* **How to write effective Test cases, procedures and definitions**

Writing **effective test cases**is a skill and that can be achieved by some experience and in-depth study of the application on which test cases are being written.

Here I will share some **tips on how to write test cases, test case procedures and some basic test case definitions.**

**What is a test case?**“A test case has components that describes an input, action or event and an expected response, to determine if a feature of an application is working correctly.” Definition by [Glossary](http://www.ee.oulu.fi/research/ouspg/sage/glossary/)

There are levels in which each test case will fall in order to avoid duplication efforts.  
**Level 1:** In this level you will write the **basic test cases from the available specification** and user documentation.  
**Level 2:** This is the **practical stage** in which writing test cases depend on actual functional and system flow of the application.  
**Level 3:** This is the stage in which you will group some test cases and **write a test procedure**. Test procedure is nothing but a group of small test cases maximum of 10.  
**Level 4:** **Automation of the project.** This will minimize human interaction with system and thus QA can focus on current updated functionalities to test rather than remaining busy with regression testing.

So you can observe a systematic growth from no testable item to a Automation suit.

**Why we write test cases?**The basic objective of writing test cases is **to validate the testing coverage of the application.** If you are working in any CMMi company then you will strictly follow test cases standards. So writing test cases brings some sort of standardization and minimizes the ad-hoc approach in testing.

**How to write test cases?**Here is a simple [test case format](http://readyset.tigris.org/nonav/templates/test-case-format.html)

**Fields in test cases:**

**Test case id:  
Unit to test:** What to be verified?  
**Assumptions:  
Test data:** Variables and their values  
**Steps to be executed:  
Expected result:  
Actual result:  
Pass/Fail:  
Comments:**

So here is a basic format of test case statement:

**Verify  
Using**[tool name, tag name, dialog, etc]  
**With** [conditions]  
**To** [what is returned, shown, demonstrated]

**Verify:** Used as the first word of the test case statement.  
**Using:** To identify what is being tested. You can use ‘entering’ or ‘selecting’ here instead of using depending on the situation.

For any application basically you will cover all the **types of test cases including functional, negative and boundary value test cases.**

Keep in mind while writing test cases that all your **test cases should be simple and easy to understand**. Don’t write explanations like essays. Be to the point.

**Try writing the simple test cases as mentioned in above test case format. Generally I use Excel sheets to write the basic test cases. Use any tool like ‘Test Director’ when you are going to automate those test cases.**

* **Tips for Writing Test Cases**

One of the most frequent and major activity of a Software Tester (SQA/SQC person) is to write Test Cases. First of all, kindly keep in mind that all this discussion is about ‘Writing Test Cases’ not about designing/defining/identifying TCs.

There are some important and critical factors related to this major activity.Let us have a bird’s eye view of those factors first.

**a. Test Cases are prone to regular revision and update:**

We live in a continuously changing world, software are also not immune to changes. Same holds good for requirements and this directly impacts the test cases. Whenever, requirements are altered, TCs need to be updated. Yet, it is not only the change in requirement that may cause revision and update to TCs.

During the execution of TCs, many ideas arise in the mind, many sub-conditions of a single TC cause update and even addition of TCs. Moreover, during regression testing several fixes and/or ripples demand revised or new TCs.

**b. Test Cases are prone to distribution among the testers who will execute these:**

Of course there is hardly the case that a single tester executes all the TCs. Normally there are several testers who test different modules of a single application. So the TCs are divided among them according to their owned areas of application under test. Some TCs related to integration of application, may be executed by multiple testers while some may be executed only by a single tester.

**c. Test Cases are prone to clustering and batching:**

It is normal and common that TCs belonging to a single test scenario usually demand their execution in some specific sequence or in the form of group. There may be some TCs pre-requisite of other TCs. Similarly, according to the business logic of AUT, a single TC may contribute in several test conditions and a single test condition may consist of multiple TCs.

**d. Test Cases have tendency of inter-dependence:**

This is also an interesting and important behavior of TCs that those may be interdependent on each other. In medium to large applications with complex business logic, this tendency is more visible.

The clearest area of any application where this behavior can definitely be observed is the interoperability between different modules of same or even different applications. Simply speaking, wherever the different modules or applications are interdependent, the same behavior is reflected in the TCs.

**e. Test Cases are prone to distribution among developers (especially in TC driven development environment):**

An important fact about TCs is that these are not only to be utilized by the testers. In normal case, when a bug is under fix by the developers, they are indirectly using the TC to fix the issue. Similarly, where the TCD development is followed, TCs are directly used by the developers to build their logic and cover all scenarios, addressed by TCs, in their code.

[](http://www.softwaretestinghelp.com/wp-content/qa/uploads/2011/07/test-cases-writing-tips.jpg)

**So, keeping the above 5 factors in mind, here are some tips to write test cases:**

**1. Keep it simple but not too simple; make it complex but not too complex:**

This statement seems a paradox, but I promise it is not so. Keep all the steps of TCs atomic, precise with correct sequence and with correct mapping to expected results, this is what I mean to make it simple.

Now, making it complex in fact means to make it integrated with the Test Plan and other TCs. Refer to other TCs, relevant artifacts, GUIs etc. where and when required. But do this in balanced way, do not make tester to move to and fro in the pile of documents for completing single test scenario. On the other hand do not even let the tester wish you had documented these TCs in some compact manner. While writing TCs, always remember that you or someone else will have to revise and update these.

**2. After documenting Test cases, review once as Tester:**

Never think that the job is done once you have written the last TC of the test scenario. Go to the start and review all the TCs once, but not with the mind of TC writer or Testing Planner. Review all TCs with the mind of a tester. Think rationally and try to dry run your TCs. Evaluate that all the Steps you have mentioned are clearly understandable, and expected results are in harmony with those steps.

The [test data](http://www.softwaretestinghelp.com/tips-to-design-test-data-before-executing-your-test-cases/) specified in TCs is feasible not only for actual testers but is according to real time environment too. Ensure that there is no dependency conflict among TCs and also verify that all references to other TCs/artifacts/GUIs are accurate because, testers may be in great trouble otherwise.

**3. Bound as well as ease the testers:**

Do not leave test data on testers, give them range of inputs especially where calculations are to be performed or application’s behavior is dependent on inputs. You may divide the test item values among them, but never give them liberty to choose the test data items themselves. Because, intentionally or unintentionally, they may use same test data and some important test data may be ignored during the execution of TCs.

Keep the testers eased by organizing TCs according to the testing categories and related areas of application. Clearly instruct and mention which TCs are inter-dependent and/or batched. Similarly, explicitly indicate which TCs are independent and isolated so that tester may manage his overall activity at his or her own will.

**4. Be a Contributor:**

Never accept the FS or Design Document as it is. Your job is not just to go through the FS and identifying the Test Scenarios. Being a quality related resource, never hesitate to contribute. Suggest to developers too, especially in TC-driven development environment. Suggest the drop-down-lists, calendar controls, selection-list, group radio buttons, more meaningful messages, cautions, prompts, improvements related to usability etc.

**5. Never Forget the End User**

The most important stakeholder is the ‘End User’ who will actually use the AUT. So, never forget him at any stage of TCs writing. In fact, End User should not be ignored at any stage throughout the SDLC, yet my emphasis so far is just related to my topic. So, during the identification of test scenarios, never overlook those cases which will be mostly used by the user or are business critical even of less frequent use. Imagine yourself as End User, once go through all the TCs and judge the practical value of executing all your documented TCs.

**Conclusion:**

Test Case Writing is an activity which has solid impact on the whole testing phase. This fact makes the task of documenting TCs, very critical and subtle. So, it should be properly planned first and must be done in well-organized manner. The person who is documenting the TCs must keep in mind that, this activity is not for him or her only, but a whole team including other testers and developers, as well as the customer will be directly and indirectly affected by this work.

So, the due attention must be paid during this activity. “Test Case Document” must be understandable for all of its users, in an unambiguous way and should be easily maintainable. Moreover, TC document must address all important features and should cover all important logical flows of the AUT, with real time and practically acceptable inputs.

* **Tips to design test data before executing your test cases**

I have mentioned importance of proper test data in many of my previous articles. Tester should check and update the test data before execution of any test case. In this article I will provide **tips on how to prepare test environment so that any important test case will not be missed by improper test data and incomplete test environment setup.**

### What do I mean by test data?

If you are writing test case then you need input data for any kind of test. Tester may provide this input data at the time of executing the test cases or application may pick the required input data from the predefined data locations. The test data may be any kind of input to application, any kind of file that is loaded by the application or entries read from the database tables. It may be in any format like xml test data, system test data, SQL test data or stress test data.

**Preparing proper test data is part of the test setup.** Generally testers call it as testbed preparation. In testbed all software and hardware requirements are set using the predefined data values.

If you don’t have the systematic approach for building test data while[writing and executing test cases](http://www.softwaretestinghelp.com/how-to-write-effective-test-cases-test-cases-procedures-and-definitions/) then there are chances of missing some important test cases. Tester can’t justify any bug saying that test data was not available or was incomplete. It’s every testers responsibility to create his/her own test data according to testing needs. Don’t even rely on the test data created by other tester or standard production test data, which might not have been updated for months! Always create fresh set of your own test data according to your test needs.

Sometime it’s not possible to create complete new set of test data for each and every build. In such cases you can use standard production data. But remember to add/insert your own data sets in this available database. One good way to design test data is use the existing sample test data or testbed and append your new test case data each time you get same module for testing. This way you can build comprehensive data set.

### How to keep your data intact for any test environment?

Many times more than one tester is responsible for testing some builds. In this case more than one tester will be having access to common test data and each tester will try to manipulate that common data according to his/her own needs. Best way to keep your valuable input data collection intact is to keep personal copies of the same data. It may be of any format like inputs to be provided to the application, input files such as word file, excel file or other photo files.

**Check if your data is not corrupted:**  
Filing a bug without proper [troubleshooting](http://www.softwaretestinghelp.com/how-to-get-your-all-bugs-resolved/) is bad a practice. Before executing any test case on existing data make sure that data is not corrupted and application can read the data source.

### How to prepare data considering performance test cases?

Performance tests require very large data set. Particularly if application fetching or updating data from DB tables then large data volume play important role while testing such application for performance. Sometimes creating data manually will not detect some subtle bugs that may only be caught by actual data created by application under test. If you want real time data, which is impossible to create manually, then ask your manager to make it available from live environment.

I generally ask to my manager if he can make live environment data available for testing. This data will be useful to ensure smooth functioning of application for all valid inputs.

**Take example of my search engine project ‘statistics testing’**. To check history of user searches and clicks on advertiser campaigns large data was processed for several years which was practically impossible to manipulate manually for several dates spread over many years. So there is no other option than using live server data backup for testing. (But first make sure your client is allowing you to use this data)

### What is the ideal test data?

Test data can be said to be ideal if for the minimum size of data set all the application errors get identified. Try to prepare test data that will incorporate all application functionality, but not exceeding cost and time constraint for preparing test data and running tests.

### How to prepare test data that will ensure complete test coverage?

Design your test data considering following categories:  
**Test data set examples:**  
**1) No data:** Run your test cases on blank or default data. See if proper error messages are generated.

**2) Valid data set:** Create it to check if application is functioning as per requirements and valid input data is properly saved in database or files.

**3) Invalid data set:** Prepare invalid data set to check application behavior for negative values, alphanumeric string inputs.

**4) Illegal data format:** Make one data set of illegal data format. System should not accept data in invalid or illegal format. Also check proper error messages are generated.

**5) Boundary Condition data set:** Data set containing out of range data. Identify application boundary cases and prepare data set that will cover lower as well as upper boundary conditions.

**6) Data set for performance, load and stress testing:** This data set should be large in volume.

This way creating separate data sets for each test condition will ensure complete test coverage.

**Conclusion:**

Preparing proper test data is a core part of “project test environment setup”. Tester cannot pass the bug responsibility saying that complete data was not available for testing. Tester should create his/her own test data additional to the existing standard production data. Your test data set should be ideal in terms of cost and time. Use the tips provided in this article to categorize test data to ensure complete functional test cases coverage.

Be creative, use your own skill and judgments to create different data sets instead of relying on standard production data while testing.

**What is your experience?**

Have you faced problem of incomplete data for testing? How you managed to create your own data then? Share your simple tips and tricks to create or use test data

* **How to find a bug in application? Tips and Tricks**

A very good and important point. Right? **If you are a software tester or a QA engineer then you must be thinking every minute to find a bug in an application. And you should be!**

I think finding a *blocker bug*like any **system crash** is often rewarding! No I don’t think like that. **You should try to find out the bugs that are most difficult to find and those always misleads users.**

**Finding such a subtle bugs is most challenging work and it gives you satisfaction of your work.** Also it should be rewarded by seniors. I will share my experience of one such subtle bug that was not only difficult to catch but was difficult to reproduce also.  
I was testing one module from my search engine project. I do most of the activities of this project manually as it is a bit complex to automate. That module consist of traffic and revenue stats of different affiliates and advertisers. So testing such a reports is always a difficult task. When I tested this report it was showing the data accurately processed for some time but when tried to test again after some time it was showing misleading results. It was strange and confusing to see the results.

There was a cron (cron is a automated script that runs after specified time or condition) to process the log files and update the database. Such multiple crons are running on log files and DB to synchronize the total data. There were two crons running on one table with some time intervals. There was a column in table that was getting overwritten by other cron making some data inconsistency. It took us long time to figure out the problem due to the vast DB processes and different crons.

**My point is try to find out the hidden bugs in the system that might occur for special conditions and causes strong impact on the system. You can find such a bugs with some tips and tricks.**

**1)** **Understand the whole application** or module in depth before starting the testing.

**2)** Prepare **good test cases** before start to testing. I mean give stress on the functional test cases which includes major risk of the application.

**3)** Create a **sufficient test data** before tests, this data set include the test case conditions and also the database records if you are going to test DB related application.

**4)** Perform repeated tests with **different test environment**.

**5)** Try to find out the **result pattern** and then compare your results with those patterns.

**6)** When you think that you have completed most of the test conditions and when you think you are tired somewhat then **do some monkey testing.**

**7)** Use your previous **test data pattern** to analyse the current set of tests.

Try some **standard test cases** for which you found the bugs in some different application. Like if you are testing input text box try inserting some html tags as the inputs and see the output on display page.

**9)** Last and the best trick is try very hard to find the bug As if you are testing only to break the application!

* **Top 20 practical software testing tips you should read before testing any application**

**I wish all testers read these software testing good practices.** Read all points carefully and try to implement them in your day-to-day testing activities. This is what I expect from this article. If you don’t understand any testing practice, **ask for more clarification in comments below.** After all you will learn all these testing practices by experience. But then why not to learn all these things before making any mistake?

**Here are some of the best testing practices I learned by experience:**

**1)** **Learn to analyze your test results thoroughly.** Do not ignore the test result. The final test result may be ‘pass’ or ‘fail’ but troubleshooting the root cause of ‘fail’ will lead you to the solution of the problem. Testers will be respected if they not only log the [bugs](http://www.softwaretestinghelp.com/why-does-software-have-bugs/" \o "Why software have bugs?)but also provide solutions.

**2)** **Learn to maximize the**[**test coverage**](http://www.softwaretestinghelp.com/sqa-processes-how-to-test-complete-application/) every time you test any application. Though 100 percent test coverage might not be possible still you can always try to reach near it.

**3)** To ensure maximum test coverage **break your application under test (AUT) into smaller functional modules.** Write test cases on such individual unit modules. Also if possible break these modules into smaller parts.  
**E.g:** Lets assume you have divided your website application in modules and ‘accepting user information’ is one of the modules. You can break this ‘User information’ screen into smaller parts for writing test cases: Parts like UI testing, security testing, functional testing of the ‘User information’ form etc. Apply all form field type and size tests, negative and validation tests on input fields and write all such test cases for maximum coverage.

**4) While**[**writing test cases**](http://www.softwaretestinghelp.com/web-testing-example-test-cases/), write test cases for intended functionality first i.e. for valid conditions according to requirements. Then write test cases for invalid conditions. This will cover expected as well unexpected behavior of application under test.

**5) Think positive.** Start testing the application by intend of finding bugs/errors. Don’t think beforehand that there will not be any bugs in the application. If you test the application by intention of finding bugs you will definitely succeed to find those [subtle bugs](http://www.softwaretestinghelp.com/how-to-find-a-bug-in-application-tips-and-tricks/) also.

**6)** Write your test cases in requirement analysis and design phase itself. This way you can ensure all the requirements are testable.

**7)** Make your **test cases available to developers prior to coding.**Don’t keep your test cases with you waiting to get final application release for testing, thinking that you can log more bugs. Let developers analyze your test cases thoroughly to develop quality application. This will also save the re-work time.

**8 )** If possible identify and **group your test cases for regression testing.** This will ensure quick and effective manual regression testing.

**9)**Applications requiring critical response time should be thoroughly tested for performance. **Performance testing is the critical part of many applications.** In manual testing this is mostly ignored part by testers due to lack of required performance testing large data volume. Find out ways to test your application for performance. If not possible to create test data manually then write some basic scripts to create test data for performance test or ask developers to write one for you.

**10) Programmers should not test their own code.** As discussed in our [previous post](http://www.softwaretestinghelp.com/developers-are-not-good-testers/), basic unit testing of developed application should be enough for developers to release the application for testers. But you (testers) should not force developers to release the product for testing. Let them take their own time. Everyone from lead to manger know when the module/update is released for testing and they can estimate the testing time accordingly. This is a typical situation in agile project environment.

**11) Go beyond requirement testing.** Test application for what it is not supposed to do.

**12)** While doing regression testing **use previous bug graph** (Bug graph – number of bugs found against time for different modules). This module-wise bug graph can be useful to predict the most probable bug part of the application.

**13)** Note down the new terms, concepts you learn while testing. Keep a text file open while testing an application. **Note down the testing progress, observations** in it. Use these notepad observations while preparing final test release report. This good habit will help you to provide the complete unambiguous test report and release details.

**14)**Many times testers or developers make changes in code base for application under test. This is required step in development or testing environment to avoid execution of live transaction processing like in banking projects. **Note down all such code changes done for testing purpose** and at the time of final release make sure you have removed all these changes from final client side deployment file resources.

**15) Keep developers away from test environment.**This is required step to detect any configuration changes missing in release or deployment document. Some times developers do some system or application configuration changes but forget to mention those in deployment steps. If developers don’t have access to testing environment they will not do any such changes accidentally on test environment and these missing things can be captured at the right place.

**16)** It’s a good practice to **involve testers right from software requirement and design phase.** These way testers can get knowledge of application dependability resulting in detailed test coverage. If you are not being asked to be part of this development cycle then make request to your lead or manager to involve your testing team in all decision making processes or meetings.

**17)** Testing teams should **share best testing practices**, experience with other teams in their organization.

**18) Increase your conversation with developers** to know more about the product. Whenever possible make face-to-face communication for resolving disputes quickly and to avoid any misunderstandings. But also when you understand the requirement or resolve any dispute – make sure to communicate the same over written communication ways like emails. Do not keep any thing verbal.

**19) Don’t run**[**out of time**](http://www.softwaretestinghelp.com/what-if-there-isnt-enough-time-for-thorough-testing/)**to do high priority testing tasks.**Prioritize your testing work from high to low priority and plan your work accordingly. Analyze all associated risks to prioritize your work.

**20) Write clear, descriptive, unambiguous bug report.** Do not only provide the bug symptoms but also provide the effect of the bug and all possible solutions.

Don’t forget testing is a creative and challenging task. Finally it depends on your skill and experience, how you handle this challenge.

**Over to you:**

Sharing your own testing experience, tips or testing secrets in comments below will definitely make this article more interesting and helpful!!

* **10 Tips you should read before automating your testing work**

I was getting too many questions on when and how to automate testing process. Instead of answering them individually I thought it would be better to have some discussion here. I will put my thoughts about **when to automate, how to automate or should we automate our testing work?** I know there some of our readers are smarter than me. So it would be always a good idea to start a meaningful discussion on such vast topic to get in-depth idea and thoughts from experts from different areas and their experience in automation testing.

**Why Automation testing?**  
**1)** You have some new releases and bug fixes in working module. So how will you ensure that the new bug fixes have not introduced any new bug in previous working functionality? You need to test the previous functionality also. So will you test manually all the module functionality every time you have some bug fixes or new functionality addition? Well you might do it manually but then you are not doing testing effectively. Effective in terms of company cost, resources, Time etc. Here comes need of Automation.  
**- So automate your testing procedure when you have lot of regression work.**

**2)** You are testing a web application where there might be thousands of users interacting with your application simultaneously. How will you test such a web application? How will you create those many users manually and simultaneously? Well very difficult task if done manually.  
**- Automate your load testing work for creating virtual users to check load capacity of your application.**

**3)** You are testing application where code is changing frequently. You have almost same GUI but functional changes are more so testing rework is more.  
**- Automate your testing work when your GUI is almost frozen but you have lot of frequently functional changes.**

**What are the Risks associated in Automation Testing?**  
There are some distinct situations where you can think of automating your testing work. I have covered some risks of automation testing here. If you have taken decision of automation or are going to take sooner then think of following scenarios first.

**1) Do you have skilled resources?**  
For automation you need to have persons having some programming knowledge. Think of your resources. Do they have sufficient programming knowledge for automation testing? If not do they have technical capabilities or programming background that they can easily adapt to the new technologies? Are you going to invest money to build a good automation team? If your answer is yes then only think to automate your work.

**2) Initial cost for Automation is very high:**  
I agree that manual testing has too much cost associated to hire skilled manual testers. And if you are thinking automation will be the solution for you, Think twice. Automation cost is too high for initial setup i.e. cost associated to automation tool purchase, training and maintenance of test scripts is very high.  
There are many unsatisfied customers regretting on their decision to automate their work. If you are spending too much and getting merely some good looking testing tools and some basic automation scripts then what is the use of automation?

**3) Do not think to automate your UI if it is not fixed:**  
Beware before automating user interface. If user interface is changing extensively, cost associated with script maintenance will be very high. Basic UI automation is sufficient in such cases.

**4) Is your application is stable enough to automate further testing work?**  
It would be bad idea to automate testing work in early development cycle (Unless it is agile environment). Script maintenance cost will be very high in such cases.

**5) Are you thinking of 100% automation?**  
Please stop dreaming. You cannot 100% automate your testing work. Certainly you have areas like performance testing, regression testing, load/stress testing where you can have chance of reaching near to 100% automation. Areas like User interface, documentation, installation, compatibility and recovery where testing must be done manually.

**6) Do not automate tests that run once:**  
Identify application areas and test cases that might be running once and not included in regression. Avoid automating such modules or test cases.

**7) Will your automation suite be having long lifetime?**  
Every automation script suite should have enough life time that its building cost should be definitely less than that of manual execution cost. This is bit difficult to analyze the effective cost of each automation script suite. Approximately your automation suite should be used or run at least 15 to 20 times for separate builds (General assumption. depends on specific application complexity) to have good ROI.

**Here is the conclusion:**

Automation testing is the best way to accomplish most of the testing goals and effective use of resources and time. But you should be cautious before choosing the automation tool. Be sure to have skilled staff before deciding to automate your testing work. Otherwise your tool will remain on the shelf giving you no ROI. Handing over the expensive automation tools to unskilled staff will lead to frustration. Before purchasing the automation tools make sure that tool is a best fit to your requirements. You cannot have the tool that will 100% match with your requirements. So find out the limitations of the tool that is best match with your requirements and then use manual testing techniques to overcome those testing tool limitations. Open source tool is also a good option to start with automation. To know more on choosing automation tools read my previous posts [here](http://www.softwaretestinghelp.com/choosing-automation-tool-for-your-organization/) and[here](http://www.softwaretestinghelp.com/will-automated-testing-tools-make-testing-easier/).

**Instead of relying 100% on either manual or automation use the best combination of manual and automation testing.** This is the best solution (I think) for every project. Automation suite will not find all the bugs and cannot be a replacement for real testers. Ad-hoc testing is also necessary in many cases.