

Development & Testing Guideline Volume 1

***Internal Document
(Not for circulation)***



©Thinksol Overseas Pvt Ltd

Development Methodology

Software Development Methodology

A recent survey¹ of 8,000 software projects found that the average custom software project exceeded its planned budget by 90 percent and its schedule by 120 percent. It doesn't have to be that way. At Thinksol Overseas Pvt Ltd (TSL) our "battle" tested methodology has allowed us to deliver quality software applications within project defined parameters.

The survey went on to say "that the most common causes of project failure were incomplete requirements, lack of user involvement, lack of resources, unrealistic expectations, lack of executive support, and changing requirements." Our methodology creates the appropriate requirements and manages the other issues.

The Blue Print for Success - Requirements

Requirements simply stated define the purpose and functionality of the application, the inputs and the outputs. They are analogous to the blue print for a home, you would not expect a builder to build your home without plans nor would you expect him to be able to provide a cost to build the home without plans. The same is true for a software application, requirements are necessary for a successful project. Requirements such as "the application needs reports or the application needs to interface with other applications" are unacceptable. Requirements are detailed statements that a programmer can use to design and program by, quality assurance can test the application and the user can understand what they are getting.

How Much Does it Cost - the Great Software Development Paradox

Naturally as a businessperson you need to know the cost of a project. However, how do we provide you with a cost if we don't have requirements, which they themselves cost to produce? And what if after the requirements are done, the cost is higher than your budget? These are the paradoxes in software development, we believe our approach removes the paradox and creates a win-win situation for both you and TSL.

¹ The survey was done by the Standish Group International, Inc. They are a market research and advisory firm specializing in mission-critical software and electronic commerce. The Standish Group uses proven primary research techniques and extensive real world experience to achieve results. Figure 1. Development shouldn't be a dream.

TSL incorporates a 6-Phase process for project development & implementation.

- ✓ **Discovery Phase** We meet with you and go over your basic requirements. We spend a brief time with you learning about your business and your business needs. From this meeting, we produce the **Order of Magnitude** document describing the project and giving you a rough picture of what will be involved with the project development.
- ✓ **Requirements Phase** In this phase we go over your requirements in detail, really getting the information that is necessary to design the proper solution for you. The time that is spent up front in this phase is saved ten-fold in the Development phase. This involves effort from you as well, but helps to get all of your processes ironed out. Upon the completion of this phase, you will be provided with a **Requirements Document** (your requirements), a **Logistics Document** (Physically describing what needs to be done, and how implementation will occur), and a **Risk Assessment Document** (Possible project Risks, if any, and how they will be handled).
- ✓ **Design Phase** This is where we do the actual system design. This equates to the blueprints for building a house. We will design features and solutions to all of the requirements from the Requirements phase. This phase might include system prototypes and a testing plan. Upon the completion of this phase, there will be a **Project Plan** (describing project implementation and schedules), a **Support Plan** (detailing a proper support plan for the project), and a methodology for how we will handle system modifications and enhancements.
- ✓ **Development Phase** In this phase we build the application. This phase includes both Internal and External testing. We will revisit our **Risk Assessment Document**, and make sure that something unexpected will not come up during implementation.
- ✓ **Implementation Phase** We will install the application and train users. If necessary we will perform any data conversions for the new system.
- ✓ **Support Phase** We will provide continued support based on the agreement from the design phase. This is a tried and true methodology that we have developed from our 10 years of experience. This process is aimed at providing you the most comprehensive solution to fulfill your business needs. Our process also provides open communication channels in every development phase and incorporates signed and documented plans throughout implementation of the project.

Quality Assurance

A major concern for customers looking to outsource software development is quality. Can they trust the cost and time estimates they have been given? Will the finished product be to a high standard? There are three ways in which Thinksol addresses this concern.

1. Process Quality
2. Systems Engineer (staff) Quality
3. Product Warranty

Process Quality

Business System Overview

- Conduct System Overview
- Document business system Requirements.
- Study Existing system Documentation.
- Understand and document client expectations and requirements.
- Deliver appreciation document And Obtain sign-off.
- Deliver Maintenance Process Document.

Migrate application offshore

- Test Connectivity
- Test all development and maintenance processes.
- Introduce offshore and onshore teams.

Periodic Team reviews

- Conduct periodic quality Reviews
- Improve cycle time and response times

Systems Engineer Quality

Thinksol always retains top class talents to maintain quality production. This knowledge retention process ensures immaculate solutions within the given time frame. Systems Engineer (staff) Quality is addressed through our competency and staff certification scheme.

Product Warranty

Product Warranty is addressed by warranting all of our products and services during contract negotiations.

When we engage in a client project, we are committed to deliver a quality solution on time and on budget. Meeting this commitment takes more than just good intentions. – It needs experienced project managers and the latest development tools. We couple this extensive experience with Thinksol's base modules. These modules are re-usable frameworks, which can be efficiently and effectively tailored to meet specific client requirements; thereby increasing productivity and reducing implementation time considerably.

Systems Engineer Quality

People are at the core of Thinksol's success by providing a challenging and innovating work environment, Thinksol helps, attracts, develops and retain top talent. Our project managers, team leaders and systems engineers have over several hundred thousand hours of solution design and implementation experience. Systems Engineer (staff) Quality is addressed through our competency and staff certification scheme.

Product Warranty

Product Warranty is addressed by warranting all of our products and services during contract negotiations.

MANAGING THE COST OF CHANGE IN SOFTWARE DEVELOPMENT

TSL's methodology combines the best features of **Rational Unified Process (RUP)** and **Extreme Programming (XP)**. These methodologies provide the necessary tools to quickly respond to the problems that might arise during the project and assure, as much as possible, the success of the project. The main focus of the methodology is the achievement of quick results while preserving high quality standards and allowing quick response to sudden changes during the development process.

WHY IS THE METHODOLOGY BETTER?

- Achievement of quick results while preserving high quality standards and allowing quick response to sudden changes during the development process.
- Methodology for teams developing software in the face of vague or rapidly changing requirements
- Risk Management
 - Short release cycles
 - Smallest release that makes the most business sense
 - Comprehensive suite of tests that are run and rerun after every change.
 - Address highest priorities first.

THE DEVELOPMENT PROCESS

- Features are developed
- Iterations are performed
- A fully functional application is delivered at the end of each iteration Figure 3

Next page schematic diagram shows our approach towards Development Methodology

Developing cost effective solutions

We are in the threshold of a new era. The entire global economy is under recession and prime business houses and industries across the universe are thriving to migrate towards a cost effective quality software development process. This consciousness leads towards the immergence of offshore software development. Looking into the feasibility, factually India becomes the prime location for outsourcing the software developments.

Outsourcing with Thinksol can achieve more than just cost savings. It can help organizations meet strategic business objectives and effect radical change. Thinksol can help to accelerate cost reductions and cope with increasing demand for the business to improve the speed and quality of IT services.

Offshore Software Development Centre (OSDC)

The purpose of OSDC is to provide efficient and feasible remote extensions to client's own IT team, maintain synergistic leverage to client's core competence with our highly specialized IT skills, and ensure a smooth transition for complete application outsourcing (maintenance) to offshore dedicated, responsive teams.

Product companies and end clients keen on complete applications. OSDCs are meant for situations where large continuous and planned resources are primarily required for the success of the product.

ODC is client's software development team's virtual extension in India. It comprises:

- Dedicated team of development staff.
- Dedicated hardware, software and network infrastructure.
- Dedicated high-speed communication or Internet links.
- IP protection mechanisms.
- Well defined, ISO certified offshore Software Development Methodologies.

OSDC primarily consists of two services;

- OSDS - Offshore Software Development Services
- OMS - Offshore Maintenance Center

Domain Expertise

Thinksol offers OSDC approach for catering to development and maintenance outsourcing requirements; we focus on broad-based long-term requirements. We have the skills to handle evolving technology. We maintain qualified staff to deliver quality network services.

OSDCs combine the cost-effectiveness of the offshore single sourced approach with the value proposition of focused methodologies and solutions.

These OSDCs concentrate on software applications or products. We create value and lie down or improve IT procedures and refine details.

Approach & Responsibility:

Thinksol approach follows a sequential chain in identifying the levels for final Offshore Development Process. A Client can tap into our capacity at whatever level required.



Quantum responsibility: Thinksol undertakes responsibility for deployment of smaller modules of the application/product involving understanding of client requirements, design, development and implementation.

Direct interaction with the end customer: Thinksol's team directly interacts with the end user to analyze and understand their requirements. In this phase Thinksol takes complete responsibility of Life Cycle Development for the complete application or product. This means satisfactorily meeting the end user requirements and at the same time reducing the effort on the part of the customer towards the above process.

Improvement and suggestions by OSDC for product/service enhancement and efficiency: The Thinksol team actively identifies areas for improvement and subsequently takes corrective measures. At this stage, the OSDC team utilizes the vast expertise available within Thinksol and the best practices from various information sources.

Achieve business objectives for the client: At this stage, the Thinksol team is fully aware of the business objectives of the customer. The OSDC team is now in a position to contribute significantly towards meeting the joint business objectives of the customer and Thinksol. Thus, Thinksol believes in moving up the value chain and ensuring fruitful long-term relationship for the client.

OMS (Offshore Maintenance Services)

Virtual extension of client's IT department or support department. We carry the responsibility of maintenance and support for existing applications or products.

Significant Features of the Maintenance Services:

- Seamless Offshore/Onshore Team Model with a high degree of Ownership
- Well defined ISO certified Processes for Quality Management
- Change Requests & Release based mode of operations
- 24x7 application and product support from Offshore.
- Maintenance Plans
- Service Level Agreements (SLA)

Approach:

Client's team hands over production support, maintenance and enhancement activities to the Thinksol team. This process extends over three distinct phases:

1. System understanding
2. Hand over
3. Maintenance

Phase 1) System Understanding

- Obtain initial functional and system overview from client.
- Study technical environment of the system.
- Study standards and conventions.
- Obtain system inventory.
- Study existing system documentation.
- Study and document important system characteristics.
- Conduct due-diligence and document application details.
- Determine and document QA, Acceptance and Production Migration procedures.
- Establish offshore procedures.
- Prepare draft project plan consisting of resource, scheduling, risk management, contingency plan and configuration management.
- Obtain sign-off from the client for the project plan.

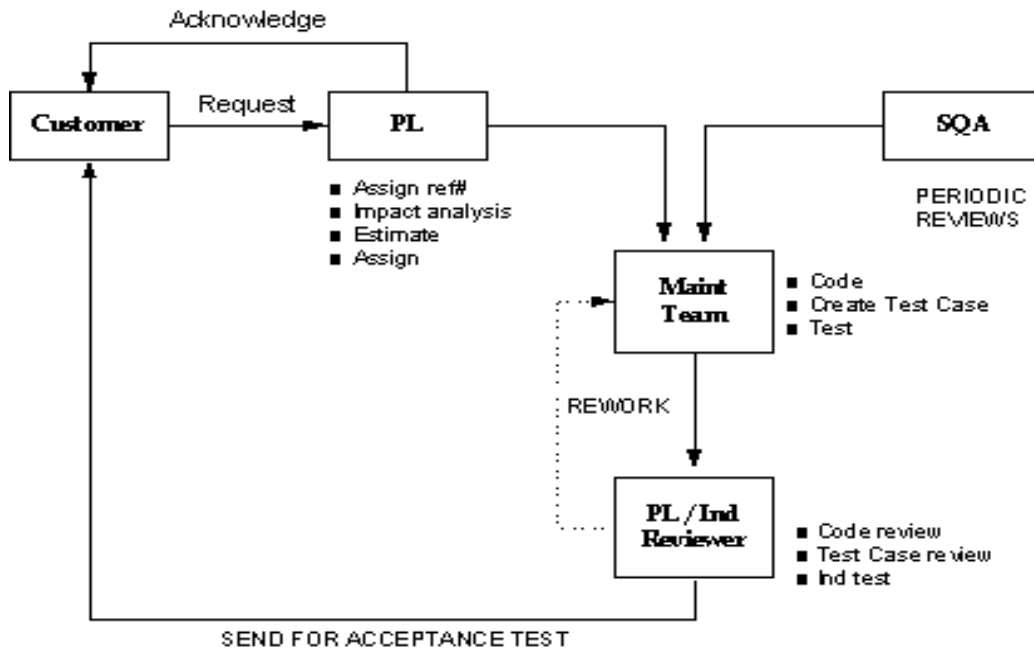
Phase 2) Hand Over

- A team is deputed onsite for this phase.
- The onsite team establishes interfaces with offshore team.
- The onsite team logs change requests, defines change specifications, and carries out coding, unit testing and acceptance testing.
- Set up offshore team with requisite infrastructure.
- Orientation of offshore team for the project.
- Trial runs with the change requests being passed to the offshore team, specifications, coding and unit testing being preformed offshore and acceptance testing being performed onsite.
- Establish confidence with the client on onsite/offshore maintenance model.

Phase 3) Maintenance Activities Phase

- The onsite team will continue to provide production support.
- The onsite team will log requests, carry out preliminary estimation and interface with offshore team.
- The offshore team will design change specifications, carry out coding and testing and interface with onsite team.

Schematic Request Flow-



Our Specialization	Technical Strength
<p>SAP IMPLEMENTATION/ ABAP PROGRAMMING</p> <p>WEB SERVERS (WEBLOGIC, WEBSHERE, iPLANET, ORACLE 9iAS)</p> <p>JAVA/J2EE FRAMEWORK</p> <p>OBJECT ORIENTED COMPUTING TECHNOLOGY AND GUI</p> <p>N-TIER CLIENT SERVER ARCHITECTURE</p> <p>CRM SYSTEMS</p> <p>MICROSOFT MIDDLEWARE TOOLS</p> <p>MOBILE COMMERCE SOLUTIONS</p> <p>WAP TECHNOLOGIES</p> <p>PERFORMANCE TESTING AND TUNING</p> <p>SOFTWARE QUALITY ASSURANCE</p> <p>VPN, LDAP</p> <p>VOIP</p> <p>NETWORKING</p> <p>NETWORK MANAGEMENT: SNMP, JAMPI, WBEM</p> <p>SAS PROGRAMMING</p> <p>DATA WAREHOUSING</p> <p>INTERNET TECHNOLOGY</p> <p>INTERNET PAYMENT GATEWAY</p> <p>DISTRIBUTED OBJECT TECHNOLOGY (CORBA, UML, JAVA)</p> <p>EMBEDDED SYSTEMS</p> <p>LEGACY SYSTEM MIGRATION SERVICES</p> <p>EURO COMPLIANCE SERVICES</p> <p>REAL TIME OPERATING SYSTEM (RTOS)</p> <p>REAL TIME DISTRIBUTED SYSTEM, TRAINING</p> <p>BACK OFFICE OPERATIONS</p> <p>BUSINESS MANAGEMENT SOFTWARE- ACCOUNTING</p> <p>DATABASE REENGINEERING</p>	<p>Hardware Intel, Power PC, Himalaya, Alpha, Pro liant, AS 400, RS6000, HP-PA, HP3000, HP9000, SUN</p> <p>OS Windows 9x/NT/ME/ Advanced 2000/2000/XP, Unix (HP-UX, AIX, SCO, TRUE64, Solaris, VMS), Linux, OS2</p> <p>Development Environment Visual C++, Java, J2EE, Cobol, RPG SQL, ABAP, PL/SQL, C++, SAS BI, Data warehousing, OLAP, CRM (Siebel), Powerbuilder, Delphi, Visual Basic, MS.net, Developer 2000, IBM Websphere, BEA Weblogic, Oracle9iAS, iplanet</p> <p>RDBMS Oracle, MS-SQL, Informix, DB2, DB2/400, Sysbase, Ingres</p> <p>Groupware MS-Office, Lotus Notes Domino</p> <p>Designing Tools UML, Rational Rose, RUP, Jdeveloper, Jbuilder</p> <p>ERP BPCS, JD Edwards, SAP, BaaN, PeopleSoft, MFG/Pro</p> <p>OLAP Tool Cognos PowerPlay, Informatca, SAS entire BI suits, Oracle Discoverer, SeaGate entire BI suite, Brio etc.</p> <p>CRM Sieble, Vantive etc.</p>

Testing Overview

Testing Overview

Need for Testing

Software Testing is an integral part of the software development process. It is associated with any process that produces a product and used to determine the status of the product during and after the build. The primary benefit of testing is that it results in improved quality.

Testing is advantages in several ways. Firstly, the defects found guide corrections to the software. Secondly, even if the defects found are not corrected, testing gives an idea as to how reliable the software is. Thirdly, over time, the record of defects found reveals the most common kinds of defects, which can be used for developing appropriate preventive measures such as training, proper design and reviewing.

Types of Testing

Testing is to be undertaken at various stages throughout the development process and after. The 11 Step Testing Process is highlighted at the end.

Functional Test

Testing that ignores the internal mechanism of a system or component and focuses solely on the outputs generated in response to selected inputs and execution conditions. In other words black box testing.

1.1.1 Non-Functional Test

Testing that ignores the functional aspect of the system and concentrates on the performance of the system like the response time, speed of execution etc.

Level of Testing

- ✓ Unit Testing
- ✓ Integration Testing
- ✓ System Testing
- ✓ Limit testing
- ✓ Volume testing
- ✓ Performance testing
- ✓ Load testing
- ✓ Stress testing
- ✓ Recovery testing
- ✓ Regression testing
- ✓ Security testing
- ✓ Compatibility testing
- ✓ Consistency testing
- ✓ Reliability testing
- ✓ Installation Testing
- ✓ User Interface Testing
- ✓ Disaster Testing
- ✓ Acceptance Testing

Unit Testing

Testing of individual pieces of a process or program is called module testing, unit testing or element testing. It is usually performed by developer.

To test a unit of code (program or set of programs) using the Unit Test Specification, after coding is completed. Involves the basic testing of a piece of code, the size of which often undefined in practice, although it is usually a function or a subroutine.

This is white box testing in the sense that the developer knows the internal structure of the unit under test and is concerned with how the testing affects the internal operation.

1.1.2 Integration testing

It is the testing of related programs, modules or units of code. It also validates that multiple parts of the system interact according to the system design.

High level testing includes System testing and Acceptance testing. It determines the extent to which the requirements of the system are met.

The process of testing interfaces and data flows between the programs within a sub-system, and between the sub-systems within a system.

Integration testing means that the tester must look for bugs in the relationship and the interfaces between pairs of components and groups of components under test.

If a system or product is a set of stand alone utilities that don't share data or invoke one another, there is no need for integration testing.

1.1.3 System testing

It is a test, executed by the developer or independent test team in a laboratory environment that should demonstrate that the developed system or subsystems meet the requirements set in the functional and quality specifications

The process of proving that the system meets its stated design specifications (design documents) w.r.t criteria such as recoverability, maintainability and security.

1.1.4 Limit Testing

One test is developed for each of the documented system limits (such as memory, number of files open, number of users). These tests are designed to investigate how the system reacts to data which is maximal or nominal in the sense of attaining limit specified either in the software requirements specification or in the user manual.

The system is also tested beyond the limits specified to verify whether sufficient safety margins have been built in.

For example, if the system is developed for ten users, then ten or more users use the system and the effect is studied.

1.1.5 Volume Testing

Such tests submit the system to large volume of data. For example, if the software requirements specification indicates that the system can store up to 10MB of data, the system is tested with 10MB and more.

1.1.6 Performance testing

Performance Testing includes testing that the system will perform as specified at predetermined levels, including wait times, static processes, dynamic processes, transaction processes. Performance is also tested at the Client/Browser and Server levels.

1.1.7 Load Testing

Load Testing is boundary condition testing. This type of testing checks the size of files a product can work with, the number of printers it can drive, and the number of terminals, modems, bytes of memory that it can manage. Load Testing increases the uptime of mission-critical systems by spotting bottlenecks in systems under large user stress scenarios before they happen in a production environment.

1.1.8 Stress Testing

Stress testing is designed to determine if the system can function when subjected to large volumes or larger than would be normally expected. This will test input transaction, internal tables/database, disk space and computer capacity. If the application functions adequately under test, it can be assumed that it will function properly with normal volume of work.

1.1.9 Recovery testing

Recovery testing involves testing how the system copes with failures in parts of the supporting infrastructure.

Recovery testing is a vital area of testing for safety-critical and similar systems.

System response to power failure is a critical area is tested.

For example, try switching off the power when the system is active and then switch on the power and study the impact.

Failure conditions are set up to determine the response of the system.

1.1.10 Regression Testing

Regression testing is re testing unchanged segments of the application system, it normally involves re-running test that have been previously executed to ensure that the same results can be achieved currently as were achieved when the segment was last tested, it is also on in which cost/benefit needs to be carefully evaluated or large amounts of effort can be expended with minimum pay back.

1.1.11 Security testing

Tests are performed that attempt to break the system's security, such as the access of database-held data by unauthorized users. For example, if the system is password protected, try testing by giving wrong passwords or try to break the password.

1.1.12 Compatibility testing

Compatibility of existing data to the higher version begin developed is preferred by users. This is especially true for packaged products where an enhancement should be compatible with the old data used by users.

1.1.13 Consistency testing

The data and results are verified to be consistent across the application. For example, the sales column in stock reports and the total sales in regionwise reports must match.

1.1.14 Reliability testing

Testing for reliability requires periods of prolonged use at varying loading levels, or, possibly, constant value at peak level.

If the reliability level (mean time between failure) is desired to be for long periods such as years, the test is conducted for short periods and the results are extrapolated to years.

1.1.15 Installation Testing

The ease of installation is checked here. Wherever installation counts are applicable, the numbers of installations specified are tested. The system is checked for higher number of installations than specified, and also uninstalled to check if the installation count is reduced.

Migration of data from older version to the new version is verified.

If the software to work on multiple platforms, then all platforms, or at least the critical platforms, are tested.

1.1.16 User Interface Testing

The user interface is checked against the design or the requirements Specification.

The user interface are then tested the user manual, on-line help and the software requirements specification together, to find areas which comply with the stated functionality. it also determines where the user manual does not describe the interface correctly or in sufficient detail.

Tests of invalid input and attempts to break the system are created.

Interfaces are evaluated for ease of understanding and use, as defined by the quantified measures given in the software requirements specification.

1.1.17 Disaster Testing

Disaster testing is a mechanism that simulates problems in the original environment so that an alternative processing environment can be tested. While it is not possible to stimulate all environments into which an application system may be moved, knowing that it can transfer between two different environments provides the high probability that other moves will not cause major complications.

1.1.18 Acceptance Testing

It is a test, executed by the user(s) and system manager(s) in an environment simulating the operational environment to the greatest possible extent, that should demonstrate that the developed system meets the functional and quality requirements.

Phases of Testing

- ✓ Test Planning
- ✓ Test Design
- ✓ Test Execution
- ✓ Test Reporting
- ✓ Test Management activities

Test Planning

It is a contract between the testers and the project team describing the role of testing in the project

Test Plan is an evolving document, which provides background information of the software to be tested, testing objectives, scope, approach and techniques, resources and schedules necessary for completion of testing, risks, testing tasks, environment, tools and individual responsibilities of the testing team.

The status report and final summary report are prepared based on the test plan.

Test plan

- ✓ Defines 'what' & 'how' to test
- ✓ Information of the software to be tested
- ✓ Defines testing objectives, scope, approach and techniques
- ✓ Effort and schedules
- ✓ Risks and risk mitigation plan
- ✓ Project management
- ✓ Test environment set up requirements.
- ✓ Pass/fail criteria

The test plan provides an overview of the testing effort for the product and should typically contain the following items :

Introduction - Include references to all relevant policy and standards documents, and high-level product plans.

Test Plan Identifier - A unique name or number.

Test Items - A test item is a software item (function, module, feature, whatever) that is to be tested. Include references to specifications (e.g. requirements and design) and manuals (e.g. user, operations, and installation).

Features to be tested - Cross-reference them to test design specifications.

Features not to be tested. Which ones and why not.

Approach - Describe the overall approach to testing that does it, main activities, techniques, and tools used for each group of features.

Item pass/fail criteria - How does a tester decide whether the program passed or failed a given test?

Suspension Criteria and resumption requirements – List anything that would cause your to stop testing until it's fixed. What would have to be done to get you to restart testing? What tests should be redone at this point?

In other words, the test plan should contain

- ✓ Test deliverables
- ✓ Testing tasks
- ✓ Environmental needs
- ✓ Responsibilities
- ✓ Staffing and training needs
- ✓ Schedule
- ✓ Risks and contingencies
- ✓ Sign Off Documents

Test Design

After analysing the business requirements, the scenarios to be tested are identified according to the test plan. For each scenario various test cases are derived and corresponding test conditions and test data are prepared. Test cases should be designed to ensure maximum coverage of the system to be tested. Testing environment should be setup as detailed in the test plan.

The test design should include specifications for the test design, test cases and the testing procedure.

Typical activities include:

- ✓ Gathering and studying the testing framework
- ✓ Identifying the test scenarios and the test cycles
- ✓ Preparing the test cases, test scripts
- ✓ Preparing the test data
- ✓ Determining if Automation is required - Scope, no of runs, frequency
- ✓ Test environment management

Test Execution

The objective of test execution is to determine whether the system performs as per the specifications in an executable mode. According to the test cases test data are prepared for execution. If automated the required scripts are prepared. The inputs include Test Plan, Test Specification (Test Case, Test Condition and Test data), Test Script, Test procedure and the output include the test log and defect log.

The test cases are executed according to the test plan. The expected and actual results are captured with the relevant evidence. After verifying the results, the unexpected results are entered into defect log. In case a fault occurs preventing the further execution of the test plan the test execution is suspended till fault is fixed. Retesting of the fixed faults is performed, if within the scope of the test plan.

In short the Test execution phase includes

- ✓ Execute test scripts - manual or automated
- ✓ Defect reporting and tracking
- ✓ Capture Test results

- ✓ Compile Test reports
- ✓ Causal Analysis
- ✓ Re-test

Test Reporting

The final deliverable from a Test is the Test Report, that summarizes:

- ✓ Test Requirements
- ✓ Features tested
- ✓ Metrics
- ✓ Issues & Resolutions
- ✓ Defect Log
- ✓ Recommendations

1.1.19 Test Management activities

Responsibilities for test planning, design, execution, and evaluation are explicitly assigned. The test objectives, types of tests to be conducted and test schedule are documented in the test plan. The test work products are stored in a configuration-controlled repository. The test work products and test activities are audited periodically. Summary report summarizes and compares all the testing activities with numerical and graphical representation

Test Management Activities include

- ✓ Configuration
- ✓ Environment
- ✓ Test Selection, Re-test, and Test Data Management

TSL Approach to Testing Assignments

- ✓ Start-up Phase
- ✓ Knowledge Transition Phase
- ✓ Service Phase

Start-up Phase

The startup phase includes the following activities:

- ✓ System Scope and Size
- ✓ Schedules and Deliverables
- ✓ Acceptance Criteria
- ✓ Knowledge Transfer Plan
- ✓ TSL, Unisys Responsibilities
- ✓ Escalation Procedure

1.1.20 Knowledge Transition Phase

The key phase where Business Analysts and Test Analysts gather information about a client's system and then pass on the information across the team. This phase involves extensive study and documentation of client processes and methodologies. Typical activities include:

- ✓ Business Knowledge Transfer

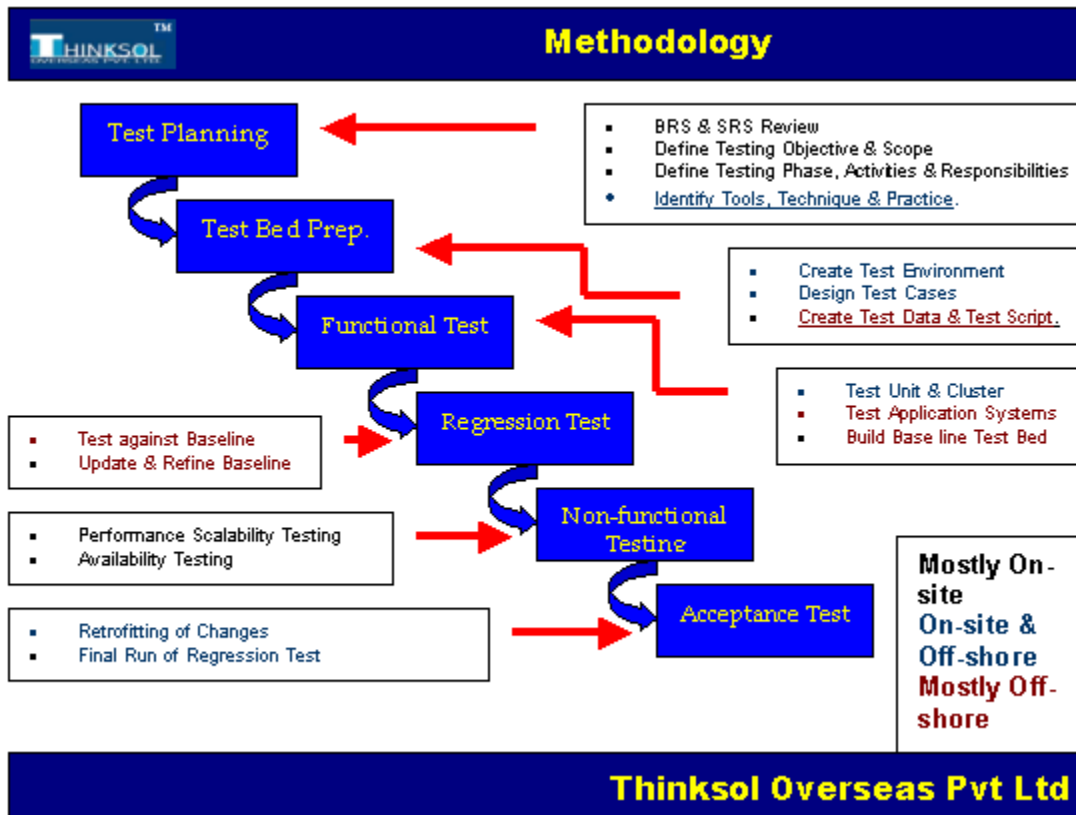
- ✓ Application Systems Knowledge Transfer
- ✓ Set up Standards & Processes
- ✓ Independent Test covering parts of the system
- ✓ Quality Improvement Plan

1.1.21 Service Phase

This phase succeeds the Knowledge transition phase and accounts for the actual execution of the project. Typically a very lean onsite/ offshore ratio is maintained during this phase. Activities include:

- ✓ Long-term Planning
- ✓ Test Requirement Gathering
- ✓ Resource Planning
- ✓ Test Plans
- ✓ Test Case/Script build
- ✓ Obtain s/w upgrades
- ✓ Execute Test Cases
- ✓ Compile Test results
- ✓ Submit Test Report
- ✓ Quality improvements

TSL Methodology



Key activities in the methodology include:

- ✓ Test Planning
- ✓ Test Bed Creation
- ✓ Functional Test
- ✓ Non-Functional Testing
- ✓ Regression Testing
- ✓ Acceptance Test

Test Planning

Test planning includes the following steps

- ✓ BRS and SRS Reviews
- ✓ Define Testing Objectives & Scope
- ✓ Define Testing Phases, Activities, Responsibilities
- ✓ Identify Tools, Techniques and Practices

1.1.22 Test Bed Creation

Use of Test Beds greatly improves the productivity and quality of testing activities on a system thereby reducing cost of maintenance in the long run.

The core phases of the Test bed creation methodology are

- ✓ Create a Test Data baseline by extracting from Production using tools

To extract the data one needs to know what to extract, how much to extract. 'What to' is described by defining the various data elements in the system to a repository and also the relationship between the data elements.

- ✓ The fact that the data is extracted from Production ensures that real-life scenarios are built into the test data.
- ✓ Structured methods to ensure completeness of test data
- ✓ Perform Coverage Analysis using agent-based tools
- ✓ This phase pertains to finding about the coverage of program logic, by the data that has been extracted. This is achieved mostly by using tools.
- ✓ The tool should have the ability to store and compare the coverage results from two different runs. The comparison report includes information on what parts of the program were common to two runs, which paths were covered in which run etc.
- ✓ Compress or Complete the Test Data
- ✓ The coverage reports available at this point would help find
- ✓ The extent to which each of the programs has been exercised
- ✓ Detailed information on what statements have missed execution
- ✓ Based on the coverage obtained, two alternatives exist to improve the test bed. One concerns with the efficiency of the test data and the other with sufficiency. The two alternatives are described below.

Test Bed Compression

To arrive at an optimum size of the test data, which does not decrease the coverage of programs, is called Test Bed Compression. The idea here is to slice the test data, based on some selection criteria and to run the application on the different slices. The coverage reports of the different runs are compared. The comparison would identify redundant or nearly redundant slices, thereby helping to create a smaller test bed.

1.1.22.1 Test Bed Completion

The coverage report identifies statements that missed execution. This would normally be the consequence of lack of necessary input data. Also, most of the error handling logic would not have been exercised, because the data was extracted from Production.

A White-box analysis would help in identifying the data requirements, to complete the test bed. Here we can use the TSL tool DARPRAN that performs Static Analysis of programs.

1.1.22.2 Generic Test Cases

Extracting test cases, which are general to a particular line of business. For example, the New Business functionality's test cases that are common across all life products can be reused.

Functional Test

As discussed earlier, Functional testing is undertaken to ensure that a system meets its intended functionality. It is primarily "black box" in nature. Key activities include:

- ✓ Test Units & Clusters (Unit, Integration)
- ✓ Test application systems (System)
- ✓ Build Baseline Test Bed

1.1.23 Non-Functional Testing

This is done after ensuring that the system meets its functional requirements. The goal is to look at the performance of the system under varying conditions of load, volume, stress etc. Key tests include:

- ✓ Performance/Scalability Testing
- ✓ Availability Testing

1.1.24 Regression Testing

This test as described earlier is to validate that no existing functionality is impacted due to the introduction of new code (for adding a new module/ function etc). This type of testing is the best candidate for automation.

- ✓ Test against Baseline
- ✓ Update and Refine Baseline

1.1.25 Acceptance Test

The final stage in the testing process, this looks at key functional/ performance features. Activities include:

- ✓ Retrofitting of changes
- ✓ Final run of Regression Test

Standards

To ensure high quality of testing, adherence to benchmarks or standards is of high importance. Commonly adhered standards include

- ✓ TSL Standards
- IEEE Standards.
- TSL-QMS-34
- ✓ Client Standards
- ✓ Domain specific/ technology specific standards (HIPAA etc.)

1.2 Review

BRS and SRS Reviews

- ✓ Review test designs

Programmers and testers can benefit from a second review. Reviews of test designs needn't be as elaborate as product design reviews, but a short check of the testing approach and the resulting tests can find significant omissions at low cost.

- ✓ Review the documentation and test installation procedures.

Reviewing the documentation means checking that all the procedures and examples in the documentation work. Testing installation procedures is a good way to avoid making a bad first impression.

- ✓ Review of test plan.
- ✓ Review of test results.