

# Vital Importance of Data in Functional Testing

White Paper - Version 1.0

## The Roles of Data in Functional Testing

Testing consumes and produces large amounts of data. Data describes the initial conditions for a test, forms the input, and is the medium through which the tester influences the software. Data is manipulated, extrapolated, summarized and referenced by the functionality under test, which finally spews forth yet more data to be checked against expectations. Data is a crucial part of most functional testing. This paper sets out to illustrate some of the ways that data can influence the test process, and will show that testing can be improved by a careful choice of input data. In doing this, the paper will concentrate most on data-heavy applications; those which use databases or are heavily influenced by the data they hold. The paper will not consider areas where data is important to non-functional testing, such as operational profiles, massive datasets and environmental tuning.

## A System is Programmed by its Data

Many modern systems allow tremendous flexibility in the way their basic functionality can be used. Configuration data can dictate control flow, data manipulation, presentation and user interface. A system can be configured to fit several business models, work (almost) seamlessly with a variety of cooperative systems and provide tailored experiences to a host of different users. A business may look to an application's configurability to allow them to keep up with the market without being slowed by the development process; an individual may look for a personalized experience from commonly-available software.

## Functional Testing Suffers if Data is poor

Tests with poor data may not describe the business model effectively, they may be hard to maintain, or require lengthy and difficult setup. They may obscure problems or avoid them altogether. Poor data tends to result in poor tests, which take longer to execute.

- **Good Data is Vital for Reliable Results:** An important goal of functional testing is to allow the test to be repeated with the same result, and varied to allow diagnosis. Without this, it is hard to communicate problems to coders, and it can become difficult to have confidence in the QA team's results, whether they are good or bad. Good data allows diagnosis, effective reporting, and allows tests to be repeated with confidence
- **Good Data can Help Testing stay on Schedule:** An easily comprehensible and well-understood dataset is a tool to help communication. Good data can greatly assist in speedy diagnosis and rapid re-testing. Regression testing and automated test maintenance can be made speedier and easier by using good data, while an elegantly-chosen dataset can often allow new tests without the overhead of new data.

## Problems which can be caused by poor data

Most testers are familiar with the problems that can be caused by poor data. The following list details the most common problems. Most projects experience these problems at some stage - recognizing them early can allow their effects to be mitigated. Unreliable test results.

- Degradation of test data over time.
- Increased test maintenance cost
- Reduced flexibility in test execution
- Obscure results and bug reports
- Larger proportion of problems can be traced to poor data
- Less time spent hunting bugs
- Confusion between developers, testers and business
- Requirements problems can be hidden in inadequate data
- Simpler to make test mistakes
- Unwieldy volumes of data
- Business data not representatively tested
- Inability to spot data corruption caused by bugs
- Poor database/environment integrity

## Organising the data

A key part of any approach to data is the way the data is organised; the way it is chosen and described, influenced by the uses that are planned for it. A good approach increases data reliability, reduces data maintenance time and can help improve the test process. Good data assists testing, rather than hinders it.

## Permutations

Most testers are familiar with the concept of permutation; generating tests so that all possible permutations of inputs are tested. Most are also familiar with the ways in which this generally vast set can be cut down. Pairwise or combinatorial testing addresses this problem by generating a set of tests that allow all possible pairs of combinations to be tested. Typically, for non-trivial sets, this produces a far smaller set of tests than the brute-force approach for all permutations. The same techniques can be applied to test data; the test data can contain all possible pairs of permutations in a far smaller set than that which contains all possible permutations. This allows a small, easy to handle dataset - which also allows a wide range of tests.

## Partitioning

Partitions allow data access to be controlled, reducing uncontrolled changes in the data. Partitions can be used independently; data use in one area will have no effect on the results of tests in another. Data can be safely and effectively partitioned by machine / database / application instance, although this partitioning can introduce configuration management problems in software version, machine setup, environmental data and data load/reload. A useful and basic way to start with partitions is to set up, not a single environment for each test or tester, but to set up three shared by many users, so allowing different kinds of data use. These three have the following characteristics:

### Safe area

- Used for enquiry tests, usability tests etc.
- No test changes the data, so the area can be trusted.
- Many testers can use simultaneously

### Change area

- Used for tests which update/change data.
- Data must be reset or reloaded after testing.
- Used by one test/tester at a time.

### Scratch area

- Used for investigative update tests and those which have unusual requirements.
- Existing data cannot be trusted.
- Used at tester's own risk!

Testing rarely has the luxury of completely separate environments for each test and each tester. Controlling data, and the access to data, in a system can be fraught. Many different stakeholders have different requirements of the data, but a common requirement is that of exclusive use. While the impact of this

requirement should not be underestimated, a number of stakeholders may be able to work with the same environmental data, and to a lesser extent, setup data - and their work may not need to change the environmental or setup data. The test strategy can take advantage of this by disciplined use of text / value fields, allowing the use of 'soft' partitions.

'Soft' partitions allow the data to be split up conceptually, rather than physically. Although testers are able to interfere with each others tests, the team can be educated to avoid each others work. Data partitions help because:

- Allow controlled and reliable data, reducing data corruption / change problems
- Can reduce the need for exclusive access to environments/machines

### Involving 'The Business'

Bringing QA into the development process early is a known way of reducing the cost of fault detection and resolution, and can improve time to market. The development process is initiated outside IT or QA teams by 'The Business', generally with some sort of business opportunity and associated requirements generation. A variety of different roles are encompassed by this vague and non-standard term, including those of regulator, customer, user and 'boss' - but one quality they have in common is that they know about the data rather better than they are likely to know about the technical internals of the system. Data can enable effective communication between QA and 'The Business' in this early stage. 'The Business' is good at looking at data. Data can be compared with – and is often equivalent to – data used by existing process and systems. It can be easier to understand than tests, and it is often simpler to make a link between input and output data than it is to describe the internal process. Sharing data with 'The Business' can help to illustrate and model situations. This has the following advantages:

- Helps focus when requirements are vague
- Increases trust and understanding by improving ease of communication
- Helps early user identification of problems
- Improves familiarity and effectiveness of User Acceptance Testing

However, sharing the data has disadvantages. In sharing data, it is important to recognise and control the following problems:

- Data creep
- Vague requirements can lead to vague data.
- Incomplete data can lead to incomplete testing

### Data Load and Data Maintenance

An important consideration in preparing data for functional testing is the ways in which the data can be loaded into the system, and the possibility and ease of maintenance.

## Loading the data

Data can be loaded into a test system in three general ways.

- Using the system you're trying to test.
- Using a data load tool-Data load tools directly manipulate the system's underlying data structures
- Not loaded at all -some tests simply take whatever is in the system and try to test with it.

**Environmental data** tends to be manually loaded, either at installation or by manipulating environmental or configuration scripts. Large volumes of setup data can often be generated from existing datasets and loaded using a data load tool, while small volumes of **setup data** often have an associated system maintenance function and can be input using the system. **Fixed input data** may be generated or migrated and is loaded using any and all of the methods above, while **consumable input data** is typically listed in test scripts or generated as an input to automation tools. When data is loaded, it can append itself to existing data, overwrite existing data, or delete existing data first. Each is appropriate in different circumstances, and due consideration should be given to the consequences.

## Data Maintenance

All data needs work to keep it complete, clean, and up-to-date. The following list shows some of the reason for data maintenance. By monitoring these, time for data maintenance can be factored into the plans.

- Replacing consumed data
- Repairing broken data
- Responding to change - database schema, code, requirements
- New test requirements

Tasks to refresh the data should be built into the test plans. Repairs can be classified in the same way as bugs and fixes prioritised appropriately. Changes to the data can generally be planned for, but be aware that if the test scope has missed some tests, or if investigative tests need new data, the changes required may be shortnotice and urgent. Data maintenance can cause significant problems. Manual data maintenance is prone to error, and can introduce faults that lead to test failures, which are then logged and must be painstakingly resolved.

Data maintenance is a sizeable task, and can make up a substantial fraction of overall test maintenance costs. However, the most common cause of problems is that data maintenance is generally (and often necessarily) performed by more than one group - different people tend to be responsible for environmental data than are responsible for fixed input data. Synchronising changes can be hard and effective management is needed to deal with differing priorities.

## Solutions

Forewarned is forearmed with data problems, and recognising and preparing for problems can help in their resolution. The lists above may help with this task. Change control and measurement can help to isolate changes and alert team members when they happen. Monitoring environmental variables, saving database contents and using configuration management can all be used to avoid or mitigate problems. Maintaining data with automated, repeatable (and debuggable) scripts reduces the number of data maintenance faults. Using small, well-known and comprehensive datasets as described above makes test maintenance simpler and quicker, and may also reduce the likelihood of errors.

## Conclusion

Data can be influential on the quality of testing. Well-planned data can allow flexibility and help reduce the cost of test maintenance. Common data problems can be avoided or reduced with preparation and automation. Effective testing of setup data is a necessary part of system testing, and good data can be used as a tool to enable and improve communication throughout the project. The following points summarise the actions that can influence the quality of the data and the effectiveness of its usage:

- Plan the data for maintenance and flexibility
- Know your data, and make its structure and content transparent
- Use the data to improve understanding throughout testing and the business
- Test setup data as you would test functionality

## About STC

STC ThirdEye Technology (India) Pvt Ltd is India's largest Independent software testing organization providing End-to-End testing Services.

We build and operate dedicated India-based testing centers for our customers with the latest computing and data communication technologies, and deliver our services, with high standards of security and confidentiality. Consistent qualities of deliverables under compressed time schedules enable us to get repeat business.

We help Fortune 500 ERP, BFSI, Healthcare, Gaming and Telecom solution providers We are ISO 9001:2000 certified organization. For more details, please visit us at [www.stcthirdeye.com](http://www.stcthirdeye.com)

## Disclaimer

The Whitepaper series presents reports on subjects in the sphere of activities of STC ThirdEye Technology (India) Pvt Ltd that are to be considered in the interest of wider public. These papers are part of the ongoing studies and authors will be glad to receive your comments.

The views expressed in these papers are to be regarded as those of the author and should not be interpreted as reflecting the views of the management of STC ThirdEye Technology (India) Pvt Ltd.

STC ThirdEye Technology (India) Pvt Ltd assumes no responsibility for any actions taken by Anybody based on the information provided in this paper.