



Toolkit for Defect Detection

Ashok Kumar T
Satyam Computer Services Ltd,
Hyderabad, India

ashokkumar_t@satyam.com

Abstract:

Cost of detecting a defect in successive stages of SDLC is very high when compared with detecting the same defect in the same stage of SDLC where the defect is originated. Techniques of early detection of defects and preventing the same from happening are very big challenges now-a-days. Sometimes Designers/Developers/Testers think there could have been *some more* features and methods in their respective tools to use effectively to find the defects or bugs. But it is difficult to define what that *some* is unless seen or explored. This paper's effort is to compile that *some* and calling as Toolkit. This paper details various upcoming and available features from different tools which can help defect prevention and defect detection techniques. The features in stages of *Requirements Engineering, Test Data preparation, Unit & Integration Testing, Functional & Regression Testing approach* are some interesting features to improve defect detection and defect prevention. Objective is not to short list the tools but to bring to the notice of Solution Architects for their academic interest and the features which can be considered to have in their Toolkit.

Keywords:

Visual Modeling, Slice Technology, Automatic Test Case Generation, Business Logic, Pure Path, Knowledge Sensors, Script free, Model Based Testing, UML

1. Introduction

It is common that many defects are unearthed during Testing and Maintenance stage and also team surprises that why these could not be detected though supposed to be and could have been detected in their respective stage. By the time they realize that it's a slippage and it becomes too late and too costly also to repair the defect. Though the teams are using different tools now-a-days to prevent the defects injection, still the slippage of defects is happening and the surprise is continuing. The reasons could be the testing methods implemented in respective stages of SDLC are not effective or teams may be having different challenges in detecting defects.

Let this paper short list what are challenges the teams are facing in each stage of SDLC.

2. Challenges in Software Development Life Cycle

Stage	Challenges
Requirements	<ul style="list-style-type: none"> i. Lack of visual representation of business logic causing complexity in building Test Cases ii. Managing Test Coverage & Traceability iii. Change Management & Assessing Impact Analysis iv. Standardization, Centralization, Completion & Correctness of project artifacts
Unit & Integration Testing	<ul style="list-style-type: none"> i. Understanding Design & Generating Unit Test Cases ii. Effective Profiling (Code Coverage, Memory Leaks & Performance)
Test Data	<ul style="list-style-type: none"> i. Test data generation on business logic ii. Replication & Data Refreshing iii. Conditional and Combinational data generation iv. Speed and accuracy
Functional & Regression Testing	<ul style="list-style-type: none"> i. Developing scripts with business logic ii. Working in collaboration and Understanding of automated script by non-programmers iii. Maintenance of automated scripts as and when changes occurs in application iv. Tracing to root cause (Design, Requirements, code etc) of defects from test results
Performance Testing	<ul style="list-style-type: none"> i. Deep Diagnostics ii. Identifying the bottle necks in at pace iii. Fine tuning the Application as per desired performance level

Table1. Current Challenges in SDLC

Now the above table leads to a debate on these two topics:

- I. Which stage in SDLC is most prone to defects?
- II. With what tools' features or testing methods these defects can be detected early or prevented to great extent and that too in the same stage of defect originated.

Let this paper discuss the above two topics in detail.

Just analyze the following two images which can address the topic I:

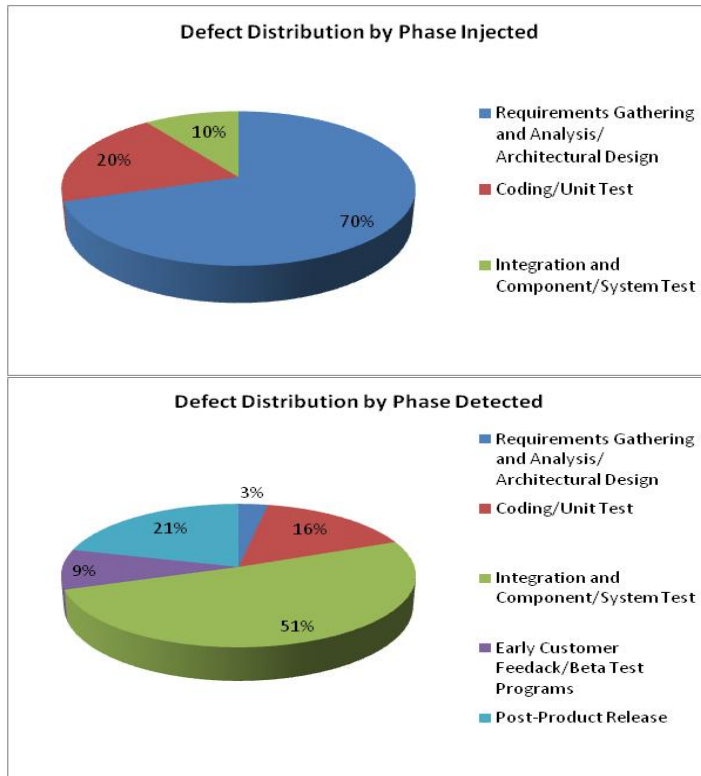


Fig. 1 Defect Introduction and Detection

Fig. 1 from NIST analysis reveals that majority of defects are introduced in Requirements itself, but majority of defects are detected in Integration and System Testing. This late detection of defects proving is very costly in the software industry. From NIST analysis, the following Table2 explains the relative cost (also referred to as cost factors) of repairing defects found at different stages of software development increases the longer it takes to find a bug. Defects introduced during this stage and found in the same stage cost 1X times to fix. But if the same defect is not found until the integration and component/RAISE system test stage, it costs 10 times more to fix.

Requirements Gathering and Analysis/ Architectural Design	Coding/Unit Test	Integration and Component/RAISE System Test	Early Customer Feedback/Beta Test Programs	Post-product Release
1X	5X	10X	15X	30X

Table2. Cost of correcting defects

The following Fig. 2, pie chart illustrates defects distribution by type from *Bug Statistics by Otto Vinter* from *Bug Taxonomy and Statistics* by Boris Beizer.

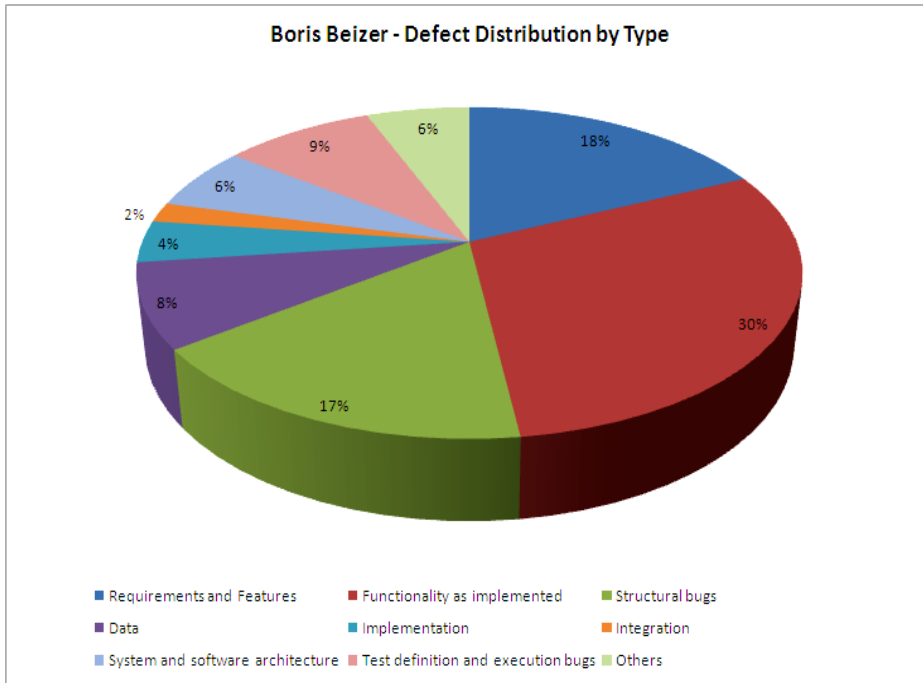
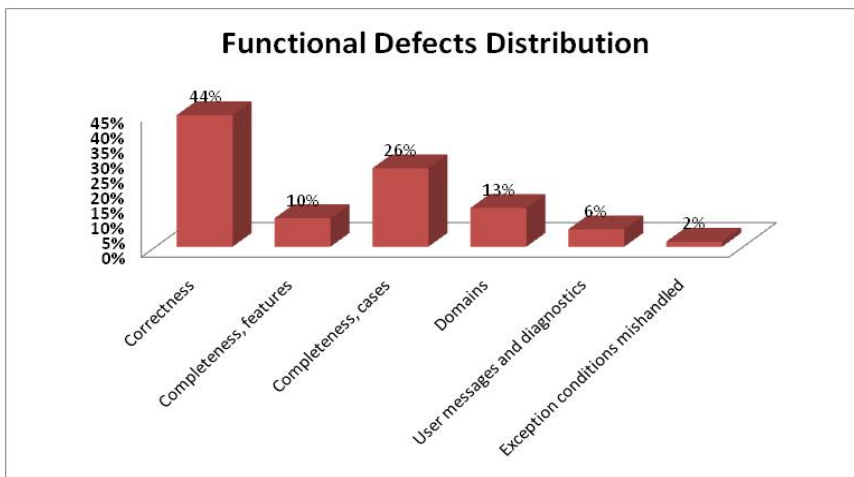


Fig. 2 Boris Beizer Defect Distribution

The distribution of defects by percent wise, the first major three are Functionality as Implemented, Requirements and Features, Structural bugs and data. The below pictures details root causes of distribution of defects in each category.



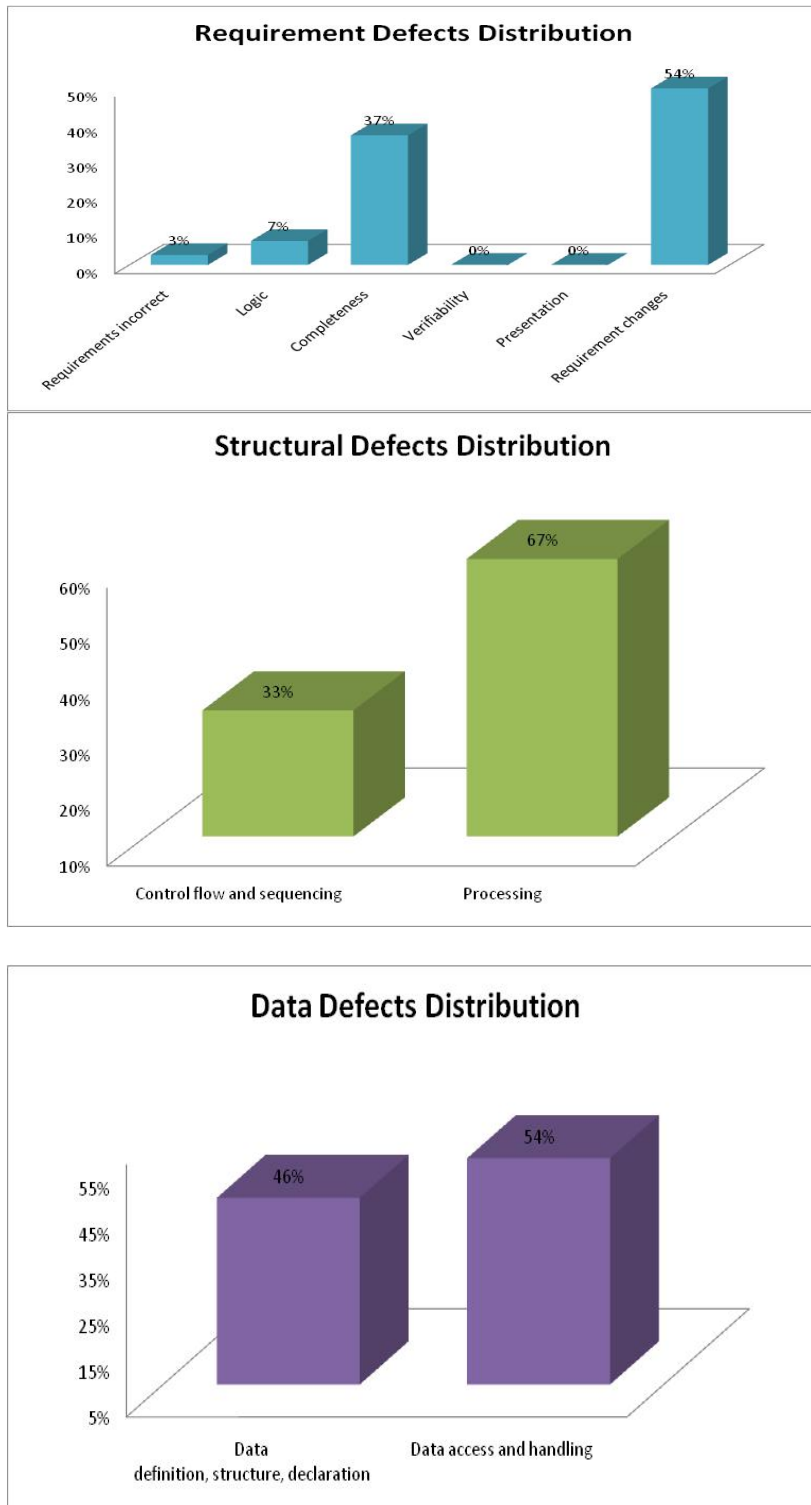


Fig. 3. Root causes of Defects Distribution in the above said four stages

By summarizing the root causes of defects from above Fig. 3. , the majority of defects are because of the following:

- a. Requirement Changes
- b. Completeness
- c. Correctness
- d. Data Access and Data Definition
- e. Processing & Control Flow

In going to the details of the above using Boris Beizer's Bug Taxonomy, it's any one's question and interest to see what the factors are grouped under the above categories.

- a. Requirement Changes
 - i. *Features changes, Domain Changes, User messages, diagnostics and External interfaces*
- b. Completeness
 - i. *Missing Feature, Duplicated, Overlapped feat, Missing Case, Duplicated & Overlapped Case and Extraneous Output data*
- c. Correctness
 - i. *Feature misunderstood, wrong, Feature interactions*
- d. Data Access and Data Definition
 - i. *Initial, default value , Duplication and aliases , Type, Value, Duplication and aliases , Resources, and Access*
- e. Processing & Control Flow
 - i. *General structure, Control logic and predicates , Loops and iterations , Terminal value or condition , Iteration variable processing, Control initialization and/or state , Algorithmic, fundamental , Expression evaluation, Arithmetic expressions, Sign, Logic or Boolean, not control, Initialization, Cleanup, Precision, accuracy and Execution time*

3. Solution based Tools

By considering the above factors, a solution can be thought of if there are tools and testing methods which can address these all factors effectively to detect the defects early. If so, those tools and methods can be "Toolkit for Defect Detection". Also it can be categorized that the toolkit can be used to address the five challenges listed in Table1 in Session 1. i.e. Requirements Engineering, Unit & Integration, Test Data Preparation, Functional Testing and Performance Testing. The toolkit's aim is to detect the defects early and cost can be reduced to great extent. It's nothing but addressing the topic II discussed in Session 2. Now the immediate challenge is to short list the tools and testing methods with features those can address the above five challenges.

The main features desired to have in the tools and methods are Automation, Visual Representation, and understanding design on its own and can track any change. These features can reduce the manual intervention and improve the understanding capabilities among the team members, also improve collaboration and finally can reduce the birth of defects. Here is the list of desired features:

- a) Requirement Management tool:
 - i. Which can address Change Management
 - ii. Having visual representation so that all team members in a project can understand business logic easily and Test Cases can be generated without deviations from Requirements.
 - iii. A Requirement Verification Tool which can verify Completion and Correctness of Requirements so that broken links can be addressed.
- b) Unit & Integration Testing
 - i. Visual representation of code
 - ii. Can address profiling like code coverage, memory leaks etc
 - iii. Can generate Test Cases automatically based on code by understanding application design
 - iv. Dashboard for Management review to track implementation and changes
- c) Functional Testing Tool
 - i. Which can generate Automated Scripts based on business logic
 - ii. Script free, so that coding effort and errors can be avoided
 - iii. Ease of Script Maintenance whenever the requirements & UI are changed
- d) Diagnostics Tool
 - i. Can trace to the code from defect
 - ii. Can trace to the design of the application
 - iii. Can address profiling issues
 - iv. Can improve the Application Performance and helps in Application tuning
- e) Test Data tool
 - i. Which can address correctness of data
 - ii. Can generate data on business logic
 - iii. Easy replication/cloning of test data
 - iv. Re-Usable Project –specific business functions
- f) Testing Model/Approach/Technique
 - i. Testing can be based on Application’s Functionality and Architecture
 - ii. Test Cases can be generated right from begin by understanding Application’s design & model
 - iii. Interfaces can be tested effectively

4. Some Interesting Tools’ Features and Testing Methods

Now let this paper share and introduce some interesting tools features come across in exploring new tools for doing different from the current which may be mapped to the desired features in session 3 based on Boris Beizer’s defect distribution and challenges discussed in session 2(Fig 2 & 3) and in Table1. Combination of these tools and testing methods can be a “Toolkit for Defect Detection” gives more freedom to dig into the causes for defects or to prevent from defect injections.

4.1. Requirement Tools Features –Benefits

The tools which can help and make different in Requirements Management and Verification are listed in the below table.

Tool Name	Feature	Benefits
<p style="text-align: center;">DOORS (Telelogic) for Requirements Management</p>	<p style="text-align: center;">UML & Visual Modeling & Collaboration</p>	<ul style="list-style-type: none"> i. In a single view, DOORS can record and display requirements text, graphics, tables, requirements attributes, change bars, traceability links, and more. ii. DOORS/Analyst's visual modeling of use cases provides an excellent communication with customers and end users in requirements capture. iii. By augmenting textual requirements with pictures or models, DOORS/Analyst helps clarify requirements and create a common understanding between all team members, particularly because the modeling language used is UML, which is widely understood by systems engineers and software developers. iv. DOORS/Analyst provides a 'model aware' environment, which enables the tool to guide in creating error-free models. Models are stored within the DOORS requirements database, allowing them to be 'baselined' in DOORS.

Tool Name	Feature	Benefits
DOORS (Telelogic) for Requirements Management	Interfacing with Test Management Tools & Project communication	<ul style="list-style-type: none"> i. Can be interfaced with HP-Mercury Quality Center. ii. Integrates with TestDirector for QualityCenter provides users visibility of the requirements to create test cases for traceability.
	Intelligent Traceability & Drag & Drop	<ul style="list-style-type: none"> i. Corresponding visual model or text is changed as and when changes occur. ii. Links from DOORS textual requirements to UML model elements ensure full traceability with drag and drop between items on the screen. iii. Change Management & Impact Analysis
Raven (RavenFlow) for Requirements Verification & Validation	Automation of Specification documents & Visual Model Representation	<ul style="list-style-type: none"> i. RAVEN automatically generates clear graphical models of the requirements from the text. This makes reviews accurate and Test Case development too. ii. Generates Test Cases from Use Cases brings 100% test coverage. iii. Finds missing requirements iv. Automatic generation of specification documents
	Collaboration	<ul style="list-style-type: none"> i. Collaboration Server enables teams of business analysts to work together. It enables creation of a corporate library of validated specifications and best practices that all analysts have access to via the web.
	Interfacing with Requirement Management & Modeling tools	<ul style="list-style-type: none"> i. Interfacing with IBM-Rational Requisitepro, DOORS, HP-Quality Center & Visual Studio 2005 ii. RAVEN exports requirements models into popular UML modeling tools such as Rational Software Architect.

Table3. Requirements Management & Verification

4.2. Unit and Integration Testing Tools Features - Benefits

AgitarOne from Agitar, DevPartner & OptimalAdvisor from Compuware can be thought of implementing at unit testing for detection of defects at development stage.

Tool Name	Feature	Benefits
AgitarOne (Agitar)	Automated Tests & Built-in Continuous Integration Testing	<ul style="list-style-type: none"> i. Creates Junit tests with excellent coverage those complements hand-written Junit Tests. ii. Automatic coverage analysis measures test effectiveness iii. With Code Enforcement it detects common coding errors and standard violations iv. Supports continuous integration testing validating the continued correct behavior of the code after any revisions.
	Project Dashboard	<ul style="list-style-type: none"> i. Project summary shows complexity, usage, risk, test quality, coverage, and test pass/fail status over time at a glance
DevPartner Java edition & Optimal Advisor (Compuware)	Design Analysis	<ul style="list-style-type: none"> i. Visualize the dependencies between packages and classes with UML class diagrams ii. Detect cycles in the dependency graph iii. Recover intended architectural layering from a polluted implementation iv. Suggest which dependencies should be removed to improve the structure v. Refactor the source code and the source model vi. View the effects of the refactoring on the dependency structure vii. Verify source code against a design model

Table4. Unit & Integration Tools

4.3. Features of Functional & Diagnostics Tools

This table explains the features which can make testing different in terms of checking defect count and cycle time.

Tool Name	Feature	Benefits
SmarteScript	Script free	<ul style="list-style-type: none"> i. It does not need programming skill set. It reduces the customization time. ii. Script is built with business logic and helps in working collaboration model with business team iii. Generates automated test scripts early in software development process, regardless of the development methodology used. It's designed to keep pace with rapid software and UI changes. Reduces rewriting scripts.

dynaTrace Diagnostics	PurePath Technology	i. Deep tracing of individual transactions critical to the business from the end-users' perspective across several servers and tiers down to code level at runtime
	Failed Transaction Capturing	i. Failed transactions are automatically captured and recorded and the underlying causes for failure are diagnosed.
	Knowledge Sensors	i. Measures the performance-relevant data of an execution path for every single transaction down to the method level. ii. Captures contextual information (e.g., method arguments, exceptions, log events) in order to enable a precise root-cause analysis besides performance metrics (e.g., response- times, CPU usage)
	Load Testing Integration	i. Integration with load-testing tools such as Mercury Load Runner, JMeter and other products.

Table5. Functional & Diagnostics Tool

4.4. Test Data Generation Tools Features - Benefits

Here is a list of Test Data tools with some interesting features come across can make an impact in reducing defect count.

Tool Name	Feature	Benefits
GS Data Generator (GSApps)	Automated data generation	i. Creates test data for software quality assurance testing (QA testing), performance testing, usability testing and database load testing. ii. To generate random test data, meaningful test data and business intelligent test data for system integration testing, ERP, CRM and data warehouse development, and software marketing.
	Project specific functions	i. Project-specific functions with business logic can be easily build and shared among development team members, avoiding effort data replication and saving development time.
TestBase (Softbase)	Slice Technology	i. Enables all Application Developers and QA Personnel in a shop to test together within one set of DB2 objects without impacting each others' results. ii. Enabling Application Developers to run far more tests with small subsets of realistic business data from DB2, VSAM and QSAM data sources iii. Test together from only one copy of test data - Application Developers can retrieve and refresh their

		test data whenever they like without having to contact their DBAs!
IBM Optim	Test Data Management	<ul style="list-style-type: none"> i. Offers comprehensive test data management capabilities for creating realistic, right-sized test databases, protecting data privacy in vulnerable development and testing environments and validating test results. ii. Eliminates costly cloning processes and correct defects early in the development process – where they are cheapest and easiest to fix. iii. Offers Solution Accelerators for Enterprise Applications with business rules.

Table6. Test Data Tools

4.5. Model Based Testing Tool Features

Model based Testing is a new Testing Method, this approach defines the expected behavior of the application and test cases are developed based on the model from Requirements for application under test. With automation features it can reduce manual efforts and can ensure Coverage and Traceability. This model/method can identify the design defects in design stage itself to reduce the defects count.

4.5.1. *Conformiq Qtronic*

- a. Is a true design model driven test automation tool for automatic test derivation, execution and analysis. The tool integrates with major CASE tools and UML editors. This helps in finding the design flaws early in process.
- b. It automatically generates tests from models of the system under test, executes the tests and analyses the results.
- c. It is designed to be used in conjunction with industry-leading model-driven CASE tools such as Borland Together Architect, IBM-Rational Software Architect, and Telelogic TAU. Conformiq Qtronic automatically generates and executed tests based on design models.

4.5.2. *LEIRIOS Test Designer*

- a. Generates Test Cases from functional model of specifications (eg. UML) which eliminates manual efforts
- b. Generates Coverage reports
- c. As Requirements changes, change the model and let Test Designer generate new Test Cases
- d. Traceability, Metrics & Export Managements are additional features

- e. With Model Based Testing with Smart Testing feature, it's possible to generate Test Cases and Executable Test Scripts automatically.
- f. Generated Test Cases can be exported to leading test execution environments, one of those is HP-Mercury's Quality Center and with adopted framework into test automation tool HP-Mercury's QuickTest Professional.

5. Conclusions

Of Course it is understood from Session4 that a single tool does not have all the features to meet requirements, but to mention that there are some interesting tools features and testing methods available around to explore more. To short list the benefits from Toolkit:

Feature	Benefits
Visual Representation	i. Improves collaboration and understanding capabilities. In turn reduces the defect injections
Automation Features	<ul style="list-style-type: none"> i. Reduces manual effort ii. Reduces Defect Injections iii. Improves Traceability iv. Automatic Test Cases generation, Test Data preparations with business logic and Automatic generation of Dashboards improves team's output.
Diagnostics	i. Improves Testing capabilities to root cause the defect and helps Performance Testing and interfacing with leading testing tools improves collaboration in project team
Model Based Testing	i. Test Cases are generated based on design of the application helps in reducing defect count.

Table7. Conclusions

6. References

- [1] NIST Report on "The Economic Impacts of Inadequate Infrastructure for Software Testing"
- [2] Bug Statistics by Otto Vinter from Bug Taxonomy and Statistics by Boris Beizer
- [3] Requirement Management:
<http://www.telelogic.com/Products/doors/doors/index.cfm>
<http://www.ravenflow.com>
- [4] Unit & Integration Testing:
<http://www.agitar.com>
http://www.compuware.com/products/devpartner/5053_ENG_HTML.htm



[5] Test Data:

<http://www.softbase.com>

<http://www.optimsolution.com/Solutions/TestDataManagement.asp>

<http://www.gsapps.com/products/datagenerator/index.html>

[6] Functional & Regression:

http://www.smartsoft.com/products_smartscript.php

[7] Performance & Diagnostics:

http://www.dynatrace.com/en/quality_assurance.aspx

[8] Model Based Testing:

http://www.verifysoft.com/en_qtronic_testautomation.html

<http://www.leirios.com/index.php>

PureConferences

www.pureconferences.com

Pure Conference Solution Pvt Ltd., A-108B, Sector 58, NOIDA 201301, India; Tel: +91 120 4621080

Page 14 of 15

Biography



Ashok Kumar T
Test Architect
Satyam Computers, Hyderabad, India

Ashok Kumar T has been working in Satyam Computers, Hyderabad in India for the last 5 years in Testing Competency unit as a Test Architect. Before that he had worked as consultant in US and returned back to India in 2002. He had handled various projects in VAX and Unix platforms in development and maintenance before migrating to Software Testing. He is a postgraduate in Mathematics from Osmania University with distinction. He had started his career with NRSA, Hyderabad in 1985. His passions are teaching and mentoring in Software Testing, Testing Tools design and Evaluation, Test Architecture and Test Automation.