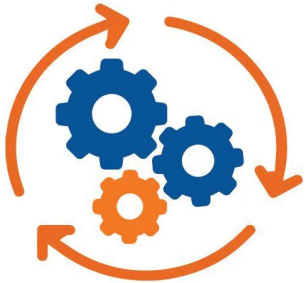
Chapter 8

Test Development



Lecturer: Nguyen Thanh Quan (MEng)
Email: tg_nguyenthanhquan_cntt@tdtu.edu.vn

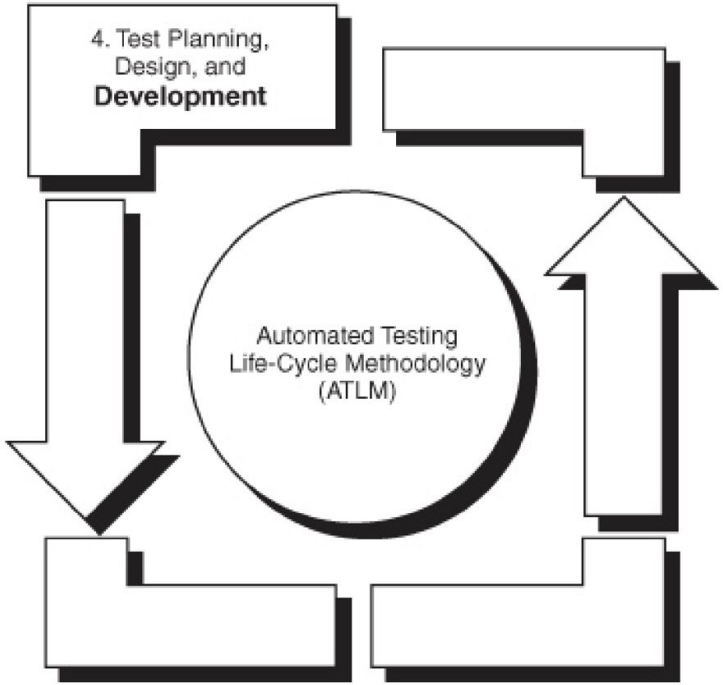
Content

1. **Test Development Architecture**
2. **Test Development Guidelines**
3. **Automation Infrastructure**

Introduction

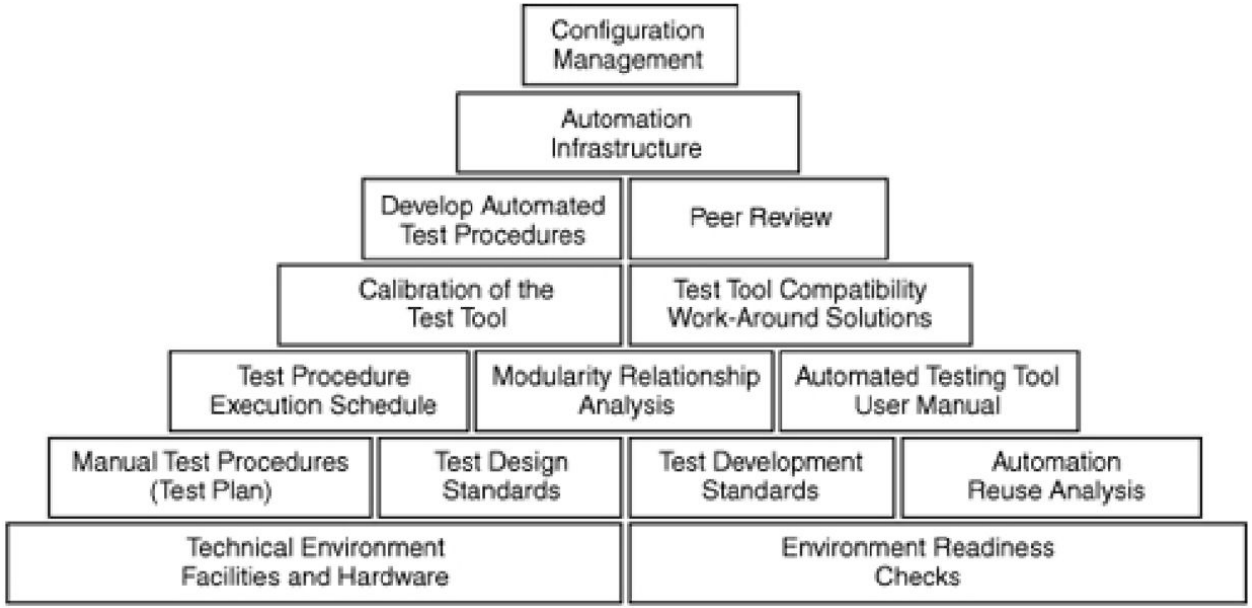
Many preparation activities need to take place before test development can begin

- Test environment **setup activities**.
- Identify the **potential for reuse** of already-existing test procedures and scripts within the automation infrastructure.
- Performs a **modularity relationship analysis.**
- Perform **configuration control** for the entire testbed (test design, test scripts, and test data, test procedure).
- **Maintainable, reusable, simple, and robust.**
- Standards and guidelines should be prepared on **context independence** (where a test procedure should start and where it should end)



Test Development Architecture

Test Development Architecture



Building Blocks of the Test Development Architecture

Test Development Architecture

- Test team members responsible for test development need to be prepared **with the proper materials.**
  - test procedure development.
  - execution schedule, test design information, automated test tool user manual.
  - test procedure development guidelines, documentation.

=> Foundation of information => cohesive and structured set of test procedures.

=> Demonstrate the strength of a test program.

Test Development Architecture

Technical Environment

- Must be set up and ready to go **before such development begins.**
- May include **facility resources** as well as the **hardware** and **software** necessary.
- Ensure enough **workstations are available** to support the entire team.

=> Need to be outlined within the test plan (Chapter 6).

- Also include **the use of an environment.**
- The **calibration** of the test tool to match the specific environment.
- When test tool **compatibility** problems arise with the AUT, work-around solutions must be identified.

Test Development Architecture

Technical Environment

- Be **consistent** with the test execution **schedule**.
- **Follow** test procedure development **guidelines**.
- Proper test **room** or **laboratory facilities** are reserved and set up.
- All necessary **equipment** is **installed** and **operational**.
- Operational support activities have been properly scheduled and must **monitor progress of these tasks.**
- **Potential issues:** network installation, network server configuration and allocated disk space, network access privileges, required desktop computer processing speed and memory, number and types of desktop computers (clients), video resolutions, browser compatibility.

Test Development Architecture

Technical Environment

- The **hardware configuration** needs to support: processing, storage, and retrieval activities across a local or wide area network.
- Must account for **stress testing.**
- Data preparation activities include:
  - The identification of **conversion data** (data mapping criteria, data element definitions, primary keys, and defining data-acceptable parameters)
  - Preprocessing of **raw data files.**
  - Loading of **temporary tables.**
  - Performance of **consistency checks.**

Test Development Architecture

Environment Readiness Checks

- Include **a review** of the organization's automation **infrastructure**.
- Check the **status** of any software **functionality**.
- Ascertain the **stability** of the AUT (should be stable).
- Should follow the **iterative** in ATLM which involves a "design a little, code a little, test a little".

Test Development Architecture

Automation Framework Reuse Analysis

TP Number	Design Component	Test Technique	SR ID	SWR ID	TR ID	A/M	Reuse Asset
2330	TV1016	Functional	3.2.3c	TV029	2220	A	—
2331	TV1016	Functional	3.2.3c	TV030	2221	A	MMS2079
2332	TV1016	Functional	3.2.3c	TV031	2412	M	—
2333	TV1017	Functional	3.2.3d	TV032	2222	A	—
2334	TV1017	Functional	3.2.3d	TV033	2412	A	—
2335	TV1018	Functional	3.2.3e	TV034	2223	A	LW2862
2336	TV1018	Functional	3.2.3e	TV035	2412	M	—
2337	TV1019	Functional	3.2.3f	TV036	2224	A	—
2338	TV1019	Functional	3.2.3g	TV037	2412	A	ST2091
2339	TV1019	Functional	3.2.3g	TV038	2225	A	ST2092

Test Development Architecture

Test Procedure Development/Execution Schedule

The schedule takes into account various factors, such as the following:

- The individuals **responsible** for each **particular** test activity are identified.
- Setup activities are **documented**.
- **Sequence** and **dependencies** are included.
- Testing will accommodate the **various** processing **cycles** that pertain to the application.
- Test procedures that could potentially **conflict** with one another are **documented**.
- Test engineers are allowed to **work independently.**
- Test procedures can be **grouped** according to specific **business functions.**

Test Development Architecture

Test Procedure Development/Execution Schedule

The schedule takes into account various factors, such as the following:

- Test procedures will be organized in such a way that effort is **not duplicated.**
- The test procedures' organization considers **priorities and risks assigned to test.**
- A plan for various testing phases and each activity in a particular phase exists.

Test Development Architecture

Test Procedure Development/Execution Schedule

- ***Identifying the individual(s) responsible*** for each particular test activity => **prevent duplication** of test effort by personnel.
- ***Setup activities*** *need to be documented.*
- Test procedure *sequence and dependencies* need to be documented.
- The test procedure development/execution schedule must ***accommodate testing that examines the various processing cycles pertaining to the application.***
- The test procedure development/execution schedule will also need to ***document all test procedures that could potentially create conflicts.***

Test Development Architecture

Test Procedure Development/Execution Schedule

- ***Test engineers will need to be able to work independently*** and must be able to share the same data or database.
- Tests may be **grouped** according to **the specific business function.**
- Automated test procedures be organized in such a way that **effort is not duplicated.**
- Must take into consideration the **priorities** and **risks** assigned to the various tests.
- Need to document the breakdown of the various testing phases and discuss the activities in each phase (that is, functional, regression, or performance testing).

Test Development Architecture

Test Procedure Development/Execution Schedule

- Must allow **time** to accommodate **multiple iterations** of test execution.
- The test team must **monitor changes** and alter the test schedule accordingly.
- The test procedure execution schedule can be created using a **project scheduling tool** or developed **within a spreadsheet or a table** using a word-processing package.

Test Development Architecture

Modularity-Relationship Analysis

- To identify any data **dependencies** or workflow dependencies between automated test procedures.
- The test engineer needs to be **aware of the state of the data** when performing tests.
- Must determine which scripts need to be run together in a particular sequence using the **modularity-relationship matrix.**
- Enables the test team to plan for **dependencies** between tests.
- **Avoid failure of one test procedure affects others.**

Test Development Architecture

Modularity-Relationship Analysis

- Allow test engineers to identify **common scripts.**
- Creating the **modularity-relationship matrix:**
  - Creation of a visual flow using boxes for each section.
  - Breaks down each box into smaller boxes or lists each component or screen for each section of the AUT

=> will bring to light **dependencies** (preconditions and post-conditions).

=> allows test engineers to identify for script **reuse** => minimizing the effort.

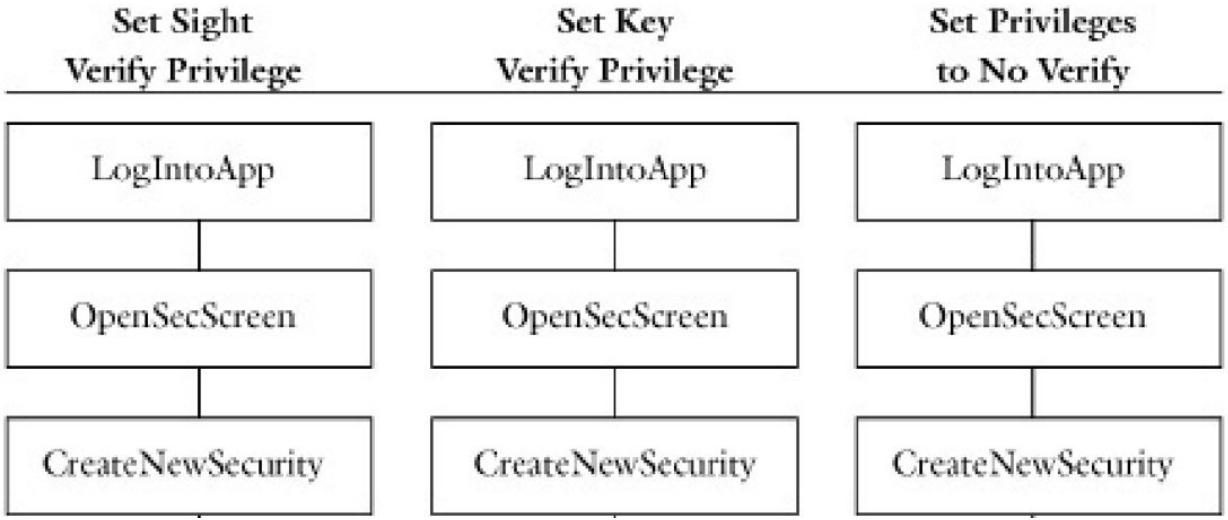
=> organize the proper **sequence** of test execution.

=> **associate** test procedures to the application design (convention).

=> matrix shows how the various test procedures **fit together**.

Test Development Architecture

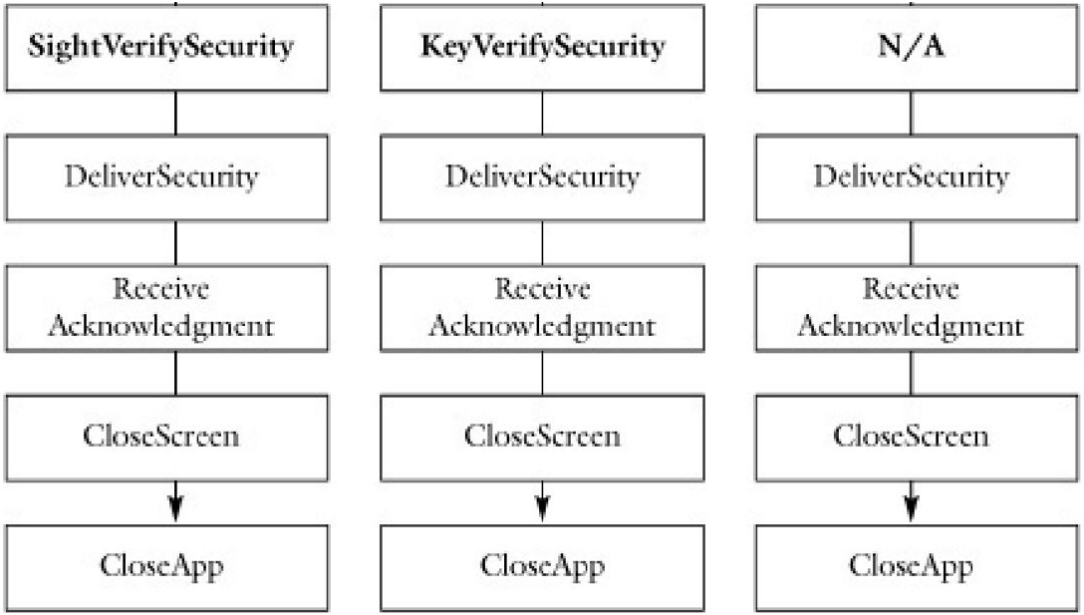
Modularity-Relationship Analysis



Sample Modularity-Relationship Matrix

Test Development Architecture

Modularity-Relationship Analysis



Sample Modularity-Relationship Matrix

Test Development Architecture

Calibration of the Test Tool

- When calibrating the automated test tool to match the environment, it will have to decide upon the tool ***playback speed.***
- A common problem for unattended testing involves **failures** that have the **domino effect.**
- Successive failures will require the tests to be **restarted repeatedly.**

=> It is beneficial if the chosen test tool provides for **automatic handling** of **unexpected** active windows and other unexpected application states.

Test Development Architecture

Compatibility Work-Around Solutions

- Consider **work-around solutions** for any **incompatibility** problems.
- A good understanding of the AUT is beneficial when contemplating a work-around solution to an **incompatibility problem**.
- Important to test for **compatibility** between the **automated tools and the third-party controls or widgets.**
- Some of these fixes might **take time to develop** => need work-around.

Test Development Architecture

Manual Execution of Test Procedures

- When **functionality** is **missing** within the application, test automation can be **inefficient and unproductive.**
- Streamline the overall test effort by investing some time in this **manual verification** process.
- After executing all of the steps in the test procedure **without any problems => start automating the test script for later reuse.**
- if the steps in the test procedure **didn’t pass => manipulate it to record the “expected result.”**

Test Development Architecture

Test Procedure Inspections - Peer Reviews

- Detect defects.
- Test coverage issues.
- As review test programming code.
- Uncover any discrepancies with the test procedures, test requirements, or system requirements.
- Functional requirements can serve as a resource for developing test requirements and subsequent test procedures.

Test Development Architecture

Test Procedure Configuration Management

- Need to be baselined using a configuration management (CM) tool.
- Groups of reusable test procedures and scripts are usually maintained in a catalog or library of test data comprising a **testbed**.
- When the test team uses a requirement management tool, such as DOORS, baseline control of test procedures is provided automatically.
- When the test team creates test procedures in a simple word-processing document, it needs to ensure that the documents are baselined.
- SQL scripts or playback automated scripts for **cleanup activities.**

Test Development Architecture

Test Procedure Configuration Management

Testbed management also includes:

- management of changes to software code.
- control of the new-build process.
- test procedures.
- test documentation.
- project documentation.
- all other test-related data and information.

Test Development Architecture

Test Procedure Configuration Management

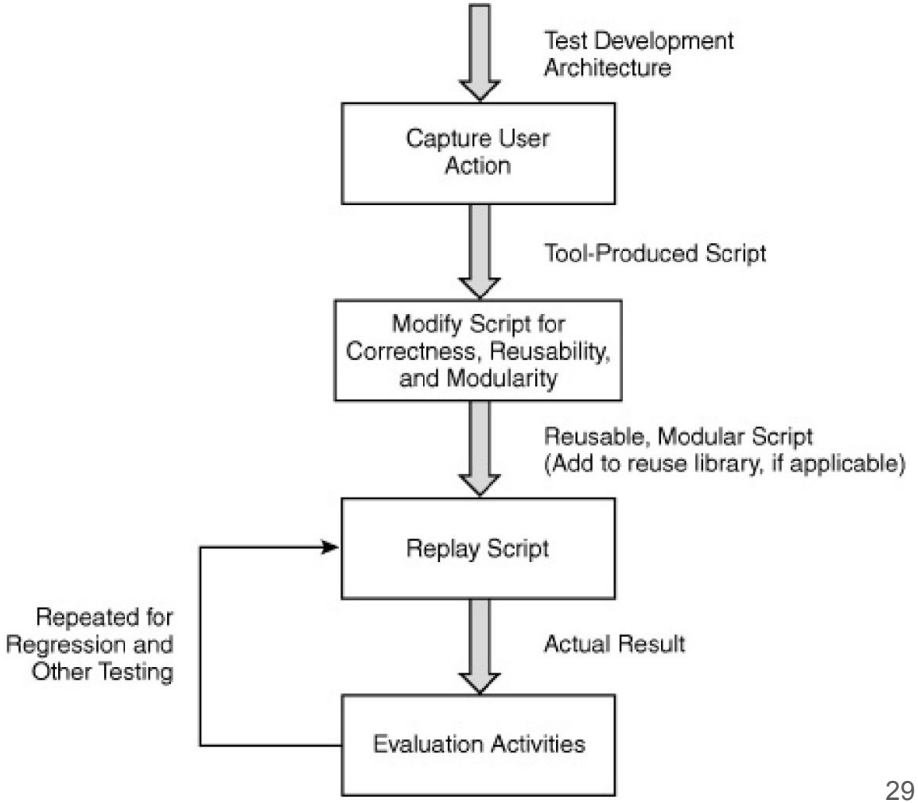
- **Specific changes that may alter the test environment:**
  - a change to a network connection.
  - modification of disk space.
  - the implementation of a faster processor or other AUT upgrades.

=> **Without test procedure management,** all test engineers affected might **not be notified of these changes.**

Test Development Guidelines

Test Development Guidelines

- **Follow** the development **standards**.
- **Simultaneously** develop test procedures that are **consistent**, **reusable**, and **maintainable**.



Test Development Guidelines

Development Guideline Topics	Description
Design-to-Development Transition	Specify how design and setup activities will be translated into test development action
Reusable Test Procedures	Test procedures need to be reusable for highest test program return on investment
Data	Avoid hard-coding data values into scripts, rendering them not reusable
Application Navigation	Standard navigation method needs to be deployed for reusable test scripts
Bitmap Image Recording	Addresses the use of a bitmap image recording method for test procedure development
Automation Wildcards	Development guidelines supporting reusable test procedures
Capture/Playback	Outlines on how to apply the use of capture/playback recording

Test Development Guidelines

Test Development Guidelines

Maintainable Test Procedures	A test procedure whose defects are easy to remove and can easily be adapted to meet new requirements
Cosmetic Standards	Standards defined to promote test program code that is easy to read and comprehend
Test Script Comments	Specifies where and how comments are used within procedures and scripts
Test Script Documentation	Specifies that test script documentation is important for test procedure maintainability
Test/Application Synchronization	How to synchronize the server/GUI/AUT with the test script
Test Procedure Index	Guidelines supporting the maintenance of an index to find test procedures of interest
Error Handling	Guidelines for how test procedures will handle errors
Naming Standards	Defines the standard naming convention for test procedures
Modularity	Guidelines for creating modular test scripts
Looping Constructs	Looping constructs support script modularity
Branching Constructs	Branching constructs support script modularity
Context Independence	Directs development of test procedures given test procedure relationships
Global Files	Globally declared functions are available to any procedure and support maintainability
Constants	Guidelines addressing use of constants to support maintainable test procedures

Test Development Guidelines

Other Guidelines	Other test development guidelines
Output Format	Users need to define what the test procedure results output format should look like
Test Procedures/Verification Points	Guidelines can specify which verification points to use most often and which ones to avoid
User-Defined Verification	Addresses the use of script programming for user-defined verification
API Calls, Dynamic Link Libraries (.dll), and Other Programming Constructs	Addresses test automation using application programming interfaces and .dll files as part of the user-defined verification methods

Test Development Guidelines

Test Development Guidelines

Design-to-Development Transition

- The test team needs to ensure that test development represents **a natural transition from test design.**
- The test design baseline consists of:
  - the test program model.
  - test architecture.
  - test procedure definition.
  - test procedure matrices.
- Define **the order of execution.**

=> should hold a **meeting** to discuss the specific way => document the outcome.

Test Development Guidelines

Reusable Test Procedures

- The most important issue when developing or modifying a test procedure is **reusability** => improve test team efficiency.
- The best practice calls for **the separation of functionality such as data read/write/validation, navigation, logic, and error checking.**

Test Development Guidelines

Reusable Test Procedures / Data

- Use a **data-driven approach** to test development.
- It is recommended that the test engineer **not hard-code data values** into a test procedure => replace such hard-coded values with variables and, whenever possible and feasible, read data from a .csv or ASCII file, spreadsheet, or word-processing software.
- Identifies the **data elements, data value names, or data variable names.**
- Whenever the test procedure should be played back with a different set of data, the data file simply can be updated.

Test Development Guidelines

Reusable Test Procedures / Application Navigation

- Navigation functionality **should reside in a separate module** in the test library (tabs, keyboard accelerator keys - hotkeys, mouse clicks, capturing the x, y coordinates the object names assigned to a particular object).
- If test procedure scripts use the tab method to navigate => tab-order-dependent.
- If accelerator keys are employed as a navigation method and new accelerator keys are later assigned, the recording again will become useless.
- If the mouse is used, a field could change and the mouse click might occur on a nonexistent field.

Test Development Guidelines

Reusable Test Procedures / Bitmap Image Recording

- **Known as screen shot recordings.**
- It allows a test procedure to compare a screen or an edit box with a baseline recording of the screen or edit box.
- The resulting test is overly **sensitive to any change**, including those that are appropriate => too **burdensome**, given the **significant amount of effort required.**
- Different video **resolution** or **different color** setting for screen display => fail.

Test Development Guidelines

Reusable Test Procedures / Automation Wildcards

- Such as the use of the asterisk “*”.
- To seek a **match of a certain value** or to **identify when conditions meet certain criteria.**
- When using an automated test tool to record test procedures, a wildcard can be employed to look for a caption of a window so as to identify the window.

Test Development Guidelines

Reusable Test Procedures / Capture/Playback

- Reusable test development guidelines should address **the use of capture/playback** methods of test procedure recording.
- A major **drawback** for the capture/playback method is that values are **hard-coded.**
- Clearly, test procedures created using the capture/playback **method are not reusable and, as a result, are not maintainable.**

Test Development Guidelines

Maintainable Test Procedures

- It is important that the test team **follow guidelines** that support the creation of **maintainable** test procedures.
- By carefully designing and developing **modular procedures**, the team can **simplify the maintenance** of the test scripts.
- The time spent actually building and maintaining tests can be **minimized** by designing carefully and by **understanding the interdependence** of the test procedures.

Test Development Guidelines

Maintainable Test Procedures

- Follow cosmetic standards
- Document test scripts
- Allow for synchronization
- Use test procedure index
- Allow for error handling
- Use naming standards
- Create modularity scripts
- Use looping constructs
- Use branching constructs
- Allow for context independence
- Use global files

Test Development Guidelines

Maintainable Test Procedures / Test Script Comments

- Standards should be **easy to read, understand, and maintain.**
- Specify how to represent if/then/else statements and case statements within code, how to align the indentation.

Sample Test Script Using No Layout Standard

```
Sub Main
Dim sInput As String
Dim word, numeric
Dim Counter As Integer
'Initially Recorded: 12/01/97  23:52:43
Open "C:\temp\wordflat.txt" For Input As #1
Counter = 1
Do While Counter < 5000
```

Sample Test Script Using Layout Standard

```
Sub Main

    Dim sInput As String
    Dim word, numeric

    Dim Counter As Integer
```

Test Development Guidelines

Maintainable Test Procedures / Test Script Comments

- **Comments** are meant to clarify **the scope of the test.**
- Enable other individuals to more easily **understand** the **purpose** and **structure of the test scripts.**
- **No effect on script performance** => the maintainability of the scripts in the long term.
- Comments can be in both **manual and automated** test procedures.
- **Guidelines** should specify how **comments** are to be **incorporated**.
- **Avoid confusion**, **easier** to **complete** test scripts, **revise** & **reuse** scripts.
- Help to create a special report **outlining the scope** of the entire suite of test scripts.

Test Development Guidelines

Maintainable Test Procedures / Test Script Documentation

- **Beneficial** for any individual reviewing the test script who **must understand the test script**, yet **doesn't have the programming background** to be able to read the scripting language.
- Supports the greater use of the script within a **reuse library** or automation infrastructure.

Test Development Guidelines

Maintainable Test Procedures / Test Procedure Header

- Used as an introduction that defines the purpose of the test procedure.
- May include the **test procedure ID, test procedure name**, preconditions, data criteria, input arguments, test conditions, expected results, status, and related requirements, **authors, date.**
- All test scripts will have **the same look and feel** (format).

=> Allows test engineers and project personnel to **share the same interpretation of test documentation.**

=> Makes all such procedures easy to read and understand.

Test Development Guidelines

Maintainable Test Procedures / Synchronization

- The **synchronization** between **test procedure execution and application execution.**
- **Wait times** need to be incorporated into the script that allow for synchronization.
- The test procedure scripts should remain synchronized with the AUT (a complex query => takes extra milliseconds => synchronize with scripts)
- **WaitState** is preferred instead of **DelayFor (hardcoded) - Why?**

Test Development Guidelines

Maintainable Test Procedures / Test Procedure Index

- The test team could also develop and maintain a test dictionary.
- The test dictionary contains information associated with test procedures, test scripts, and pass/fail results, document names, windows/screen names, fields/objects, vocabulary for tests, and cross references.

=> **Improve maintainability.**

Test Development Guidelines

Maintainable Test Procedures / Error Handling

- Errors need to be reported from the test procedure that **detects the error and knows what the error is.**
- Error handling can increase the **maintainability** and **stability** of the test procedures.
- Error recovery must be built in and must take into consideration the **various types of errors** that the test script could encounter.
- **Logic** within a test script can be implemented to **allow** for **branching** to a **different script** or **calling** a script that **cleans up the error condition.**
- The error message should **be self-explanatory.**

Test Development Guidelines

Maintainable Test Procedures / Error Handling

Test Station Check

```
DataSource = SQAGetTestStationName()

   'Check for valid test station name
   If DataSource = "" Then
      MsgBox "No Test Station Name Found"
      Exit Sub
   End If
```

```
'Make sure files exist
      DataSource = SQAGetDir(SQA_DIR_PROCEDURES) & "CUSTPOOL.CSV"
      If Dir(DataSource) = "" Then
         MsgBox "Cannot run this test.  File not found: " & DataSource
         Result = 0
         Exit Sub
      End If
   Else
      WriteLogMessage "Found " & DataSource
   End If
```

Test Development Guidelines

Maintainable Test Procedures / Naming Standards

Naming standards improve script **maintainability**:

- **Standardize and decode** the structure and logic.
- Variables are **self-documenting** as to the data they represent.
- Variables are **consistent** within and across applications.
- Script is **precise, complete, readable, memorable, and unambiguous.**
- **Consistent** with **programming language conventions.**
- A greater opportunity for **longer or fuller variable names, procedure names, function names, and so on.**
- Bad naming conventions: *String a; int var1…*
- Good naming conventions: *String customerName…*

Test Development Guidelines

Maintainable Test Procedures / Modularity

- A modular script increases **maintainability**, **easier to understand and debug.**
- The scripting **tasks can be divided** among test engineers in the group.
- Each module can consist of several small functions. For example login:
  - Start the AUT.
  - Input the login user ID.
  - Verify the login user ID (error checking).
  - Input the login password.
  - Verify the login password (error checking).
  - Hit OK.

Test Development Guidelines

Maintainable Test Procedures / Modularity

- Test procedures should be **short and modular.**
- Should **focus on a specific area of testing.**

=> Modular test procedures can be c**alled, copied, combined.**

=> **Easily modified** if changes are made with AUT.

=> **Easier to debug.**

=> **Maintenance** of modular test procedures **becomes easier.**

Test Development Guidelines

Maintainable Test Procedures / Looping Constructs

- Looping constructs support modularity and thus maintainability.
- Almost all test tool scripting languages support two kinds of loops:
  - **counted** loop acts as a FOR loop.
  - **continuous** loop performs as a WHILE loop.
- application requirements specify a specific number or when the test engineer can determine a specific number of times that a task needs to be performed => FOR loop.
- Data addition, expected changes handling => looping constructs.

Test Development Guidelines

Maintainable Test Procedures / Branching Constructs

- Branching constructs **support modularity and thus maintainability.**
- ***If..then..else, case, or GOTO***
- If..then..else statements are most often used for error checking and other conditional checking.
- The use of GOTO routines often inspires debate => create spaghetti code => unstructured code => hard to follow.

Test Development Guidelines

Maintainable Test Procedures / Context Independence

- Context independence is facilitated by the implementation of the **modularity** principles.
- The required **sequencing** for test procedure execution could be **documented** within each test procedure as a **precondition** for execution.

Test Development Guidelines

Maintainable Test Procedures / Global Files

- Automated test procedures can be **maintained more easily** by adding global files.
- Global files offer a benefit because the globally **declared procedures** are available for any test procedure.

Test Development Guidelines

Maintainable Test Procedures / Constants

- Use of **constants enhances the maintainability** of test procedures.
- Should **store** constants in **one or more global files** => maintain.
- To make life simpler, the test engineer could put the time in a constant.
- Examples:
  - MILLI_SEC = 1000
  - MILLI_MIN = 60000
  - MILLI_HOUR = 360000

Test Development Guidelines

Other Guidelines / Output Format

- During the test planning phase, the test result output should be identified.
- Should obtain **customer** or **end-user approval** for the planned test results output format.

Other Guidelines / Test Procedures/Verification Points

- Test procedures can **examine the object properties** of the various controls on a GUI screen.
- The test engineers might have a preference as to which verification points they use most often and which ones they avoid.

Test Development Guidelines

Other Guidelines / User-Defined Verification Methods

- The test procedure requires to determine whether a test has **passed** or **failed**.
- The test team **defines the exact** string that should appear in order to pass a test. For example, any date is acceptable or the exact date.
- The test may **incorporate** a custom-built verification method.
- The test engineer can also take advantage of application programming interfaces (API) calls and dynamic link library (.dll) files.

Test Development Guidelines

Other Guidelines / API Calls and .dll Files

- The test team needs to verify whether its **chosen test tool supports the use of API calls.**
- Can consult the test tool vendor to identify the various APIs that are **compatible** with the tool.
- To create an API call, the test engineer declares the API function within the test procedure code.

Automation Infrastructure

Automation Infrastructure

Table-Driven Test Automation

- Data, controls, commands, and expected results are incorporated in testing => **test script code is separated from data.**
- Distinguish between the action of determining ***“what requirements to test”*** and the effort of determining ***“how to test the requirements”.***

Automation Infrastructure

Table-Driven Test Automation

Table Used to Generate Automated Testing Script

Window (VB Name)	Window (Visual Text)	Control	Action	Arguments
StartScreen	XYZ Savings Bank	—	SetContext	—
PrequalifyButton	Prequalifying	PushButton	Click	—
frmMain	Mortgage Prequalifier	—	SetContext	—
FrmMain	File	—	MenuSelect	New Customer

Automation Infrastructure

PC Environment Automated Setup Script

- The state of the PC **environment** should **exactly match** the state during the recording stage to permit a script to be played back successfully.
- The test team should develop a **common and often reused test environment** setup script => verify the PC configuration.
- The test environment should be **configured** in **the same way.**
- The computer drive mapping is the same for all pertinent PCs and verify all network connections.

Automation Infrastructure

PC Environment Automated Setup Script

- **Synchronize** the date and time for each desktop computer.
- Verify that the **correct version** of the AUT is in place.
- Checks the installation of all dynamic link libraries (.dll) and verifies registry entries.
- Could turn on or off the operation of system messages that might interfere with the development or playback of the scripts.
- **Disk space/memory.**

Automation Infrastructure

Automated Recording Options

- Most automated test tools **allow** the test engineer to set up **script recording options** in various ways (a number of parameters, such as mouse drags, certain objects, window settings, and an object window)
- Most automated test tools will provide the **capability** to set up script playback options.

Automation Infrastructure

Login Function

- **Login script can start and end at a specific point.**
- Can start the AUT and verifies the login ID and password.
- Can verify the available memory before and after the application starts, among other things.
- Can be called at the beginning of all procedures.

Automation Infrastructure

Exit Function

- Includes the selection of an **Exit button**, selection of the Exit value from the File pull-down menu, selection of the Close value under the File pull-down menu, and double-clicking on the Window Control menu box…

Navigation

- **Navigation** test procedure sare **not designed to validate** specific functional requirements, but rather to **support user interface actions.**

Automation Infrastructure

Verifying GUI Standards

- GUI standards (such as those examining font, color, and tab order choices) that could be run as **part of a regression test,** rather than as part of the system test (can write rules in .csv files).

Smoke Test (build verification)

- Focuses on automating the **critical** high-level functionality of the application.
- If the testing result **does not meet expectations** and the failures do not result from config/script… => the software as **not ready for test.**

Automation Infrastructure

Error-Logging Routine

- The test engineer can create an **error checking routine** by recording any error information that the tool might not already collect.

Help Function Verification Script

- A help verification script can verify that the help functionality is accessible, that the content is correct, and that correct navigation is occurring.

Automation Infrastructure

Timed Message Boxes Function

- This function allows message boxes to be displayed and then disappear after a specified amount of time, so as to avoid script failure.

Advanced Math Functions

- Advanced math functions, once created, can be reused whenever applicable for any AUT.

Chapter summary

- ★ Test development involves the creation of test procedures that are maintainable, reusable, simple, and robust, which in itself can be as challenging as the development of the AUT.To maximize the benefit derived from test automation, test engineers need to conduct test procedure development, like test design, in parallel with development of the target application.
- ★ The test development architecture provides the test team with a clear picture of the test development preparation activities (building blocks) necessary to create test procedures. This architecture graphically depicts the major activities to be performed as part of test development. The test team must modify and tailor the test development architecture to reflect the priorities of the particular project.

Chapter summary

- ★ Test procedure development needs to be preceded by several setup activities, such as tracking and managing test environment setup activities, where material procurements may have long lead times.
- ★ Prior to the commencement of test development, the test team needs to perform an analysis to identify the potential for reuse of existing test procedures and scripts within the automation infrastructure (reuse library).
- ★ The test procedure development/execution schedule is prepared by the test team as a means to identify the timeframe for developing and executing the various tests. This schedule identifies dependencies between tests and includes test setup activities, test procedure sequences, and cleanup activities.

Chapter summary

★ Prior to the creation of a complete suite of test procedures, the test team should perform modularity-relationship analysis. The results of this analysis help to clarify data independence, plan for dependencies between tests, and identify common scripts that can be repeatedly applied to the test effort. A modularity-relationship matrix is created that specifies the interrelationship of the various test scripts. This graphical representation allows test engineers to identify opportunities for script reuse in various combinations using the wrapper format, thereby minimizing the effort required to build and maintain test scripts.

Chapter summary

- ★ As test procedures are being developed, the test team needs to peform configuration control for the test design, test scripts, and test data, as well as for each individual test procedure. Groups of reusable test procedures and scripts are usually maintained in a catalog or library of test data called a testbed. The testbed needs to be baselined using a configuration management tool.
- ★ The team next needs to identify the test procedure development guidelines that will apply on the project to support the various test development activities. These guidelines should apply to the development of both manual and automated test procedures that are reusable and maintainable.

Chapter summary

★ An automation infrastructure (or automation framework) is a library of reusable functions that may have been originally created for test efforts on different projects or to test multiple or incremental versions of a particular application. The key to these functions is their potential for reuse, which minimizes the duplication of the test procedure development effort and enhances the reusability of the resulting test procedures.

References

[1] “Analyze a little, design a little, code a little, test a little” has been attributed to Grady Booch.

[2] Linz, T., Daigl, M. GUI Testing Made Painless, Implementation and Results of the ESSI Project Number 24306. Moehrendorf, Germany, 1998. www.imbus.de

[3] http://www3.sympatico.ca/michael.woodall/sqa.htm.

[4] http://www3.sympatico.ca/michael.woodall.

[5] VB Flavor Hungarian Notation: www.strangecreations.com/library/c/naming.txt for. Also see http://support.microsoft.com/support/kb/articles/Q110/2/64.asp.

References

[6] Linz, T., Daigl, M. How to Automate Testing of Graphical User Interfaces, Implementation and Results of the ESSI Project Number 24306. Moehrendorf, Germany, 1998. www.imbus.de.

[7] Pettichord, B. Success with Test Automation. (Web page.) 1995. www.io.com/~wazmo/qa.html.

[8] Jacobson, I., Booch, G., Rumbaugh, J. The Unified Software Development Process. Reading, MA: Addison-Wesley, 1999.