

Automation Scripting Techniques

Zhu Zhong (@钟柱)

6/30/14

► Scripting

Scripting techniques refer to the **methods** and **approaches** used to **design**, **develop**, and **execute test scripts**. Test scripts are a set of **instructions or code** that automate the testing process, allowing testers to **repeat tests**, **save time**, and **ensure consistency**.

Scripting Techniques

- Record and Playback
- Data-Driven Testing
- Keyword-Driven Testing
- Behavior-Driven Development (BDD)
- Model-Based Testing
- Hybrid Approach

Record and Playback:

This technique involves using a **tool or framework** to **record user actions** on the application under test. The **tool generates scripts based on the recorded actions**, which can be played back to repeat the same steps during subsequent test runs.

Web Application Testing

Selenium WebDriver

Mobile App Testing

Appium or UI Automator

Data-Driven Testing:

Data-driven Testing with data-driven testing, **test scripts are designed to handle multiple sets of input data.** The test script reads test data from **external sources** such as **spreadsheets or databases** and uses that data to drive the **test execution**, checking the **application's behavior** against different data combinations.

Login Functionality

Form Validation

Calculation or Processing Logic

Keyword-Driven Testing:

Keyword-driven testing is a technique where **test scripts are composed using a set of keywords or action words** that represent **specific operations or functionalities**. These keywords are mapped to functions or methods in the test automation framework, allowing testers to create test cases by combining and reusing keywords.

E-commerce Checkout Process

"AddToCart," "EnterShippingAddress," "EnterPaymentDetails," and "PlaceOrder"

User Permissions

- ▶ Test Case 1: Grant administrator access to a user Keywords: GrantPermission(User, Administrator)
- ▶ Test Case 2: Revoke read-only access from a user Keywords: RevokePermission(User, ReadOnly)

Behavior-Driven Development (BDD):

BDD is an **agile software development** technique that emphasizes **collaboration** between **developers, testers, and business stakeholders**. BDD uses a **natural language syntax** called **Gherkin** to describe the desired behavior of the software. Test scripts are then created based on these specifications, making them more understandable to non-technical stakeholders.

1. User Registration: Scenario: User Registration Given the user is on the registration page When they enter valid registration details And click the Register button Then they should be successfully registered
2. In this example, the scenario describes the behavior of the user registration process. The "Given," "When," "And," and "Then" keywords represent the different steps of the scenario. Test scripts can be created based on this scenario, mapping each step to the corresponding automation code or manual test instructions.

Model-Based Testing:

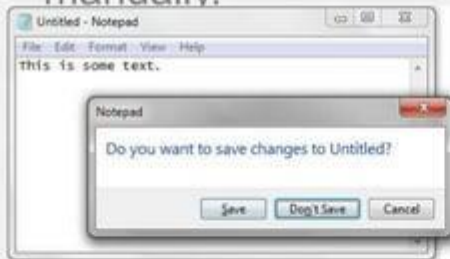
In model-based testing, **test scripts are derived from models** that represent the **expected behavior or structure of the system** under test. These models can be in the form of **state diagrams, flowcharts, or other visual** representations. The test scripts are generated automatically from the models, ensuring comprehensive test coverage.

Hybrid Approach:

A combination of scripting techniques is used to **achieve effective test automation**. Testers may use a hybrid approach by combining **record and playback** with **data-driven** or **keyword-driven techniques** to leverage the advantages of each method and address specific testing requirements.

Linear Scripts

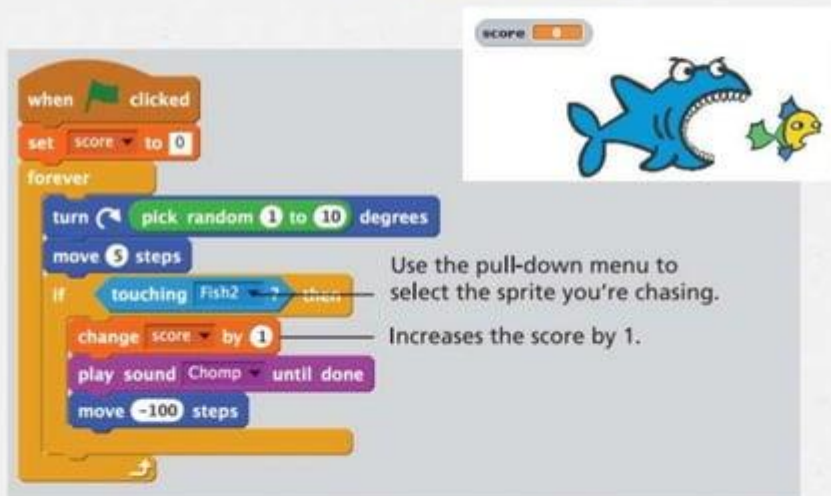
- A Linear script is what you end up with when you record a test case performed manually.



```
1 Run("notepad.exe")
2 WinWaitActive("Untitled - Notepad")
3 Send("This is some text.")
4 WinClose("Untitled - Notepad")
5 WinWaitActive("Notepad", "Do you want to save changes to Untitled?")
6 Send("In")
```

Structured Scripts

= Linear scripts + if + for/while loop



The image shows a Scratch script and a game state. The script is a 'when green flag clicked' event that sets a 'score' variable to 0. It then enters a 'forever' loop. Inside the loop, the shark sprite turns a random angle between 1 and 10 degrees, moves 5 steps, and checks if it is touching a fish sprite named 'Fish2'. If it is touching 'Fish2', the score is increased by 1, a 'Chomp' sound is played until done, and the shark moves -100 steps. The game state shows a score of 3, a blue shark sprite, and a small green fish sprite.

```
when green flag clicked
  set score to 0
  forever loop
    turn pick random 1 to 10 degrees
    move 5 steps
    if touching Fish2 then
      change score by 1
      play sound Chomp until done
      move -100 steps
```

score 3

Use the pull-down menu to select the sprite you're chasing.

Increases the score by 1.

(scratch)

Shared Scripts

- Shared scripts are scripts that are used (or shared) by more than one test case.
- May be called reusable script/function library
- Hard-coded values -> variables

Shared Scripts

– an example

checkLinuxVersion

Steps

No.	Object	Action	Parameter
1		function	demo
2	myTelnet	open	\$myHost
3	myTelnet	command	uname -a
4		query	-query id:version, -responmappingfile demo.toml, -expect .*, -save \$version
5		if	\$version != \$myVersion
6		fail	version is wrong!
7	myTelnet	close	

Convert **shared script** to
HOST.checkLinuxVersion



Service Browser

HOST
SNMP.Get
SNMP.Set
checkLinuxVersion

tc

Steps

No.	Object	Action	Parameter
1		function	checkLinuxVersion
2	myLinuxServer	checkLinuxVersion	-myVersion 2.6.32

Test case is simplified: just need to call shared script **HOST.checkLinuxVersion**

(EasyTest)

Do you have a question for me?

Email zhongzhu@ymail.com
Or follow me on Sina Weibo @钟柱

