iSQI
CTAL-TAE_V2
ISTQB Certified Tester Advanced Level - Test Automation Engineering CTAL-TAE (Syllabus v2.0)

For More Information – Visit link below:

https://www.examsempire.com/

Product Version

1. Up to Date products, reliable and verified.
2. Questions and Answers in PDF Format.



Latest Version: 6.0

Question: 1

You are working for a clothing retailer that has big stores in many cities. Your TAS, for the first time, has just been used for reliability and stress testing of the overnight batch processing that collects sales transactions from the stores and adds them to a data warehouse; functional suitability tests have been reused for these runs.
All tests have finished normally but their results have not been satisfactory, and it has been difficult to understand the nature of the defect or defects that have been causing the failures.
What data, if added to the test logs, would MOST likely help with debugging in this situation?
a) Dynamic data about the status of the test platforms
b) Screenshots taken during test execution
c) Crash dumps and stack traces
d) Values of randomly generated test input data
e) Counts of the quantity of test cases executed during each run

A. c and d
B. a and e
C. a and d
D. b and c

Answer: B

Explanation:
For reliability and stress testing, CTAL-TAE specifically recommends capturing dynamic information about the SUT and recording execution counters. Dynamic platform information can expose conditions such as resource utilization, memory leaks, or other runtime degradation that may explain failures occurring during intensive overnight processing. The syllabus also states that when reliability or stress testing involves numerous test cycles, a counter should be logged so investigators can determine how many times the test cases were executed and correlate failures with prolonged execution. Crash dumps and stack traces are particularly relevant when the SUT crashes, but the scenario explicitly states that the tests finished normally. Random input values are important only when randomized testing is used, which is not stated. Screenshots provide little diagnostic value for backend overnight batch reliability testing. Therefore, a and e are the most useful data.

Question: 2

In which way would a feature toggle configuration be used?

A. To identify which test suites to execute
B. To identify the test data sets needed for a test suite
C. To identify a test environment configuration
D. To identify which oracle to use for verification

Answer: A

Explanation:
A feature toggle configuration can be defined for each SUT release or test environment to indicate which features are active. The automated testware can associate specific test cases and test suites with those features and then use the toggle configuration to determine which suites should execute for the applicable release or environment. This prevents automated tests for inactive functionality from being executed unnecessarily or producing misleading failures. Feature toggles therefore support test-selection and release-configuration management. They do not primarily determine which data sets are needed, define the test environment itself, or choose the test oracle used for verification. The CTAL-TAE syllabus explicitly identifies selection of the appropriate test suites as the test automation use of feature toggle configurations.

Question: 3

A release candidate of a SUT, after being fully integrated with all other necessary systems, has successfully passed all required functional tests (90% were automated tests and 10% were manual tests). Now, it is necessary to perform reliability tests aimed at evaluating whether, under certain conditions, that release will be able to guarantee an MTBF (Mean Time Between Failures) in the production environment higher than a certain threshold (expressed in CPU time). Which of the following test environments is BEST suited to perform these reliability tests?

A. Local development environment
B. Build environment
C. Integration environment
D. Preproduction environment

Answer: D

Explanation:
Reliability testing (e.g., long-duration runs, endurance/soak, stability measurements, MTBF assessment) requires an environment that closely resembles production in terms of configuration, resource allocation, deployment topology, integrations, and operational characteristics. TAE guidance emphasizes that measurements like MTBF are highly sensitive to environmental differences such as CPU quotas, background load, database sizing, network topology, virtualization settings, and monitoring agents. A local development environment is unsuitable because it is not representative, is often unstable, and typically lacks full system integration. A build environment focuses on building/packaging and fast verification, not production-like reliability evaluation. An integration environment can validate that systems work together, but it is frequently shared, changes often, and may not match production sizing and operational constraints; it is also commonly disrupted by other teams' deployments. Preproduction (often called staging) is designed to be the closest safe approximation to production while still allowing controlled testing, including reliability and performance-related evaluations, without risking real users or live data. Therefore, preproduction is the best-suited environment to run reliability tests intended to predict production MTBF behavior with credible confidence.

Question: 4

You are evaluating the best approach to implement automated tests at the UI level for a web app. Specifically, your goal is to allow test analysts to write automated tests in tabular format, within files that encapsulate logical test steps related to how a user interacts with the web UI, along with the corresponding test data. These steps must be expressed using natural language words that represent the actions performed by the user on the web UI. These files will then be interpreted and executed by a test execution tool. Which of the following approaches to test automation is BEST suited to achieve your goal?

A. Test-driven development
B. Keyword-driven testing
C. Data-driven testing
D. Linear scripting

Answer: B

Explanation:
The described goal matches the defining characteristics of keyword-driven testing: tests are expressed using keywords (action words) that represent user operations, often arranged in tabular form with parameters/test data. TAE describes keyword-driven approaches as enabling non-programmers (e.g., test analysts) to create and maintain tests by combining high-level keywords such as “Open Browser,” “Click,” “Enter Text,” “Select,” “Verify Text,” etc., while the underlying automation framework maps those keywords to executable code. The use of files interpreted by a test execution tool is also typical: keyword tables (or similar structured specifications) are read and executed by the automation engine. Data-driven testing focuses on separating test logic from test data, typically running the same script multiple times with different datasets; it does not inherently require natural-language action words or tabular step definitions (though it can be combined). Linear scripting is code-centric and not aligned with analyst-authored natural language step tables. TDD is unrelated to the requirement of tabular, natural-language keyword specification for UI test steps. Therefore, keyword-driven testing is the best fit for the stated approach.

Question: 5

An automated test case that should always pass sometimes passes and sometimes fails intermittently (non-deterministic behavior) when executed in the same test environment, even if no code (i.e., SUT code or the test automation code) has been changed. Which of the following statements about the root cause of this non-deterministic behavior is TRUE?

A. The specified root cause is a race condition that can be identified by also analyzing the log files of the test case, the SUT, and the TAF
B. Determining the specified root cause may require, in addition to the TAE, the support of others such as developers and system engineers
C. The specified root cause must be in the instability of the test environment, since no code has been changed

D. Determining the specified root cause is certainly easier than if the automated test always fails (deterministic behavior)

Answer: B

Explanation:
TAE treats non-deterministic (flaky) test behavior as a symptom that can originate from multiple sources: timing and synchronization issues, race conditions, concurrency, environmental variability (resource contention, network latency), unstable test data, third-party dependencies, or hidden state leakage between tests. Because these causes often span boundaries—application code, infrastructure, deployment configuration, test tooling, and data pipelines—finding the true root cause frequently requires collaboration beyond the TAE role. Developers may need to inspect application logs, thread behavior, and recent architectural assumptions; system engineers may need to analyze resource saturation, container orchestration events, network anomalies, or environment drift. Option A is too specific and assertive: the root cause is not necessarily a race condition, and logs may not be sufficient to identify it. Option C is incorrect because no code change does not imply the environment is the only cause; flaky behavior can stem from hidden nondeterminism in the system or tests that is always present but only sometimes triggers. Option D is also incorrect; intermittent failures are often harder to diagnose than consistent deterministic failures because evidence is less reproducible. Therefore, the true statement is that determining the root cause may require support from developers and system engineers in addition to the TAE.

Question: 6

Your TAS is being used for automated functional suitability testing of a stock management system. Several tests have crashed. Despite the fact that the SUT creates detailed logs of all its activity, it has been difficult to determine the reason(s) for these crashes from the TAS and SUT test logs.
What actions are MOST likely to help with debugging?
a) Ensure that the TAS logs a count of how many times each test case is run.
b) Ensure that the SUT logs all available configuration data when it starts up.
c) Ensure that the SUT logs all available crash dumps and stack traces in the event of failure.
d) Ensure that the SUT logs all values of randomly generated test input data.
e) Ensure exact time synchronization of all test logs before any tests are run.

A. b and e
B. a and d
C. c and d
D. b and c

Answer: D

Explanation:
Configuration information and crash diagnostics are the most relevant additions for investigating unexplained crashes. CTAL-TAE recommends recording configuration information when the SUT starts, including software or firmware versions, SUT configuration, and operating-system configuration. This enables investigators to determine whether a particular environment or configuration contributed to

the failure. The syllabus also emphasizes preserving crash dumps and stack traces whenever failures occur because these provide direct technical evidence about the point and cause of a crash. Execution counters are particularly useful during reliability or stress testing involving repeated cycles, which is not stated here. Random input values should be logged when randomized tests are being used, but the scenario does not indicate random test generation. Therefore, actions b and c most directly improve diagnosis of the described crashes.

Question: 7

You are a TAE responsible for a TAS that is used on several products, all of which have complex UIs. A substantial proportion of the automated tests break on a regular basis, usually for reasons associated with the UI. The investigation and maintenance this requires is significantly reducing the productivity of the teams and the return on investment.
You are looking for ways to improve the test cases in this TAS. Which of the following approaches would be MOST appropriate?
a) Artificial intelligence
b) Test histograms
c) Reliability growth modelling
d) Schema validation
e) Operational profiles

A. d and e
B. a and b
C. c and d
D. b and c

Answer: B

Explanation:
Both artificial intelligence and test histograms directly address unstable UI automation. CTAL-TAE explains that test histograms visualize test-result and test-data trends and help TAEs identify fragile test cases that should be refactored or redesigned. This is particularly useful when a substantial proportion of tests repeatedly break. AI and machine-learning techniques provide an additional UI-specific improvement: tools can detect changed locator values, identify replacement selectors through ML or image-recognition techniques, and apply self-healing mechanisms that reduce locator-related maintenance. Schema validation is primarily applicable to API and database data structures rather than complex UI instability. Reliability growth modelling and operational profiles address software reliability concerns rather than recurring automated UI maintenance. Therefore, combining test histograms for identifying fragile tests with AI-based self-healing for UI locator changes gives the strongest solution.

Question: 8

What would be a goal of test automation in a Preproduction environment?

A. To build the software and check correctness of the build
B. To verify component integration

C. To verify component functionality
D. To assess non-functional characteristics

Answer: D

Explanation:
A preproduction environment is designed to resemble the production environment as closely as practicable and is therefore particularly appropriate for assessing non-functional quality characteristics. CTAL-TAE specifically identifies performance efficiency as an example of the type of characteristic commonly assessed in preproduction. Although non-functional testing can theoretically occur in other environments, preproduction provides greater realism in infrastructure, configuration, integration, and operating conditions. Building software and verifying the correctness of the resulting build belong primarily to the build environment. Component functionality is normally verified earlier, such as within local development or build environments, while broader integration testing occurs within integration environments. Preproduction may also support user acceptance testing and execution of existing automated suites, but among the listed choices its distinguishing automation objective is assessment of non-functional characteristics.

Question: 9

A TAS that performs automated testing in a single test environment was successfully manually installed and configured from a central repository, with all its components in the correct versions. It was also verified that all TAS components in this environment are capable of providing reliable and repeatable performance. The TAS will be used to run several suites of automated regression test scripts on various SUTs in the test environment. Your current goal is to complete all preliminary verifications to ensure that the TAS works correctly. Which of the following activities would you perform FIRST?

A. Create scripts to automatically install and configure the TAS in the test environment from the central repository
B. Check whether the TAS connectivity to all required internal systems, external systems, and interfaces is available
C. Run a given suite multiple times using TAS to determine whether all regression test scripts always provide the same result
D. Check whether all regression test scripts in a given suite have expected results

Answer: B

Explanation:
TAE differentiates verifying the automation environment and infrastructure (the ability of the TAS to operate) from verifying the test suites' correctness (the behavior of specific automated tests). The scenario states the TAS was installed correctly and its components perform reliably in isolation. The next preliminary verification is ensuring the TAS can actually interact with the necessary systems and interfaces required to execute tests end-to-end: SUT endpoints, browsers/devices, authentication services, databases, messaging systems, third-party integrations, and any CI/CD or artifact services it must access. If connectivity is missing or unstable, any subsequent suite executions or repeatability checks can fail for reasons unrelated to test logic, creating noise and wasted investigation. Creating

installation scripts (A) is valuable for scalability, but it is not needed to confirm the TAS works in the already-installed single environment. Checking expected results in scripts (D) and running suites repeatedly for determinism (C) are important, but they assume the TAS can reliably reach all required dependencies. TAE recommends validating connectivity and access prerequisites early as a gate for meaningful execution. Therefore, the first activity is to verify TAS connectivity to all required internal/external systems and interfaces.

Question: 10

Which of the following statements about the relationship between TAA, TAS and TAF is true?

A. A TAF can be used to implement a TAS, which is an implementation of a TAA
B. A TAS can be used to implement a TAF, which is an implementation of a TAA
C. A TAS can be used to implement a TAA, which is an implementation of a TAF
D. A TAF can be used to implement a TAA, which is an implementation of a TAS

Answer: A

Explanation:
In TAE terminology, the Test Automation Architecture (TAA) is the conceptual, high-level blueprint that describes how automation will be structured, what layers exist, how components interact, and how the automation connects to the SUT and supporting systems. The Test Automation Solution (TAS) is the concrete realization of that architecture in a specific context—tools, infrastructure, pipelines, conventions, and components assembled to deliver automated testing capability. The Test Automation Framework (TAF) is a structured set of reusable libraries, guidelines, and mechanisms that supports efficient development, execution, reporting, and maintenance of automated tests; it is commonly a key part used to build the TAS. TAE documents commonly present this relationship as: TAA (design) → implemented as TAS (solution) → constructed using one or more TAFs (framework elements) plus tools and environment components. Options B, C, and D invert these relationships and misrepresent the concept that architecture is implemented by a solution, not the other way around. Therefore, the statement that a TAF can be used to implement a TAS, which is an implementation of a TAA, is the correct relationship.

Thank You for Trying Our Product

Special 16 USD Discount Coupon: NSZUBG3X

Email: support@examsempire.com

Check our Customer Testimonials and ratings available on every product page.

Visit our website.

https://examsempire.com/