5 Test Management — 225 minutes

Keywords

configuration management, defect management, entry criteria, exit criteria, product risk, project risk, risk, risk level, risk-based testing, test approach, test control, test estimation, test manager, test monitoring, test plan, test planning, test progress report, test strategy, test summary report, tester

Learning Objectives for Test Management

5.1 Test Organization

FL-5.1.1 (K2) Explain the benefits and drawbacks of independent testing

FL-5.1.2 (K1) Identify the tasks of a test manager and tester

5.2 Test Planning and Estimation

FL-5.2.1 (K2) Summarize the purpose and content of a test plan

FL-5.2.2 (K2) Differentiate between various test strategies

FL-5.2.3 (K2) Give examples of potential entry and exit criteria

FL-5.2.4 (K3) Apply knowledge of prioritization, and technical and logical dependencies, to schedule test execution for a given set of test cases

FL-5.2.5 (K1) Identify factors that influence the effort related to testing

FL-5.2.6 (K2) Explain the difference between two estimation techniques: the metrics-based technique and the expert-based technique

5.3 Test Monitoring and Control

FL-5.3.1 (K1) Recall metrics used for testing

FL-5.3.2 (K2) Summarize the purposes, contents, and audiences for test reports

5.4 Configuration Management

FL-5.4.1 (K2) Summarize how configuration management supports testing

5.5 Risks and Testing

FL-5.5.1 (K1) Define risk level by using likelihood and impact

FL-5.5.2 (K2) Distinguish between project and product risks

FL-5.5.3 (K2) Describe, by using examples, how product risk analysis may influence the thoroughness and scope of testing

5.6 Defect Management

FL-5.6.1 (K3) Write a defect report, covering defects found during testing

5.1 Test Organization

5.1.1 Independent Testing

Testing tasks may be done by people in a specific testing role, or by people in another role (e.g., customers). A certain degree of independence often makes the tester more effective at finding defects due to differences between the author's and the tester's cognitive biases (see section 1.5). Independence is not, however, a replacement for familiarity, and developers can efficiently find many defects in their own code.

Degrees of independence in testing include the following (from low level of independence to high level):

- No independent testers; the only form of testing available is developers testing their own code
- Independent developers or testers within the development teams or the project team; this could be developers testing their colleagues' products
- Independent test team or group within the organization, reporting to project management or executive management
- Independent testers from the business organization or user community, or with specializations in specific test types such as usability, security, performance, regulatory/compliance, or portability
- Independent testers external to the organization, either working on-site (insourcing) or off-site (outsourcing)

For most types of projects, it is usually best to have multiple test levels, with some of these levels handled by independent testers. Developers should participate in testing, especially at the lower levels, so as to exercise control over the quality of their own work.

The way in which independence of testing is implemented varies depending on the software development lifecycle model. For example, in Agile development, testers may be part of a development team. In some organizations using Agile methods, these testers may be considered part of a larger independent test team as well. In addition, in such organizations, product owners may perform acceptance testing to validate user stories at the end of each iteration.

Potential benefits of test independence include:

- Independent testers are likely to recognize different kinds of failures compared to developers because of their different backgrounds, technical perspectives, and biases
- An independent tester can verify, challenge, or disprove assumptions made by stakeholders during specification and implementation of the system

Potential drawbacks of test independence include:

- Isolation from the development team, leading to a lack of collaboration, delays in providing feedback to the development team, or an adversarial relationship with the development team
- Developers may lose a sense of responsibility for quality
- Independent testers may be seen as a bottleneck or blamed for delays in release
- Independent testers may lack some important information (e.g., about the test object)

Many organizations are able to successfully achieve the benefits of test independence while avoiding the drawbacks.

5.1.2 Tasks of a Test Manager and Tester

In this syllabus, two test roles are covered, test managers and testers. The activities and tasks performed by these two roles depend on the project and product context, the skills of the people in the roles, and the organization.

The test manager is tasked with overall responsibility for the test process and successful leadership of the test activities. The test management role might be performed by a professional test manager, or by a project manager, a development manager, or a quality assurance manager. In larger projects or organizations, several test teams may report to a test manager, test coach, or test coordinator, each team being headed by a test leader or lead tester.

Typical test manager tasks may include:

- Develop or review a test policy and test strategy for the organization
- Plan the test activities by considering the context, and understanding the test objectives and risks. This may include selecting test approaches, estimating test time, effort and cost, acquiring resources, defining test levels and test cycles, and planning defect management
- Write and update the test plan(s)
- Coordinate the test plan(s) with project managers, product owners, and others
- Share testing perspectives with other project activities, such as integration planning
- Initiate the analysis, design, implementation, and execution of tests, monitor test progress and results, and check the status of exit criteria (or definition of done)
- Prepare and deliver test progress reports and test summary reports based on the information gathered
- Adapt planning based on test results and progress (sometimes documented in test progress reports, and/or in test summary reports for other testing already completed on the project) and take any actions necessary for test control
- Support setting up the defect management system and adequate configuration management of testware
- Introduce suitable metrics for measuring test progress and evaluating the quality of the testing and the product
- Support the selection and implementation of tools to support the test process, including recommending the budget for tool selection (and possibly purchase and/or support), allocating time and effort for pilot projects, and providing continuing support in the use of the tool(s)
- Decide about the implementation of test environment(s)
- Promote and advocate the testers, the test team, and the test profession within the organization
- Develop the skills and careers of testers (e.g., through training plans, performance evaluations, coaching, etc.)

The way in which the test manager role is carried out varies depending on the software development lifecycle. For example, in Agile development, some of the tasks mentioned above are handled by the Agile team, especially those tasks concerned with the day-to-day testing done within the team, often by a tester working within the team. Some of the tasks that span multiple teams or the entire organization, or

that have to do with personnel management, may be done by test managers outside of the development team, who are sometimes called test coaches. See Black 2009 for more on managing the test process.

Typical tester tasks may include:

- Review and contribute to test plans
- Analyze, review, and assess requirements, user stories and acceptance criteria, specifications, and models for testability (i.e., the test basis)
- Identify and document test conditions, and capture traceability between test cases, test conditions, and the test basis
- Design, set up, and verify test environment(s), often coordinating with system administration and network management
- Design and implement test cases and test procedures
- Prepare and acquire test data
- Create the detailed test execution schedule
- Execute tests, evaluate the results, and document deviations from expected results
- Use appropriate tools to facilitate the test process
- Automate tests as needed (may be supported by a developer or a test automation expert)
- Evaluate non-functional characteristics such as performance efficiency, reliability, usability, security, compatibility, and portability
- Review tests developed by others

People who work on test analysis, test design, specific test types, or test automation may be specialists in these roles. Depending on the risks related to the product and the project, and the software development lifecycle model selected, different people may take over the role of tester at different test levels. For example, at the component testing level and the component integration testing level, the role of a tester is often done by developers. At the acceptance test level, the role of a tester is often done by business analysts, subject matter experts, and users. At the system test level and the system integration test level, the role of a tester is often done by an independent test team. At the operational acceptance test level, the role of a tester is often done by operations and/or systems administration staff.

5.2 Test Planning and Estimation

5.2.1 Purpose and Content of a Test Plan

A test plan outlines test activities for development and maintenance projects. Planning is influenced by the test policy and test strategy of the organization, the development lifecycles and methods being used (see section 2.1), the scope of testing, objectives, risks, constraints, criticality, testability, and the availability of resources.

As the project and test planning progress, more information becomes available and more detail can be included in the test plan. Test planning is a continuous activity and is performed throughout the product's lifecycle. (Note that the product's lifecycle may extend beyond a project's scope to include the maintenance phase.) Feedback from test activities should be used to recognize changing risks so that planning can be adjusted. Planning may be documented in a master test plan and in separate test plans for test levels, such as system testing and acceptance testing, or for separate test types, such as usability testing and performance testing. Test planning activities may include the following and some of these may be documented in a test plan:

- Determining the scope, objectives, and risks of testing
- Defining the overall approach of testing
- Integrating and coordinating the test activities into the software lifecycle activities
- Making decisions about what to test, the people and other resources required to perform the various test activities, and how test activities will be carried out
- Scheduling of test analysis, design, implementation, execution, and evaluation activities, either on particular dates (e.g., in sequential development) or in the context of each iteration (e.g., in iterative development)
- Selecting metrics for test monitoring and control
- Budgeting for the test activities
- Determining the level of detail and structure for test documentation (e.g., by providing templates or example documents)

The content of test plans vary, and can extend beyond the topics identified above. Sample test plans can be found in ISO standard (ISO/IEC/IEEE 29119-3).

5.2.2 Test Strategy and Test Approach

A test strategy provides a generalized description of the test process, usually at the product or organizational level. Common types of test strategies include:

- **Analytical:** This type of test strategy is based on an analysis of some factor (e.g., requirement or risk). Risk-based testing is an example of an analytical approach, where tests are designed and prioritized based on the level of risk.
- **Model-Based:** In this type of test strategy, tests are designed based on some model of some required aspect of the product, such as a function, a business process, an internal structure, or a non-functional characteristic (e.g., reliability). Examples of such models include business process models, state models, and reliability growth models.

- **Methodical:** This type of test strategy relies on making systematic use of some predefined set of tests or test conditions, such as a taxonomy of common or likely types of failures, a list of important quality characteristics, or company-wide look-and-feel standards for mobile apps or web pages.
- **Process-compliant** (or standard-compliant): This type of test strategy involves analyzing, designing, and implementing tests based on external rules and standards, such as those specified by industry-specific standards, by process documentation, by the rigorous identification and use of the test basis, or by any process or standard imposed on or by the organization.
- **Directed** (or consultative): This type of test strategy is driven primarily by the advice, guidance, or instructions of stakeholders, business domain experts, or technology experts, who may be outside the test team or outside the organization itself.
- **Regression-averse:** This type of test strategy is motivated by a desire to avoid regression of existing capabilities. This test strategy includes reuse of existing testware (especially test cases and test data), extensive automation of regression tests, and standard test suites.
- **Reactive:** In this type of test strategy, testing is reactive to the component or system being tested, and the events occurring during test execution, rather than being pre-planned (as the preceding strategies are). Tests are designed and implemented, and may immediately be executed in response to knowledge gained from prior test results. Exploratory testing is a common technique employed in reactive strategies.

An appropriate test strategy is often created by combining several of these types of test strategies. For example, risk-based testing (an analytical strategy) can be combined with exploratory testing (a reactive strategy); they complement each other and may achieve more effective testing when used together.

While the test strategy provides a generalized description of the test process, the test approach tailors the test strategy for a particular project or release. The test approach is the starting point for selecting the test techniques, test levels, and test types, and for defining the entry criteria and exit criteria (or definition of ready and definition of done, respectively). The tailoring of the strategy is based on decisions made in relation to the complexity and goals of the project, the type of product being developed, and product risk analysis. The selected approach depends on the context and may consider factors such as risks, safety, available resources and skills, technology, the nature of the system (e.g., custom-built versus COTS), test objectives, and regulations.

5.2.3 Entry Criteria and Exit Criteria (Definition of Ready and Definition of Done)

In order to exercise effective control over the quality of the software, and of the testing, it is advisable to have criteria which define when a given test activity should start and when the activity is complete. Entry criteria (more typically called definition of ready in Agile development) define the preconditions for undertaking a given test activity. If entry criteria are not met, it is likely that the activity will prove more difficult, more time-consuming, more costly, and more risky. Exit criteria (more typically called definition of done in Agile development) define what conditions must be achieved in order to declare a test level or a set of tests completed. Entry and exit criteria should be defined for each test level and test type, and will differ based on the test objectives.

Typical entry criteria include:

- Availability of testable requirements, user stories, and/or models (e.g., when following a model-based testing strategy)
- Availability of test items that have met the exit criteria for any prior test levels

- Availability of test environment
- Availability of necessary test tools
- Availability of test data and other necessary resources

Typical exit criteria include:

- Planned tests have been executed
- A defined level of coverage (e.g., of requirements, user stories, acceptance criteria, risks, code) has been achieved
- The number of unresolved defects is within an agreed limit
- The number of estimated remaining defects is sufficiently low
- The evaluated levels of reliability, performance efficiency, usability, security, and other relevant quality characteristics are sufficient

Even without exit criteria being satisfied, it is also common for test activities to be curtailed due to the budget being expended, the scheduled time being completed, and/or pressure to bring the product to market. It can be acceptable to end testing under such circumstances, if the project stakeholders and business owners have reviewed and accepted the risk to go live without further testing.

5.2.4 Test Execution Schedule

Once the various test cases and test procedures are produced (with some test procedures potentially automated) and assembled into test suites, the test suites can be arranged in a test execution schedule that defines the order in which they are to be run. The test execution schedule should take into account such factors as prioritization, dependencies, confirmation tests, regression tests, and the most efficient sequence for executing the tests.

Ideally, test cases would be ordered to run based on their priority levels, usually by executing the test cases with the highest priority first. However, this practice may not work if the test cases have dependencies or the features being tested have dependencies. If a test case with a higher priority is dependent on a test case with a lower priority, the lower priority test case must be executed first. Similarly, if there are dependencies across test cases, they must be ordered appropriately regardless of their relative priorities. Confirmation and regression tests must be prioritized as well, based on the importance of rapid feedback on changes, but here again dependencies may apply.

In some cases, various sequences of tests are possible, with differing levels of efficiency associated with those sequences. In such cases, trade-offs between efficiency of test execution versus adherence to prioritization must be made.

5.2.5 Factors Influencing the Test Effort

Test effort estimation involves predicting the amount of test-related work that will be needed in order to meet the objectives of the testing for a particular project, release, or iteration. Factors influencing the test effort may include characteristics of the product, characteristics of the development process, characteristics of the people, and the test results, as shown below.

Product characteristics

- The risks associated with the product
- The quality of the test basis

- The size of the product
- The complexity of the product domain
- The requirements for quality characteristics (e.g., security, reliability)
- The required level of detail for test documentation
- Requirements for legal and regulatory compliance

Development process characteristics

- The stability and maturity of the organization
- The development model in use
- The test approach
- The tools used
- The test process
- Time pressure

People characteristics

- The skills and experience of the people involved, especially with similar projects and products (e.g., domain knowledge)
- Team cohesion and leadership

Test results

- The number and severity of defects found
- The amount of rework required

5.2.6 Test Estimation Techniques

There are a number of estimation techniques used to determine the effort required for adequate testing. Two of the most commonly used techniques are:

- The metrics-based technique: estimating the test effort based on metrics of former similar projects, or based on typical values
- The expert-based technique: estimating the test effort based on the experience of the owners of the testing tasks or by experts

For example, in Agile development, burndown charts are examples of the metrics-based approach as effort is being captured and reported, and is then used to feed into the team's velocity to determine the amount of work the team can do in the next iteration; whereas planning poker is an example of the expert-based approach, as team members are estimating the effort to deliver a feature based on their experience (ISTQB-AT Foundation Level Agile Tester Extension Syllabus).#

Within sequential projects, defect removal models are examples of the metrics-based approach, where volumes of defects and time to remove them are captured and reported, which then provides a basis for estimating future projects of a similar nature; whereas the Wideband Delphi estimation technique is an example of the expert-based approach in which groups of experts provides estimates based on their experience (ISTQB-ATM Advanced Level Test Manager Syllabus).

5.3 Test Monitoring and Control

The purpose of test monitoring is to gather information and provide feedback and visibility about test activities. Information to be monitored may be collected manually or automatically and should be used to assess test progress and to measure whether the test exit criteria, or the testing tasks associated with an Agile project's definition of done, are satisfied, such as meeting the targets for coverage of product risks, requirements, or acceptance criteria.

Test control describes any guiding or corrective actions taken as a result of information and metrics gathered and (possibly) reported. Actions may cover any test activity and may affect any other software lifecycle activity.

Examples of test control actions include:

- Re-prioritizing tests when an identified risk occurs (e.g., software delivered late)
- Changing the test schedule due to availability or unavailability of a test environment or other resources
- Re-evaluating whether a test item meets an entry or exit criterion due to rework

5.3.1 Metrics Used in Testing

Metrics can be collected during and at the end of test activities in order to assess:

- Progress against the planned schedule and budget
- Current quality of the test object
- Adequacy of the test approach
- Effectiveness of the test activities with respect to the objectives

Common test metrics include:

- Percentage of planned work done in test case preparation (or percentage of planned test cases implemented)
- Percentage of planned work done in test environment preparation
- Test case execution (e.g., number of test cases run/not run, test cases passed/failed, and/or test conditions passed/failed)
- Defect information (e.g., defect density, defects found and fixed, failure rate, and confirmation test results)
- Test coverage of requirements, user stories, acceptance criteria, risks, or code
- Task completion, resource allocation and usage, and effort
- Cost of testing, including the cost compared to the benefit of finding the next defect or the cost compared to the benefit of running the next test

5.3.2 Purposes, Contents, and Audiences for Test Reports

The purpose of test reporting is to summarize and communicate test activity information, both during and at the end of a test activity (e.g., a test level). The test report prepared during a test activity may be referred to as a test progress report, while a test report prepared at the end of a test activity may be referred to as a test summary report.

During test monitoring and control, the test manager regularly issues test progress reports for stakeholders. In addition to content common to test progress reports and test summary reports, typical test progress reports may also include:#

- The status of the test activities and progress against the test plan
- Factors impeding progress
- Testing planned for the next reporting period
- The quality of the test object

When exit criteria are reached, the test manager issues the test summary report. This report provides a summary of the testing performed, based on the latest test progress report and any other relevant information.

Typical test progress reports and test summary reports may include:

- Summary of testing performed
- Information on what occurred during a test period
- Deviations from plan, including deviations in schedule, duration, or effort of test activities
- Status of testing and product quality with respect to the exit criteria or definition of done
- Factors that have blocked or continue to block progress
- Metrics of defects, test cases, test coverage, activity progress, and resource consumption. (e.g., as described in 5.3.1)
- Residual risks (see section 5.5)
- Reusable test work products produced

The contents of a test report will vary depending on the project, the organizational requirements, and the software development lifecycle. For example, a complex project with many stakeholders or a regulated project may require more detailed and rigorous reporting than a quick software update. As another example, in Agile development, test progress reporting may be incorporated into task boards, defect summaries, and burndown charts, which may be discussed during a daily stand-up meeting (see ISTQB-AT Foundation Level Agile Tester Extension Syllabus).

In addition to tailoring test reports based on the context of the project, test reports should be tailored based on the report's audience. The type and amount of information that should be included for a technical audience or a test team may be different from what would be included in an executive summary report. In the first case, detailed information on defect types and trends may be important. In the latter case, a high-level report (e.g., a status summary of defects by priority, budget, schedule, and test conditions passed/failed/not tested) may be more appropriate.

ISO standard (ISO/IEC/IEEE 29119-3) refers to two types of test reports, test progress reports and test completion reports (called test summary reports in this syllabus), and contains structures and examples for each type.

5.4 Configuration Management

The purpose of configuration management is to establish and maintain the integrity of the component or system, the testware, and their relationships to one another through the project and product lifecycle.

To properly support testing, configuration management may involve ensuring the following:

- All test items are uniquely identified, version controlled, tracked for changes, and related to each other
- All items of testware are uniquely identified, version controlled, tracked for changes, related to each other and related to versions of the test item(s) so that traceability can be maintained throughout the test process
- All identified documents and software items are referenced unambiguously in test documentation

During test planning, configuration management procedures and infrastructure (tools) should be identified and implemented.

5.5 Risks and Testing

5.5.1 Definition of Risk

Risk involves the possibility of an event in the future which has negative consequences. The level of risk is determined by the likelihood of the event and the impact (the harm) from that event.

5.5.2 Product and Project Risks

Product risk involves the possibility that a work product (e.g., a specification, component, system, or test) may fail to satisfy the legitimate needs of its users and/or stakeholders. When the product risks are associated with specific quality characteristics of a product (e.g., functional suitability, reliability, performance efficiency, usability, security, compatibility, maintainability, and portability), product risks are also called quality risks. Examples of product risks include:

- Software might not perform its intended functions according to the specification
- Software might not perform its intended functions according to user, customer, and/or stakeholder needs
- A system architecture may not adequately support some non-functional requirement(s)
- A particular computation may be performed incorrectly in some circumstances
- A loop control structure may be coded incorrectly
- Response-times may be inadequate for a high-performance transaction processing system
- User experience (UX) feedback might not meet product expectations

Project risk involves situations that, should they occur, may have a negative effect on a project's ability to achieve its objectives. Examples of project risks include:

- Project issues:
  - Delays may occur in delivery, task completion, or satisfaction of exit criteria or definition of done
  - Inaccurate estimates, reallocation of funds to higher priority projects, or general cost-cutting across the organization may result in inadequate funding
  - Late changes may result in substantial re-work
- Organizational issues:
  - Skills, training, and staff may not be sufficient
  - Personnel issues may cause conflict and problems
  - Users, business staff, or subject matter experts may not be available due to conflicting business priorities
- Political issues:
  - Testers may not communicate their needs and/or the test results adequately
  - Developers and/or testers may fail to follow up on information found in testing and reviews (e.g., not improving development and testing practices)
  - There may be an improper attitude toward, or expectations of, testing (e.g., not appreciating the value of finding defects during testing)
- Technical issues:
  - Requirements may not be defined well enough
  - The requirements may not be met, given existing constraints
  - The test environment may not be ready on time
  - Data conversion, migration planning, and their tool support may be late
  - Weaknesses in the development process may impact the consistency or quality of project work products such as design, code, configuration, test data, and test cases
  - Poor defect management and similar problems may result in accumulated defects and other technical debt
- Supplier issues:
  - A third party may fail to deliver a necessary product or service, or go bankrupt
  - Contractual issues may cause problems to the project

Project risks may affect both development activities and test activities. In some cases, project managers are responsible for handling all project risks, but it is not unusual for test managers to have responsibility for test-related project risks.

5.5.3 Risk-based Testing and Product Quality

Risk is used to focus the effort required during testing. It is used to decide where and when to start testing and to identify areas that need more attention. Testing is used to reduce the probability of an adverse event occurring, or to reduce the impact of an adverse event. Testing is used as a risk mitigation activity, to provide feedback about identified risks, as well as providing feedback on residual (unresolved) risks.

A risk-based approach to testing provides proactive opportunities to reduce the levels of product risk. It involves product risk analysis, which includes the identification of product risks and the assessment of each risk's likelihood and impact. The resulting product risk information is used to guide test planning, the specification, preparation and execution of test cases, and test monitoring and control. Analyzing product risks early contributes to the success of a project.

In a risk-based approach, the results of product risk analysis are used to:

- Determine the test techniques to be employed
- Determine the particular levels and types of testing to be performed (e.g., security testing, accessibility testing)
- Determine the extent of testing to be carried out
- Prioritize testing in an attempt to find the critical defects as early as possible
- Determine whether any activities in addition to testing could be employed to reduce risk (e.g., providing training to inexperienced designers)

Risk-based testing draws on the collective knowledge and insight of the project stakeholders to carry out product risk analysis. To ensure that the likelihood of a product failure is minimized, risk management activities provide a disciplined approach to:

- Analyze (and re-evaluate on a regular basis) what can go wrong (risks)
- Determine which risks are important to deal with
- Implement actions to mitigate those risks
- Make contingency plans to deal with the risks should they become actual events

In addition, testing may identify new risks, help to determine what risks should be mitigated, and lower uncertainty about risks.

5.6 Defect Management

Since one of the objectives of testing is to find defects, defects found during testing should be logged. The way in which defects are logged may vary, depending on the context of the component or system being tested, the test level, and the software development lifecycle model. Any defects identified should be investigated and should be tracked from discovery and classification to their resolution (e.g., correction of the defects and successful confirmation testing of the solution, deferral to a subsequent release, acceptance as a permanent product limitation, etc.). In order to manage all defects to resolution, an organization should establish a defect management process which includes a workflow and rules for classification. This process must be agreed with all those participating in defect management, including designers, developers, testers, and product owners. In some organizations, defect logging and tracking may be very informal.

During the defect management process, some of the reports may turn out to describe false positives, not actual failures due to defects. For example, a test may fail when a network connection is broken or times out. This behavior does not result from a defect in the test object, but is an anomaly that needs to be investigated. Testers should attempt to minimize the number of false positives reported as defects.

Defects may be reported during coding, static analysis, reviews, dynamic testing, or use of a software product. Defects may be reported for issues in code or working systems, or in any type of documentation including requirements, user stories and acceptance criteria, development documents, test documents, user manuals, or installation guides. In order to have an effective and efficient defect management process, organizations may define standards for the attributes, classification, and workflow of defects.

Typical defect reports have the following objectives:

- Provide developers and other parties with information about any adverse event that occurred, to enable them to identify specific effects, to isolate the problem with a minimal reproducing test, and to correct the potential defect(s), as needed or to otherwise resolve the problem
- Provide test managers a means of tracking the quality of the work product and the impact on the testing (e.g., if a lot of defects are reported, the testers will have spent a lot of time reporting them instead of running tests, and there will be more confirmation testing needed)
- Provide ideas for development and test process improvement

A defect report filed during dynamic testing typically includes:

- An identifier
- A title and a short summary of the defect being reported
- Date of the defect report, issuing organization, and author
- Identification of the test item (configuration item being tested) and environment
- The development lifecycle phase(s) in which the defect was observed
- A description of the defect to enable reproduction and resolution, including logs, database dumps screenshots, or recordings (if found during test execution)
- Expected and actual results
- Scope or degree of impact (severity) of the defect on the interests of stakeholder(s)
- Urgency/priority to fix

- State of the defect report (e.g., open, deferred, duplicate, waiting to be fixed, awaiting confirmation testing, re-opened, closed)
- Conclusions, recommendations and approvals
- Global issues, such as other areas that may be affected by a change resulting from the defect
- Change history, such as the sequence of actions taken by project team members with respect to the defect to isolate, repair, and confirm it as fixed
- References, including the test case that revealed the problem

Some of these details may be automatically included and/or managed when using defect management tools, e.g., automatic assignment of an identifier, assignment and update of the defect report state during the workflow, etc. Defects found during static testing, particularly reviews, will normally be documented in a different way, e.g., in review meeting notes.

An example of the contents of a defect report can be found in ISO standard (ISO/IEC/IEEE 29119-3) (which refers to defect reports as incident reports).