Certified Tester Quality in DevOps (CT-QDO) Accreditation Guidelines

v1.0

International Software Testing Qualifications Board

Revision History

Version	Date	Remarks
v1.0	2026/02/13	Release
v0.4	2026/01/15	Technical Editing after Beta Release version according to template v2.4
v0.3	2025/08/25	Beta Release version according to template v2.4
v0.2	2025/07/01	TechnicalEditing version according to template v2.4
v0.1	2025/03/20	Alpha version according to template v2.4

Table of Contents

0 Copyright

This document was formally released by the General Assembly of the ISTQB® on 3rd of May 2024.

1 Introduction

Accreditation is a key part of the work that ISTQB® and the individual Member Boards perform for both Training Providers and students.

As a student, studying for your ISTQB® qualifications with an Accredited Training Provider, it will give you assurance that they have been independently assessed and accredited as a suitable organization to carry the ISTQB® Training Provider logo for the course that you are studying with. Therefore, you can study in the knowledge that the company who you select to do your training with will be able to provide you the right level and scope of information to help you prepare for the ISTQB® certification exam.

As a Training Provider, having your course accredited will mean your company name is published on the ISTQB® website and in some cases the individual Member Board that accredited you as an Accredited Trainer Provider. This will provide Training Providers with focused marketing and include additional benefits shown below.

This document provides a high-level overview of the guidelines a Training Provider or a Training Material Owner has to meet if they wish to be accredited as a provider of any ISTQB® courses.

Accreditation is undertaken by an ISTQB® member board. This guideline includes information regarding how to apply, what input to provide and describes the accreditation process.

2 Objectives

The objectives of the accreditation process are:

1. To define in general terms what Training Providers have to produce in order to have their course considered for accreditation.
2. To ensure that any course submitted for accreditation meets the requirements of the relevant ISTQB® syllabus through independent assessment.
3. To provide students with the confidence that any Accredited Training Provider they choose will have been assessed by qualified reviewers and that any Training Materials meet the individual syllabus requirements.
4. To provide ongoing governance to ensure that Training Providers continually meet the requirements of a syllabus and their exam pass rates are of an acceptable standard.

3 ISTQB® Accreditation Options

The ISTQB® accreditation process includes the following accreditation options:

- Accreditation as a Training Provider
  - As a Training Provider you will be providing training of one or more accredited courses. You may have developed the Training Material yourself or are licensing courses from another Training Provider or Training Material Owner.
- Accreditation as a Training Material Owner
  - A Training Material Owner is a supplier, who owns and develops accredited Training Material, but don't necessarily train the materials.
- Trainers
  - There are some member boards who as part of their accreditation also accredit the individual Trainers. Please approach your local Member Board for their accreditation process.

4 ISTQB® Accreditation Process

To ensure consistency of accreditation all ISTQB® Member Boards are required to follow a defined minimum accreditation process that has been developed and released by ISTQB®. As each country is unique there may be local initiatives or legislation that require slightly modified approaches in some aspects. Any Member Board shall be able to provide details of their specific accreditation process on request.

4.1 Fees

Each Member Board will provide details of their accreditation fees on request.

4.2 Application

- Each Member Board shall provide on request their application form that details all information required to enable Training Providers to apply for accreditation. This will include but may not be limited to:
  - Copies of all the proposed training slides
  - The detailed training timetable showing the trainings times compared to the syllabus including time proposed for exercises, practice exams, any homework and breaks.
  - Lecturer notes that show the information that lecturers are expected to teach for each slide.
  - Student notes (or copies of books if they are to be used in place of your own written notes)
  - Detailed exercises with the expected outcomes relating to all the course learning objectives.
  - Practice exams with answers
  - Details of the trainers including their training testing experience. Trainers will need to hold the certification they are planning to teach.
- The board will acknowledge receipt of the application and advise the latest date that a response will be received.
- For the specific list of requirements for any country, please contact the relevant ISTQB® Member Board.

4.3 Review of the Application

Each Member Board has a panel of reviewers from which resource will be selected to assess each submission. Their role is to ensure the Training Material submitted is of suitable quality and to meet the requirements of the relevant syllabus. The Accreditation panel may not use sampling methods (i.e., evaluating some sections instead of the full Training Material). All Training Material provided shall be evaluated.

If the application is from a Training Provider who license Training Material that is already accredited, the reviewers only need to assess the suitability of the training provider and instructor(s) given the Training Materials are already accredited.

If a Training Provider is accredited in one country location, their accreditation is automatically valid with all ISTQB® Member Boards. If a Training Provider wishes to deliver training in a different country to the one they are accredited in, they should seek guidance of any local issues from the local Member Board of the country they intend to train in and see what their requirements are for running exams.

Once the accreditation application has been reviewed, the response from the Member Board will fall into the following categories:

- **Approved** – approved with few or no changes required. If a few changes are required, confirmation from the Training Provider may be required once they have been completed. Details will be provided in the Member Board response.
- **Deferred** – additional information or changes to the submission may be required before being able to approve accreditation. Progression of the application will be "on hold" until all the requirements are met.
- **Decline** – the submission does not meet the accreditation requirements. If an application is declined the applicant may re-apply again at any time in the future and pay the appropriate accreditation fee.

Once the application is approved, for the duration stated by the Member Board, the Training Provider will be issued with the appropriate ISTQB® logos that can be used on marketing collateral. Details of the Training Provider and the course that has been accredited will appear on the ISTQB® website. Member Boards may also publish the Accredited Training Providers on their website.

4.4 Re-accreditation

At the end of the accreditation period a re-submission is normally required to ensure the course still meets the syllabus objectives. Each Member Board will be able to provide Training Providers with details required for re-accreditation. This may be a full set of materials as per the initial accreditation or a letter confirming that no changes have been introduced since accreditation was granted.

4.5 Surveillance

It is the duty of each Member Board to perform due diligence and ensure that courses are being taught to the correct accredited version of the materials. During the accreditation period Training Providers may be visited by a reviewer to monitor a short part of the training course. A report will be provided back to the Training Provider along with any recommendations if required.

Member Boards will as a matter of course monitor the ongoing exam results for each Training Provider and if pass rates fall lower than expected within that country, they will be contacted to offer guidance and support as required to improve the results.

5 What do We Evaluate?

5.1 Syllabus and Learning Objectives

The Training Material should cover the entire syllabus and all the learning objectives defined in it. A straight copy of the syllabus words onto training slides is not sufficient. It is expected that Training Material will expand on the syllabus words to ensure understanding.

The training material shall provide the student with the knowledge and skills of each chapter and section as outlined in the syllabus and body of knowledge (if it exists for the given syllabus), in line with the defined K-level for the given learning objective. (see K-level definitions in the syllabus appendix section).

The Training Material to be accredited shall demonstrate coverage of all applicable learning objectives. Applications for accreditation shall include a traceability matrix showing coverage of the learning objectives in terms of presentation and supporting materials.

The amount of training including hours or days should meet the requirements of the syllabus being taught. For each module, all chapters shall be covered with at least as much time as required in the syllabus. Applications for accreditation shall include a timing matrix showing the time allocated per chapter.

All learning objectives shall have lectures, examples, and exercises according to their K-level, as defined in this document or in the Accreditation Guidelines for the certification module. All exercises shall include an expected/sample solution to help students understand the exercise.

5.2 Glossary

For any Glossary term defined in the syllabus, the Training Material must be consistent with the definition of that term in the latest released version of the ISTQB® Glossary.

Note: although ISTQB® makes its best to align between the glossary and the syllabus, deviations may occur. In case of difference between the syllabus and the glossary, the glossary prevails.

5.3 Evaluation of Examples

Every K2 and K3 learning objective (LO) must contain at least one example.

Examples shall be:

- Appropriate for the material being taught,
- Drawn upon realistic software, system, or project; i.e., trainers should not use toy or non-computer-related examples. Ideally, examples should be substantial and be drawn from real life occurrences.

5.4 Evaluation of Exercises and Answers

Every K3 learning objective (LO) or hands-on objective (HO) shall have at least one exercise.

Exercises shall be:

- Appropriate for the material, complexity and K-level of the LO, or HO-level taught,

- Drawn upon realistic software or systems projects, i.e. Trainers should not use toy or non-computer-related exercises. Ideally, exercises should be substantial and be drawn from real life occurrences.
- Each exercise shall include solutions.

For live classes, all exercises must be solved by the students in class (i.e., not as optional or required homework) and a solution reviewed in class by the instructor. For e-learning or correspondence classes, an exercise solution must be provided in the course material.

5.5 Evaluation of Trainer Notes

If the slides are not self-explanatory, notes about what instructors are expected to say on each section should be available. These trainer notes can be presenter notes in the slides or a separate document.

5.6 Practice Exams

Details of the types of exam questions are available for each syllabus. It is expected that for Training Providers to prepare students for the exam, sufficient practice exams of the type expected in the final exam will be provided with solutions. Reviewers will evaluate the quality of these exams relative to the real ones that will be used.

5.7 Evaluation of Additional Material

If trainers reference additional material (such as books or articles not referenced in the syllabus), they shall provide that material to the Accreditation authority and ensure that this material is not in conflict with the syllabi being Accredited or other ISTQB® syllabi.

5.8 Hands-on Objectives (HOs)

Like K3 learning objectives (LOs), hands-on objectives (HOs) focus on practical skills and competencies. However, HOs are not examined by the multiple-choice exam questions. It is therefore important that HOs are well covered by the Training Material as instructed by the syllabus accreditation guideline document.

To make this syllabus practical, the training provider shall follow the specific hands-on objectives levels and apply exercises as follows:

- H0: This can include a live demo of an exercise or recorded video. Since this is not performed by the trainee, it is not strictly an exercise.
- H1: Guided exercise. The trainees follow a sequence of steps performed by the trainer.
- H2: Exercise with hints. The trainee is given an exercise with relevant hints to enable the exercise to be solved within the given timeframe.
- H3: Unguided exercises without hints.

Below the description, mandatory for the exercise and expected outcomes are mandatory to check in the accreditation of training material. The examples are just examples to help in creation of training material.

QDO 1.2 (H0) Summarize How DevOps Principles are Visible in a Case Study

Description

For this hands-on objective, the 15-minute H0 level exercise supports the students in preparing them for further hands-on objectives by giving a context of a case study. The imaginary organization described in the case study should be considered as only a partial DevOps organization so that the exercises based on the further hands-on objectives can implement improvements to the ways of working within this organization.

Mandatory for the Exercise

This exercise introduces students to a shared case study that will be reused in later hands-on objectives. The goal is not to improve the organization yet, but to:

- Understand the current way of working
- Identify where DevOps concepts are already visible
- Recognize where DevOps is missing, incomplete, or inconsistent

This prepares students to apply QA and testing-related improvements in later exercises.

Example of the Exercise

This is an example that helps Trainers to design their own case study.

Case Study: “BlueBank Online Services” Organization Context BlueBank is a mid-sized financial services company that provides online banking services to private customers. The company has recently experienced increased competition from digital-only banks and has started an initiative to “move towards DevOps”.

BlueBank develops a customer-facing web and mobile application that allows users to:

- View account balances
- Make payments
- Apply for loans
- Receive notifications

The organization employs around 120 people in IT.

Current Ways of Working

Team Structure

- Development Team
  - Developers and testers work together in one team
  - Testers focus mainly on system and regression testing
- Operations Team
  - Separate team responsible for infrastructure, deployments, and production monitoring
- Security Team

  - Performs security reviews late in the release cycle Developers and testers rarely interact directly with operations. Most communication happens via tickets.

Development and Testing

- Development is done in two-week iterations
- Developers use version control and perform code reviews via pull requests
- Unit tests are written by developers, but coverage is inconsistent
- Testers execute automated regression tests nightly and manual exploratory tests before release
- Non-functional testing (performance, reliability) is performed only before major releases

Deployment and Release

- A CI/CD pipeline builds the application and runs unit tests automatically
- Deployments to production are performed manually by the operations team once per month
- Failed deployments require manual rollback and often take several hours to recover
- Production incidents are investigated by operations first, with developers involved later if needed

Monitoring and Feedback

- Basic monitoring exists for availability and server health
- Customer-reported issues are the main source of feedback
- There are no clearly defined service level objectives (SLOs)
- Metrics such as deployment frequency and lead time are not actively tracked

Culture and Collaboration

- Developers are encouraged to “deliver features quickly”
- Operations prioritize stability and avoiding production changes
- Post-incident reviews exist, but they often focus on “what went wrong” rather than learning
- Knowledge about production issues is mostly kept within the operations team

Trainer guidance and debrief points in the example case study

During the debrief, guide discussion toward:

Wall of Confusion

- Separation of Dev and Ops
- Conflicting goals: speed vs. stability

CALMS

- Culture: limited shared responsibility
- Automation: CI exists, CD is limited
- Measurement: lack of DORA metrics and SLOs

- Sharing: knowledge silos around production

Three Ways of DevOps

- Flow: slowed by manual deployments
- Feedback: delayed and customer-driven
- Continuous learning: limited by blame-oriented incident reviews

QA Perspective

- Testing exists, but is not fully integrated into flow and feedback
- Non-functional testing and reliability are late activities

Emphasize that this case study will be reused and improved in later hands-on objectives.

QDO 2.1 (H2) Contribute to the Value Stream to Assist in Quality Engineering

Description

For this hands-on objective, the 60-minute H2 level exercise supports the students in learning how to create an overview of the quality engineering activities specific to DevOps for a specific case. This includes what quality assurance and testing deliverables should be defined, aimed at achieving the right quality for the three aspects of quality: products, processes and people.

Mandatory for the Exercise

Students will identify remains of a sequential model in the business case study, and suggest improvements to quality assurance and testing deliverables to align with the DevOps principles. Students can integrate test objectives for DevOps in those improvements, and apply the core principles of continuous testing and pull requests.

The students receive an incomplete quality and test policy, and a test process description that has separate testing activities at the end, e.g., without mention of static testing. They have to improve and extend these descriptions for the relevant QA and testing deliverables to make them fit for a DevOps context using the knowledge of all sections in chapter 2.1.

Expected outcomes

After completing this exercise, students should be able to:

- Describe the outline of a quality and test policy
- Describe an implementation of the principles, activities and practices related to the three aspects of quality (people, product, process)
- Describe the core elements of QA and Testing in a DevOps SDLC, including a CI/CD pipeline
- Include the core principles of continuous testing in their DevOps SDLC
- Include the process of pull requests in their DevOps SDLC

Example of the Exercise

This is an example that helps trainers to design their own exercise.

Context

The exercise builds on the case study in QDO 1.2 (H0). You are part of a DevOps team working within a financial services organization that is transitioning from a sequential way of working toward DevOps. The organization already applies some Agile practices and uses a CI/CD pipeline, but quality assurance and testing activities are not yet fully aligned with DevOps principles. Several remains of a sequential development model are still present. You have been asked to review and improve existing quality-related descriptions so they better support built-in quality, continuous testing, and collaboration across teams.

Students receive the following documents:

1. An incomplete quality and test policy
    - Focuses mainly on testing at the end of development
    - Describes roles and responsibilities in silos
    - Lacks explicit alignment with DevOps principles
    - Does not address static testing, continuous testing or pull requests
2. A Test process description
    - Describes separate test phases after development
    - Mentions automated regression testing only
    - Does not include static testing activities
    - Has limited integration with the CI/CD pipeline
    - Does not describe production feedback or monitoring

Student tasks

Task 1 – Identify gaps and sequential model remains (15 minutes)

Analyze the provided documents and identify findings in a short list:

- At least five indicators of a sequential or late-testing approach
- Missing or weak quality engineering practices related to the three aspects:
    - Products
    - Processes
    - People

Task 2 – Improve the quality and test policy (15 minutes)

Create an improved outline for the quality and test policy that fits a DevOps context.

Your improved policy should:

- Clearly support built-in quality and a test-first mindset
- Address products, processes, and people

- Define responsibilities for cross-functional DevOps teams
- Include guidance on:
  - Risk-based quality engineering activities
  - Automation across test levels
  - Continuous improvement

You do not need to write a full policy, an outline with short explanations per section is sufficient.

Task 3 – Improve the test process description (15 minutes)

Update the test process so it supports continuous testing across the SDLC.

Your improved description should include:

- Test activities integrated into:
  - Development
  - CI/CD pipeline
  - Deployment
  - Production
- A balance of:
  - Automated testing at multiple levels
  - Manual testing (including exploratory testing)
  - Static testing activities (e.g., reviews, static analysis)
  - Use of monitoring and observability data as test input

Task 4 – Integrate pull requests as a quality assurance mechanism (15 minutes)

Extend the test process by describing how pull requests (PRs) contribute to quality.

Include:

- The role of PRs in collaboration and knowledge sharing
- How PRs act as a quality gate
- The relationship between PRs and automated test execution
- How PRs support traceability and risk reduction

Trainer guidance and debrief points in the example exercise

- Enforce time limits strictly to keep the exercise within 60 minutes.
- Encourage students to think in terms of value flow, not phases
- Accept multiple valid solutions; focus on reasoning and alignment with DevOps principles
- Use discussion to highlight differences between Agile-only and DevOps-oriented QA and testing
- Emphasize that quality engineering is broader than just QA and testing

QDO 2.2 (H2) Contribute to the DevOps Loop Implementation

Description

For this hands-on objective, the 60-minute H2 level exercise supports the students in learning how to design and implement a CI/CD pipeline, what stages, and what capabilities for tooling to be included to represent a DevOps loop.

Mandatory for the Exercise

Students will design and implement a simplified CI/CD pipeline, defining stages, selecting appropriate technologies and tooling, as well as defining an implementation roadmap for the automation of the stages, and improvement metrics.

As this requires an understanding of almost all sections of the syllabus, the exercise could be done at the end of the course after all the LOs are covered.

The exercise should contain the following elements:

- Identify the different stages and steps for the flow of work items through the DevOps loop (i.e., as defined in QDO 2.2.1 - QDO 2.2.5).
- Identify deployment and release strategies to be used (i.e., as defined in QDO 2.2.3 - QDO 2.2.5).
- Create an implementation plan of what order the steps and stages would be automated (i.e., in line with the steps defined in QDO 2.2.6)
- Identify metrics to measure as the base for continuous improvement of the CI/CD pipeline.

The exercise should have two approaches combined, a gamification part and a technology oriented part:

- To design the CI/CD pipeline stages and steps, we suggest using gamification, for example a card game representing the possible steps, or a white board drawing the diagram of the CI/CD pipeline.
- To get hands-on experience on how a CI/CD pipeline is actually developed, the students should be able to modify a predefined CI/CD pipeline file, for example to rearrange the order of steps. The instructor should prepare a pre-made, working CI/CD pipeline using available technology. The instructor should demo this running pipeline, describing the major stages, and technologies, including branching strategies, IaC based environment deployment, feature toggles and observability solutions. Students should do simple modifications only.

Expected outcomes

By the end of the exercise, students will have demonstrated the ability to:

- Identify and design the flow of work across the DevOps loop (QDO 2.2.1 – QDO 2.2.5)
- Choose appropriate deployment and release strategies
- Modify CI/CD pipeline configurations safely
- Define incremental automation steps consistent with QDO 2.2.6
- Identify metrics for improving the CI/CD pipeline over time

Example of the exercise

This is an example that helps trainers to design their own exercise.

Part 1: Gamified design workshop (group activity)

Objective

Students receive the following documents:

- A card deck containing:
  - Activity cards for each DevOps loop activity (discover, plan, understand, build, commit stage tests, acceptance stage tests, deploy, smoke test, monitor, etc.)
  - Deployment/release strategy cards (blue-green, canary, dark launch, release toggles, etc.)
- Whiteboard or Miro board for pipeline drawing

Student tasks

Task 1 — Map the DevOps loop activities (10 minutes) Using the activity cards, students must collaboratively assemble the pipeline flow that covers:

- continuous discovery (discover → plan → understand)
- continuous integration (commit atage → acceptance stage)
- continuous delivery
- continuous deployment
- release on demand
- continuous monitoring Students should draw a complete “infinity loop” or pipeline diagram on a whiteboard and do the following:
- Identify QA and testing activities in each stage (e.g., ATDD/BDD examples in understand, static analysis in commit stage, acceptance tests in acceptance stage, smoke tests in CD, production testing before release).
- Place manual vs. automated steps correctly.
- Identify where quality gates must exist (coverage thresholds, code review, deployment validation, release approval, etc.).

Task 2 — Select deployment and release strategies (10 minutes)

Students draw cards (blue-green, canary, A/B testing, dark launch, feature flagging, release toggles) to select appropriate strategies and describe:

- How risk is mitigated
- What is tested in production
- How rollback is supported
- When users are exposed to the new functionality

Task 3 — Define improvement metrics (10 minutes)

Students identify 3 metrics and describe why for continuous improvement of CI/CD pipeline, e.g.,:

- Lead time of activities
- Build and test cycle time
- Deployment frequency
- Rate of failed deployments
- Mean time to recovery (MTTR)
- Test flakiness rate
- Coverage levels and risk indicators
- Frequency of manual interventions
- Observability indicators (error rates, latency, saturation)

Part 2: Technical hands-on exercise (individual or pair activity)

Objective

Students gain hands-on experience with pipeline implementation by modifying a predefined CI/CD pipeline. The focus is not on coding new pipelines but on safe, simple modifications that reinforce syllabus concepts.

Students receive the following documents: Before the course, the trainer prepares a working CI/CD pipeline, e.g. for a trivial demo application (web app, API, or microservice).

The pipeline should include at least the following:

- Commit stage (static analysis, unit tests, component test sample, vulnerability scan)
- Acceptance stage (system tests, API tests, UI tests — at least one automated example each)
- Automated environment provisioning via IaC
- Container build and push
- Automated deployment to a test environment
- Feature toggle file or environment variable
- An observability dashboard (e.g., logs, simple metrics)
- A branching strategy (e.g., trunk-based or GitHub Flow)

The trainer demonstrates how the pipeline runs from commit through deployment.

Task 1 — Modify the predefined pipeline (15 minutes)

Students receive the pipeline file (e.g., .gitlab-ci.yml, .github/workflows/main.yml, or Jenkinsfile) and must perform simple, safe modifications, e.g.,:

- Rearranging the order of steps (e.g., moving a static analysis step earlier in the commit stage)
- Adding a missing quality gate (e.g., blocking deployment unless certain tests pass)

- Adding or updating a smoke test step in the CD stage
- Adjusting the acceptance stage to run tests in parallel
- Turning a feature toggle on/off for a test deployment
- Adding a new job that logs a metric or exports an artifact
- Modifying the environment deployment logic (e.g., switching from “dev” to “test” environment)

Task 2 — Create an incremental CI/CD implementation plan (QDO 2.2.6) (15 minutes)

Students produce a short implementation roadmap following the sequence defined in the syllabus: Model the value stream & create a walking skeleton

1. Automate build and deployment
2. Automate commit stage
3. Automate acceptance stage
4. Automate release activities
5. Evolve and optimize (parallelization, reducing waste, improving test isolation, improving observability)

Students must assign:

- What to automate first
- What depends on what
- Risks, constraints, and non-functional considerations
- Metrics for each step

Trainer guidance and debrief points in the example exercise

- Enforce time limits strictly to keep the exercise within 60 minutes.
- Encourage students to think in terms of CI/CD pipeline stages.
- Accept multiple valid solutions; focus on alignment with the DevOps Loop
- Use discussion to highlight meaning of various CI/CD pipeline stages.

QDO 3.1 (H2) Describe How Automation Supports Quality Assurance in DevOps

Description

For this hands-on objective, the 60-minute H2 level exercise supports the students in learning how to implement automation to support QA in DevOps.

This exercise enables students to apply automation concepts to a known organizational context and to reason about how automation supports QA across the DevOps value stream.

Mandatory for the Exercise

The exercise should begin by listing different key metrics, test environments, test data, test basis, and testware, where automation is utilized. The concept of single source or truth should be considered. The list should be drafted into a prototype blueprint of a dashboard or equivalent view of the big picture of automation. It is mandatory to have this exercise at least as a paper exercise.

Expected outcomes

After completing this exercise, students should be able to:

- Explain how automation supports QA beyond test execution
- Apply SSOT thinking to DevOps quality information
- Describe how automation improves flow and feedback
- Visualize quality-related automation across the SDLC
- Communicate automation concepts using DevOps terminology

Example of the Exercise

This is an example that helps Trainers to design their own exercise.

Tool support can be used for this exercise, in which case the available list of topics should be predefined (and limited by number) by the instructor. An easy-to-use integrated tooling connecting different sources into e.g., a dashboard generator should be prepared.

The focus is not on test automation, but on:

- Automation of information flow
- Automation of quality visibility
- Automation supporting traceability, environments, data, and decision-making

Students create a high-level automation blueprint that visualizes how quality-related information is collected, integrated, and shared.

Exercise setup

- Students work individually or in small groups (2–4 people).
- The previously introduced case study organization is assumed as the context
- Work is primarily paper-based (flipchart, whiteboard, or worksheet)
- Optional: an instructor-prepared lightweight dashboard or visualization tool may be used

Student tasks

Task 1 – Identify quality-relevant automation areas (15 minutes)

Based on the case study context, list existing or missing automation opportunities related to quality assurance in the DevOps process.

At minimum, include items from the following categories:

- Key metrics

- Delivery and quality indicators
- Test environments
- Provisioning, consistency, availability
- Test data
- Creation, refresh, anonymization
- Test basis
- Requirements, user stories, acceptance criteria
- Testware
- Test cases, scripts, results, reports

The list should focus on what information exists or should exist, not on specific tools yet.

Task 2 – Apply the single source of truth (SSOT) concept (15 minutes)

For each listed item, determine:

- Where this information should be stored
- Whether it currently exists in multiple places
- How automation could ensure:
- One authoritative source
- Controlled change management
- Traceability to related work products

Document which elements would benefit most from being part of an SSOT architecture.

Task 3 – Select two automation focus areas (15 minutes)

Select two elements from your list (for example):

- Quality reporting
- Traceability
- Test environment management
- Test data management
- Metrics collection

For each selected element, describe:

- What is automated
- What inputs are used
- What outputs are produced
- How this automation supports:
- Faster feedback

- Better decision-making
- Improved quality assurance

Task 4 – Create an automation blueprint (big picture view) (15 minutes)

Create a prototype blueprint that visualizes the automation landscape. The blueprint should show:

- Information sources (e.g., backlog, CI/CD pipeline stages, monitoring)
- Automation flows between sources
- A consolidated view (e.g., dashboard or report)
- How stakeholders access quality-related information

This can be:

- A hand-drawn diagram, or
- A simple digital mock-up using instructor-provided tooling

Do not aim for technical detail; clarity and coverage are key.

Trainer guidance and debrief points in the example exercise

Enforce time limits strictly to keep the exercise within 60 minutes.

During the debrief, guide discussion toward:

Automation and Quality Assurance

- How automation reduces manual handovers
- How automation enables earlier and more reliable feedback
- How QA benefits from consistent, traceable information

Single Source of Truth

- Typical risks of fragmented information
- Benefits for auditability and compliance
- Impact on collaboration across roles

Quality reporting

- Difference between raw metrics and actionable insights
- Value of integrating manual and automated results
- Stakeholder-specific views of quality

Alignment with DevOps principles

- Flow: reduced delays caused by manual data handling
- Feedback: faster visibility of quality signals
- Continuous learning: data-driven retrospectives

Encourage students to compare different blueprints and discuss trade-offs.

QDO 3.2 (H2) Test Automation in DevOps Teams

Description

For this hands-on objective, the 60-minute H2 level exercise supports the students in learning how to define the extent and role of test automation in a DevOps context.

The focus is not on tools or implementation, but on strategic decisions about:

- What should be automated
- Where automation is essential, useful, or optional
- What should not be automated and why

Mandatory for the Exercise

The exercise should:

- Build directly on the case study introduced in QDO 1.2 (H0)
- Build on the quality and automation landscape created in QDO 3.1 (H2)
- Be performed in teams of about 4 students
- Start from a partially predefined
  - System architecture
  - CI/CD pipeline stages
  - Test environments
  - List of test levels and test types
- Require students to:
  - Classify automation needs
  - Make explicit trade-off decisions
  - Produce a test automation strategy in DevOps

The result will be used as input for:

- QDO 3.3 (H2) (manual and exploratory testing focus)
- QDO 4.1 (H2) (tool selection)

Expected outcomes

After completing this exercise, students should be able to:

- Distinguish between different roles of test automation in a DevOps pipeline
- Decide where automation is:
  - Essential
  - Useful
  - Optional

643 – Not appropriate
644 • Explain why some tests should not be automated
645 • Describe a layered and risk-based regression strategy
646 • Communicate a test automation strategy using DevOps and testing terminology

647 **Example of the Exercise**

648 This is an example that helps Trainers to design their own exercise. The exercise is primarily paper-based
649 (flipchart, whiteboard, or worksheet).

650 The instructor reuses:

651 • The BlueBank Online Services case study
652 • The high-level automation landscape produced in QDO 3.1 (H2)

653 **Exercise setup**

654 Students work in teams of 3–5 people. Each team receives the following documents:

655 A simplified system architecture diagram, including:

656 • Web frontend
657 • Mobile app
658 • Backend services (e.g. Order, Payment, Customer, Inventory)
659 • External payment provider
660 • APIs between components

661 A simplified CI/CD pipeline diagram with stages such as:

662 • Code commit / pre-merge
663 • Build
664 • Component tests
665 • Integration tests
666 • System tests
667 • Release
668 • Production monitoring

669 A list of available environments, for example:

670 • Developer environment
671 • CI test environment
672 • Integration test environment
673 • Production

674 The instructor also provides a predefined list of:

- Test levels (e.g. unit, component, integration, system, end-to-end)
- Test types (e.g. functional, regression, performance, security, contract)

Student tasks

Task 1 – Identify automation candidates (15 minutes)

Based on:

- The case study
- The architecture
- The pipeline
- The results from QDO 3.1 (H2)

List and mark which test levels and test types apply to which parts of the system:

- Frontend
- Individual services
- APIs between services
- External dependencies

This creates the test scope map for the system.

Task 2 – Classify the role of automation (15 minutes)

For each relevant test activity, decide whether automation is:

- Essential
- Useful
- Optional
- Not automated

And document why.

Decisions should be based on:

- Risk
- Feedback speed
- Cost and maintenance effort
- Frequency of execution
- Business impact of failures

Task 3 – Define regression strategy layers (10 minutes)

Based on the above classification, define:

- Which automated tests should run:
  - Pre-merge

- Post-merge
- Nightly
- Before release only

- Which test suites are:
  - Fast and focused
  - Broader but slower
  - Full regression

Explicitly discuss:

- Where API and contract tests are used
- How regression testing is split into multiple suites with different purposes

Task 4 – Identify non-automated areas (10 minutes)

Explicitly list:

- Which test activities are not automated
- Why they are not automated
- Which of these will require manual or exploratory testing

Mark these clearly. These areas will be used as input for QDO 3.3 (H2)

Task 5 – Create the test automation strategy (10 minutes)

Create a test automation strategy in DevOps as:

- A diagram, or
- A structured table or short document

It must show:

- Test levels and test types
- Placement in the CI/CD pipeline
- Degree of automation (essential / useful / optional / not automated)
- Regression layers and execution frequency
- Clear distinction between automated and non-automated areas

Trainer guidance and debrief points in the example exercise

Enforce time limits strictly to keep the exercise within 60 minutes.

During the debrief, guide discussion toward:

Strategic use of automation

- Why “automate everything” is not a good strategy
- Cost of maintenance vs. value of feedback

- The difference between:
  - Automation for fast feedback
  - Automation for release confidence

Regression in DevOps

- Why multiple regression suites exist
- How risk influences test selection
- Why full regression is not run on every commit

API and contract testing

- Why these are often essential in DevOps
- How they reduce the need for expensive end-to-end tests

Connection to other exercises

- Show how:
  - This strategy builds on the automation landscape from QDO 3.1 (H2)
  - The non-automated areas feed directly into QDO 3.3 (H2)
  - The automation needs will be used in QDO 4.1 (H2) for tool selection

Encourage teams to compare strategies and discuss trade-offs not “right answers”.

QDO 3.3 (H2) Implement Manual Testing in DevOps Teams

Description

For this hands-on objective, the 60-minute H2 level exercise supports the students in learning how to identify, implement and debrief manual test approach in DevOps, e.g., with exploratory testing, crowd testing, or quality hunting, to complement automated testing.

Mandatory for the Exercise

This exercise should be aligned with previous exercises and the QDO 1.2 (H0) case study used throughout. The group of students is divided into same small teams of students as in QDO 3.2 (H2) exercise. Each team should first review the results of the previous exercise QDO 3.2 (H2) to identify which areas of testing in DevOps were not using test automation. Then they should prioritize those areas and decide which of the manual testing approaches they want to utilise (e.g., exploratory testing, crowd testing or quality hunting). Finally, they should pick one of these areas and simulate a testing session of their choice, using the execution logs provided by the instructor. For the definition of exploratory testing the students can refer to [istqb:ctfl].

Expected outcomes

After completing this exercise, students should be able to:

- Identify areas requiring manual testing to complement automated testing
- Select the manual test approach to be used (e.g., exploratory testing, crowd testing or quality hunting)

- Create a charter or plan for exploratory testing, crowd testing or quality hunting
- Understand log of the tests that were executed
- Create the relevant debriefing information that their stakeholders can use to get a view on product quality

Example of the Exercise

Context

You are working in a DevOps team that already applies automated testing as part of its CI/CD pipeline. From previous exercises, the team has identified areas where test automation is limited or not feasible, especially regarding usability, user experience, and exploratory quality assessment.

The organization wants to strengthen its quality engineering approach by deliberately integrating manual testing into the DevOps workflow.

Input materials (provided to students)

- Results from the previous hands-on exercise (QDO 3.2 (H2)), including:
  - Automated test coverage
  - Identified quality risks
  - Gaps in test automation
- A simple template for:
  - Test charters
  - Debriefing notes
- Sample test logs and notes to simulate execution of test sessions

Students tasks

The exercise is executed in teams of approximately four students.

Task 1 – Identify areas requiring manual testing (15 minutes)

Review the outcomes of the previous hands-on exercise (QDO 3.2 (H2)). As a team:

- Identify areas of the system where:
  - Automated testing is missing, insufficient, or impractical
  - Human judgment is needed to assess quality
- Create a short list of candidate areas for manual testing
- Prioritize these areas based on:
  - Risk
  - Business impact
  - Stakeholder confidence needs

Task 2 – Select a manual testing approach (15 minutes)

For the highest-priority area, decide which manual testing approach you will use:

- Exploratory testing
- Crowd testing
- Quality hunting

Document why this approach is most suitable for the selected area.

Task 3 – Create a test charter or plan (15 minutes)

Create a test charter or plan for the chosen manual testing approach. The charter or plan should include:

- Test mission (what you want to learn about the system)
- Scope and focus areas
- Risks or quality characteristics to explore
- Test environment and constraints
- Time-box for the session

Task 4 – Debrief and report quality information (15 minutes)

Create debriefing information for stakeholders. (Trainer will provide the following to simulate an actual test execution: actions performed, expected outcomes, actual outcomes, observations, questions, or concerns.)

Debriefing and report should include:

- Summary of what was tested
- Key findings related to product quality
- Identified risks and uncertainties
- Areas requiring further testing
- Overall confidence level in the tested area

Focus on quality information, not just defects.

Trainer guidance and debrief points in the example exercise

- Enforce time limits strictly to keep the exercise within 60 minutes.
- Emphasize that manual testing is about learning and insight, not just defect detection
- Encourage teams to think in terms of value and confidence
- Encourage different teams to choose different manual testing approaches to compare their experiences afterwards
- Highlight how manual testing integrates with CI/CD and complements automation

QDO 4.1 (H2) Tools and Practices in DevOps

Description

For this hands-on objective, the 60-minute H2 level exercise supports the students in choosing different tools for the CI/CD pipeline including both test automation and other automation tools.

Mandatory for the Exercise

The group of students is divided into same teams of about 4 people as in QDO 2.2 (H2) and QDO 3.2 (H2) exercises. The teams should look at a source of CI/CD pipeline tools available in the internet, e.g., the "periodic-table" (https://digital.ai/learn/devsecops-periodic-table/). They should align with their outcome of the exercise QDO 2.2 (H2) where they designed a CI/CD pipeline, and QDO 3.2 (H2) about where they defined a need of test automation. The teams should select 5-10 tools that could work together as a CI/CD pipeline based on the descriptions of those tools, and the capabilities described in Chapter 4. The exercise is a paper exercise, i.e., the team doesn't need to install the tools nor implement the integration of those tools.

Expected outcomes

After completing this exercise, students should be able to:

- Identify essential tool needs in a CI/CD pipeline
- Map those needs to DevOps and testing tool capability categories
- Select a small, coherent set of interoperable tools supporting quality in DevOps

Example of the Exercise

This is an example that helps trainers to design their own exercise.

Exercise setup

- Group exercise of approximately 4 students (paper exercise).
- Same teams as in QDO 2.2 (H2) and QDO 3.2 (H2).
- Total time allocation: 60 minutes

Students must use:

- The provided BlueBank case study.
- Their CI/CD pipeline design from QDO 2.2 (H2).
- Their test automation needs identified in QDO 3.2 (H2).
- No additional assumptions should be introduced.

Student tasks

Students act as a DevOps team supporting BlueBank.

The goal of the exercise is to select a minimum viable set of DevOps and testing tools that improves automation, quality, and feedback in the CI/CD pipeline.

This is not a detailed tool comparison exercise and not a technical implementation task.

Task 1 – Identify key tool needs (10 minutes)

Based on the case study and previous exercises, each team identifies up to five key tool needs in the CI/CD pipeline.

The needs should focus on areas such as:

- Build and continuous integration
- Test automation
- Security and code quality feedback
- Deployment reliability
- Monitoring and feedback from production

Each need should be documented as one short, clear statement.

Task 2 – Select tools and map capabilities (30 minutes)

Using the capability categories described in section 4.1 and a public overview of DevOps tools (for example, the DevSecOps Periodic Table):

- Select one tool for each identified need
- Limit the total number of selected tools to five
- For each selected tool, document:
  - Which capability category it supports
  - Its primary role in the CI/CD pipeline
  - Why it is suitable for BlueBank's context

This is a paper exercise. Students do not need to install tools or design technical integrations.

Task 3 – Interoperability considerations (10 minutes)

Within the team, briefly discuss:

- Which tool would act as the main pipeline orchestrator
- How quality-related information (for example test results or metrics) would flow between tools
- One risk related to poor tool interoperability

Task 4 – Short presentation and debrief (10 minutes)

Each team presents:

- Their identified tool needs
- Their selected five tools
- One key interoperability or orchestration consideration

Trainer guidance and debrief points in the example exercise

- Enforce time limits strictly to keep the exercise within 60 minutes.
- Focus assessment on reasoning and alignment, not on tool brand knowledge

- Look for the following:
  - Clear linkage between needs, capabilities, and tools
  - Awareness of quality risks, especially security and reliability
  - Understanding of feedback loops in DevOps

QDO 4.2.3 (H0) Summarize the Concepts of Infrastructure as Code

Description

For this hands-on objective, the 15-minute H0 level exercise supports the students in understanding the Infrastructure as Code concept.

Mandatory for the Exercise

The exercise can include a live demo or a recorded video, followed by a class discussion.

Present an Infrastructure as Code example (trainer can choose to show the difference between declarative vs. imperative approaches):

- Using the declarative approach for writing a script, ensures that the server is running with specific configurations without detailing the steps to achieve it.
- Conversely, an imperative approach involves scripting each step, such as installing software, configuring settings, and starting the server.

Expected outcomes

After completing this exercise, students should be able to:

- Summarize the concept of Infrastructure as Code
- Explain declarative vs. imperative IaC
- Describe how IaC supports:
  - DevOps workflows
  - Continuous delivery
  - Infrastructure quality and reliability

Example of the Exercise

The trainer explains the following scenario: BlueBank's operations team manually configures application servers for production. Each environment (test, staging, production) is set up slightly differently, leading to:

- Configuration drift
- "Works on my machine" problems
- Long recovery times when changes fail

Trainer demonstrates one of the IaC approaches.

Example A: Declarative approach (desired state)

The trainer presents a declarative IaC example, such as:

- Ensure that:
  - Three application servers exist
  - The web service is running
  - The correct version of the application is deployed”

Key characteristics to highlight:

- Focus on what the infrastructure should look like
- No detailed steps
- Re-running the code produces the same result (idempotency)

Example B: Imperative approach (step-by-step)

The trainer presents an imperative IaC example, such as:

- Step 1: Create a server
- Step 2: Install the operating system
- Step 3: Install the web server
- Step 4: Configure the application
- Step 5: Start the service

Key characteristics to highlight:

- Focus on how to achieve the result
- Explicit instructions
- More procedural and sequential

Trainer guidance and debrief points in the example exercise

1. Which approach (declarative or imperative) better supports:
   - Consistency across environments?
   - Automated quality checks?
2. How could version-controlling these IaC scripts help BlueBank?
3. How does idempotency reduce configuration drift?
4. How could IaC improve collaboration between development, testing and operations

5.9 Evaluation of Trainers

The knowledge and experience of the Trainers will be looked at to ensure they hold the necessary skills to deliver training. The Trainers will need to hold the same qualification they are intending to teach.

6 Benefits of Accreditation

In addition to being listed on websites and use of ISTQB® logos, Accredited Training Providers will have access to ISTQB® initiatives as they are public available, such as Beta versions of new certifications, new revisions of existing syllabi, as well as other specific benefits that may be available from time to time.

Any questions about the accreditation process can be directed to the Member Board of your country or via email to *info@ISTQB.org*.