PassQueen
a.k.a dumpsqueen.com

# ISTQB Certified Tester Advanced Level-Test Automation Engineering

## BCS TAE

**Version Demo**

**Total Demo Questions: 7**

**Total Premium Questions: 79**

**Buy Premium PDF**

https://passqueen.com

support@passqueen.com

# Topic Break Down

| Topic | No. of Questions |
|---|---|
| Topic 1, Introduction and Objectives for Test Automation | 3 |
| Topic 2, Preparing for Test Automation | 13 |
| Topic 3, Test Automation Architecture | 33 |
| Topic 4, Risks and Contingencies | 7 |
| Topic 5, Test Automation Reporting and Metrics | 13 |
| Topic 6, Transitioning Manual Testing to an Automated Environment | 3 |
| Topic 7, Verifying the Test Automation Solution | 5 |
| Topic 8, Continuous Improvement | 2 |
| Total | 79 |

QUESTION NO: 1

You are working on a TAS for standalone application. The automated tests are developed based on a automation framework that allows interaction with GUI elements using on object orientated API. The GUI elements include menus, buttons, radio buttons, text toolbars and their properties.

Whilst automating a test, you have discovered that the GUI elements of some third party components are not identifiable by the automated tool you are using.

Which of the following is the FIRST step that you take to investigate this issue?

**A.** Verify the testability support with the providers of the third party components

**B.** Verify whether the GUI identification depends on the browser.

**C.** Adopt an approach that uses the coordinates of the GUI elements instead

**D.** Verify whether naming standards for variables and have been defined for the current automation solution

**ANSWER: A**

**Explanation:**

Verify the testability support with the providers of the third party components is the appropriate first action because the issue concerns whether the automation framework can access and identify controls supplied by an external component library. Object-oriented GUI automation relies on supported control types, exposed object properties, stable identifiers, and accessibility or automation interfaces. A third-party control may require a specific adapter, plug-in, version, configuration setting, or an update from its provider before it can be recognized reliably by the selected tool.

Investigating this support first establishes the technical capability and supported integration path at the source, rather than prematurely designing a workaround. The component provider can confirm which automation technologies and control properties are exposed, document any required configuration, and identify compatibility constraints between the component version and the automation tool. This aligns with the Test Automation Engineer syllabus's emphasis on testability, maintainable automation, and selecting implementation approaches that provide dependable interaction with the system under test. It also helps ensure that the resulting tests interact through intended interfaces and remain resilient to GUI changes. See the ISTQB Test Automation Engineer certification page and the ISTQB Advanced Level certification information.

**QUESTION NO: 2**

Which of the following BEST describes why it is important to separate test definition from test execution in a TAA?

**A.** It allows developing steps of the test process without being closely tied to the SUT interface.

**B.** It allow choosing different paradigms (e.g event-driven) for the interaction TAS and SUT

**C.** It allows specify test cases without being closely tied to the tool to run them against the SUT

**D.** It allows testers to find more defects on the SUT

**ANSWER: C**

**Explanation:**

It allows specify test cases without being closely tied to the tool to run them against the SUT. This separation is a core test automation architecture principle because a test's intent, conditions, data, and expected results should be expressed independently from the technical mechanisms used to drive the system under test (SUT). The test definition states what is to be verified, while the execution layer translates that definition into tool-, interface-, and platform-specific actions. This improves maintainability: changes to the automation tool, drivers, user interface technology, or execution environment can be addressed mainly in the execution implementation without requiring the underlying test cases to be rewritten. It also promotes portability and reuse, since the same logical test definitions can potentially be executed through different tools or adapters. In a Test Automation Architecture (TAA), this abstraction helps prevent automated tests from becoming tightly coupled to one particular automation solution and supports sustainable automation over the product lifecycle. The ISTQB

Advanced Level Test Automation Engineering syllabus describes the importance of layered architecture and separating test definition from the technical details of test execution. See the ISTQB Test Automation Engineer certification page and the CTAL-TAE syllabus resources.

## QUESTION NO: 3

As a TAE you are evaluating a functional test automation tool that will be for several projects within your organization. The projects require that tool to work effectively and efficiently with SUT's in distributed environments. The test automated tool also needs to interface with other existing test tools (test management tool and defect tracking tool.) The existing test tools subject to planned updates and their interface to the test automated tool may not work property after these updates.

Which of the following are the two LEAST important concerns related to the evaluation of the test automation in this scenario?

A) Is the test automation tool able to launch processors and execute test cases on multiple machines in different environments?

B) Does the test automation tool support a licensing scheme that allows accessing different sets?

C) Does the test automation tool have a large feature set, but only part of the features will be sets?

D) Do the release notes for the planned updates on existing specify the impacts on their interfaces to other tools?

Does the test automation tool need to install specific libraries that could impact the SUT?

**A.** A and C

**B.** A and E

**C.** B and E

**D.** C and D

## ANSWER: C

**Explanation:**

**B and E** is the best answer because these matters are comparatively less important than the scenario's explicit technical priorities when evaluating the tool. The stated needs focus on effective execution in distributed environments and on preserving interoperability with the organization's existing test-management and defect-tracking tools as those tools change. Tool evaluation should therefore primarily establish whether the automation solution can operate reliably across the required execution topology and whether its integrations can be maintained through planned interface changes.

A licensing arrangement that controls access to different sets is relevant to procurement, administration, and rollout planning, but it does not directly establish whether the tool will meet the required distributed-execution and integration capabilities. Likewise, whether supporting libraries must be installed on the system under test is a deployment consideration that should be recorded and managed, but it is not as central to the stated selection risks as distributed operation and interface compatibility. The Test Automation Engineer syllabus treats tool selection as a context-driven activity: evaluation criteria should be prioritized according to the project's automation requirements, constraints, and integration needs. See the ISTQB Certified Tester Advanced Level—Test Automation Engineering certification information and the BCS software testing certifications overview.

## QUESTION NO: 4

You have been asked to automate a set of functional tests at system Test level via the CLI of the SUT for the first release of a software system. The automated tests will be delivered to the learn in change of maintenance testing, who will use them for part of the regression testing. They have the following requirements.

1. The automated tests must be as fast and cheap to maintain as possible

2. The cost of adding new automated tests must be as low as possible

3. The automated tests must have a high level of independence from the tool itself

Which of the following scripting techniques would be MOST suitable?

**A.** Data-driven scripting

**B.** Keyword-driven scripting

**C.** Linear scripting

**D.** Structured scripting

**ANSWER: B**

**Explanation:**

Keyword-driven scripting is the most suitable technique because it separates the description of a test from the technical implementation that executes it. Tests are assembled from business-meaningful keywords, while the code that maps each keyword to CLI actions is held in a reusable automation layer. This separation lets maintenance testers update test flows, test data, and expected outcomes without routinely changing low-level automation code. Consequently, regression tests can be maintained efficiently when the system changes.

New automated tests can also be created at relatively low cost by combining existing keywords in new sequences. Only genuinely new CLI operations require implementation of an additional keyword; once implemented, that keyword is reusable throughout the suite. The approach therefore supports scalable growth of the regression pack while avoiding unnecessary duplication.

Importantly, the keyword layer provides abstraction from a particular automation tool. The test assets express intended actions and checks rather than tool-specific commands. If the underlying execution technology must change, the keyword implementations can be adapted while preserving the higher-level test definitions to a substantial extent. This aligns with the Test Automation Engineer syllabus's emphasis on maintainable, reusable testware and suitable abstraction layers. See the ISTQB Test Automation Engineer certification page and the CTAL-TAE syllabus resources.

**QUESTION NO: 5**

Consider a TAS deployed into production. The SUT is a web application and the test suite consists of a set of automated regression tests developed via GUI. A keyword-driven framework has been adopted for automating the regression tests. The tests are based on identification at low-levels of the web page components (e.g class indexes, tab sequence indexes and coordinates) in the next planned release the SUT will be subject to significant corrective maintenance (bug-fixes) and evolution (new features) Maintenance costs to update the test scripts should be as low as possible and the scripts must be highly reusable.

Which of the following statements is most likely to be TRUE?

**A.** The keyword-driven framework is not suitable, it would be better to adopt a structured-scripting approach

**B.** False positive errors are likely to occur when running the automated tests on the new releases without modifying the test

**C.** The total execution time of the automated regression test suite will decrease for each planned release.

**D.** The keyword-driven framework introduces a level abstraction that is too high and makes it difficult what really happens

**ANSWER: B**

**Explanation:**

False positive errors are likely to occur when running the automated tests on the new releases without modifying the test. The automated GUI checks identify components through fragile, implementation-specific properties, including class indexes, tab ordering, and screen coordinates. Corrective maintenance and new functionality can legitimately alter page layout, control ordering, styling, or the generated structure of a page even when the business behaviour that the regression test intends to verify remains correct. Consequently, an existing script can fail because it can no longer find or interact with the expected component, and the failure may be reported as a test result despite no defect existing in the SUT. This is a false positive and requires investigation before it can be classified correctly. Keyword-driven automation can support reuse by

separating business-level test intent from the technical implementation of keywords, but that benefit depends on stable, maintainable interfaces beneath those keywords. The TAE syllabus emphasises maintainability and appropriate abstraction in a test automation architecture, including reducing dependence on volatile SUT details. Stable, semantic identifiers and centralized object recognition would make the affected technical layer easier to update and reduce these spurious failures. See the ISTQB Certified Tester Advanced Level Test Automation Engineering syllabus and W3C guidance on identification of user-interface elements.

## QUESTION NO: 6

The GUI of a Customer Relationship Management (CRM) application has been delivered through internet Explorer with proprietary Active X and Java controls. This implementation enables rich client capabilities, but specific commercial automation tools are necessary to automate test cases at GUI of functional test cases. This is to demonstrate whether a small set of the commercial are able to properly recognize actions takenby a tester when interacting with GUI of the CRM application.

Which of the following scripting techniques would be MOST suitable in this scenario?

**A.** Data-driven scripting

**B.** Keyword-driven scripting

**C.** Linear scripting

**D.** Structure scripting

## ANSWER: C

**Explanation:**

**Linear scripting** is most suitable because the immediate objective is a technical feasibility demonstration: establish whether each candidate commercial tool can recognise and automate interactions with the CRM GUI, including its proprietary ActiveX and Java controls. A linear script can be created quickly by recording, then replaying, a representative sequence of tester actions. This directly exposes whether the tool can identify the relevant GUI objects, capture user actions reliably, and reproduce those actions against the application. It therefore provides prompt, practical evidence for selecting a tool or identifying integration limitations before investing in a more maintainable automation architecture.

The ISTQB Test Automation Engineer syllabus discusses linear scripting as an approach that is particularly applicable to small-scale automation and record-and-playback situations. In this case, maintainability, broad data coverage, and business-readable abstraction are not the primary decision criteria; recognition of the rich-client interface is. A short linear proof-of-concept gives an efficient basis for evaluating that capability using realistic functional workflows. See the ISTQB Certified Tester Advanced Level Test Automation Engineer certification page and the ISTQB Advanced Level certification overview.

## QUESTION NO: 7

An automated UI test verifies that an order can be submitted. The test passes whenever a confirmation message is displayed, even if the application creates an order with an incorrect total amount. The order total is available through an API after submission.

Which of the following would MOST improve the test oracle?

**A.** Verify the persisted order total through the API

**B.** Increase the display time of the confirmation message

**C.** Replace the expected order total with the actual displayed total

**D.** Remove the confirmation-message check from the test

## ANSWER: A

**Explanation:**

Verifying the persisted order total through the API would most improve the test oracle because it checks a business-relevant outcome of the submission. The confirmation message demonstrates that a UI response was displayed, but it is not sufficient evidence that the order was calculated and stored correctly.

A useful automated oracle compares an observable actual result with an expected result that represents the test objective. Combining the UI confirmation with an API-level verification of the created order provides stronger observability and can identify defects that a presentation-only check would miss. See the ISTQB Glossary definition of test oracle.