

# Test Language

## *Introduction to Keyword-Driven Testing*

Ayal Zylberman and Aviram Shotten

### Introduction

KDT is the next generation test automation approach that separates the task of automated test case implementation from the automation infrastructure.

Test Language is not a test automation approach. Test Language is a comprehensive test approach that hands over the responsibility of automation plan, design and execution to the functional testers by using a KDT-based solution.

### Background

Several years ago, I was Test Manager for a large organization in the defense industry. The team consisted of 10 test engineers, 2 of which were test automation experts, the others were functional testers. The test automation experts were focused on developing test automation scripts that covered some of the functional processes of the application.

It took me only a few days to realize that things were not efficient.

The same tests that were executed automatically were also tested manually. The functional testers had low confidence in test automation scripts, since the test automation experts did not have the required knowledge about the application under test. The test automation development could start only after the application was ready for testing, and thus was relevant only for a small part of the tests (mainly regression tests).

I presented the problem to the team. After many discussions, we came up with the idea of letting the functional testers create test automation scripts.

The idea may be simple, but we faced a number of challenges in trying to implement it. The main obstacle was that functional testers did not have programming skills or test automation knowledge. To solve this, the test automaters suggested that the functional testers would create their test cases using pre-defined words, which they will “translate” into automated scripts. This approach was favorably accepted by the entire team.

We proceeded to create a keyword dictionary. The keywords were mutually defined by the automation experts and functional testers. During the first phase, the functional testers wrote their test cases using the keywords, and the automation experts translated them into automation scripts. At a later stage, we developed an automated application that took care of the translation. This enabled the functional testers to create and execute the automated tests and saved even more time.

White Paper

The keyword dictionary combined with the automated translation application generated a fast return on investment. This allowed us to position test automation as a basis of our testing strategy rather than a luxury.

**This was the beginning of Test Language.**

**Test Automation**

Let's review the way Test Automation is commonly approached.

The testing process includes a number of phases: test planning, test definition, bug tracking and test closure activities. The actual test execution phase is often recursive, repetitive and manual. This phase is usually described as a tedious and boring and can be carried out automatically.

To automate the execution phase, you need to create test automation scripts that are usually implemented in commercial test automation tools, such as HP's QTP, IBM's Robot, Automated QA's Test Complete, MS VS 2010 team system and freeware tools such as Selenium etc..

Each test automation script represents a test case or complementary steps for manual test cases. The scripts are created by test automation experts after the completion of the manual test design.

Contrary what test tool vendors would have you believe, test automation is expensive. It does not replace the need for manual testing or enable you to "down-size" your testing department.

Automated testing is an addition to your testing process. According to Cem Kaner, "Improving the Maintainability of Automated Test Suites", it can take between 3 to 10 times longer to develop, verify, and document an automated test case than to create and execute a manual test case. This is especially true if you elect to use the "record/playback" feature (contained in most test tools, as your primary automated testing methodology).

Sanity check: Measure the success of your Testing Automation project.

Step 1: Measure how many man hours were invested in test automation in the past 6 months (include Test Automation experts, functional testers, management overhead etc.). Mark this as **A**.

Step 2: Measure how many working hours would have needed in order to manually execute all tests that were executed automatically in the past 6 month. Mark this as **B**.

**If A > B than most likely your Test Automation project is a failure candidate**

---

## White Paper

### Why Do Test Automation Projects Fail?

These are the common reasons for test automation failures:

1. **Process** – Test Automation requires changes in the test process. The changes apply to the
  - a. Test design approaches - Test Automation requires more specific design.
  - b. Test coverage - Test Automation allows more scenarios to be tested on a specific function.
  - c. Test execution - functions in the application that are tested automatically shouldn't be tested manually.Many organizations fail to plan or implement the required changes.
2. **Maintenance** – Automated tests require both functional maintenance (i.e. updating the scripts after a functional change in the AUT) and technical maintenance (i.e. AUT UI changed from MFC to C#).
3. **Expertise** – In order to write an efficient automated test, test automation experts are required to be a techno-geek version of superman – good test engineers, system experts and great SW developers.

Obviously, a different approach is required to make test automation work.

### What is Test Language?

Test Language is a dictionary of keywords that helps testers to communicate with one another and with other subject-matter experts. The keywords replace the common English as the basis and create an approach called keyword-driven testing (KDT).

KDT can be used to achieve a number of goals:

- **Improve communication** between testers
- **Avoid inconsistency** in test documents
- Serve as the infrastructure for **Test Automation** based on Keyword-Driven Testing.

### Test Language structure

The Test Language is based on a dictionary, which is comprised of words (keywords) and parameters.

### Test Cases

A test case is a sequence of steps that tests the behavior of a given functionality/ feature in an application. Unlike traditional test approaches, Test Language uses pre-defined keywords to describe the steps and expected results (see example in Figure 2 below).

### The Keywords

Keywords are the basic functional sub-procedures for the test cases of the application under test. A test case is comprised of at least one keyword.

## White Paper

**Types of Keywords**

**Item Operation (Item)** – an action that performs a specific operation on a given GUI component. For example: set value "Larry Swartz" in "customer name" control, verify that the value "3" appears in the "result" field. When performing an operation on a GUI item, the following parameters should be specified: Name of GUI item, what operation to perform, and the values.

**Utility Functions (Function)** – a script that executes a certain functional operation that is hard/non-effective to implement as a sequence. For example: wait X seconds, retrieve data from DB etc.

**Sequence** – a set of keywords that produces a business process, such as “create customer”. We recommend collecting frequently used functional processes such as login, addition of new records to the system as sequences, instead of implementing them as items in test cases.

**Parameters**

In most cases, parameters should be defined for the created keywords. The parameters are additional information required in order to generate the test conditions. For example: failed authentication by-passing username with illegal password, number for mathematical calculation, etc.

Examples for sequence parameters:

create\_customer(FirstName, LastName, BirthDate, email)

Keyword

Parameters

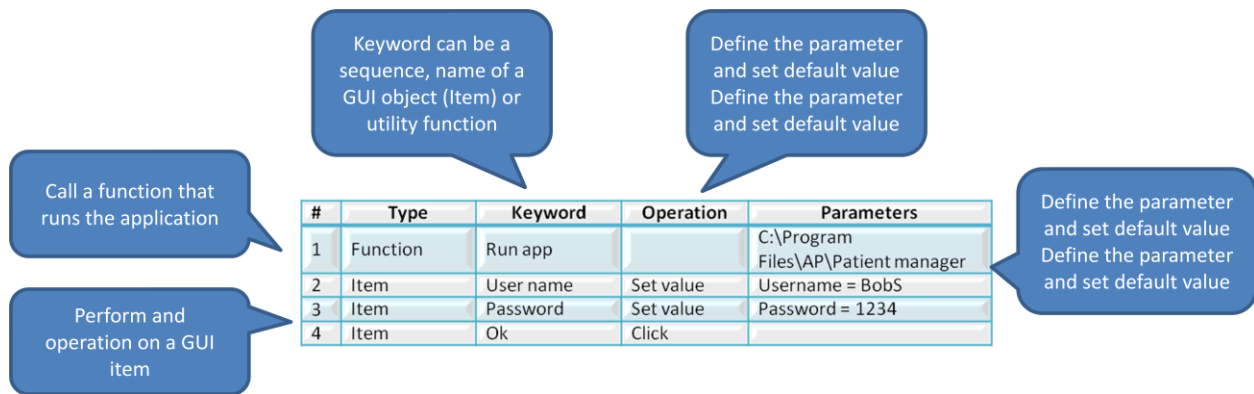
When the user wants to create a new customer, the following syntax is used:

**create\_custmer (Bob,Dylan,1/1/2000,bobdylan@gmail.com)**

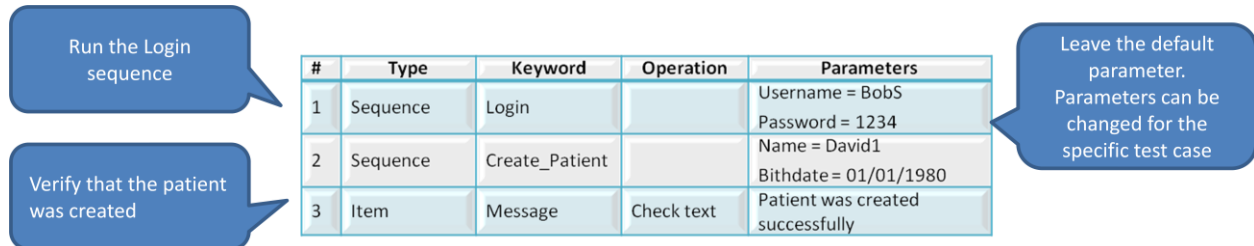
White Paper

**Using default parameters**

Some of the keywords may contain dozens of parameters. In order to simplify the test creation, all parameters should contain default values. The tester should be able to change each one of the default parameters according to the context of the test. For example, if the tester would like to create a new customer that is older than 100 years, only the birth date will be changed and all other parameters will remain the same. Obviously, a specific change will not affect the default parameters being used for other tests.



**Figure 1: Login sequence**



**Figure 2 – Create Patient Test Case**

It is extremely important to plan the keywords well and to optimize the amount of keywords by creating multi-function keywords (for example, creating "Change\_customer\_status" keyword is a better approach than creating 2 special keywords for "activate\_customer" and "deactivate\_customer")

## Keyword-Driven Testing (KDT)

KDT is the mechanism used in order to implement Test Language.

Keyword-Driven Testing can be divided into two main layers:

**Infrastructure Layer (KDT Engine)** - a combination of Item Operation, Utility Functions and User-Defined Functions (also called Sequence), used as an “engine” that receives inputs (keywords) and performs operations on the application under test.

**Logical Layer (KDT Test Case)** - used by manual testers and other subject matter experts to build and execute test scripts by using a pre-defined keyword.

## KDT Principles

### Simplicity of test case implementation and reviewing

Functional testers should create the automated test cases directly in a simple, non-code, elite language. This eliminates the need for the time-consuming two-step process, in which testers create manual test cases which are then converted into automated scripts by the test automation team.

### Automation infrastructure and test cases coexistence

The test automation infrastructure should be separated from the logical layer (which is the test cases). This allows testers to create automated test cases at an early stage in the development life-cycle before the application is ready and even before the automation infrastructure is developed.

### Plan and control

In order to maximize the benefit from KDT, a plan used to assess the on-going effectiveness of the automation program.

## What to Automate

***Test Language should focus on testing new features rather than on regression testing, which is the common practice in traditional test automation***

In traditional test automation implementations, automation scripts cannot be created before the application is ready for testing. As a result, sanity and regression tests are the main areas, which are automated. This approach usually leads to parallel executions (both manually and automatically) of the same tests.

In addition, tests, written for manual execution, are phrased in a way that does not take into account the strengths of automation scripting. The main gap between the two approaches are:

## White Paper

- Detail level: Test Automation requires more details (for example, after log-in, the automated script should define what is expected to happen, while in many manual scripts, this is left to the tester’s intuition).
- Coverage: When performing test automation, more test cases can be created. For example: when testing a certain numeric field that get values in the range of 1-10, usually boundary analysis will be used to test the following numbers: 1, 2 , 10 and 11, while for automated tests, more scenarios can be planned.

**KDT allows functional testers to plan test automation before the application is ready.** This enables organizations to broaden the automated testing spectrum.

We recommend that new tests rather than translation of existing tests should be the focus of automated testing using KDT. All new tests can be planned using the KDT approach. We believe that this approach eases the KDT implementation process and greatly improves the ROI of automated testing.

### Organization Structure and Responsibilities

The organizational structure of a KDT-based organization is similar to traditional test automation based organizations. It consists of Core Test Automation teams and Functional Testers. However, the responsibilities are much different:

Activity	Automation Experts	Functional Testers	Notes
Define Keywords		+	Keywords should be defined by functional testers under the supervision and control of the test automation experts
Develop Utility Functions	+		
Develop Sequences (user-defined functions)	+	+	Sequences are developed by functional testers in small-scale organizations and for private use (functions that will not be used by other testers).
Develop Test Cases		+	
Execute Test Cases		+	

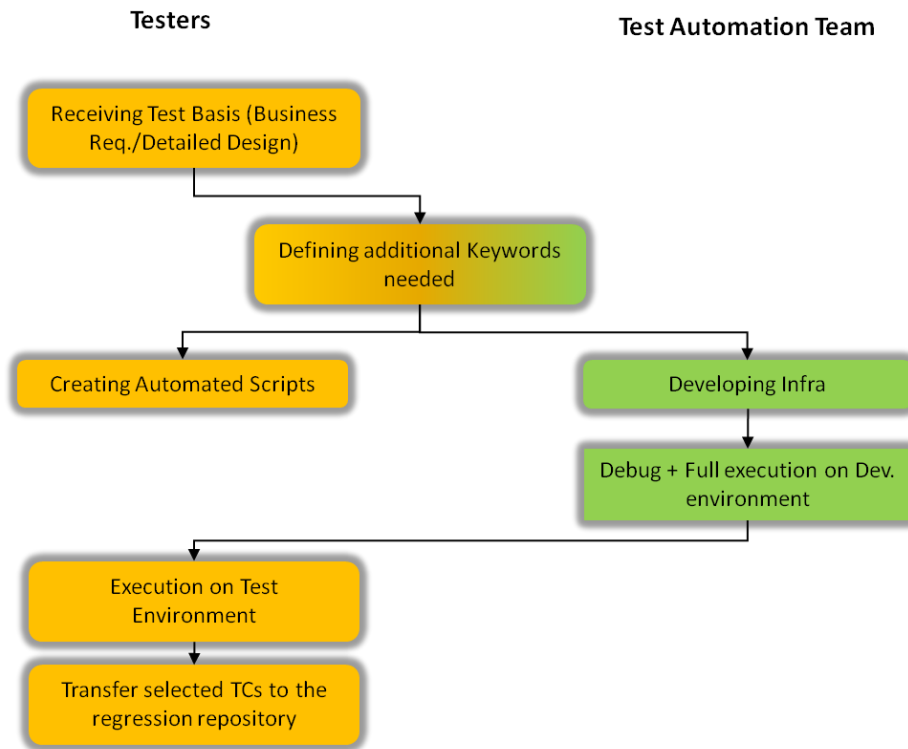
Figure 3: KDT Responsibilities Matrix

White Paper

**KDT Process**

The key factor to fully enjoy the benefit of the Keyword-Driven approach is by fully integrating it throughout the entire test process.

The following diagram shows the KDT process:



**Figure 4: Keyword-Driven Testing Process**

---

## White Paper

### About QualiTest

QualiTest Group is a leading global provider of Quality Assurance and Software Testing services. With over 800 Software Testing professionals, 10 branches and a strong presence in the USA, Europe, Israel and Asia, QualiTest provides services to global Fortune 500 companies, large and small organizations. The company is committed to improving quality by providing superior solutions to customers and partners worldwide as well as advancing QA and testing methodologies.

QualiTest offers a wide variety of staffing services such as: onshore testing, quality assurance (QA) and testing outsourcing and flexible staffing. QualiTest expert services promote global practices including: Test Process Improvement (TPI), Keyword Driven Testing (KDT), Scrum Based Testing and Test automation.

### Professional History and Credentials:

**Ayal Zylberman** is a senior Software Test Specialist and QualiTest Group co-founder. He has been in the field of Software Testing since 1995 testing several disciplines such as military systems, billing systems, SAP, RT, NMS, networking and telephony. He was one of the first in Israel to implement automated testing tools and is recognized as one of the top-level experts in this field worldwide. Ayal was involved in Test Automation activities in more than 50 companies. During his career, he published more than 10 professional articles worldwide and is a frequently requested lecturer at Israeli and international conferences.

**Aviram Shotten** is QualiTest's program manager for the "Automation Planner®" suite supporting KDT, and a senior test manager responsible for over 80 test engineers in over 25 projects; among them more than 10 have successfully implemented KDT automation via the Automation Planner®.

### References

Keith Zambelich, White paper: [Totally Data-Driven Automation Testing](#)