TestOps Data Standardization

Overview

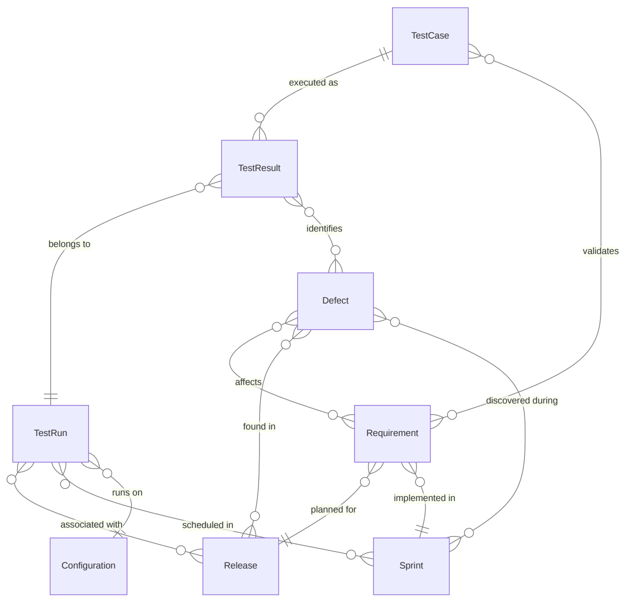
This document defines the standardized approach for data models, entity relationships, filtering rules, and metric calculations used in TestOps reports and dashboards. It ensures consistency across all reporting components and enables the future development of a customizable reporting framework.

DISCLOSURE: This document serves as a reference framework only, not an action plan. A separate action plan will be developed to address existing inconsistencies and dataset mismatches. This framework establishes the target state for data standardization that engineering teams should work toward, provides QE with expected data structures for test planning, and guides designers in creating consistent reporting interfaces.

Document Owner	@Ha Nguyen
Contributors	@Que Tran @Huy (Henrik) Dao @Nhu (Kara) Nguyen
Informed	@Huy Nguyen @Long (Leo) Bui @Man Hua @Khue (Mike) Nguyen @Tien Nguyen @Nhan Tran

Core Entities & Relationships

The following diagram illustrates the primary relationships between key entities:



Association Highlights

Test Case → Requirement Association

- **Association Type**: Many-to-many (test cases can be linked to multiple requirements and vice versa)
- **Usage**: Used for requirement coverage calculations
- **Scope Impact**: Determines which test cases belong to which release/sprint based on requirement associations

Test Case → Test Result Association

- **Association Type**: One-to-many (a test case can have multiple results)
- **Latest Result**: Determined by the most recent startTime within the selected scope
- **Scope Impact**: Latest result may differ between time-based and iteration-based scopes

Test Result → Test Run Association

- **Association Type**: Many-to-one (many results belong to one test run)
- **Usage**: Used for grouping related test executions
- **Scope Impact**: Test run's release/sprint association propagates to its results

Test Run → Release/Sprint Association

- **Association Type**: Many-to-many (test runs can be linked to multiple releases/sprints)
- **Usage**: Used to determine which test results belong to which release/sprint

Defect → Test Result Association

- **Association Type**: Many-to-many (defects can be linked to multiple test results and vice versa)
- **Usage**: Used to track defects found during specific test executions
- **Scope Impact**: Determines which defects belong to which release/sprint based on test result associations

Core Entity Definitions

- **Project Scoping**: All entities should be scoped within a project as you've noted
- **Soft Deletion**: We should explicitly note handling of soft-deleted items across all entities

Entity	Definition	**Identifiers**	**Timing Information**	**Status Values**	**Relationships**	**Additional Properties**	**Derived Values**
Test Case	The fundamental test unit that can be executed to validate specific functionality	Test ID, Name	Creation Date, Last Update, Last Execution Date	Workflow Status (Draft, Ready to Review, In Review, Published, Inactive)	Linked Requirements, Related Defects, Author	Implementation Type (Manual, Automated, Manual & Automated)	**Latest Result**: For a test case within a release/sprint, use the most recent applicable result within scope boundaries **Not Executed:** If a test case has no results within the selected scope, its result-related columns should display "Not Executed" or equivalent
Test Result	Represents the outcome of a test case execution within a test run	Result ID, Executed Test Case, Parent Test Run	Start Time, End Time	Execution Status (Passed, Failed, Error, Blocked, Skipped, Incomplete)	Parent Test Run, Executor, Configuration, Execution Profile, *Environment*	Execution Method, Duration	
Test Run	A collection of test executions performed together	Run ID, Run Name	Start Time, End Time, Scheduled Time	Execution Stage (Todo, Running, Processing, Finished, Terminated) Execution Status (Passed, Failed, Incomplete)	Associated Release, Associated Sprint, Run Executor, Test Configurations, Execution Profile, *Environments*	Execution Type (Manual, Automation), Duration	Results Summary (pass/fail/other counts)

Requirement	A specification of functionality that needs to be tested	Requirement ID, Title	Creation Date, Last Update	ALM Workflow Status, ALM Status Category	Planned Release, Target Sprint		Associated Test Cases (Published only), Test Coverage Status
Defect	An issue found during testing	Defect ID, Title	Creatied Date, Last Update, Resolved Date	ALM Workflow Status, ALM Priority, ALM Severity (Critical, High, etc.), ALM Status Category	Associated Test Results, Related Requirement, Affected Release, Found During Sprint, Affected Environments		
Configuration	Represents the environment where tests are executed	Config ID, Name	-	-		Configuration Category, Operating System, Browser Details, Device Information, *Environment (QA, Staging, Production)*	Test Execution Count, Test Case Count, Pass Rate, Fail Rate
Release	A planned software deployment	Release ID, Name	Start Date, End Date	ALM Workflow Status		Description	Readiness, Passed Criteria
Sprint	A time-boxed development iteration	Sprint ID, Name	Start Date, End Date	ALM Workflow Status		Goal	Readiness, Passed Criteria

Status Classifications

Status values are grouped into categories for consistent calculations:

Test Result Status Categories

Status	Category	Description	Default Inclusion
Passed	Executed	Test executed successfully	Execution Progress, Pass Rate, Execution Count
Failed		Test executed but did not meet expectations	Execution Progress, Failure Rate, Execution Count
Error	Non-executed	Test could not complete due to technical issues	Execution Progress, Error Rate
Incomplete		Test executed but did not have `endTime` in execution log	
Blocked		Test could not be executed due to dependencies	Execution Progress
Skipped		Test was intentionally skipped	
Not Run		Test has not been executed	

Test Case Type Classifications

Classification	Definition	Filter Criteria
Manual Test Case	Test case with only manual implementation	TestCase.type = "Manual"
Automated Test Case	Test case with only automated implementation	TestCase.type = "Automated"
Dual Implementation	Test case with both implementations	TestCase.type = "Manual & Automated"
Manual Execution	Test executed manually	TestRun.executionType = "Manual"
Automated Execution	Test executed via automation	TestRun.executionType = "Automated"

Requirement Coverage Status

Status	Definition	Criteria
Fully Validated	All tests published, executed, and passed	All linked test cases are published AND all executed with "Passed" status
Partially Validated	Some tests incomplete or failed	Tests linked but some not published OR not executed OR not passed
Not Covered	No published tests linked	No published test cases linked to requirement

Test Case Status Classification

Status	Definition	Included in Analysis
Draft, Ready to Review, In Review	In development, not ready for execution	No (unless explicitly included)
Published	Ready for execution	Yes
Archived	No longer actively used	No (unless explicitly included)

Test Run Status Classification

Stage Group	Stage Category	Status	Definition	Default Inclusion
Executed	Finished	Passed, Failed	Represents completed test runs	Included in execution rate calculations
Incomplete	Terminated, Cancelled, Incomplete	Incomplete	Represents prematurely ended test runs	Excluded from execution rate calculations
	Running, Processing/Importing		Groups in-progress test runs	
	Not Started		Represents scheduled but not started test runs	Available in Execution module only, not visible in reports as no record exists yet

Scope Rules

TestOps supports two primary analysis scopes with distinct entity inclusion rules:

Time-Based Scope

Time-based analysis focuses on activities within a specific time range:

Entity	Primary Timestamp	Inclusion Rule	Usage Context
TestCase	updatedAt	Created or modified within time range	Test case activity tracking
TestResult	startTime	Execution started within time range	Result analysis, trending
TestRun	startTime	Execution started within time range	Execution tracking, scheduling
Requirement	updatedAt	Created or modified within time range	Requirement tracking
Defect	updatedAt	Created or modified within time range	Defect tracking, resolution time

Release/Sprint Scope

Release/Sprint scope focuses on items associated with a specific iteration:

- **Start Date/End Date**: The defined time boundaries of a sprint or release
- **Latest Result**: For a test case within a release/sprint, use the latest test result that falls within the scope definition (not necessarily the absolute latest)
- Only **Published** test cases should be counted in coverage metrics

Entity	Release Scope Inclusion	Sprint Scope Inclusion
TestCase	Linked to release requirements OR executed in release test runs	Linked to sprint requirements OR executed in sprint test runs
TestResult	From test runs marked for release	From test runs marked for sprint
TestRun	Explicitly associated with release	Explicitly associated with sprint
Requirement	Explicitly associated with release	Explicitly associated with sprint
Defect	Linked to release test results OR directly associated with release	Linked to sprint test results OR directly associated with sprint
Configuration	• Configuration entities themselves don't have time or iteration-based scoping • They are referenced by test runs and test results	

Standard Filtering Framework

The filtering system follows a hierarchical approach:

```
[SCOPE SELECTION] → [GLOBAL FILTERS] → [ENTITY-SPECIFIC FILTERS] → [AGGREGATION RULES] → [METRIC CALCULATION]
```

Filtering Standardization

Standard Metrics & Calculations

Any Metric = Total matching condition / Total base condition

For example:

- Execution Rate = Total executed test cases / Total test cases
- Manual Pass Rate = Total passed manual executed test cases / Total manual executed test cases

- Automation Pass Rate = Total passed automated executed test cases / Total automated executed test cases

Test Execution Metrics

Executed Tests = Total Tests - Non-executed Tests

Metric	Definition	Calculation	Notes
Pass Rate	Percentage of test cases that passed	(Passed Tests / Executed Tests) * 100	• Success Group Tests, a.k.a Passed • Exclude Not Executed to focus on actual results
Failure Rate	Percentage of test cases that failed	((Failed Tests) / (Executed Tests) * 100	• Failure Group Tests, a.k.a Failed • Exclude Not Executed to focus on actual results
Execution Progress	Percentage of test cases executed compared to planned	(Executed Tests / Total (Planned) Test Cases) × 100	• For release/sprint scope: Only test cases within the scope
Error Rate *(Not available yet)*	Percentage of test cases that errored	(Error Tests / (Total Tests - Not Executed)) * 100	• Technical errors only • Exclude Not Executed to focus on actual results

Defect Metrics

Metric	Definition	Calculation	Notes
Open Defects.	Count of unresolved defects	COUNT(Defects WHERE resolvedAt = NULL)	
Blocking Defects	Count of open defects with configured blocking criteria	COUNT(Defects WHERE resolvedAt = NULL AND priority IN ([Configured Priorities]))	Key release/sprint blocker metric
Defect Density	Number of defects per test case	Open Defects / Total Test Cases	Measure of quality
Resolution Rate	Percentage of defects resolved	(Resolved Defects / Total Defects) * 100	Measure of progress
Average Resolution Time	Average time to resolve defects	AVG(Resolution Date - Reported Date)	

Requirement Coverage Metrics

Metric	Definition	Calculation	Notes

Test Coverage	Percentage of requirements covered by tests	(Requirements with linked Published Test Cases / Total Requirements) * 100	Only count published test cases
Test Execution Coverage	Percentage of requirements with executed tests	(Requirements with executed Test Cases / Total Requirements) * 100	
Pass Coverage	Percentage of requirements with passing tests	(Requirements with all tests passing / Total Requirements) * 100	

Configuration Coverage Metrics

Metric	Definition	Calculation	Notes
Platform Coverage	Percentage of test cases executed per platform	(Test Cases Executed on Platform / Total Test Cases) * 100	Can be broken down by version
Browser Coverage	Percentage of test cases executed per browser	(Test Cases Executed on Browser / Total Test Cases) * 100	
OS Coverage	Percentage of test cases executed per OS	(Test Cases Executed on OS / Total Test Cases) * 100	

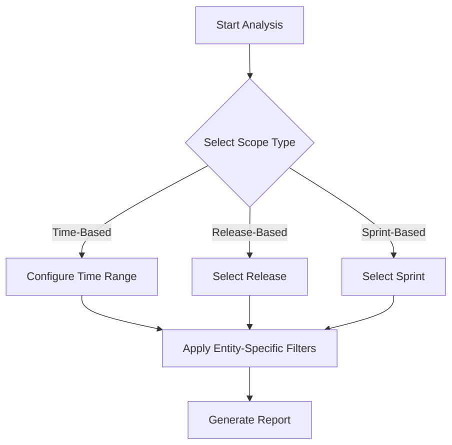
Edge Case Handling

Scenario	Handling	Rationale
TestResult from a TestRun marked for a release, but TestCase not linked to any requirement in that release	Include in release scope	Direct assignment takes precedence
TestCase linked to multiple requirements spanning different releases/sprints	Include in all relevant scopes	Ensures comprehensive coverage tracking
Defect linked to multiple TestResults across different releases/sprints	Include in all relevant scopes	Ensures comprehensive defect tracking
Test executed outside the time range of its assigned sprint/release	Includes in sprint scope/release	• Maintains data completeness - prevents gaps in coverage analysis • Reflects reality of testing activities not always aligning with sprint boundaries • Supports regression testing needs and defect verification • **Note:** UI should visually distinguish out-of-bounds executions

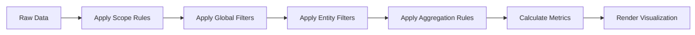
		and provide filtering options
Configuration with no test executions	Exclude from coverage calculations	Cannot calculate meaningful metrics
Related Defects in Test cases vs. Defects under Release/Sprint scope	• **Test cases view:** Shows all test cases in release scope with all related defects. • **Defects view:** Shows only defects directly associated with the release via test results.	These complementary views serve different purposes - test case view provides complete defect visibility per test case, while defect view focuses on release-specific issues only.
Test Execution Progress vs. Pass Rate reporting	Maintain separate metrics and views for Execution Progress (using Test Case Datasets) and Pass Rate (using Test Execution Datasets)	These metrics serve different purposes - Execution Progress tracks completion status against planned work, while Pass Rate measures quality of executed tests only
	Quality Dashboard: Split into separate widgets for Execution Progress and Test Execution Pass Rate, each with dedicated reports	Prevents confusion between completion metrics and quality metrics, allowing stakeholders to focus on relevant data for decision-making

Appendix: Reference Flowcharts

Analysis Scope Selection Flow



Report Data Processing Flow



Future Considerations

Property/Dataset	Use Case	User Needs	Example Calculated Properties
Test Case Quality Properties	Evaluate and improve test case effectiveness	Identify low-quality or problematic test cases for maintenance or refactoring	• P95 Execution Duration • Flakiness Score (% of inconsistent results) • Defects Found Count • Quality Score (composite metric) • Maintenance Frequency • Complexity Score
User Productivity	Measure team performance and resource allocation	Track individual and team efficiency metrics for capacity planning	• Tests Created per Period • Tests Executed per Period • Defects Found per Period • Average Time to Create/Execute Tests • Documentation Completeness Score
License vs. Usage Utilization	Optimize license costs and allocation	Understand if licenses are properly allocated and utilized	• Active Users vs. Licensed Users Ratio • Usage Hours per License

			• Feature Utilization per License Type • ROI per License • Unutilized License Cost
Activity Tracking	Monitor system usage patterns	Understand how reports and dashboards are being used	• Most Viewed Reports • Dashboard Usage Frequency • Time Spent per Report • Filter Usage Patterns • Export/Share Frequency • Custom Report Creation Rate

When Core team implements direct link between Test Case and Release/Sprint → Revisit the rules of Total Planned Test Cases.

References

https://docs.getxray.app/display/ON/The+Different+Types+of+Statuses+and+how+they+connect

https://docs.getxray.app/display/XRAYCLOUD/Coverage+Analysis

https://docs.getxray.app/display/XRAYCLOUD/Understanding+the+calculation+of+coverage+status+and+the+status+of+Tests