Literature Review

M.Y. Suhaila, W.K. Wan Mohd Nasir, Member, IAENG, 2011 In this paper, the numerous recent contribution of web application testing approaches reflect the rising awareness and concern for assuring web application’s functionality and performance. The functionality of a web application cannot be compromised, especially if dealing with critical web-based applications such as online banking system, ticketing system and file hosting services. This paper aims to systematically investigate state-of-the-art functional testing approaches for web applications. Focus will be given on testing techniques that address the functional aspect of a web application. In this paper we firstly perform exhaustive literature reviews on state-of-the-art approaches in functional testing of web application. Next, a comparative evaluation framework that consists of several related criteria is developed, and subsequently applied to the selected prominent approaches. Finally, we present a critical discussion on the results of the comparative evaluation. Based on our observations, the evaluation results may indicate the current and future direction in web application functional testing and the viability of these approaches.

Raed Shatnawi and Ahmad Alzu’bi, IMECS 2011 In this paper author describe model-driven development, design understandability is very important to maintain software systems. Software developers use the design models in their endeavor to understand and maintain the final product. Typically, software developers expect consistency between design and implementation artifacts of a software system. However, software systems may deviate from design. Software verification is an important phase to prevent this deviation and to ensure that developers are building the product right. There are few research
studies that were conducted to assess the correspondence of a software implementation to its design. In this paper, author validates the use of a hierarchal quality model to verify the correspondence between design and implementation artifacts.

**Juraj Ivanko , 2011** The goal of this paper is to describe quality assurance and software testing domains. In this, work is focused on small organizations, that develop commercial, off-the-shelf software and have some troubles to perform (or even establish) regression testing process. In this work problems are identified and information and best practices are provided to help such organizations with regression testing, and thereby with overall quality of their products.

**Na Zhang, Xiaoan Bao, Zuohua Ding, 2010** In this paper Author has been gathered significant momentum in both academia and industry in recent years. SCA, which describes a model for building applications and systems using a SOA (Service Oriented Architecture), extends and complements prior approaches to implementing services. However, it is very difficult to test if the service components integration satisfies the requirements. In this paper they propose a formal service component architecture model according to the specification of service component issued as a standard in Mar. 2007. A formulism, called Port Algebra, is developed to describe architecture. Test generation formulas are created to generate test cases for each service component.

**Radoslaw Hofman, 2009** This article analysis non-technical aspects of software quality perception and proposes further research activities for this subject. Cognitive science, psychology, micro economics and other human-oriented sciences
do analyze human behavior, cognition and decision processes. On the other hand engineering disciplines, including software engineering, propose many formal and technical approaches for product quality description.

**Usman Azmat, 2009** This thesis describes Testing is one of the critical processes in software development life cycle. It plays key role in the success of software product by improving its quality. Web-based applications are emerging and evolving rapidly; their importance and complexity is also increasing. Testing is one of the key processes to achieve and ensure the quality of these software or Web-based products. There are many testing challenges involved in Web-based applications. But most importantly interoperability and integration are the most critical testing challenges associated with Web-based applications. There are number of challenging factors involved in both integration and interoperability testing efforts. These integration and interoperability factors have almost 70 percent to 80 percent impact on overall quality of Web-based applications. In software industry different kind of testing approaches are used by practitioners to solve the issues associated with integration and interoperability, which are due to ever increasing complexities of Web-based applications. It is fact that both integration and interoperability are inter-related and it is very helpful to cover all the possible issues of interoperability testing that will reduce the integration testing effort. It will be more beneficial if a dedicated testing team is placed to perform the both integration and interoperability testing.

**Kiran Lakhotia, 2009** This paper is concerned with the problem of automatic test data generation for structural testing criteria, in particular the branch coverage adequacy criterion,
using search based techniques. The primary objective of this paper is to advance the current state-of-the-art in automated search based structural testing. Despite the large body of work within the field of search based testing, the accompanying literature remains without convincing Solutions for several important problems, including: support for pointers, dynamic data structures, and loop assigned flag variables. Furthermore, relatively little work was done to extend search based testing to multi objective problem formulations. One of the obstacles for the wider uptake of search based testing was the lack of publicly available tools, which may have contributed to the lack of empirical studies carried out on real world systems. This paper presents AUSTIN, a prototype structural test data generation tool for the C language. The tool is built on top of the CIL framework and combines a hill climber with a custom constraint solver for pointer type inputs. AUSTIN has been applied to five large open source applications, as well as eight non trivial, machine generated C functions drawn from three real world embedded software modules from the automotive sector. Furthermore, AUSTIN has been compared to a state-of-the-art Evolutionary Testing Framework and a dynamic symbolic execution tool, CUTE. In all cases AUSTIN was shown to be competitive, both in terms of branch coverage and efficiency. To address the problem of loop assigned flags, this paper presents a testability transformation along with a tool that transforms programs with loop assigned flags into flag free equivalents, so that existing search based test data generation approaches can successfully be applied. The paper concludes by introducing multi objective branch coverage. It presents results from a case study of the twin objectives of branch coverage and dynamic memory consumption for both real and synthetic programs.
MIRZA MAHMOOD BAIG, 2009 In this Paper author describe that Software Testing is regarded as an integral part of software development process. It is directly related to software quality and reliability. The main purpose of software testing is to reveal bugs and eliminate them but also to serve as a tool for verification, validation and certification; with the increased complexity of software development; Software Testing is one of the most widely known and essential fields in software development. It may be regarded as a tradeoff between budget, time and quality. Most popular approach to stop testing process is when some or one of the resources i.e., time, budget and test cases are exhausted. It is an accepted fact that proper testing improves the effectiveness and efficiency; it also reduces overall development cost, time, budget and efforts for a system development process. With the concept of systematic formal testing gaining popularity, organizations are now trying to implement standards for properly testing their products before taking them to market. The research undertaken facilitates for testing process proceed from system specification level down to final database development level. The most popular generic existing methodologies or frameworks to develop software are: Water Fall Model, Rapid Application Development (RAD), Boehm Spiral, Capability Maturity Model, Incremental Development, Joint Application Model (JAD), Proto Type, Object Oriented and Structured Model. In each of these system development methodologies, testing is defined as a single phase activity following implementation phase therefore, in the subsequent phases, errors, time and correcting cost increase exponentially. Information system testing is a means of establishing the confidence that the test process undertaken is correct (verification) and meets the desired requirements (validation). Correctness deals with syntactic and semantic
soundness, while validation examines the functionality of the system against the requirements. The entire idea of software testing is based on the concepts of test strategy, test plan, test cases, test data, and test environment. In software testing, the hardware and software requirements are known as Test bed or Test environment. A test case is a proper way (formally called Test bed) of specifying inputs and corresponding expected outputs for the system under testing process. Based on methods to identify test cases, testing process is broadly divided in two domains i.e. Functional testing (i.e. Black box testing) and Structural testing (i.e. White box testing). In the research project we have proposed and developed three types of Software testing strategies, namely:

- Metric Space Based Software Testing Strategy
- Hypothetical Database Testing Strategy
- Quantum Computing Based Software Testing Strategy

Justus Randolph, 2008 In this paper Methodological reviews have been used successfully to identify research trends and improve research practice in a variety of academic fields. Although there have been three methodological reviews of the emerging field of computer science education research, they lacked reliability or generalizability. Therefore, because of the capacity for a methodological review to improve practice in computer science education and because the previous methodological reviews were lacking, a large scale, reliable, and generalizable methodological review of the recent research on computer science education is reported here. The purpose of this methodological review is to provide a methodologically rigorous basis on which to make recommendations for the improvement of computer science education research and to promote informed
dialogue about its practice.

**Mikko Aleksi Mäkinen, 2007** This paper is describing how user actions and system states in an application relate to each other. If the model is thorough enough, describing every user action and corresponding system response, it should be possible to automate the test case creation and not just the test execution. This technique is known as model based testing. In order to deploy model based testing successfully into an organization the following issues must be solved: First, a way is needed to bring model based testing to an organization “softly”, so that the initial efforts required for deployment are not too high. Second, a model based testing tool that creates test cases automatically from the models must be available to be able to exploit the model based testing in an organization. And last, a clear proof of financial benefits must be shown in order to make things happen in the organization.

**Dafydd Vaughan, 2007** This paper describes Today’s software systems have become increasingly complex with more risks associated with failure. This has meant that a greater importance has been placed on testing to ensure the software works correctly and matches the specification. This document looks at software testing and new technologies and techniques that can be used to both improve the efficiency and rigors of testing software.

**Fiona Warman, 2006** In this paper author wants to describes about distribute system testing framework. Distributed systems are becoming increasingly common. However, the testing of these systems is difficult due to their non-linear, stochastic and dynamic behaviors, and limited application-level testing support.
In this thesis, a prototype cluster computing-based test harness has been developed that can be used for performance testing on a variety of distributed systems. Its usefulness is demonstrated through tests conducted on an example distributed system, including using the test harness to perform a parameter search on the system in an iterative fashion.

**Eda Marchetti, 2004** In this paper, author describes that, Software Testing is a critical part of the whole process of development, on which the quality of the products delivered strictly depends. In this paper author present their journey through the world of Software Testing, ranging over many fields from definition to organization, from its applicability to analysis of its effectiveness, because testing activity is not limited to the detection of “bugs”. The general aims are proposing both a global view of the testing phase, which exploits and unifies the knowledge from the industrial reality and the research context and putting research in practice. For this, starting from test planning they analyze the different testing stages pointing out their characteristic problems and presenting our original proposals (methods, tools or new approaches based on UML specifications) for solving them. In particular they also define procedural strategies, which support suitable testing choices since the first phases of development, and provide the reader with practical and quantitative guidance all along the testing phase. All the proposals presented are the result of a strict collaboration with software developers looking for solutions for their problems and improvements in the different activities of the testing process. This collaboration imposes them two important constraints: usability, i.e. the methodologies as far as possible must adapt themselves to the modeling notations and procedures commonly used by industries and real environments and not vice
versa; and automation, i.e., increasing as much as possible the mechanization in test cases derivation, execution and validation, consequently reducing the manual labor. We adopt therefore the leading principle of providing readers with some easy-to apply and low-cost methodologies, which maximize the automation and minimize as much as possible the required additional formalism or ad-hoc effort specifically for testing purposes.

Sarfraz Khurshid, 2003 This paper describes a method for systematic constraint-based test generation for programs that take as inputs structurally complex data, presents an automated SAT-based framework for testing such programs, and provides evidence on the feasibility of using this approach to generate high quality test suites and find bugs in non-trivial programs. The framework tests a program systematically on all no isomorphic inputs. Test inputs are automatically generated from a given input constraint that characterizes allowed program inputs. In unit testing of object-oriented programs, for example, an input constraint corresponds to the representation invariant; the test inputs are then objects on which to invoke a method under test.

Walt Scacchi, 2001 In this paper Software systems come and go through a series of passages that account for their inception, initial development, productive operation, upkeep, and retirement from one generation to another. This article categorizes and examines a number of methods for describing or modeling how software systems are developed. It begins with background and definitions of traditional software life cycle models that dominate most textbook discussions and current software development practices. This is followed by a more comprehensive review of the alternative models of software
evolution that are of current use as the basis for organizing software engineering projects and technologies.