Certified Tester
Finance Testing (CT-FT)
Syllabus

v1.0

International Software Testing Qualifications Board

Copyright Notice

Revision History

Version	Date	Remarks
v1.0	2025/09/30	Alpha review
v1.0	2026/02/01	Beta review
v1.0	2026/03/13	Release Version

Table of Contents

Acknowledgements

This document was formally released by the General Assembly of the ISTQB® on 27. May 2026

It was produced by a team from the International Software Testing Qualifications Board: Florian Fieber (chair), Richard Seidl (vice chair), Nishan Portoyan (product owner), Adam Roman, Ilia Kulakov, Julija Jankeviciene, Mantas Aniulis, Patricia McQuaid, Pavel Sharikov, Raunak Maskay, Renzo Cerquozzi, Webstar Ongaki, Mitko Mitev, Baris Sarialioglu.

The team thanks the Member Boards for their suggestions and input.

The following persons participated in the reviewing, commenting and balloting of this syllabus: Adam Roman (CaSQB), Anna Petrosyan (ArmSTQB), Amanda Logue (CSTB), Andrew Pollner (ASTQB), Arda Ender Torçuk (BNTQB), Bíró Ádám (HTB), Claude Zhang (CSTQB), Dietmar Gehring (LTB), Erhardt Wunderlich (GTB), Ferdinand Gramsamer (STB), Florian Fieber (GTB), Gary Mogyorodi (Glossary WG), Gergely Agnecz (HTB), Grzegorz Holak (PTB), Judy McKay (ASTQB), Laura Albert (HTB), Liang Ren (CSTQB), Lucjan Stapp (CaSQB), Mantas Aniulis (LTSTQB), Mattijs Kemmink (BNTQB), Meile Posthuma (BNTQB), Michael Humm (GTB), Mohammed Kamel Abu Thawabeh (JOSTQB), Péter Sótér (HTB), Ralf Pichler (STB), Renzo Cerquozzi (STB), Shun Tsunoda (JSTQB), Stephan Christmann (STB), Tal Pe'er (Exam WG), Tanja Tremmel (GTB), Taz Daughtrey (ASTQB), Vinicius Pazutti Correia (DSTB), Walter Eder (ATB)

0 Introduction

0.1 Purpose of this Syllabus

This syllabus forms the basis for the *ISTQB® Certified Tester – Finance Testing* (CT-FT) qualification. It is intended for individuals who work in or with financial systems and are involved in software testing and quality assurance within this domain. The ISTQB® provides this syllabus to:

- Member boards, to translate into their local language and to accredit training providers. Member boards may adapt the syllabus to their particular language needs and modify the references to adapt to their local publications.
- Certification bodies, to derive examination questions in their local language adapted to the learning objectives for this syllabus.
- Training providers, to produce training materials and determine appropriate teaching methods.
- Certification candidates, to prepare for the certification exam (either as part of a training course or independently).
- The international software and systems engineering community, to advance software and systems testing profession and as a basis for books and articles.

0.2 The ISTQB® Certified Tester - Finance Testing in Software Testing

The ISTQB® Certified Tester – Finance Testing (CT-FT) qualification targets professionals involved in software testing or quality assurance within financial institutions, financial technology (fintech) companies, or regulatory bodies. It is suitable for example for testers, business analysts, developers, compliance officers, auditors and regulators who want to understand the specific testing and compliance requirements of financial systems.

The certification provides a domain-focused extension to the Foundation Level (CTFL) qualification, enhancing the candidate's understanding of regulatory, risk, data, security, automation and performance topics specific to the finance industry.

0.3 Career Path for Testers

The ISTQB® scheme supports testing professionals at all stages of their careers offering both breadth and depth of knowledge. Individuals who achieve the ISTQB® Certified Tester - Finance Testing certification may also be interested in the Core Advanced Levels (Test Analyst, Technical Test Analyst and Test Management, Test Automation Engineer) and thereafter Expert Level (Test Management or Improving the Test Process). The Specialist stream offers a deep dive into areas that have specific test approaches and test activities e.g. in Agile Testing, AI Testing, or Mobile App Testing, or which cluster testing know-how for certain industry domains e.g. Gambling or Gaming. Please visit www.istqb.org for the latest information on the ISTQB® Certified Tester scheme.

0.4 Business Outcomes

This section lists the Business Outcomes expected of a candidate who has achieved the Certified Tester - Finance Testing certification.

A Certified Tester in Finance Testing can...

BO1	understand the structure of the financial industry, its institutions, systems and processes and explain the specific challenges of testing within this domain.
BO2	apply test practices that support compliance with financial regulations and standards, determining auditability, traceability and documentation requirements are met.
BO3	apply risk-based testing to financial systems, prioritizing test activities according to business, technical and regulatory risks.
BO4	understand test practices that determine data accuracy, consistency and protection across financial systems, including the use of masking and anonymization techniques.
BO5	apply functional and non-functional test techniques typically used in the financial domain, with particular emphasis on non-functional aspects such as performance, reliability and security.
BO6	understand the role of test automation in financial environments and apply test automation strategies effectively in both legacy and modern systems, while considering regulatory constraints.

(See full mapping in Appendix B).

0.5 Learning Objectives and Cognitive Level of Knowledge

Learning objectives support the business outcomes and are used to create the Certified Tester - Finance Testing exams.

The exam questions will confirm knowledge of keywords at K1 level (see below) or learning objectives at the respective level of knowledge.

The specific learning objectives and their levels of knowledge are shown at the beginning of each chapter and classified as follows:

- K1: Remember
- K2: Understand
- K3: Apply

Further details and examples of learning objectives are given in Appendix A.

For all terms listed as keywords just below chapter headings, the correct name and definition from the ISTQB® glossary shall be remembered (K1), even if not explicitly mentioned in the learning objective.

0.6 The Certified Tester - Finance Testing Certificate Exam

The Certified Tester - Finance Testing Certificate exam will be based on this syllabus. Answers to exam questions may require the use of material based on more than one section of this syllabus. All sections of the syllabus are examinable, except for the Introduction and Appendices. Standards and books are included as references, but their content is not examinable, beyond what is summarized in the syllabus itself from such standards and books.

Refer to Certified Tester - Finance Testing Exam Structures and Rules V1.0 document for further details.

To be eligible for the Finance Testing (CT-FT) exam, candidates must hold a valid ISTQB® Certified Tester Foundation Level (CTFL) certificate.

Familiarity with financial systems or software projects in the financial domain is beneficial but not mandatory.

0.7 Accreditation

An ISTQB® Member Board may accredit training providers whose course material follows this syllabus. Training providers should obtain accreditation guidelines from the Member Board or body that performs the accreditation. An accredited course is recognized as conforming to this syllabus and is allowed to have an ISTQB® exam as part of the course.

The accreditation guidelines for this syllabus follow the general Accreditation Guidelines published by the Processes Management and Compliance Working Group.

0.8 Handling of Standards

International standardization organizations such as IEEE and ISO have issued standards associated with quality characteristics and software testing. Such standards are referenced in this syllabus. The purpose of these references is to provide a framework (as in the references to ISO 25010 regarding quality characteristics) or to provide a source of additional information if desired by the reader. Please note that ISTQB® syllabi are using the standards documents as reference. Standards documents are not intended for examination.

The syllabus references international financial regulations, domain-specific standards and software testing best practices. Key sources include:

- European Union's Revised Payment Services Directive 2 (PSD2), PCI DSS, Basel III / IV and other European guidelines and regulatory frameworks
- ISO 20022 Financial services – Universal financial industry message scheme
- ISO 25010 – Product quality model (SQuaRE)
- Open Worldwide Application Security Project (OWASP), ISO 27001 for security
- ISTQB® syllabi

A full list of references can be found in Chapter 9.

0.9 Level of Detail

The level of detail in this syllabus allows internationally consistent courses and exams. To achieve this goal, the syllabus consists of:

- General instructional objectives describing the intention of the Finance Testing Level
- A list of terms that students must be able to recall
- Learning objectives for each knowledge area, describing the cognitive learning outcome to be achieved
- A description of the key concepts, including references to sources such as accepted literature or standards

The syllabus content does not describe the entire knowledge area of software testing; it reflects the level of detail to be covered in the Certified Tester Finance Testing training courses. It focuses on test approaches and test techniques that can apply to all finance-related software projects, including those following Agile software development. This syllabus does not contain any specific learning objectives related to Agile testing, but it does discuss how these test approaches and test techniques apply in Agile software development projects and other types of projects.

The syllabus uses the terminology (i.e. the name and meaning) of the terms used in software testing and quality assurance according to the ISTQB® Glossary.

0.10 How this Syllabus is Organized

There are six chapters with examinable content. The top-level heading for each chapter specifies the time for the chapter; timing is not provided below the chapter level. *For accredited training courses, the syllabus requires a minimum of 8.75* hours of instruction, distributed across the six chapters as follows:

- **Chapter 1: 50 minutes – Introduction to the Financial Services Industry**
  - Recall the primary objectives of finance testing
  - Summarize the unique characteristics and technical requirements of various financial test environments
  - Identify key financial systems and their associated testable workflows.
  - Explain how domain knowledge is applied to mitigate common testing challenges in the financial sector
- **Chapter 2: 80 minutes – Compliance Testing**
  - Summarize the objectives of compliance testing and the relationship between its primary test approaches
  - Explain the testing relevance of key financial regulations and standards
  - Recall consequences of non-compliance to financial regulations
  - Summarize the unique characteristics of finance testing
  - Understand the principles and implementation mechanisms of Continuous Automated
  - Compliance Validation (CACV) within the software delivery lifecycle
- **Chapter 3: 45 minutes – Risk-Based Testing**
  - Explain the main categories of risk in financial systems
  - Explain the principles of risk-based testing in finance
  - Explain how risk-based prioritization affects test activities in financial applications
- **Chapter 4: 90 minutes – Data Testing and Data Management**
  - Explain the importance of data accuracy and consistency across financial systems
  - Apply data reconciliation techniques to system of systems financial processes
  - Explain the importance of data privacy and data protection in finance testing

- **Chapter 5: 165 minutes – Functional and Non-Functional Testing**
  - Apply functional testing to validate financial systems
  - Explain the performance risks in financial systems during peak periods
  - Apply load testing and stress testing in finance-specific contexts
  - Explain security testing and availability testing needs in financial systems
  - Explain how GenAI can support and improve finance testing
- **Chapter 6: 105 minutes – Test Automation**
  - Role of Test Automation in Finance Testing
  - Explain how test automation supports test efficiency in finance
  - Test Automation Strategies
  - Apply test automation strategies in legacy and modern financial IT landscapes
  - Challenges in Regulated Test Automation
  - Explain the challenges of test automation in regulated environments
  - Challenges in Different Types of Test Automation
  - Explain the challenges of test automation on mobile, frontend and backend systems

1 Introduction to the Financial Services Industry – 50 minutes

Keywords

Domain-Specific Keywords

financial institution, regulatory standard

Learning Objectives for Chapter 1:

1.1 Fundamentals of Software Testing in Finance Testing

FT-1.1.1 (K1) Recall the primary objectives of finance testing

FT-1.1.2 (K2) Summarize the unique characteristics and technical requirements of various financial test environments

FT-1.1.3 (K2) Identify key financial systems and typical testable workflows associated with them

1.2 Challenges and the Role of Domain Knowledge in Finance Testing

FT-1.2.1 (K2) Explain how domain knowledge is applied to mitigate common testing challenges in the financial sector

Introduction to the Financial Services Industry

The financial services industry acts as the primary engine of global commerce, moving beyond simple money management to encompass a sophisticated ecosystem of banking, insurance, and investment. In this high-stakes environment, software systems are not just tools but the actual infrastructure upon which economic stability and customer trust depend. As digital transformation accelerates, the software powering a single institution has become inseparable from the stability of the global economy. This convergence makes rigorous software testing the only real safeguard against a domino effect of systemic failure.

1.1 Fundamentals of Software Testing in Financial Services

1.1.1 Definition and Objectives of Finance Testing

Finance testing is the process of verifying and validating software applications utilized within the financial services sector, including banking, financial technology (fintech), non-banking financial institutions (NBFIs)—such as insurance, pension funds, and leasing companies – and other specialized financial services. This domain-specific testing evaluates whether financial systems function correctly, securely, and in accordance with business logic and regulatory requirements. It involves validating the entire ecosystem required to execute financial services, specifically the software-driven workflows that support internal processes, external user interfaces, and third-party integrations.

The primary objectives of finance testing include:

- Functional correctness – verifying that financial transactions, complex mathematical calculations (e.g., interest rates, amortization), and account balances are processed with defined precision rules and tolerance levels to prevent financial loss.
- Regulatory compliance – verifying the software complies with required global and regional legal standards and to mitigate risk of legal liability and financial penalties.
- Data integrity – confirming that transaction records and financial data remain consistent, accurate and uncorrupted across databases and integrated systems.
- Security assurance – validating that robust security controls are in place to protect sensitive customer data and financial assets from cyber threats, fraud and unauthorized access.
- Operational resilience and performance – assessing system stability and performance under high load by validating throughput and response times, and verifying failover mechanisms to maintain business continuity.
- Customer trust – building confidence that the product operates reliably, protects users' interests, and preserves the institution's reputation to support long-term customer retention.

1.1.2 Types of Financial Test Environments in Financial Institutions

Testing in the financial sector can occur in several types of technical environments depending on system architecture and business domain. The following examples illustrate common categories rather than an exhaustive list:

Omnichannel consumer environment (retail and commercial banking). This environment prioritizes a unified and seamless customer journey across integrated digital and physical touchpoints, including mobile applications, web portals, branches, automated teller machines (ATMs) and Interactive Voice

Response systems. Test objectives focus on evaluating consistency in brand experience, data synchronization across channels, and interoperability between front-end interfaces and legacy core banking systems.

High-throughput and low-latency environment (investment and trading). Typical of stock exchange engines and algorithmic trading, these environments must process millions of transactions per second with sub-millisecond response times. Performance testing is the primary concern, focusing on throughput, minimal latency, and execution determinism to support handling massive volumes with consistent speed.

Logic-dense and resilient API environment (fintech).

Fintech systems utilize microservice architecture to manage complex business logic through high-dimensional inputs. Testing focuses on operational resilience, ensuring failures are isolated to prevent cascading impacts on dependent services. Validation requires simulating intricate transaction workflows, which can be achieved through model-based testing, service virtualization, or AI-driven test orchestration.

Schema-driven messaging environment (specialized inter-bank systems). This environment facilitates inter-bank communication using standards such as ISO 20022 and protocols such as FIX (Financial Information Exchange). Test activities emphasize schema validation for message types including PAIN (payment initiation), PACS (credit transfers) and CAMT (cash management) reports.

Time-dependent and actuarial environment (insurance and mortgages). Focused on the long-term lifecycle of policies and loans, these environments validate complex mathematical models for premium ratings, risk assessment, and compound interest amortizations. Key test concerns include precision in multi-variable calculations and **time travel testing** — a testing technique in which the system clock or simulated time is intentionally modified to verify the correct behavior of time-dependent functionality.

1.1.3 Financial System Architectures and Testable Workflows

The architecture of a financial institution is typically composed of multiple interconnected layers, each introducing distinct testing concerns:

- **Core Banking Systems** act as the centralized backend for processing financial transactions, maintaining customer records and managing accounts. Testing these systems requires addressing specific architectural complexities, transactional dependencies and critical non-functional requirements.
- **Integration and middleware layer** functions as the interconnecting infrastructure that links front-end interfaces. Testing focuses on verifying the reliable exchange of financial data, maintaining data integrity across subsystems and isolation of failures to prevent cascading impact on dependent services.
- **Digital front-end and channels** include mobile applications, web portals, and ATM or point of sale (POS) terminals. Coverage addresses cross device behavior, biometric authentication, offline stability, and correct transaction routing, with device independent UI architectures requiring consistent rendering across all target platforms.

Testable workflows in finance include:

- **Transactional workflows.** Testing financial transactions focuses on verifying the complete lifecycle of a payment instruction across multiple subsystems. The scope includes correct state transitions, settlement logic, and rollback behavior under both normal processing and failure conditions.

- **Compliance and identity workflows**. Customer onboarding integrity is maintained through verification of identity proofing mechanisms, including document authentication and biometric tests, in accordance with Know Your Customer (KYC) regulatory standards. Anti-Money Laundering (AML) workflows are assessed to confirm accurate detection of suspicious activities and effective screening of entities against global sanctions lists while controlling false positive rates.
- **Investment and treasury workflows.** Testing investment systems involves validating the end-to-end trade lifecycle – from order routing in OMS/EMS (Order Management System/Execution Management System) to final settlement – across diverse order types and global exchanges. It requires specialized verification for asset-specific logic, such as derivative pricing models and algorithmic risk "kill switches," alongside rigorous performance testing for latency and real-time risk metrics.
- **Operational and back-office workflows.** Operational workflow coverage addresses reconciliation integrity, ensuring consistent transaction data flow from front-office systems to the General Ledger and alignment with external records. Coverage includes periodic processing cycles—such as End-of-Day, End-of-Month, and End-of-Year—to validate settlement accuracy, tax reporting, and the maintenance of complete audit trails for governance and compliance.
- **Specialized insurance workflows.** Policy lifecycle coverage focuses on correct state transitions and accurate premium calculation based on underwriting and risk rules. Claims workflows are designed to facilitate systematic adjudication, trigger fraud detection protocols, and execute settlements. These processes require consistent data exchange between core insurance systems to maintain data integrity.

1.2 Challenges and the Role of Domain Knowledge in Finance Testing

In the financial services sector, generic software testing skills are often insufficient due to the complexity of the test basis, the precision required in transaction processing and the severity of regulatory constraints. Domain knowledge functions as a critical enabler for mitigating the following specific testing challenges associated with financial systems:

- Resolution of the test oracle problem. A primary challenge in finance testing is the test oracle problem, where determining the correct expected result for a given input is non-trivial due to complex underlying mathematical models (e.g., derivatives pricing, actuarial risk calculations, or yield curves). Domain expertise allows for the independent derivation of expected results to verify functional correctness.
- Translation of regulatory mandates into test conditions. Financial regulations are often abstract and open to interpretation, making it difficult to translate directly into executable test cases. Domain fluency allows testers to decompose high-level legal text into precise, testable acceptance criteria.
- Data dependencies. Financial inputs possess strict dependencies between data fields (e.g., a specific transaction type requiring a distinct tax code). Domain knowledge enables the definition of these constraints to generate valid test data that passes initial system validation logic.
- Validation of end-to-end transaction lifecycles. Financial transactions often traverse multiple heterogeneous systems, moving from front-office delivery channels through middle-office settlement engines to back-office general ledgers. Maintaining data integrity and state consistency across these systems is a primary testing challenge. Domain expertise is applied to verify the complete transaction lifecycle by determining that:

  - transactions are correctly captured and validated at the point of entry,
  - data remains consistent as it undergoes processes such as clearing, settlement, or currency conversion between different systems,
  - the final state of the transaction is accurately reflected in the accounting records, audit trails, and reporting components.
- Effective prioritization in risk-based testing. Due to time constraints and high volumes of regression tests, exhaustive testing is impossible. Generic risk assessment may fail to identify high-impact financial failure modes. Domain expertise facilitates accurate impact analysis. Testers can distinguish between cosmetic defects and critical failures that could lead to financial loss, regulatory fines, or reputational damage.

2 Compliance Testing – 80 minutes

Keywords

auditability, compliance, continuous automated compliance validation, control testing, negative testing, substantive testing, traceability

Domain-Specific Keywords

regulatory compliance, General Data Protection Regulation (GDPR), Sarbanes-Oxley Act (SOX), Digital Operational Resilience Act (DORA)

Learning Objectives for Chapter 2:

2.1 Compliance Testing Objectives and Approaches

FT-2.1.1 (K2) Summarize the objectives of compliance testing and the relationship between its primary testing approaches

2.2 Financial Regulations and Their Impact on Testing

FT-2.2.1 (K2) Explain the testing relevance of key financial regulations and standards

FT-2.2.2 (K1) Recall consequences of non-compliance to finance regulations

2.3 Distinctive Characteristics of Regulated Finance Testing

FT-2.3.1 (K2) Summarize the unique characteristics of finance testing

2.4 Auditability and Continuous Automated Compliance Validation

FT-2.4.1 (K2) Understand the fundamental principles of auditability in compliance testing

FT-2.4.2 (K2) Understand the principles and implementation mechanisms of Continuous Automated Compliance Validation (CACV) within the software delivery lifecycle

Compliance Testing

As software becomes the backbone of the global financial infrastructure, the intersection of engineering and legal oversight has become a critical operational frontier. This chapter explores the rigorous framework of compliance testing, a discipline that moves beyond traditional defect detection to focus on institutional integrity and regulatory adherence. By examining the specialized methodologies, mandatory non-functional requirements, and the evolution toward automated, continuous auditability, we define how modern systems maintain trust in an increasingly scrutinized and complex legislative environment.

2.1 Compliance Testing Objectives and Approaches

Compliance testing (also referred to as conformance or regulatory testing) is the process of testing to determine whether a component or system complies with defined standards, laws, regulations, and contracts. While it is often categorized as non-functional testing, in the financial sector it frequently involves validating functional business logic and data against specific regulatory mandates. It assesses whether activities, financial transactions, and information comply, in all material respects, with the authorities and criteria governing the audited entity.

The primary objectives of compliance testing include:

- verification of adherence – to obtain reasonable assurance that the system and its underlying transactions comply with relevant regulatory frameworks and internal policies,
- risk identification –to identify and assess the risks of material non-compliance, determining that violations of legal or regulatory mandates are detected early to mitigate potential penalties, financial loss, or reputational damage,
- control validation – to evaluate the effectiveness of internal control systems in preventing, detecting, and correcting instances of non-compliance,
- evidence provision – to generate sufficient, relevant, and reliable audit evidence that demonstrates due diligence and adherence to standards for stakeholders and regulatory bodies.

Control testing and substantive testing represent two complementary testing approaches used to achieve compliance assurance:

- Control testing evaluates whether internal controls are designed appropriately and operate effectively to prevent, detect, or correct material misstatements or instances of non-compliance. The primary objective of control testing is to obtain evidence that controls operate as intended, are applied consistently, and remain effective over the period under review. Control testing focuses on the process and mechanisms, such as approval workflows, segregation of duties, access controls, and automated validation rules, rather than on individual transaction results.
- Substantive testing consists of procedures designed to detect material misstatements or non-compliance by directly examining transactions, account balances, and outputs. The primary objective of substantive testing is to verify the accuracy, completeness, and validity of data and results produced by the system. Unlike control testing, which validates how processing is performed, substantive testing validates what the system produces, regardless of whether controls are present or effective.

2.2 Financial Regulations and Their Impact on Testing

2.2.1 Key Financial Regulations and Their Testing Relevance

The compliance landscape in finance is shaped by regulations, standards, and guidelines at different jurisdictional levels. Some are legally binding, while others guide industry best practices. Their priority depends on legal status and potential business impact. Table 1 provides a structured overview of major financial regulations and their testing relevance:

Table 1. Structured overview of major regulations relevant to financial systems and their testing relevance

Focus area	Regulation	Compliance impact on testing
Data protection and privacy	General Data Protection Regulation (GDPR)	Verify lawful processing via consent handling, data-subject rights, access controls, privacy-by-design; using masked/synthetic PII (Personally Identifiable Information) and respect retention and jurisdiction limits.
Financial transparency	Sarbanes-Oxley Act (SOX)	Validate internal controls through segregation of duties, approval workflows, immutable audit logs and preserved financial history using anonymized datasets.
Payment security	The Payment Card Industry Data Security Standard (PCI DSS)	Confirm protection of cardholder data via encryption, strong authentication and secure transaction flows, using tokenized Primary Account Numbers (PAN) and masked logs only
Operational resilience	The Digital Operational Resilience Act (DORA)	Assess information and communication technology (ICT) resilience through disaster recovery and business continuity planning (DR/BCP), failover and scenario-based testing using anonymized replicated datasets.
Payments and open banking	Payment Services Directive 2 (PSD2)	Determine that there are secure open-banking operations by testing Strong Customer Authentication (SCA), Multi-Factor Authentication (MFA), application programming interface security and consent lifecycle handling using synthetic accounts.
Bank capital and risk	Basel III/IV	Support capital adequacy compliance by validating credit, market and liquidity risk models through stress testing and scenario-based testing using large masked or synthetic portfolios.
Anti-money laundering	Fifth and Sixth EU Anti-Money Laundering Directives (AMLD5/AMLD6)	Verify anti-money laundering controls via KYC flows, sanctions screening, alert generation and risk-flagging logic using realistic synthetic customer and transaction patterns.
Crypto assets	The Markets in Crypto-Assets (MiCA) Regulation	Validate crypto-asset compliance by testing custody safeguards, order integrity and market-abuse detection using synthetic wallets and anonymized crypto transactions.

Financial regulations vary across regions and jurisdictions, requiring adaptation of compliance frameworks, control mechanisms, and reporting standards within the test process.

2.2.2 Consequences of Non-compliance

Failure to meet regulatory and compliance requirements in the financial industry can lead to significant consequences across financial, legal, reputational and operational dimensions:

- **Financial penalties and economic impact.** Regulatory bodies have the authority to impose substantial monetary fines and financial penalties for non-compliance. These fines may relate to

violations of anti-money laundering rules, data privacy laws, or misreporting under financial directives.

- **Legal and regulatory sanctions.** Non-compliance may also trigger legal enforcement actions beyond monetary penalties, such as lawsuits, criminal liability, mandatory restructuring, or the appointment of external management. In severe cases, regulators may impose enforcement measures such as mandatory asset restructuring or the appointment of external management.
- **Reputational damage and market position.** Incidents of non-compliance can significantly damage public and stakeholder trust, resulting in negative media coverage, investor concern and customer attrition.
- **Business disruption.** Compliance failures may trigger service suspensions, license restrictions, or regulatory intervention. These can lead to revenue losses, market share declines and customer loyalty erosion. In severe cases, regulators may impose enforcement measures such as mandatory asset restructuring or the appointment of external management.
- **Operational risks and technical debt.** Non-compliance introduces systemic operational risks and technical debt, undermining the stability and maintainability of software systems. The failure to implement mandated security controls exposes organizations to security vulnerabilities, including data breaches and fraud.

These risks underline the importance of integrating compliance validation into software development and testing activities.

2.3 Distinctive Characteristics of Regulated Finance Testing

Testing in regulated financial systems has distinct characteristics that significantly influence how the test process is performed and controlled:

- **Regulatory change-driven testing**. Testing is frequently triggered by external legislative mandates rather than internal business objectives. Unlike general software development, where requirements evolve based on market feedback, financial systems must adapt to a continuous stream of new and updated regulations. This dynamic regulatory landscape requires a test strategy characterized by continuous adaptation, rigorous impact analysis, and rapid responsiveness to maintain ongoing legal compliance.
- **Regulatory-aware requirement analysis** is the specialized process of identifying, interpreting, and mapping legal and compliance mandates into testable software specifications. Unlike standard requirement engineering, which focuses on business utility, this process prioritizes adherence to external regulations and internal governance policies.
- **Negative testing as a regulatory necessity**. Financial systems must actively resist misuse, failures, and non-compliant behavior. Negative testing helps identify security weaknesses, fraud paths, and logic gaps by exercising valid-looking but prohibited inputs. This verifies that the system correctly blocks unauthorized access, limit breaches, and violations of regulatory rules such as sanctions controls.
- **Compliance-critical test data management**. The handling of test data is governed by legal regulations that fundamentally constrain how testing is performed, distinguishing it from non-regulated industries, where production data might be copied for testing purposes (see Chapter 4).
- **Evidence-driven testing** is characterized by the systematic collection, retention, and management of testware to satisfy external auditors and regulatory bodies. The primary objective

shifts from detecting defects to demonstrating due diligence and validating that specific controls are executed effectively.

- **Mandatory non-functional testing** is a compulsory activity driven by statutory frameworks, establishing compliance as a built-in quality characteristic rather than a business preference. It helps mitigate security vulnerabilities that could lead to financial fraud or data breaches. Performance efficiency and reliability testing are mandated to verify system stability under high-frequency transaction loads and to confirm operational resilience through rigorous disaster recovery and business continuity validation.

2.4 Auditability and Continuous Automated Compliance Validation

2.4.1 Auditability

Auditability in the context of compliance testing involves the capability to independently reconstruct, review, and verify the test activities and the behaviors of the system under test (SUT). It encompasses not only the functional verification of the software but also the rigorous documentation of the testing process itself. Such documentation provides evidence for regulators to verify the effective execution of regulatory controls and confirm that the system operates within defined legal and contractual boundaries.

Key components of auditability include:

- **Bidirectional traceability**. Auditability requires the establishment of a verifiable lineage connecting high-level regulatory mandates to specific technical artifacts. This component enables linking each regulatory requirement to corresponding test cases (forward traceability) and tracing each defect or test result back to the regulatory rule or control objective it relates to (backward traceability).
- **Immutable evidence retention**. Compliance testing requires the systematic collection and preservation of testware. This includes test plans, signed-off requirements, detailed execution logs, and screenshots of critical steps. These artifacts must be stored in a tamper-proof format to serve as irrefutable evidence of due diligence.
- **Evidence integrity and privacy**. Systems must generate secure, timestamped audit trails that record all user activities, access events, and transaction states. These logs must be protected against unauthorized alteration to serve as valid legal evidence during forensic investigations or regulatory reviews. Crucially, when capturing this metadata, the process must incorporate data masking techniques to prevent sensitive financial information from being exposed in the audit trail, thereby supporting compliance with global privacy regulations.
- **Reproducibility**. Auditability requires that test processes and system behaviors be reproducible. The system must capture sufficient environmental configuration data and input parameters to allow an independent auditor to re-execute a test or replay a transaction and achieve the same result. This capability confirms that compliance controls operate consistently and are not subject to transient or undocumented variables.

2.4.2 Continuous Automated Compliance Validation

Continuous Automated Compliance Validation (CACV) is the practice of embedding regulatory and security controls programmatically throughout the software delivery lifecycle. Unlike traditional audit models that rely on periodic, point-in-time assessments, CACV employs test automation to provide real-time, continuous assurance of regulatory compliance. This approach transforms compliance from a reactive, manual verification activity into a proactive, built-in feature of the system architecture.

Mechanisms of implementation

CACV relies on specific technical implementations to enforce and monitor regulatory requirements:

- **Compliance as Code** (CaC). Regulatory requirements are translated into machine-readable code or scripts. This enables compliance rules to be version-controlled, tested, and automatically executed against infrastructure and application code.
- **Pipeline integration and quality gates**. Compliance checks are integrated directly into the Continuous Integration/Continuous Deployment (CI/CD) pipeline. "Compliance gates" are established to automatically block the promotion of non-compliant code to production environments.
- **Continuous Control Monitoring** (CCM): Beyond the build phase, CACV involves the continuous monitoring of the production environment to detect “drift.” Configuration drift occurs when a compliant system is altered post-deployment (e.g., a firewall port is opened manually), rendering it non-compliant.

CACV workflow integrates regulatory verification into the software delivery lifecycle through four iterative phases:

- **Elaboration and definition**. Regulatory frameworks are analysed to derive precise, testable requirements. These are grouped into compliance profiles to establish specific criteria for the software under test.
- **Development**. Compliance requirements are implemented as executable code. Functional test cases are developed and explicitly mapped to regulatory rules to establish bi-directional traceability between requirements and test work products.
- **Automated execution and inspection**. Automated compliance tests run continuously within the CI/CD pipeline. Quality gates enforce adherence by blocking the promotion of non-compliant artifacts to production environments.
- **Evaluation and reporting**. Test outcomes are assessed to generate immutable audit evidence and violation reports. Continuous monitoring detects configuration drift in production, triggering remediation workflows to restore compliance.

3 Risk-Based Testing – 45 minutes

Keywords

risk, risk mitigation, risk assessment, risk-based testing, traceability matrix

Domain-Specific Keywords

Compliance risk, dynamic risk assessment, risk acceptance, risk avoidance, risk transfer

Learning Objectives for Chapter 3:

3.1 Categories of Risk in Financial Systems

FT-3.1.1 (K2) Explain the main categories of risk in financial systems

3.2 Principles of Risk-Based Testing in Finance

FT-3.2.1 (K2) Explain the principles of risk-based testing in finance

3.3 Risk-Based Activities

FT-3.3.1 (K2) Explain how risk-based prioritization affects test activities in financial applications

Introduction to Risk-Based Testing in Financial Systems

Risk-based testing is a strategic software testing approach that prioritizes test efforts based on the identified risks associated with different features or functionalities of a software application, calculated as the product of the probability of failure and the potential business impact. Examples include failures to meet standards (e.g., GDPR, PSD2, PCI DSS, or MiCA) or defective risk calculations or financial models. Risk-based testing is covered in Section 5 of the ISTQB Foundation-Level Syllabus [ISTQB_CTFL_SYL].

3.1 Categories of Risk in Financial Systems

In finance, risks are identified across three primary dimensions:

- **Business and financial risks**. These can include risks associated with sensitive data migration, data handling defects, incorrect calculations, and incorrect application of rules.
- **Technical and operational risks**. These include scope creep, inadequate tool support, or more technical dependency risks, such as overreliance on specific tools, test automation frameworks, or APIs.
- **Regulatory and compliance risks**. There are vendor and outsourcing risks, such as third-party service providers failing to meet compliance, security, or performance obligations (especially relevant under DORA). There can be delays or misunderstandings in implementing new legal or compliance requirements. These risks apply to systems developed internally, also.

The ISTQB® Foundation Level syllabus distinguishes between project and product risks.

Project risks relate to the management and control of the project and may affect its schedule, budget, or scope, thereby impairing its ability to achieve its objectives.

Product risks are related to product quality characteristics. Product risks may include defective commercial loan calculations or noncompliance with applicable electronic funds transfer protocols. The ISO 25000 standard family defines two key quality models: the product quality model, which evaluates the software itself (ISO/IEC 25010, 2023), and the quality-in-use model, which evaluates the effect of software when used in a specific context (ISO/IEC 25019, 2023). The focus here is on the product quality model, with Table 2 illustrating the nine quality characteristics and sub-characteristics (ISO/IEC 25010).

Table 2. ISO/IEC 25010 product quality model

SOFTWARE PRODUCT QUALITY								
Functional Suitability	**Performance Efficiency**	**Compatibility**	**Interaction Capability**	**Reliability**	**Security**	**Maintainability**	**Flexibility**	**Safety**
Functional Completeness Functional Correctness Functional Appropriateness **Source: ISO2500.com**	Time Behavior Resource Utilization Capacity	Co-Existence Interoperability	Appropriateness Recognizability Learnability Operability User Error Protection User Engagement Inclusivity Self-Descriptiveness	Faultlessness Availability Fault Tolerance Recoverability	Confidentiality Integrity Non-repudiation Accountability Authenticity Resistance	Modularity Reusability Analyzability Modifiability Testability	Adaptability Scalability Installability Replaceability	Operational Constraint Risk Identification Fail Safe Hazard Warning Safe Integration

While all nine characteristics are essential, maintaining security and preserving the privacy of personal and financial information are critical in this sector. Data breaches can have significant impacts, including exposure of client information to attackers, damage to a company's reputation, substantial financial

losses, and legal penalties and lawsuits. Depending on the country, an organization may be liable to pay for credit monitoring due to increased phishing and scam attempts against individuals.

Cybersecurity is explored in detail in both the ISTQB® Security Tester [ISTQB_CT-SEC_SYL] and ISTQB® Security Test Engineer [ISTQB_CT-STE_SYL] syllabi.

3.2 Principles of Risk-Based Testing in Financial Systems

Risk-based testing prioritizes test activities based on risk level. In financial systems, where the cost of defects can be extremely high (e.g., monetary loss, regulatory penalties, reputational damage), risk-based testing is especially critical. While financial institutions differ in their structures, they share similar functions. Business processes, financial transactions, compliance requirements, and security vulnerabilities need to be analyzed. Teams aim to prioritize their tests based on risk. In addition to calculating interest on deposits and mortgages, financial institutions perform many other functions, including privacy and security.

Teams apply risk-based testing by prioritizing areas that pose the highest compliance or business risk. Once identified, each risk is assessed for risk likelihood and risk impact. A risk level value, called risk level, can be calculated (e.g., risk likelihood * risk impact) to prioritize test focus areas. Risk levels can be expressed quantitatively (e.g. a 20% chance of delay) or qualitatively (e.g., as low, medium, or high).

A risk matrix (also called a likelihood-impact matrix) is a visual risk management tool that maps identified risks on a grid, using risk likelihood (probability of occurrence) on one axis and risk impact (consequences if they occur) on the other. Risks with the highest likelihood and highest impact receive top test priority. In dynamic risk assessment, risk levels in financial systems evolve as new features, integrations, or regulations are introduced. Periodic risk reassessment verifies that test priorities remain aligned with current business and regulatory realities.

Reasons for risk-based testing include:

- **efficient resource allocation** – teams can plan for and allocate resources to areas that have the highest risk likelihood and risk impact,
- **early defect detection** – enables teams to identify critical defects early in the SDLC (shift left), providing the opportunity to repair defects in critical code during the current sprint/iteration,
- **optimized coverage** – testing can be focused on critical functionalities and high-priority areas, to determine that the most important aspects of the software are thoroughly tested,
- **improved business value** – helps deliver software that meets both business requirements and customer expectations, helping to increase customer satisfaction and revenue.

3.3 Test Activities Related to Risk-Based Testing

Both functional risks (e.g., defective interest calculations) and non-functional risks (e.g., poor performance under high load) need to be considered. To identify and prioritize the risks, stakeholders such as business analysts, compliance officers, technical team, and risk managers need to be involved.

Identification of which risks need immediate attention and which can be monitored, help teams focus resources effectively.

Prioritize test efforts. There are multiple techniques to prioritize risks, including:

- Failure Mode and Effect Analysis (FMEA) – a systematic, proactive approach used to identify, prioritize, and mitigate potential failures in a design, process, or service before they occur.
- Fault Tree Analysis (FTA) - a top-down, deductive failure analysis technique used to identify the root causes of an undesired system-level event by graphically mapping component failures using Boolean logic gates (AND, OR).
- PRISMA (Product RISk MAnagement) – an approach for identifying the areas that are most important to test, i.e., identifying the areas that have the highest level of business or technical risk (Veenendaal, 2012).

Define test conditions. Create test conditions that cover the identified high-risk areas to verify coverage aligns with risk levels and addresses specific risks, providing comprehensive coverage and mitigation of identified risks. To help determine that completeness and compliance, traceability matrices can be created by linking risks to test conditions and the test conditions to test results.

Execute and monitor tests. Execute the test suites created from the test cases, based on test conditions, closely monitoring the test results, and collecting relevant data.

Report and mitigate risks. Document and report any defects discovered during testing. Risk mitigation strategies identify, address, reduce, or manage identified risks that could negatively impact a project, business, or operation. Standard risk mitigation strategies are:

- risk avoidance – taking deliberate action to eliminate a specific risk by not engaging in the activity that could cause it,
- risk mitigation by testing – reducing the risk likelihood or risk impact of the risk by test activities,
- risk transfer – shifting the risk to a third party, such as insurance,
- risk acceptance – acknowledging and accepting the risk.

ISO 31000 (2018) provides additional guidance on integrating risk management into organizational core activities and decision-making. They are: integrated, structured and comprehensive, customized, inclusive, dynamic, use the best available information, consider human and cultural factors, and promote continual improvement.

4 Data Testing and Data Management – 90 minutes

Keywords

data accuracy, data consistency, data reconciliation, test data, test technique

Domain-Specific Keywords

audit trail, core banking system, customer trust, data privacy, data profiling, general ledger, settlement system

Learning Objectives for Chapter 4:

4.1 Data Accuracy and Consistency

FT-4.1.1 (K2) Explain the importance of data accuracy and consistency across financial systems

4.2 Data Reconciliation Across Financial Systems

FT-4.2.1 (K3) Apply data reconciliation techniques to multi-system financial processes

4.3 Data Privacy and Data Protection in Finance **Testing**

FT-4.3.1 (K2) Explain the importance of data privacy and data protection in finance testing.

Introduction to Data Testing and Data Management

In the financial sector, data consistency, accuracy and integrity are essential in making sound decisions and complying with regulations. Data testing verifies accuracy, completeness, timeliness, and lineage of financial data across its lifecycle, ensuring compliance with data governance frameworks. Financial institutions process massive volumes of transactional and analytical data, so testing is needed to verify the accuracy and reliability of data pipelines, reports, and financial models. This section covers data quality dimensions, validation rules, test types such as reconciliation, completeness, transformation logic, and regression testing. The financial data lifecycle is the focus, starting with data creation, then reporting and identifying data abnormalities that could affect risk levels, compliance, or customer experience.

4.1 Data Accuracy and Data Consistency

Data accuracy refers to how correctly data represents real financial information. Accurate data reflects the true value of transactions, balances, and customer details. It is complete, current and free from defects that could distort system behavior or outputs.

Data consistency refers to the uniform representation of data across systems, components, and databases. Consistent data appears in the same format, value, and meaning wherever it is used, ensuring that systems do not present conflicting information.

Importance in financial systems

- Financial reporting. Minor inaccuracies or inconsistencies can cascade into defective calculations, misleading summaries, and unreliable financial insights.
- Risk and decision support. Risk assessments and analytical models depend on large volumes of historical and transactional data. Inconsistent or defective inputs can weaken forecasts, distort risk profiles, and reduce confidence in analytical results.
- System integration and automation. Financial platforms typically comprise multiple interconnected components, including transaction processing, customer management, and reconciliation systems. Consistent data enables smooth information flow across interfaces, integration layers, and transformation processes, reducing manual intervention and operational friction.

Typical data accuracy and data consistency defects include the following:

- duplicate transactions appearing in statements,
- balance differences between ledger and sub-ledger systems,
- customer data variations across onboarding and identity systems,
- currency mismatches in cross-border transaction processing,
- data loss or corruption during migration or transformation activities.

Testing focus areas. Testing activities focus on determining data accuracy and data consistency as it moves, transforms, and is stored across systems. This involves comparing source and target datasets, validating reconciliation results, confirming business rule logic, and evaluating the impact of system changes on existing data flows. Understanding master and reference data definitions is essential to maintain consistent data interpretation across applications.

Below are some best practices to follow to help support data accuracy and data consistency:

- apply well-defined data validation rules and exception handling,

- use automated comparison techniques for large datasets,
- perform data profiling to assess structure, completeness, and quality,
- identify unusual patterns or anomalies at an early stage,
- maintain clear documentation for data definitions and metadata.

4.2 Data Reconciliation Across Financial Systems

Data reconciliation addresses the alignment of financial data as it travels across multiple systems within a financial ecosystem. Financial transactions typically pass through several platforms, including transaction processing systems, settlement engines, accounting platforms, and reporting layers. Reconciliation activities support confidence that each system reflects the same financial outcome, despite differences in processing logic, timing, or data transformations.

This area is particularly relevant during system changes, including core banking migrations, platform upgrades, and large-scale data transfers. Even when systems function correctly at a technical level, reconciliation gaps can lead to defective balances, missing transactions, or inconsistent reporting if data is not aligned across systems.

Data migration and reconciliation context. During data migration initiatives, financial institutions transfer large volumes of historical and active data, including accounts, balances, transactions, and customer records. Reconciliation in this context focuses on confirming that data extracted from legacy systems is fully and correctly represented in the target environment after transformation and loading.

Validation techniques commonly include record sampling, aggregation comparisons, checksum verification, and balance matching between source and target systems. Post-migration reconciliation compares key financial indicators such as opening balances, transaction totals, and posting statuses to identify discrepancies introduced during migration or transformation.

Reconciliation in integrated financial environments. In operational environments, reconciliation supports consistency across interconnected systems such as core banking platforms, payment gateways, clearing and settlement systems, and data warehouses. The scope extends beyond individual records to complete business workflows. For example, a transaction initiated in a front-end channel is expected to follow a consistent lifecycle authorization, settlement, reversal, or rejection across all downstream systems.

Effective reconciliation also supports reliable financial and operational reporting by verify that aggregated figures align with underlying transaction-level data.

Key reconciliation areas

- **System-to-system data alignment**. Reconciliation examines how transactions are reflected as they move between systems, such as from transaction processing platforms to accounting and settlement systems. Transaction states, amounts, and timestamps are expected to remain aligned across interfaces.
- **Reconciliation scope definition**. Clear identification of reconciliation points such as balances, transaction counts, posting dates, and statuses verify coverage of critical financial processes, including deposits, withdrawals, transfers, and loan repayments.
- **Tolerance levels and exceptions**. Certain variances, such as rounding differences caused by currency conversion, may fall within predefined tolerance limits. Reconciliation distinguishes acceptable variances from genuine discrepancies, with exceptions identified for further analysis.

- **Batch processing and cut-off handling**. Many financial systems rely on scheduled batch cycles, particularly during end-of-day or end-of-period processing. Reconciliation considers transaction handling around cut-off times (such as month-end) to verify completeness and prevent duplication.
- **Ledger and balance reconciliation**. Transaction-level data is aligned with sub-ledger and general ledger postings. Differences, including suspense or unmatched balances, are tracked until resolved.
- **Reconciliation frequency and volume handling**. Reconciliation may be performed in real time, intra-day, or end-of-day, depending on business needs. System behavior under peak transaction volumes is observed to verify reconciliation processes remain stable and reliable.
- **Data transformation and mapping validation**. Financial data often undergoes transformations such as fee application, interest calculation and currency conversion. Reconciliation confirms that transformation rules produce consistent outcomes across systems.
- **Multi-currency and cross-border transactions**. International transactions introduce additional reconciliation complexity due to exchange rates, settlement currencies, and timing differences. Alignment of converted amounts and settlement values is a key focus.

Examples of transaction-specific reconciliation scenarios include:

- **Card transactions**. Card payments involve issuing systems, acquiring platforms, networks, and settlement engines. Reconciliation focuses on transaction amounts, fees, reversals, chargebacks and settlement postings.
- **ATM transactions.** ATM activity spans switches, processors and core banking systems. Reconciliation covers cash withdrawals, fees, reversals, partial dispensations and exception scenarios

4.3 Data Privacy and Data Protection in Finance Testing

Financial systems process highly sensitive information, including personal data, transaction details and institutional information. During testing, such data shall be protected against unauthorized access, disclosure, alteration or misuse. Testing shall also confirm that the system handles data in accordance with applicable privacy and protection requirements.

Data privacy concerns the proper collection, use, storage, sharing and deletion of personal and financial information in line with legal, regulatory and organizational requirements.

Examples of privacy-related areas to be tested include:

- access to sensitive data according to user roles and responsibilities
- restriction of sensitive data visibility in screens, reports, exports and error messages
- use of personal data only for authorized purposes and according to consent

- correct implementation of data retention and deletion rules
- controlled sharing of data through interfaces and external systems
- logging of access and actions on sensitive data to support accountability and auditability

Typical privacy risks in finance testing include unintended exposure of personal or financial data, leakage of sensitive data through logs or reports, inconsistent application of privacy controls and system behavior not aligned with privacy requirements.

Data protection concerns the safeguarding of sensitive information, especially in test environments. Real production data should be avoided unless it is strongly masked or anonymized under strict controls. Where possible, synthetic, anonymized, pseudonymized or masked data should be used for testing.

Examples of data protection-related areas to be tested include:

- classification of data according to sensitivity so that appropriate controls can be applied
- masking or anonymization of sensitive values while preserving test usability
- replacement of identifiers through pseudonymization to reduce exposure of personal data
- restriction of access to authorized personnel only
- retention of test data only for the required period, followed by secure removal
- protection of logs and monitoring outputs so that confidential data is not shown in plain text

Testing of privacy and data protection in financial systems helps reduce compliance, security and operational risks, while supporting trustworthy handling of sensitive information.

5 Functional and Non-Functional Testing – 165 minutes

Keywords

availability, back-to-back testing, calculation testing, exploratory testing, functional testing, load testing, non-functional testing, performance testing, security testing, stress testing

Domain-Specific Keywords

GenAI

Learning Objectives for Chapter 5:

5.1 Functional Testing of Financial Systems

FT-5.1.1 (K3) Apply functional testing to validate financial systems

5.2 Performance Risks in Finance

FT-5.2.1 (K2) Explain the performance risks in financial systems during peak periods

5.3 Performance Testing in Financial Contexts

FT-5.3.1 (K3) Apply load testing and stress testing in finance-specific contexts

5.4 Security Testing and Availability Testing

FT-5.4.1 (K2) Explain security testing and availability testing needs in financial systems

5.5 GenAI in Finance Testing

FT-5.5.1 (K2) Explain how GenAI can support and improve finance testing

Introduction to Functional and Non-Functional Testing

Testing of financial software is crucial for supporting the security, functionality, reliability, and compliance of financial applications. It typically involves various testing types, including functional, security, performance, and compliance testing, to safeguard against risks and enhance software reliability.

5.1 Functional Testing of Financial Systems

Since financial systems handle sensitive financial data and real monetary transactions, the cost of failure is very high. Testing requires a comprehensive, compliant, and risk-based testing covering critical business flows, demanding solid domain knowledge and awareness of regulations and risks.

In the financial industry, functional testing (especially black-box test techniques) plays a key role in supporting functional correctness and satisfying functional requirements. This includes:

- verifying core financial functions such as account management, transaction processing, and reporting,
- defining the scope of functional objectives and coverage strategy,
- performing risk-based prioritization,
- determining compliance with regulations, such as GDPR or PCI DSS (see Chapter 2).

Several test techniques are particularly relevant for validating core financial features (see [ISTQB_CTFL_SYL], Section 4.2):

- equivalence partitioning – verifying transaction functional correctness and consistency,
- boundary value analysis – verifying amount boundaries with currency precision (e.g., 0.01 increments), including rounding at the smallest unit and just-below/at/just-above limit values,
- decision table testing – validating test results for combinations of business rules and conditions, because decision table testing is used to test combinations,
- state transition testing – testing account states such as active/inactive/blocked.

Table 3 provides examples of additional functional test techniques/test approaches suitable for financial software systems.

Table 3. Examples of test techniques used in finance testing

Test technique / test approach		Advantages, disadvantages, and challenges
Calculation testing	This test technique involves recalculating values using known functions and formulas. This is needed when defining correct formulas and boundary values. Some examples include calculating cash flows, net present value of future cash flows, and option prices. Verification through calculation testing focuses on identifying formalized	**Advantages:** reliable test results, easily formalized and documentable. **Disadvantages:** limited applicability for complex analytical testing due to extensive documentation and implementation requirements. **Challenges:** requires deep domain expertise.

	relationships among variables rather than directly computing outputs.	
Exploratory testing	This test approach focuses on identifying behavioral patterns in financial values without requiring explicit calculations. Tests cases verify input-output relationships, validate variable assumptions, apply limit values (e.g., exceeding the transfer amount limit), and use exploratory testing to discover calculation defects,	**Advantages:** quick execution, broad coverage, low overhead. **Disadvantages:** impossibility of integrating into automation frameworks or CI/CD pipelines. **Challenges:** requires strong qualifications and domain-specific knowledge.
Back-to-back testing		**Advantages:** effective for regression testing. **Disadvantages:** setting up test environments and synchronizing input parameters can be difficult, especially when maintaining multiple application versions. **Challenges:** analysing discrepancies demands deep domain knowledge and familiarity with both systems.

Based on time, coverage needs, and team skills, choosing the right set of test techniques is key to achieving efficient and complete test results. For example, ATM testing includes basic techniques (boundary value analysis for min/max transfers) and specialized ones: back-to-back testing, functional correctness testing of fee and currency calculations, and exploratory testing for new ATM models.

To maximize the effectiveness of functional testing, consider adopting the following best practices:

- collect actual user feedback,
- use real devices – emulators and simulators cannot be a long-term solution for device testing,
- test with production-like data (e.g., anonymized real data or realistic synthetic test data),
- coordinate and align test efforts between business experts and testers to achieve a common understanding and joint accountability.

5.2 Performance Risks in Finance

Financial institutions depend on reliable software systems to process high volumes of financial transactions, support payment services, manage investments, and comply with regulatory requirements. These systems must maintain acceptable performance efficiency under varying operational conditions, particularly during peak transaction periods, while preserving data integrity. Performance risks refer to the potential for unacceptable response times, processing delays, or throughput degradation when transaction volumes or system load exceed normal operating levels.

Common performance challenges in financial institutions include:

- Scalability – systems unable to scale effectively with increasing user loads. For example, a payment processing system experiences performance failures during peak times (public holidays, Christmas, Black Friday, market crashes).

- Concurrency defects – failure to manage simultaneous requests. For example, after massive failures, trading platform employees start reloading the application page to begin service.
- Infrastructure limitations – outdated hardware or inefficient configurations hindering optimal performance. For example, investment management systems are experiencing performance failures, but they cannot quickly move to cloud-based external solutions.
- Data integrity violations – loss or corruption of transactional data under heavy load. For example, when a bank prepares annual reports for regulators, it may encounter performance failures that lead to data loss and regulatory sanctions.
- Technical debt risks – the need to update both the software and outdated hardware components, which can fail prematurely, significantly increase maintenance costs, and lead to the unavailability of highly loaded financial systems.
- Third-party dependencies – dependence on external software vendors, cloud services, and payment gateway providers, whose unavailability can affect internal performance operations.

Ignoring software performance risks exposes financial institutions to financial loss, severe reputational harm (e.g., public customer complaints on social media causing a loss of trust). The activities to minimize these problems, including:

- continuous monitoring,
- employee training programs,
- regular audits,
- static test activities include analyzing system-load metrics and comparing them to an agreed baseline with business analysts, to determine whether changes to the load profile are required,
- reliability-focused test processes, including regularly scheduled performance testing and availability testing.

5.3 Performance Testing in Financial Contexts

Performance testing (e.g., stress testing, load testing) plays an important role in assessing the reliability and resilience of financial systems and should be mandatory in regulated environments. The test strategy can specify a separate, dedicated environment for load testing that closely reflects the characteristics of the production environment. This requires strong isolation and a high degree of production similarity, including aligned configurations, synchronization procedures, dependency management, and consistent versions of external services. The same applies to the data, its structure, and quantity. Performance testing is often conducted daily. Alternatively, teams may use continuous performance monitoring or nightly performance tests. More information about general concepts in performance testing can be found in [ISTQB_CT-PT_SYL].

Examples of events that require performance testing in financial systems include:

- initial production deployment or a major release (e.g., adding integration with a fraud-monitoring service to a banking application),
- significant system changes, such as software upgrades (e.g., migrating to a new major database version) or architectural shifts (e.g., moving from a monolith to microservices),

- expected changes to the load profile (e.g., opening new offices leading to an increased client base for a trading company).

For high-load financial systems, typical performance testing tasks may include the following steps:

- initiate,
- gathering and analyzing volumetric historical peak metrics,
- prepare the performance test plan and define acceptable metrics (SLAs),
- prepare the test environment, test data, and performance testing tools,
- execute performance testing,
- analyze the test results and prepare the performance test report.

The performance testing tasks are covered in detail in [ISTQB_CT-PT_SYL], Chapter 4.

Financial systems often experience peak transaction volumes (e.g., tax deadlines, retail peaks, trading sessions). To address the performance risks described in Section 5.2, Table 4 provides examples of additional performance tests applicable to financial software systems.

Table 4. Additional activities that support testing financial software systems"

Activities	Description
Load testing	Validates system functionality under specified load levels using metrics, as transactions per second.
Stress testing	Check system behavior under loads exceeding normal or expected peak conditions (e.g., up to 200%). Expected result: System may experience degraded performance or partial failures, but it should fail gracefully (e.g., return meaningful errors, not crash completely) and recover to acceptable performance levels once the load returns to normal. Metrics: request latency, error rate, throughput, and resource utilization (CPU, memory) during and after the stress period.
Endurance	Determine if the SUT operates stably under sustained load over time. Monitoring metrics such as central processor unit (CPU) and resource utilization can help detect degradation (e.g., memory leaks). This applies to endurance testing.
External system availability	Measures system performance and stability (response times, error rates, resource consumption) when dependent external systems are slow, timeout, or unavailable. Helps ensure that failure handling logic (circuit breakers, retries) does not introduce performance degradation or resource leaks. Metrics: CPU/Memory usage, request latency percentiles, number of timeout exceptions.
Failover	Verifies system stability upon restart, crash resilience, recovery speed and back-up. Cloud-based systems may delete pods during testing. Applicable for spike testing, which determines system resilience against sudden peak load spikes and recovery to stable operation.

Data center availability	Tests correct request balancing between data centers during downtime, using metrics such as failure rate, transactions, and HTTP requests per second. Useful for scalability testing, spike testing, and stress testing.

By implementing regular performance testing procedures and investing in reliable infrastructure, financial institutions can reduce the risk of costly service interruptions.

5.4 Security Testing and Availability Testing

Security testing of financial software verifies that the system protects the confidentiality, integrity, and availability of data, thereby helping prevent fraud and financial loss. These activities should be integrated throughout the development lifecycle, for example, by identifying gaps in standards early and applying threat modelling to mitigate financial risks, or by conducting expert security reviews of designs and architectures.

Types of security tests include:

- vulnerability scanning,
- penetration testing,
- Static/Dynamic Application Security Testing (SAST/DAST),
- dependency and third-party component analysis.

Key finance testing areas include:

- testing cloud configurations and secrets management for compliance (e.g., for payment processing),
- testing data encryption algorithms (e.g., accounting and banking sectors),
- validating multi-factor authentication, role-based access controls, token lifecycles, and determining that there is secure routing of API traffic through dedicated gateways (e.g., mobile, web banking applications),
- verifying security log integrity, configuration, management procedures, and alerting (e.g., protecting policyholders' personal information in the insurance sector).

Several best practices for performing security testing include:

- using specialized tools,
- following applicable standards (e.g. OWASP),
- integrating security into the development process (DevSecOps),
- incorporating threat modelling,
- managing third-party and supply chain risks,
- designing secure test environments and handling test data responsibly,
- aligning with compliance testing (see Chapter 2).

For more information, see [ISTQB_CT-SEC_SYL].

Availability testing determines if there is continuous operation during peak loads to prevent transaction losses, revenue impact, and loss of customer trust. Key objectives and related metrics include:

- recovery targets: Recovery Time Objective (RTO) for IT services and Recovery Point Objective (RPO) for key transactions,
- performance under load: execution time, response time, and throughput for key transactions,
- system reliability: failure rate, Mean Time Between Failures (MTBF), Mean Time to Failure (MTTF), and availability,
- resilience validation: load balancing, failover tests, and Mean Time to Repair (MTTR).

Test strategies for mitigating availability challenges include:

- scalability: perform load testing and stress testing using peak-hour scenarios and validate automated scaling behavior,
- infrastructure: conduct systematic failover testing to verify recovery behavior and continuity,

Security testing and availability testing are essential for reliable financial software systems. Security testing protects against malicious attacks, while availability testing verifies service continuity and recovery under adverse conditions. Together, these activities contribute to overall system reliability and resilience.

5.5 GenAI in Finance Software Testing

The integration of Generative Artificial Intelligence (GenAI) into the testing of financial systems can improve efficiency and consistency in a highly regulated environment. By applying prompt engineering techniques, GenAI can support test activities across the test process, including test analysis, test design, test implementation, test monitoring and test control, and test completion.

Financial organizations can adopt the common AI assistants listed in Table 5 into their SDLC.

Table 5. Common AI assistants applicable in finance testing

Test activities	Assistant aspect of AI
Testing in development phase	Built-in AI assistants can help developers generate unit tests and automatically resolve minor defects by using models tailored to the organization's coding standards and established patterns.
Test analysis	The AI assistant supports testers by analyzing APIs and suggesting requirements and test conditions for review and refinement.
Test design and test implementation	The AI assistant: - builds and synchronizes test models from code, generates test cases within development environments, and uploads test results to test management tools, - creates and reviews automated tests based on existing test cases, and generates UI and API test cases from requirements, - checks test cases for violations of information security requirements, - enables low-code/no-code test automation by helping testers create AI-assisted automated test scripts using visual interfaces and pre-built components, reducing manual coding effort.

Test monitoring and test control	The AI assistant analyzes test logs to detect anomalies and identify defects during testing.
Test completion	The AI assistant supports the creation and validation of test reports and defect reports.

GenAI can enhance testing of financial software, but it also introduces challenges, particularly the limited availability and high cost of high-quality labeled data. To maximize GenAI's value, financial institutions should prioritize data integrity, adherence to regulations, skills development, and infrastructure improvements. Economic risks should be managed through pilot initiatives and realistic planning. Success should be measured using clear metrics (e.g., test cycle time and defect rates) and by evaluating opportunities such as autonomous test generation and continuous compliance monitoring. Quality control roles are likely to expand to include proactive risk mitigation, continuous learning, oversight of AI model training, adoption of best practices, and alignment with business goals. Advanced analytics and AI debugging skills will become increasingly important.

6 Test Automation – 105 minutes

Keywords

test automation

Domain-Specific Keywords

DTO, privacy

Learning Objectives for Chapter 6:

6.1 Role of Test Automation in Finance Testing

FT-6.1.1 (K2) Explain how test automation supports test efficiency in finance

6.2 Test Automation Strategies

FT-6.2.1 (K3) Apply test automation strategies in legacy and modern financial IT landscapes

6.3 Challenges of Test Automation in Regulated Environments

FT-6.3.1 (K2) Explain the challenges of test automation in regulated environments

6.4 Challenges in Different Types of Test Automation

FT-6.4.1 (K2) Explain the challenges of test automation on mobile, frontend and backend components

Introduction to Test Automation

Test automation plays an important role in improving the stability and efficiency of testing for financial systems. In the financial sector, where precision and speed are essential (e.g., calculating compound interest, verifying credit scores, or executing extensive regression test suites), automation offers significant advantages over manual testing. Using specialized tools and frameworks, test engineers can efficiently validate complex calculations, simulate high transaction volumes, and verify compliance with regulatory requirements. For additional information, refer to the [ISTQB_CTAL-TAE_SYL].

6.1 Role of Test Automation in Finance Testing

Test automation provides significant benefits for testing financial systems, such as shorter test cycles, improved accuracy, and better scalability. Examples include:

- Automation enables consistent test execution and delivers precise test results, minimizing discrepancies caused by manual input errors.
- Automation supports quick regression testing after updates (e.g., credit or payment functions), freeing manual testers to focus on complex scenarios.
- Automated test scripts can be scheduled for frequent execution aligned with deployments, providing fast feedback and supporting iterative delivery.
- Automated tests can simulate peak-load scenarios and run consistently across multiple environments with aligned configurations.
- Improved compliance and security – test automation can systematically verify compliance with standards (e.g., PCI DSS for card payments and AML/CFT controls) and generate auditable evidence. Automated tests can also be executed in restricted environments without granting manual access by running repeatable test scripts through controlled pipelines.
- Automated processes can create complex end-to-end test data (e.g., customers with credit history, invoices, cards, and transaction templates).

Despite its advantages, implementing test automation presents unique obstacles specific to the financial domain:

- **Complexity** – financial systems often involve complex workflows that require advanced algorithms to accurately simulate real-world scenarios. For example, automating tests for a mortgage registration process may require coverage of scoring, income verification, payment calculations, and integrations with external registries.
- **Cost** – the upfront effort and expense to design, implement, and maintain effective test automation can be significant. In financial systems, the return on investment is often driven less by reducing manual testing and more by reducing the risk likelihood and risk impact of high-cost failures, such as service outages, failed transactions, and regulatory penalties.
- **Skill gap** – a lack of professionals who understand both financial business logic and the required technical test automation skills can limit the adoption and scalability of test automation efforts.

Addressing these concerns requires:

- strategic planning,
- investment in training programs (competency matrices, individual development plans, and mentoring),

- standardized test frameworks and test tools (e.g. for simulates external APIs)
- integration of test automation into the development lifecycle through CI/CD pipelines,
- collaboration among stakeholders at all test levels, and these elements should be reflected in the test automation strategy.

6.2 Test Automation Strategies

Test automation offers significant advantages, including time and cost savings. However, designing and implementing an optimal test automation strategy requires careful planning and consideration of several factors. Common test automation strategies include:

- Different test distribution patterns (pyramid, ice cream cone, hourglass, umbrella) guide test automation levels, helping assess current states and set improvement targets. Legacy systems require distinct strategies compared to new implementations.
- Shift left prioritizes early test automation and compliance in development pipelines, while shift right monitors real-time quality, transaction integrity, and fraud detection in production environments.
- In a financial organization with a complex infrastructure landscape (e.g., a legacy processing system integrated with a modern mobile payment application), different test approaches are used depending on the software development lifecycle model, such as:
    - agile software development practices, with an emphasis on test automation,
    - legacy SDLC models (e.g., sequential software development), where testing is often planned and executed in distinct phases,
    - devOps practices, enabling continuous testing through CI/CD pipelines.

For example, initially, giving preference to test automation all layers of the testing pyramid – e.g., component tests, API/service tests, and UI tests – a bank can free testing capacity to focus on validating real devices such as ATMs and POS terminals, which are costly and challenging to automate. In this example, the bank implements a set of test approaches: it uses a test distribution scheme, focuses on test automation, and gives priority to early test automation.

For additional information about common test approaches when defining a test automation strategy, refer to the [ISTQB_CT-TAS_SYL], section 3.1.1.

In addition to common approaches, it is necessary to note the specifics inherent to the financial sector. When planning a test automation strategy for the financial sector, it is important to consider the following factors that can influence decisions:

- High accuracy demands – defects in financial calculations can have serious consequences. For example, financial systems may store currency values using different data types and perform calculations with varying levels of precision. When defining the test strategy, it is important to account for these differences and verify that tests apply stricter precision and verification than would typically be required in non-financial domains.
- Implementing end-to-end automated tests across multiple interconnected financial systems introduces significant technical complexity, particularly in areas such as accounting systems with complex billing workflows.

- Use of real-time data (or as close as possible) in automated test scripts – the structure and content of input data may change over time. For example, customer transaction patterns can vary by season and by product offerings. Automated tests should reflect real-world scenarios as closely as possible by using anonymized pre-production data or regularly updated test data.
- Automated test oracle (or pseudo-oracle) – when developing a new system, test results can be compared with a reference application. This approach may be limited because keeping both systems aligned is challenging, and differences in business rules or data processing can affect test results.
- Approach for automated regression testing – compare current application calculations with past versions or reference test results to verify correctness, accounting for maintenance costs of dual test environments.

6.3 Challenges of Test Automation in Regulated Environments

Organizations operating in finance must comply with multiple sets of rules, both national and international, such as GDPR, SOX, PCI DSS, and DORA. These rules impose additional challenges that need to be considered when planning a test automation strategy:

- **Complexity of regulations** – regulations introduce complexity by imposing strict data protection, privacy, and security requirements, impacting test automation designs. An example is adhering to PCI DSS standards when testing ATM transactions.
- **Maintenance problems** – supporting test automation entails regular tasks such as modifying application code, managing test scripts and libraries, updating external integrations, and adjusting test suites.
- **Logging and provability** – business transaction logs must be maintained and retained, along with test logs, for auditor validation. Regulators demand proof of testing efficacy and reproducibility of transaction paths for investigations.
- **Specific automated test tools** – financial organizations frequently rely on proprietary test automation tools tailored to their unique needs. These tools lack widespread market representation and centralized support, for example, test tools for biometric testing and POS/ATMs.

Activities for overcoming challenges include:

- Complexity of regulations
  - Creating a testing map for the complexity of regulations, i.e., a table in which the area of impact, test data requirements, and critical test conditions are defined for each regulatory requirement.
  - Data Anonymization (Masking). Modify real data by removing or obfuscating personal identifiers so that the data remains structurally intact and realistic for testing but is no longer considered personal data under regulations such as GDPR.
  - Synthetic Test Data Generation. Create artificial datasets that mimic real-world characteristics, ensuring they are free from any regulatory constraints.
- Maintenance problems
  - Use a common data transfer object (DTO), if such are defined by the interaction contract.

- Within the test automation framework, the isolation of test logic from resources (UI details, test data) should be implemented.
- Implement automatic startup of automated test scripts with each code change to quickly identify defects.

- Logging and provability
  - Implement end-to-end test logging. Each test run should automatically generate a detailed, structured testware.
  - Automate audit reporting. Develop automated reports that demonstrate coverage of compliance-critical functions, their execution history, and proof of execution before release.
- Specific automated test tools
  - Placing the source code of internal tools in shared internal repositories accessible across the organization (inner source).
  - When developing internal tools, it is necessary to consider the possibility of integrating them through standard protocols (REST API).

6.4 Challenges in Different Types of Test Automation

As the complexity of financial systems increases due to advances in technologies such as cloud computing, service-oriented architectures, AI, and mobile platforms, manual test approaches face numerous challenges. From mobile banking apps to cloud financial systems, these applications require rigorous testing to determine reliability, security, and regulatory compliance. Test automation can streamline this process but also presents challenges, particularly in mobile, front-end, and back-end compenent testing.

While test automation brings considerable advantages, certain challenges persist:

- **Automating mobile applications** – in the financial sector, it's important to consider security and compliance concerns, usability, and user experience, including network conditions variance, battery life optimization, device fragmentation, and compliance-specific challenges related to security or encryption.
- **Automating frontend web-based applications** – focus on determining usability, accessibility, and compliance within financial applications, including cross-browser compatibility, dynamic content handling, GUI automation limitations/flakiness, and test automation with headless browsers.
- **Automating backend** – focus on data integrity, integration, and business logic in financial systems, including API integration complexities, data integrity assurance, and load balancing and scalability defects.
- **Automated security testing** – in financial sector related to:
  - mandatory masking of sensitive data,
  - high level of false-positive and false-negative test results, and the need for periodic manual verification of the test results of security tools,
  - working with encrypted traffic and authentication.

Test automation brings immense benefits to financial software development, but it entails significant challenges related to mobile, frontend, and backend components.

Activities for overcoming challenges include:

- adopting a risk-based test pyramid,
- managing test data and isolating environments in CI/CD,
- reducing flakiness with reliable locators, explicit waits, idempotent setup/teardown,
- defining a device/network matrix, leveraging device clouds, and preferring platform frameworks for stability - applicable for mobile testing.

Test automation in financial software has great potential to improve efficiency and help support the quality of financial services. GenAI holds promise for financial test automation, offering comprehensive test coverage, defect detection, task automation, and quick feedback. Realizing this potential requires overcoming the outlined challenges.

7 List of Abbreviations

	Definition
AML	Anti-Money Laundering
Basel III / IV	Internationally agreed set of measures by the Basel Committee on Banking Supervision
CAC	Continuous Automated Compliance
CACV	Continuous Automated Compliance Validation
CAMT	Cash Management (ISO 20022 message category)
CCM	Continuous Compliance Management
CFT	Combating the Financing of Terrorism
CTFL	Certified Tester - Foundation Level
CT-FT	Certified Tester - Finance Testing
CPU	Central Processor Unit
DAST	Dynamic Application Security Testing
DORA	Digital Operational Resilience Act
DTO	Data Transfer Object
EMS	Execution Management System
GDPR	General Data Protection Regulation
GenAI	Generative Artificial Intelligence
IEEE	Institute of Electrical and Electronics Engineers
ISO	International Organization for Standardization
KYC	Know Your Customer (or Client)
MiCA	Markets in Crypto-Assets Regulation
MFA	Multi-Factor Authentication
MTBF	Mean Time Between Failures
MTTF	Mean time to failure
MTTR	Mean time to repair
OMS	Order Management System
OWASP	Open Worldwide Application Security Project
PACS	Payments Clearing and Settlement (ISO 20022 message category)
PAIN	Payment Initiation (ISO 20022 message category)
PAN	Device Primary Account Number
PCI DSS	Payment Card Industry Data Security Standard
PII	Personally Identifiable Information
POS	Point of Sale
PSD2	European Union's Revised Payment Services Directive 2
RTO	Recovery Time Objective

RPO	Recovery Point Objective
SAST	Static Application Security Testing
SCA	Strong Customer Authentication
SOX	Sarbanes-Oxley Act
SUT	System Under Test

8 Non-Testing Domain Specific Terms

Term Name	Definition
audit trail	A system that traces the detailed transactions relating to any item in an accounting record.
Basel III / IV	An international regulatory framework for banks that strengthens capital requirements, tightens risk and liquidity rules and demands accurate, auditable reporting.
compliance risk	The risk of legal sanctions, financial losses, or damage to reputation that an institution may suffer as a result of failure to comply with laws, regulations, codes of conduct, and standards of best practice.
customer trust	The faith a consumer has in a company to deliver on its commitments and uphold its promises.
data accuracy	A measure of how closely information represents the objects or events being recorded.
data breach	A security incident where confidential or sensitive information is illegally accessed, viewed, disclosed, copied, or stolen by an unauthorized individual or group, compromising its security, integrity, or confidentiality.
data consistency	The accuracy, completeness, and correctness of data stored in a database
data privacy	Data privacy refers to the protection of personal information from unauthorized access, disclosure, alteration, or destruction. It involves policies, procedures, and technologies that govern how individuals' private data is collected, stored, processed, and shared by organizations.
data profiling	The systematic analysis, examination, and summarization of datasets to understand their structure, content, relationships, and quality.
data reconciliation	The systematic process of comparing data from different sources to determine accuracy, consistency, and completeness.
Digital Operational Resilience Act (DORA)	A regulation that enhances the operational resilience of information and communication technology (ICT) and third-party providers in the EU financial sector.
Data Transfer Object (DTO)	A design pattern used in software development to efficiently transfer data between different layers or components of an application. It typically contains only the necessary fields for transferring information without any business logic or behavior, ensuring separation of concerns and improving performance by minimizing network traffic.
financial institution	A company or organization that engages in financial and monetary transactions, acting as an intermediary to provide

	financial services such as loans, deposits, investments, and insurance to individuals, businesses, and governments.
GenAI	Generative Artificial Intelligence (GenAI) is a branch of artificial intelligence that uses large, pre-trained models to generate human-like output, such as text, images, or code. Large language models (LLMs) are GenAI models that are pre-trained on large textual datasets, enabling them to determine context and produce relevant responses according to user prompts.
General Data Protection Regulation (GDPR)	A European Union (EU) regulation that establishes comprehensive rules for data privacy and the protection of personal data for residents.
general ledger	The central, primary accounting system for a business, used to record all its financial transactions across various accounts such as assets, liabilities, equity, revenue, and expenses.
know your customer (KYC)	A mandatory process for financial institutions to verify the identity of their clients and assess potential risks.
Open Worldwide Application Security Project (OWASP)	A non-profit foundation that works to improve software security.
Payment Card Industry Data Security Standard (PCI DSS)	A set of security standards that any business handling credit or debit card data must follow to prevent fraud.
periodic risk assessment	A scheduled, recurring review of an organization's operations, systems, or projects to identify, analyze, and mitigate evolving threats.
privacy	The right to control your personal information, space, and communication from unauthorized intrusion and observation.
PSD2	Payment Services Directive 2. An EU directive that regulates payment services to increase competition, improve security, and open the market to new companies, known as third-party providers.
regulatory compliance	The process of an organization following applicable laws, regulations, standards, and rules set by governments and regulatory bodies to support legal, ethical, and safe operations.
regulatory standard	An official rule or requirement set by a government or other authority to determine minimum levels of quality, safety, or performance in an industry or process.
risk transfer	
settlement system	A set of processes, rules, and infrastructure that enables the final transfer of money or securities to fulfil a financial obligation between parties.
Sarbanes-Oxley Act (SOX)	A United States federal law requiring public companies to establish and maintain accurate financial records and strong internal controls to prevent corporate fraud and protect investors.

9 References

9.1 Standards

ISO 20022 Financial services – Universal financial industry message scheme (2025)

https://www.iso20022.org/

ISO/IEC 25010 Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) System and software quality models (2023)

https://iso25000.com/index.php/en/iso-25000-standards/iso-25010

ISO/IEC 25019:2023 Systems and software engineering

https://www.iso.org/standard/78177.html

ISO/IEC 27001 Information security, cybersecurity and privacy protection – Information security management systems – Requirements (2022)

https://www.iso.org/standard/27001

IS0 31000 Risk management – Principles and guidelines (2018)

https://www.iso.org/standard/65694.html

MiFID II Markets in Financial Instruments Directive II, European Union (2018)

https://eur-lex.europa.eu/eli/dir/2014/65/oj/eng

OWASP The Open Worldwide Application Security Project (2021)

https://owasp.org/

PSD2 Revised Payment Services Directive 2, European Union (2019)

https://www.ecb.europa.eu/press/intro/mip-online/2018/html/1803_revisedpsd.en.html

9.2 ISTQB® Documents

[ISTQB_CTAL-TA_SYL] ISTQB® Advanced Level Test Analyst Syllabus, V4.0, 2025, https://istqb.org/wp-content/uploads/sdm-uploads/ISTQB-CTAL-TA-Syllabus-v4.0-EN-4.pdf

[ISTQB_CTAL-TAE_SYL] ISTQB® Certified Tester Advanced Level Test Automation Engineering, v2.0, 2024, https://istqb.org/certifications/certified-tester-advanced-level-test-automation-engineering-ctal-tae-v2-0/

[ISTQB_CT-ATLaS_SYL] ISTQB® Certified Tester Agile Test Leadership at Scale (ATLaS) v2.0, 2023, https://istqb.org/wp-content/uploads/2024/11/ISTQB_CT-ATLaS_Syllabus_v2.0.pdf

[ISTQB_CT-GenAI_SYL] ISTQB® Certified Tester Testing with Generative AI v1.0, 2025, https://istqb.org/wp-content/uploads/sdm-uploads/CT-GenAI-Syllabus-v1.0.pdf

[ISTQB_CTFL_SYL] ISTQB® Foundation Level Syllabus v4.0.1, 2024, https://istqb.org/wp-content/uploads/2024/11/ISTQB_CTFL_Syllabus_v4.0.1.pdf

[ISTQB_CT-MAT] ISTQB® Certified Tester Mobile Application Testing, https://istqb.org/wp-content/uploads/2024/11/ISTQB-CT-MAT_Syllabus_v1.0_2019.pdf

[ISTQB_CT-PT_SYL] ISTQB® Certified Tester Performance Testing, v1.0, 2018, https://istqb.org/wp-content/uploads/2024/11/ISTQB-CT-PT_Syllabus_v1.0_2018.pdf

[ISTQB_CT-SEC_SYL] ISTQB® Security Tester v1.0, 2016, https://istqb.org/wp-content/uploads/2024/11/ISTQB-CT-SEC_Syllabus_v1.0_2016.pdf

[ISTQB_CT-STE_SYL] ISTQB® Security Test Engineer v1.0.1, 2025, https://istqb.org/certifications/certified-tester-security-test-engineer/

[ISTQB_CT-TAS_SYL] ISTQB® Certified Tester Test Automation Strategy v1.0, 2024, https://istqb.org/wp-content/uploads/2024/11/ISTQB_CT-TAS_Syllabus_v1.0.pdf

9.3 Glossary References

- Agile Alliance https://www.agilealliance.org/agile-practice-guide/
- IREB-CPRE for Requirements Engineering https://www.ireb.org/en/cpre/glossary
- ISTQB® Glossary https://glossary.istqb.org/

9.4 Books, Articles and Web Pages

Books

Adzic, G. (2009). Bridging the Communication Gap: Specification by Example and Agile Acceptance Testing. Neuri Limited.

Adzic, G. (2009b). Test Driven Development: By Example. Addison-Wesley.

Enders, A. (1975). An Analysis of Errors and Their Causes in System Programs. IEEE Transactions on Software Engineering, pp. 140-149.

Marick, B. (2003). Exploration through Example. Retrieved from http://www.exampler.com/old-blog/2003/08/21.1.html#agile-testing-project-1

Webpages

Dieu (2024) Linh Chu Dieu, “Best QA and Testing Practices for Financial Systems”, from https://smartdev.com/best-qa-and-testing-practices-for-financial-systems/

Hayes (2025) Adam Hayes, “What Is a Financial Institution”, from https://www.investopedia.com/terms/f/financialinstitution.asp

Joseph (2020) Timothy Joseph. “Financial Domain Testing: The Breakdown”, from https://blog.qasource.com/financial-domain-testing-the-breakdown/

Veenendaal (2012) “PRISMA: Product Risk Assessment for Agile projects”, from https://www.erikvanveenendaal.nl/NL/files/testing_experience_no_20_column_van_veenendaal.pdf

The previous references point to information available on the Internet and elsewhere. Even though those references were checked at the time of publication of this syllabus, the ISTQB® cannot be held responsible if the references are not available anymore.

Appendix A – Learning Objectives/Cognitive Level of Knowledge

The specific learning objectives applying to this syllabus are shown at the beginning of each chapter. Each topic in the syllabus will be examined according to its learning objective.

The learning objectives begin with an action verb corresponding to their cognitive level of knowledge as listed below.

Level 1: Remember (K1)

The candidate will remember, recognize and recall a term or concept.

Action verbs: Recall, recognize.

Examples
Recall the concepts of the test pyramid.
Recognize the typical objectives of testing.

Level 2: Understand (K2)

The candidate can select the reasons or explanations for statements related to the topic, can summarize, compare, classify and give examples for the testing concept.

Action verbs: Classify, compare, differentiate, distinguish, explain, give examples, interpret, summarize

Examples	**Notes**
Classify test tools according to their purpose and the test activities they support.	
Compare the different test levels.	Can be used to look for similarities, differences, or both.
Differentiate testing from debugging.	Looks for differences between concepts.
Distinguish between project and product risks.	Allows two (or more) concepts to be separately classified.
Explain the impact of context on the test process.	
Give examples of why testing is necessary.	
Infer the root cause of defects from a given profile of failures.	
Summarize the activities of the work product review process.	

Level 3: Apply (K3)

The candidate can carry out a procedure when confronted with a familiar task, or select the correct procedure and apply it to a given context.

Action verbs: Apply, implement, prepare, use

Examples	Notes
Apply boundary value analysis to derive test cases from given requirements.	Should refer to a procedure / technique / process, etc.
Implement metrics collection methods to support technical and management requirements.	
Prepare installability tests for mobile apps.	
Use traceability to monitor test progress for completeness and consistency with the test objectives, test strategy and test plan.	Could be used in a LO that wants the candidate to be able to use a technique or procedure. Similar to 'apply'.

Reference

(For the cognitive levels of learning objectives)

Anderson, L. W. and Krathwohl, D. R. (eds) (2001) A Taxonomy for Learning, Teaching and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives, Allyn and Bacon

10 Appendix B – Business Outcomes traceability matrix with Learning Objectives

This section lists the traceability between the Business Outcomes and the Learning Objectives of Certified Tester - Finance Testing.

Business Outcomes: Certified Tester - Finance Testing			BO1	BO2	BO3	BO4	BO5	BO6
BO1	understand the structure of the financial industry, its institutions, systems, and processes, and explain the specific challenges of testing within this domain.		4					
BO2	apply testing practices that support compliance with financial regulations and standards, ensuring auditability, traceability, and documentation requirements are met.			7				
BO3	apply risk-based testing approaches to financial systems, prioritizing testing activities according to business, technical, and regulatory risks.				3			
BO4	understand test practices that ensure data accuracy, consistency and protection across financial systems, including the use of masking and anonymization techniques.					4		
BO5	apply functional and non-functional test techniques in the financial domain, with particular emphasis on performance, reliability, and security.						5	
BO6	understand the role of test automation in financial environments and apply automation strategies effectively in both legacy and modern systems, while considering regulatory constraints.							4
Unique LO	Learning Objective	K-Level						
1	Introduction to the Financial Services Industry							
1.1	Fundamentals of Software Testing in Finance Testing							
FT-1.1.1	Recall the primary objectives of finance testing	K1	X					

FT-1.1.2	Summarize the unique characteristics and technical requirements of various financial test environments	K2	X					
FT-1.1.3	Identify key financial systems and typical testable workflows associated with them	K2	X					
1.2	Challenges and the Role of Domain Knowledge in Finance Testing							
FT-1.2.1	Explain how domain knowledge is applied to mitigate common testing challenges in the financial sector	K2	X					
2	Compliance Testing							
2.1	Compliance Testing Objectives and Approaches							
FT-2.1.1	Summarize the objectives of compliance testing and the relationship between its primary testing approaches	K2		X				
2.2	Financial Regulations and Their Impact on Testing							
FT-2.2.1	Explain the testing relevance of key financial regulations and standards	K2		X				
FT-2.2.2	Recall consequences of non-compliance to finance regulations	K1		X				
2.3	Distinctive Characteristics of Regulated Finance Testing							
FT-2.3.1	Summarize the unique characteristics of finance testing	K2		X				
2.4	Auditability and Continuous Automated Compliance Validation							
FT-2.4.1	Understand the fundamental principles of auditability in compliance testing	K2		X				
FT-2.4.2	Understand the principles and implementation mechanisms of Continuous Automated Compliance Validation (CACV) within the software delivery lifecycle.	K2		X				
3	Risk-Based Testing							
3.1	Categories of Risk in Financial Systems							
FT-3.1.1	Explain the main categories of risk in financial systems	K2			X			
3.2	Principles of Risk-Based Testing in Finance							
FT-3.2.1	Explain the principles of risk-based testing in finance	K2			X			

3.3	Risk-Based Activities							
FT-3.3.1	Explain how risk-based prioritization affects test activities in financial applications	K2			X			
4	Data Testing and Data Management							
4.1	Data Accuracy and Consistency							
FT-4.1.1	Explain the importance of data accuracy and consistency across financial systems	K2				X		
4.2	Data Reconciliation Across Financial Systems							
FT-4.2.1	Apply data reconciliation techniques to multi-system financial processes	K3				X		
4.3	Data Privacy and Data Protection in Financial Testing							
FT-4.3.1	Explain the importance of data privacy and data protection in finance testing	K2				X		
5	Functional & Non-Functional Testing							
5.1	Functional Testing of Finanial Systems							
FT-5.1.1	Apply functional test techniques to validate financial systems	K3					X	
5.2	Performance Risks in Finance							
FT-5.2.1	Explain the performance risks in financial systems during peak periods	K2					X	
5.3	Performance Testing in Financial Contexts							
FT-5.3.1	Apply load testing and stress testing in finance-specific contexts	K3					X	
5.4	Security and Availability Testing							
FT-5.4.1	Explain security testing and availability testing needs in financial systems	K2					X	
5.5	GenAI in Finance Software Testing							
FT-5.5.1	Explain how GenAI can support and improve financial testing	K2					X	
6	Test Automation							
6.1	Role of Test Automation in Finance Testing							

FT-6.1.1	Explain how test automation supports test efficiency in finance	K2						X
6.2	Test Automation Strategies							
FT-6.2.1	Apply test automation strategies in legacy and modern financial IT landscapes	K3						X
6.3	Challenges of Test Automation in Regulated Environments							
FT-6.3.1	Explain the challenges of test automation in regulated environments	K2		X				X
6.4	Challenges in Different Types of Test Automation							
FT-6.4.1	Explain the challenges of test automation on mobile, frontend and backend components	K2						X

11 Appendix C – Release Notes

The ISTQB® Certified Tester – Finance Testing (CT-FT) Syllabus v1.0 is the first release of this Specialist syllabus. It introduces a structured approach to testing in the finance domain, addressing the unique challenges of financial systems, processes and regulatory requirements.

The creation of this syllabus was driven by the increasing reliance on complex IT landscapes within finance, the growing importance of regulatory compliance and the demand for risk-based and data-focused test approaches.

This release aims to provide finance organizations, testers and quality professionals with practical, domain-specific knowledge that complements the existing ISTQB® portfolio.

Purpose and benefits

The CT-FT syllabus strengthens the ISTQB® portfolio by addressing one of the world's most regulated and critical industries. It enables testers to gain valuable domain expertise, supports organizations in meeting compliance and quality requirements and contributes to professional recognition of finance testing worldwide.

As a first-time release, CT-FT establishes the baseline for future updates. Evolutions will follow industry trends, regulatory developments and emerging technologies to help ensure the syllabus remains relevant for both practitioners and organizations.

12 Appendix D – Index