certain zip code may be habitually delayed due to a GPS module providing faulty or sub-optimal routing for that zip code. In such a case there is often no visible system failure to be investigated. Instead, business process logs may need to be examined to identify such failures by looking for systematic differences between process logs that result in good and bad process outcomes. The focus of our research is to investigate approaches using data mining to discover fault patterns in business process logs. We propose an approach that relies on leveraging faults in business process outcomes (e.g. a package arriving late), a mapping of how software modules impact specific business processes, and the use of data mining techniques to learn patterns to identify the potential sources of error. We extend our STEP 2008 paper (Hevner and Padmanabhan 2008) in two significant ways. First we present a detailed methodology that can be applied on binary
process logs. Second, we present detailed results from a simulation of business process logs that shows the potential of the approach.

**Title:** Aligning Testing & Development: Empirical Evidence  
**Authors:** Colin Onita and Jasbir Dhaliwal  
**Abstract:** Efficient and effective development of high quality software requires that developer and tester groups work in an aligned and harmonious manner. This paper builds on previous research to empirically investigate a specified Development-Testing (DT) Alignment model initially proposed by Dhaliwal and Onita [2]. This model decomposes DT Alignment based on structural components (strategy and capabilities) and flow components comprising internal coherence of both testing and development strategy and capabilities; strategy and capabilities alignment; and alignment pertaining to the ability to execute for the testing and development units. Data collected through a national level survey of development and testing professionals indicates that respondents view testing and development to be aligned with the exception of some components of testing strategy and capability alignment. The main sources of misalignment were identified as the resource endowment of the testing group and the skill sets, tools and techniques used in testing and development.

**Title:** Developing Simulations of Six Models for Engaging Testers Early and Throughout the Systems Development Process  
**Authors:** Mark Gillenson, Michael Racer, Sandra Richardson, and Xihui Zhang  
**Abstract:** In the early history of systems development, testing was confined to test the finished code. However, errors can be introduced at all stages of the systems development process and it is a well-accepted fact that the earlier in the process an error is found, the less costly it will be to fix. Thus, as the practice of systems development has evolved, there has been increasing interest in expanding the role of testing into the earlier systems development stages. While testing finished code is accomplished with established black box testing, white box testing, and code review techniques, testing in the earlier systems development stages involves less well-defined design inspection and walkthrough exercises. This paper describes the development of the simulation of a set of six models for embedding testing throughout the systems development process.

**Title:** An Analysis of the Impact of Testing within Agile and Lifecycle Development Methodologies  
**Authors:** Xihui Zhang and Jasbir Dhaliwal  
**Abstract:** Software testing is generally done within the contexts of two categories of development methodologies: the traditional systems development lifecycle and variations of a newer agile framework. There has been little empirical evidence to-date as to how these impact the effectiveness of software testing. This research considers the impact of the use of agile methods, one-to-one matching between developers and testers, and the existence of a distinct corporate testing unit on dependent variables that represent impacts. These dependent variables include the levels of relational and task conflict between developers and testers, job satisfaction, and software quality. Our results suggest that
the use of agile methods does not impact job satisfaction or software quality but does increase task conflict between developers and testers. One-to-one matching of developers and testers has positive impact on both job satisfaction and software quality. The existence of a dedicated organizational unit for software testing also has a positive impact on the quality of software developed. These results need to be considered by software development organizations when making critical decisions pertaining to the selection of software development methodologies and the organizational roles for testing professionals.

STEP 2008

Workshop Articles

Title: Sources of Conflict between Developers and Testers in Software Development: A Preliminary Investigation

Authors: Xihui Zhang, Mark Gillenson, Jasbir Dhaliwal, and Gertrude Moeller

Abstract: Interpersonal conflict between software developers and testers is inevitable and pervasive. This conflict is likely to be negatively associated with software quality and job satisfaction. This study addresses one major research question: What are the sources of interpersonal conflict between developers and testers in software development? Using a qualitative approach, we collect and analyze fifty developer-tester conflict scenarios from professional developers and testers. Preliminary results indicate that conflict sources between software developers and testers fall into three major categories: Process, people, and communication. Conflict sources are presented in a category-subcategory-example format. Implications for research and practice are discussed

Title: Developing Simulations of Six Models for Engaging Testers Early and Throughout the Systems Development Process

Authors: Mark Gillenson, Sandra Richardson, Michael Racer, and Xihui Zhang

Abstract: In the early history of systems development, testing was confined to test the finished code. However, errors can be introduced at all stages of the systems development process and it is a well-accepted fact that the earlier in the process an error is found, the less costly it will be to fix. Thus, as the practice of systems development has evolved, there has been increasing interest in expanding the role of testing into the earlier systems development stages. While testing finished code is accomplished with established black box testing, white box testing, and code review techniques, testing in the earlier systems development stages involves less well-defined design inspection and walkthrough exercises. This paper describes the development of the simulation of a set of six models for embedding testing throughout the systems development process.

Title: A Systematic Methodology for Aligning Testing and Development

Authors: Colin Onita and Jasbir Dhaliwal
Abstract: Alignment between testing and development has been raised as an issue for successful systems development. Missing however have been actionable how-to methodologies for assessing and enhancing such alignment. This paper attempts to fill this gap by describing a systematic methodology for this purpose that can be pragmatically used by systems development organizations. This methodology considers alignment at both strategy and execution levels. By dissecting alignment into internal (within) and external (between) categories, it outlines pragmatic mechanisms by which the coherence between the internal components of developer-tester alignment can be assessed and managed. At the strategy level the aspects of these components that have to be aligned include scope, governance and resources. At the execution-capability level, aspects that have to be aligned include processes, architecture and skills. Specific mechanisms and tools for ensuring that the execution capabilities of testing groups are aligned with their stated strategies are identified. Systematic mechanisms for tying testing strategy and capabilities to development strategies and capabilities are also discussed.

Title: Integration of Testing and Other Software Development Tools

Authors: Thomas Meservy, Chen Zhang, and Bill Kettinger

Abstract: Numerous software development and testing methodologies, tools, and techniques have emerged over the last few decades promising to enhance software quality. While it can be argued that there has been some improvement it is apparent that many of the techniques and tools are isolated to a specific lifecycle phase or functional area. Tool integration advocates promise to further increase software quality throughout the organization though literature remains primarily focused on technical solutions rather than on socio-economic factors associated with tool integration. This study draws on literature in information systems, organization science, and software engineering to develop a conceptual model of factors that influence an organizations’ need for enterprise-level integration of information systems development tools.

Title: Representing Test Cases for Semantic Web-based Software Testing Systems

Authors: Nor Adnan Yahaya

Abstract: Recent advances in the World Wide Web technologies have brought about the emergence more powerful tools and applications that can support collaborative activities. Now with the advent of the Semantic Web as the future Web, it is timely to explore the potential use of emerging Semantic Web technologies in developing knowledge-based software testing systems (KbSTs) that would also be collaborative in nature. This paper uses test case management as a basis for studying the problem of developing KbSTs over the Semantic Web framework and explores the representational issues pertaining to test cases that are foundational to the development of what we refer to as Semantic Web-based software testing systems (SWbSTs).

Title: Testing Using Program Instrumentation and Visualization

Author: V.Lakshmi Narasimhan
**Abstract:** This paper presents a mechanism to perform software testing using program instrumentation and visualization. Through program instrumentation, the problem of test data generation is treated entirely as a numerical optimization problem. As a consequence, this method does not suffer from the disadvantages of symbolic execution or a constraint simplifier in order to analyze a program. Instead, program instrumentation is used to solve a set of constraints without explicitly knowing their form. The system supports not only the generation of integer and real numerical data types, but also non-numerical discrete types such as characters and enumerated types. Thus, the approach combines the fundamental elements of symbolic execution and program instrumentation to produce a system, which offers superior performance for program testing. A suitable visualization environment has also been implemented along with several metrics generator.

**Title:** A Study on Fault Injection based Approach for Dependability Evaluation

**Authors:** S. M. Masum, S. Saha and M. Yeasin

**Abstract:** Fault Injection is an effective solution to the problem of validating highly reliable systems. Fault Injection is the process of corrupting a data state during program execution. Fault injection based testing is the process of determining the effect of that corruption. The testing may consist of simply measuring whether the corrupted state affected a particular output, or the testing may determine whether system attributes such as safety, security, or survivability have been affected. Fault injection based testing is often used in large system development projects. This paper presents a study on fault-injection-based approach for dependability evaluation.

**Title:** Data Mining of Business Processes to Detect Software Errors

**Authors:** Alan Hevner and Balaji Padmanabhan

**Abstract:** It is a challenge to detect software errors in operational systems when no obvious error conditions are evident. Business processes may perform poorly or incorrectly due to software system defects. For example, packages delivered to a certain zip code may be habitually delayed. However, linking these operational conditions to the responsible software defects may be very difficult. The focus of our research is to investigate the use of data mining to discover fault patterns in business processes and then to link these faults with their cause in a software module. We propose an approach that relies on leveraging faults in business process outcomes (e.g. a package arriving late), a mapping of how software modules impact specific business processes, and the use of data mining techniques to learn patterns to identify the potential sources of error.

**Title:** Predicting Defects in Large Open Source Software

**Authors:** Teodora R. Stoian, Vasile Rus, and Sajjan S. Shiva

**Abstract:** Defect prediction is an important topic in software quality processes. We present in this paper our research on defect prediction based on numerical models. Numerical models predict how many defects are in software units as opposed to just predicting the presence or absence of defects performed in binary model. The proposed models were evaluated using experimental data collected
from approximately 3 million lines in 3 Mozilla’s Firefox releases. To generate the experimental data, we collected problem reports (PRs) from Bugzilla, Firefox' defect tracking system, and linked the PRs to modification reports (MRs) obtained from Mozilla’s Concurrent Versioning System (CVS). Various linear and nonlinear machine learning algorithms were then used to induce predictors based on size and complexity metrics. The results obtained with the various algorithms are discussed and compared.

**Title:** Machine Learning-based Approach for Testing Large Scale Software: A Survey  
**Authors:** M. S. Sorower, A. K. M. M. Rahman and M. Yeasin

**Abstract:** Software engineering process incorporates different theoretical and applicable computer science domains (i.e. theoretical computer science, graph modeling etc.) to assure appropriate behavior of the developed software. Software testing is particularly devoted to find defects in a software system. Inadequate test plans and procedures are often positively correlated with time delays, cost overruns and security vulnerabilities. To avoid time delay and cost overruns, significant manpower and efforts are often employed into software testing. However, in cases of overbudget and behind schedule scenarios, software testing efforts are often cut short resulting in software security vulnerabilities. Hence, the modular automated software testing system that can detect common programming errors and security bugs, which are otherwise expensive to find, will translate to optimized and safe code. This paper reviews the challenges in using machine learning techniques for identifying software defects. The foci of the paper is to present a comprehensive analysis and review on approaches suitable for large scale software testing and will facilitate a modular and highly scalable architecture that may be used across a variety of software products with minimal changes to the testing package. While reviewing past achievements, this paper also outlines the implementation strategies currently under investigation.

**Title:** Conceptualizing Software Testing as a Service: the TESTQUAL Measurement for Testing Service Quality  
**Authors:** Yang Yang, Colin Onita, and Jasbir Dhaliwal

**Abstract:** Testing has emerged as a distinct responsibility within software development. Rather than being viewed as a sequential line responsibility in software development, this paper argues that it may be better conceptualized as a service responsibility. Testing as a service has three key aspects: 1) as a service to developers, 2) as an indirect service to end users of software products, and 3) as the service component of software products. This paper draws on the SERVQUAL literature in information systems and the International Standards Organization (ISO) standards for software quality to propose a structured quality measurement for testing as a service. In the paper, this TESTQUAL framework is described and its components defined and exemplified. It also discusses practical applications of TESTQUAL and research model for investigating its antecedents and impacts is presented.

**Title:** Applying TDD to Web-Application Development – Does Variety Help or Harm?  
**Authors:** Wolf-Gideon Bleek and Sebastian Pappert
Abstract: This paper presents a case study which covers experiences gained from a Web development project employing agile techniques. Web development involves a number of technologies introducing a variety which is difficult to handle. Beside other agile techniques the project’s participants followed test-driven development (TDD) in the project’s various technology areas. TDD is used in addition to unit testing to positively influence the design of program artifacts. In doing so, they encountered problems, pitfalls and failures and developed strategies to have an overall test-driven approach. We report on the stony way exercising Test-Driven Development in a typical technology mix of a Web development project to learn about obstacles surfacing while pursuing design through Test-Driven Development. The case study reveals three strategies that programmers follow: avoiding to subclass from a framework, moving code to easy testable classes and keeping hard-to-test artifacts small.

Title: Defect Management System for Software Development Process

Author: K. V. Dinesha

Abstract: Defect management forms an integral part of every quality system. Defects tell about the health of the system. One of the main focuses of SEI CMM level 5 is defect prevention and use of defects for improving software process. In this article we discuss defect management life cycle from a practitioner’s perspective. We hope that this article provides background, guidelines and framework for setting a defect management system for the software system environment of an organization. This article discusses what defects are and their different aspects, organization attitude towards defects, different methods of analyzing defects, how defect analysis and classification can help in improving the ongoing project and organization process, and what to expect from defect management software. The objective of this article is to assist an organization in building process part of its Defect Management System.

Title: Wikis for KM in Software Testing: Benefits and Success Factors

Authors: Ted Lee and Bill Kettinger

Abstract: Software testing involves the process of detecting software discrepancies so that they can be corrected before they are installed into a live environment supporting operational business unit. Considering the importance of conscious KM practice for software testing project management and better support for the complex and multi-staged task of software testing, it seems logically natural to adopt a KM approach for software testing projects. Because of a high initial and on-going commitment involved in full scale KM frameworks and methodologies, an alternative and incremental approach to KM for software testing is called for. We suggest to use wikis as a promising technology, wikis have been touted for low-risk, cost-effective tool for knowledge sharing and collaboration. Based on literature review, we try to identify typical activities in software testing for wiki use, potential benefits of using wikis, and success factors of adoption and usage of wikis. As a future study, we plan to conduct a set of qualitative case studies of wikis in real software testing organizations to explore the role of these success factors of wikis and confirm the potential benefits in a natural setting.

Title: Perspectives of Offshore Testing Vendors
Authors: Robin Poston, Judith C. Simon, and Bill Kettinger

Abstract: Global offshoring is not new. Steady improvements in information and communication technologies and technical skill levels globally have led to a boom in offshore outsourcing for software development and testing. While considerable research exists examining specific factors related to selecting and working with offshore vendors, little research has taken a best practices perspective on how to evaluate and sustain a successful long-term relationships with an offshore vendor. Now that many firms have been outsourcing globally for many years, it is time to examine those best practices that result in successful outsourcing partnerships over the long term. This paper addresses this gap in the literature by examining three major testing firms in India that have differing levels of success in sustaining a long term relationship with their Fortune 500 client. Key findings which incorporate direct quotes from the vendors are offered to shed light on the best practices in building successful long-term relationships between offshore vendors and global enterprises.

Title: Offshore Vendor Management: Results from a Recent Survey of Managers

Authors: Judith C. Simon and Robin Poston

Abstract: This paper represents the second phase of an offshore vendor management research project. Phase one was a qualitative phase involving interviews with internal offshoring management personnel of a Fortune 500 company to gain insights about offshoring, with results presented at the 2007 Workshop on Advances and Innovations in Systems Testing. Those results served as a basis for the content of a survey instrument developed for Phase two. In this second phase, we have gathered data through surveys of additional organizations, both large and small. The results provide levels of agreement on several factors regarding offshore vendor management

STEP 2007

Workshop Articles

Title: A Tool for Risk-Based Testing

Authors: Linda Sherrell, Sarah Bowen, and Hima Puppala

Abstract: This paper describes research in heuristic risk-based testing. The aim of this research is to design and implement a risk assessment tool called QUART-ET (Quick Assessment of Risks for Engineering Testing) to facilitate the risk management process. The purpose of QUART-ET is to help software development teams manage risks specifically related to testing and to aid management in discovering an acceptable risk appetite for software maintenance. The tool provides automated assistance to members of the testing team during the process steps of risk identification and risk analysis. The researchers use an outside-in approach to heuristic-based testing by furnishing users with predefined product and process risk lists within the tool.

Title: A General Framework for Information Processing with Application to Quantitative Software Testing
**Authors:** Vasile Rus and Sajjan Shiva

**Abstract:** We propose in this paper a general framework for collecting and processing data from various sources of information. An advantage of our framework over traditional data mining frameworks is the ability to collect nuggets of information from sources encoded in different formats. We can handle documents or fragments of knowledge written in natural language (open structure documents), XML-like data (semi-structured documents), and database repositories (structured documents). Our proposed framework leads to an integrated technology for collecting data from these various sources as opposed to a technology in which there is a set of isolated components that deal with format-specific data sources. We exemplify how this framework can help with the problem of finding duplicate defects. Before reporting a defect, a tester must check the defect has not been reported before. This checking step is time consuming because previous descriptions of same or similar defects may be worded differently, may appear in more than one repository (mailing lists, bug repository, known issues list), and may be encoded in different formats. Our information processing framework can provide an easy to use and efficient solution to this particular problem and other information seeking needs.

**Title:** Hercules: An Environment for Large-Scale Enterprise Infrastructure Testing

**Authors:** Lan Wang, Charles Ellis, Wei Yin, and Dung Dinh Luong

**Abstract:** Failures in Internet-based services often stem from problems in the underlying enterprise infrastructures that provide the services. We propose an environment called Hercules to test the reliability and performance of large-scale enterprise infrastructures. Hercules contains two major components: (1) a methodology to build a virtual testbed that can accurately emulate any infrastructure topology, as well as simulate failures, attacks and other types of stresses on the infrastructure to identify defects and bottlenecks; and (2) a realistic traffic model and a tool to automatically generate different traffic loads based on the model. Systems testers can use Hercules to evaluate whether an existing or proposed enterprise infrastructure provides adequate support to its targeted applications.

**Title:** Fault Injection: A Method for Validating Fault-Tolerant System

**Author:** Salahuddin Mohammad Masum, Bhanu Chander Reddy Vanteru, and Mohammed Yeasin

**Abstract:** Fault Injection is an effective solution to the problem of validating highly reliable systems. Fault Injection is the process of corrupting a data state during program execution. Fault injection based testing is the process of determining the effect of that corruption. The testing may consist of simply measuring whether the corrupted state affected a particular output, or the testing may determine whether system attributes such as safety, security, or survivability have been affected. Fault injection based testing is often used in large system development projects. This paper presents a fault injection technique that injects faults into computer system to measure system dependability. Our simulation shows the performance of the fault injection model.

**Title:** Automated Unit Testing with AgitarOne

**Authors:** Eamon McCormick
Abstract: Unit testing is the first line of defense in the battle to produce high quality and highly maintainable software. When developers create unit tests along with their code, they both ensure that code is written correctly and provide executable documentation that protects code against regressions. A thorough set of unit tests allow engineering teams to add capabilities to their applications and change existing behavior confidently. AgitarOne is Agitar Software’s automated unit test solution. AgitarOne allows development teams to more thoroughly unit test Java code through automation. AgitarOne combines a revolutionary exploratory testing technique called Software Agitation with a powerful regression testing approach built on JUnit. During this presentation we'll discuss the value of unit testing and the additional value that can be added by an automated solution like AgitarOne.

Title: A Framework for Aligning Testing and Development

Authors: Jasbir Dhaliwal and Colin Onita

Abstract: Strategic alignment between an organization’s business strategy/capabilities and those of its information technology (IT) unit has been researched extensively and has found significant application in industry in the last decade. Given the critical interdependencies between development and testing groups within a corporate IT unit, this paper presents a similar alignment model for ensuring that these two functions work together effectively in meeting corporate IT goals pertaining to building new systems. This development-testing alignment (DTA) model is described and an overarching research framework for investigating its value and application is presented.

Title: Generating Test Inputs for Fault-Tree Analyzers using Imperative Predicates

Authors: Saša Misailović, Aleksandar Miličević, Sarfraz Khurshid, and Darko Marinov

Abstract: This paper presents a case study on how Korat can be used in system testing, specifically in testing a large fault-tree analyzer developed for NASA. A fault-tree analyzer takes as input a fault tree that models how combinations of failures in the components of a system produce overall failures of the system. Testing a fault-tree analyzer requires generating fault trees. Korat is a previously developed testing tool that automates generation of structurally complex test inputs. Fault trees are structural in that they can be represented as graphs, and the nodes in the graphs need to satisfy certain complex constraints. Korat allows the user to express these constraints in widely used imperative programming languages such as Java. Previous research has shown how to test a fault-tree analyzer using another testing tool that requires the user to express constraints in a declarative language. This paper compares these two approaches. The results show that Korat generates a larger number of inputs but does not prune out non-equivalent inputs and thus can generate inputs that reveal errors in the system under test.

Title: The Division of Knowledge: Knowledge Transfer Practices in Outsourcing

Authors: Thomas F. Stafford, Mark L. Gillenson, and Sandra Richardson

Abstract: The Division of Knowledge deals with the key distinction between the implicit and the explicit bodies of knowledge about firm processes, both within the client firm and across outsourcer vendors.
This concerns the explicit codification for the client, by the vendor, of the tacit and informally transferred systems testing knowledge base of the firm. The primary theoretical contribution of this formalized explication of the parallel division of knowledge development lies in building an understanding of the knowledge transfer process across an outsourcing context in which the vendor is tasked with developing, documenting and sharing an explicit knowledge base of testing processes as both a working tool and a specific customer service owed to and owned by the client, while the client maintains an extensively tacit approach to skill development in its systems testing organization.

Title: Applying Knowledge Management Approach for Software Testing

Author: Ted Lee

Abstract: Software testing involves the process of detecting software discrepancies so that they can be corrected before they are installed into a live environment supporting operational business units. To better support this complex task of software-testing, this study proposes identifying and applying a knowledge management (KM) approach to software testing. Based on literature review, three frameworks are identified, each of which was used in a project based environment. An integrated hybrid KM framework for software testing is developed by incorporating desirable aspects of the first two frameworks into the third one. To assess the effectiveness of the hybrid framework, an empirical study needs to be conducted.

Title: Can Employees’ Personality and Cultural Characteristics Be Used to Predict their Best Fit With Software Testing Job Tasks?

Authors: Judith C. Simon, Charles J. Campbell, Judith C. Brown, and Sandra Richardson

Abstract: This paper presents a review of characteristics that might be appropriate factors to predict job success, followed by a preliminary model that could be incorporated into a more specific model once data becomes available for testing the model. The eventual model could add value to the process of making software testing task assignments by considering certain employee traits that would lead to an increased chance of putting the right employee in the right job at the right time, resulting in improvements in quality and productivity of software testing activities.

Title: Knowledge Partitioning and Knowledge Transfer Mechanisms in Software Testing: An Empirical Investigation

Authors: Radhika Jain and Sandra Richardson

Abstract: The software development industry spends more than half of its budget on maintenance related activities. Software testing provides a means to reduce errors, cut maintenance and overall software costs. Given the importance of software testing it is surprising that there is very little emphasis on formal software testing education. This suggests that most software testers are then either self-taught or they acquire needed skills on the job through informal and formal knowledge sharing mechanisms. We explore the role of knowledge management and specifically the various knowledge sharing and transfer mechanisms that enable software testers to acquire software testing skills. This is a
multi part study. This paper reports our preliminary findings from the first phase. First phase data was collected at a large multi-national company. We expect that these guidelines can help organizations to devise a program for improving skills of their employees involved in software testing and institute a culture of effective use of knowledge transfer mechanisms.

Title: The CollegeCareerCorps Model

Authors: Laurie Craig, Kevin Avent, and Jeff Brittain

Abstract: Universities across the nation report a significant decline in the number of students enrolling in Information Technology centric majors. According to a recent March 2007 study, this decline is approximately 50% as compared to the 2000/2001 academic year, with many departments reporting a 14% decline in the last year alone. To help stem this decline locally, the FedEx Institute of Technology at the University of Memphis has teamed with Cook Systems International to create the CollegeCareerCorps model. While this issue will require a multifaceted approach, the CollegeCareerCorps model is one key tool now available to communities. This innovative model provides students with practical technology experiences, provides corporations and governments with cost-effective solutions, provides specific funding for additional scholarships and fellowships to STEP, and provides faculty with additional channels for research opportunities. In fact, the CollegeCareerCorps’ 2004 pilot effort at FedEx was an important catalyst in the actual creation of STEP. One key client of the CollegeCareerCorps is Hilton Hotels Corporation. Hilton has been using the model for over a year to help test a set of Hilton’s key applications. Come hear from Laurie Craig, Manager at Hilton on how the CollegeCareerCorps enhances their software testing organization, processes, and results.

Title: Managing Offshore Outsourcing of Software Testing

Authors: Robin S. Poston, Judith C. Simon, and Radhika Jain

Abstract: Offshore outsourcing is becoming more lucrative in the light of organizations’ desires to become low-cost yet high quality service providers. Within the context of software development, much of the outsourcing research has broadly focused on outsourcing without special attention to the nature of what is being outsourced and whether it has any impact on clientvendor relationships. Also extant research has paid little attention to the challenges brought by offshore outsourcing. With this concern, the current research reports our preliminary findings on various aspects of an offshore outsourcing relationship for software testing. Data collection for this research was conducted at a large multinational service organization.

Title: Importance of Test Automation in System Testing

Author: Sudheer Mamadapur

Abstract: System testing is an integral, costly, and timeconsuming activity in the software development life cycle. As is true for software development in general, reuse of common artifacts can provide a significant gain in productivity. In addition, because testing involves running the system being tested...
under a variety of configurations and circumstances, automation of execution-related activities offers another potential source of savings in the testing process.

**Title:** Interpersonal Conflict between Developers and Testers in Software Development

**Authors:** Xihui Zhang, Jasbir Dhaliwal, and Mark L. Gillenson

**Abstract:** Software development is a complex process that necessitates interaction between diverse individuals playing differentiated roles. These include users, business analysts, system analysts, designers, programmers, and testers. The trend in software development methodologies is towards those that require more frequent and sophisticated interaction between developers (a category that includes system analysts, designers and programmers) and testers. Given the inherent differences between developers and testers and their differentiated roles, the increased interaction is likely to lead to a greater level of interpersonal conflict between them. This research explores the theoretical and pragmatic nature of such developer-tester conflict in terms of both task and relationship elements as well as its impact on software development outcomes. The effectiveness of various conflict management styles that can help regulate the level of developer-tester conflict is also explored. Based on an extensive literature review, this paper advances a theory-driven research model for conflict in developer-tester relations and presents a set of hypotheses that are to serve as the basis for empirical studies of this critical phenomenon.

**Title:** Generalization of MOPS for Software Testing using Finite State Machine

**Authors:** Mohammad S. Sorower, Mohammad F. Ahmed, and Mohammed Yeasin

**Abstract:** This paper augments MOPS [1] by adding specific formal rules for finding potential security vulnerabilities in programs, generally written in C. The key idea is to use simple generalized finite state automata for different categories of security problems with software coding. A set of finite state machines for individual vulnerabilities (i.e., unreachable code, ignored return values) are defined and merged to have a generalized solution. One major advantage of such solution is that security bugs can be eliminated before code is deployed. It is also scalable to adapt new rules. The proposed approach has been tested with the existing Model checking Programs for Security Properties (MOPS) which shows a reasonable improvement of the system.

**Title:** Service Oriented Architecture (SOA) Testing

**Authors:** Khais Khan, Satheesh Kumar, and Prabahakran Sundaram

**Abstract:** Service-Oriented Architecture (SOA) is an architectural style. Applications built using an SOA style deliver functionality as services that can be used or reused when building applications or integrating within the enterprise or trading partners.

- A service is an implementation of well-defined business functionality.
- Application functions are modularized and presented as services.
• Services are loosely coupled
• Service interface is independent of the implementation.
• Services have platform independent, self describing interfaces (XML)
• Messages are formally defined
• Services can be discovered
• Services have quality of service characteristics defined in policies
• Services can be provided on any platform

With today’s enterprises focusing on Service Oriented Architecture (SOA), which is more than just a new name for web services and the reality is that SOA implementations are a composite mix of web components, mid-tier components and exposed legacy and backend systems. Unlike traditional testing approaches, SOA testing approach should encompass all the aspects of business processes and also its integration framework. SOA Testing Strategy should not only focus on just the functionality or the front-end clients but also on the integration layers. SOA initiative has thrown complexities in the integration framework that requires complete testing of business workflows across every heterogeneous technology layer of the SOA at both system and component level. This poses a challenge on testing methodologies, which are currently used for an end-to-end testing scenario.

The paper offers:
• An overview of Service Oriented Architecture (SOA) framework
• SOA implications for Businesses (ROI) and Key Drivers for SOA
• SOA and Web Services
• SOA and OOPS and impact on Software Development Life Cycle (SDLC)
• SOA Governance and Center of Excellence (CoE)
• Test strategies for Service Oriented Architecture (SOA) testing
• Challenges of testing the SOA and addressing those unique challenges of testing SOA integration
• A structured approach to SOA automated testing and its best practices
• Q&A Session

Title: Testing in the Early Phases of the Software Development Life Cycle

Authors: Sarah B. Lee, Hima Puppala, Linda B. Sherrell, and Sajjan Shiva
**Abstract:** The aim of this research is to design a testing process for FedEx Corporation that more thoroughly integrates testing activities throughout the software development life cycle. With the new process, testing personnel will be actively involved in the planning, requirements, and design of the project. This should have the side-benefit of increasing morale among testing personnel as they are committed to the project from its inception to completion. In other words, testing personnel are not treated as a necessary evil “to find bugs” but as critical contributors to the software team.

**Title:** Engaging Testers Early and Throughout the SDLC

**Authors:** Mark L. Gillenson, Xihui Zhang, and Sandra Richardson

**Abstract:** Early in the history of software development, testing was confined to testing the finished code. However, as the practice of software development has evolved, there has been increasing interest in expanding the role of testing upwards in the SDLC stages. This paper presents a set of models for expanding testing throughout the SDLC. It includes comparisons of the models as well as personnel requirements for each one. It also addresses the roles of the testers at the various SDLC stages.

**Title:** Development of Decision Models for Best Use of Software Testing Resources

**Authors:** Charles J. Campbell, Judith C. Simon, and Ronald B. Wilkes

**Abstract:** This focus of this paper is to present the first phase of development of decision models that could be used to determine the best uses of software testing resources. This model is a prototype that will evolve. As data and process information becomes available, the preliminary model presented in this paper will be expanded, with multiple tools used to determine the most appropriate and complete model. The ultimate model could be used to reduce overall costs while applying resources where they provide the greatest value throughout the systems development process.