5.ManagingtheTestActivities–335minutes

Keywords

defectmanagement,defectreport, entrycriteria,exitcriteria,product risk,projectrisk, risk,riskanalysis, risk assessment, risk control, risk identification, risk level, risk management, risk mitigation, risk monitoring, risk-based testing, test approach, test completion report, test control, test monitoring, test plan, test planning, test progress report, test pyramid, testing quadrants

LearningObjectivesforChapter5:

5.1 TestPlanning

FL-5.1.1 (K2)Exemplifythepurposeandcontentofatestplan

FL-5.1.2 (K1)Recognizehow atesteraddsvaluetoiterationandreleaseplanning FL-

5.1.3 (K2) Compare and contrast entry criteria and exit criteria

FL-5.1.4 (K3)Useestimationtechniquestocalculatetherequiredtesteffort FL-

5.1.5 (K3) Apply test case prioritization

FL-5.1.6 (K1)Recalltheconceptsofthetestpyramid

FL-5.1.7 (K2)Summarizethetestingquadrantsandtheirrelationshipswithtestlevelsandtesttypes

5.2 Risk Management

FL-5.2.1 (K1)Identifyrisklevel byusing risklikelihoodandriskimpact FL-

5.2.2 (K2) Distinguish between project risks and product risks

FL-5.2.3 (K2)Explainhow productrisk analysismayinfluencethoroughness andscopeoftesting FL-

5.2.4 (K2) Explain what measures can be taken in response to analyzed product risks

5.3 TestMonitoring,TestControlandTestCompletion

FL-5.3.1 (K1)Recallmetricsusedfortesting

FL-5.3.2 (K2)Summarizethepurposes, content,andaudiencesfortestreports FL-

5.3.3 (K2) Exemplify how to communicate the status of testing

5.4 ConfigurationManagement

FL-5.4.1 (K2)Summarizehowconfigurationmanagementsupportstesting

5.5 DefectManagement

FL-5.5.1 (K3)Prepareadefectreport

5.1. TestPlanning

5.1.1. PurposeandContentofaTestPlan

Atestplandescribestheobjectives,resourcesandprocessesfora testproject.Atestplan:

- Documentsthemeansandscheduleforachievingtestobjectives
- Helpstoensurethattheperformedtestactivitieswillmeettheestablishedcriteria
- Servesasameansofcommunicationwithteammembersandotherstakeholders
- Demonstratesthattestingwilladheretothe existingtest policy andteststrategy (orexplainswhy the testing will deviate from them)

Test planning guides the testers' thinking and forces the testers to confront the future challenges related torisks,schedules,people, tools,costs,effort, etc. Theprocessofpreparinga testplanisausefulwayto think through the efforts needed to achieve the test project objectives.

Thetypicalcontentof atestplanincludes:

- Contextof testing(e.g.,scope,testobjectives,constraints,testbasis)
- Assumptionsandconstraintsofthetestproject
- Stakeholders(e.g.,roles,responsibilities,relevancetotesting,hiringandtrainingneeds)
- Communication(e.g.,formsandfrequencyofcommunication,documentationtemplates)
- Riskregister(e.g.,productrisks, projectrisks)
- Testapproach(e.g.,testlevels,testtypes, testtechniques,testdeliverables,entry criteriaand exit criteria, independence of testing, metrics to be collected, test data requirements, test environment requirements, deviations from the organizational test policy and test strategy)
- Budgetandschedule

Moredetailsaboutthetestplananditscontentcanbefoundinthe ISO/IEC/IEEE29119-3standard.

5.1.2. Tester'sContributiontoIterationandReleasePlanning

IniterativeSDLCs,typicallytwokindsofplanningoccur:releaseplanninganditerationplanning.

Release planning looks ahead to the release of a product, defines and re-defines the product backlog, and may involve refining larger user stories into a set of smaller user stories. It also serves as the basis forthe testapproachandtest planacrossalliterations.Testersinvolvedinrelease planningparticipatein writing testable user stories and acceptance criteria (see section 4.5), participate in project and quality risk analyses (see section 5.2), estimate test effort associated with user stories (see section 5.1.4), determine the test approach, and plan the testing for the release.

Iteration planning looks ahead to the end of a single iteration and is concerned with the iteration backlog. Testers involvedin iterationplanning participateinthedetailedrisk analysis ofuserstories, determinethe testabilityof userstories, break downuserstoriesintotasks(particularlytestingtasks), estimatetesteffort for all testing tasks, and identify and refine functional and non-functional aspects of the test object.

5.1.3. EntryCriteriaandExitCriteria

Entry criteria define the preconditions for undertaking a given activity. If entry criteria are not met, it is likelythatthe activitywillprovetobemoredifficult,time-consuming,costly,andriskier.Exitcriteriadefine what must be achieved in order to declare an activity completed. Entry criteria and exit criteria should be defined for each test level, and will differ based on the test objectives.

Typicalentrycriteriainclude:availabilityofresources(e.g.,people,tools, environments,test data, budget, time), availability of testware (e.g., test basis, testable requirements, user stories, test cases), and initial quality level of a test object (e.g., all smoke tests have passed).

Typical exit criteria include: measures of thoroughness (e.g., achieved level of coverage, number of unresolveddefects,defectdensity,numberoffailedtest cases),andcompletioncriteria (e.g.,planned tests have been executed, static testing has been performed, all defects found are reported, all regression tests are automated).

Running out of time or budget can also be viewed as valid exit criteria. Even without other exit criteria beingsatisfied,itcanbe acceptableto endtestingunder suchcircumstances,if thestakeholdershave reviewed and accepted the risk to go live without further testing.

In Agile software development, exit criteria are often called Definition of Done, defining the team's objectivemetricsforareleasableitem.Entrycriteriathat auserstorymustfulfilltostartthedevelopment and/or testing activities are called Definition of Ready.

5.1.4. Estimation Techniques

Testeffortestimation involvespredictingthe amountof test-relatedwork neededto meet theobjectivesof atestproject.It isimportant tomakeitcleartothestakeholdersthatthe estimateisbasedon anumberof assumptionsand isalwayssubjecttoestimationerror.Estimationforsmall tasksisusuallymoreaccurate than for the large ones. Therefore, when estimating a large task, it can be decomposed into a set of smaller tasks which then in turn can be estimated.

Inthissyllabus,thefollowingfourestimationtechniquesaredescribed.

Estimationbasedonratios.Inthis metrics-basedtechnique,figuresarecollectedfromprevious projects within the organization, which makes it possible to derive “standard” ratios for similar projects. The ratios of an organization's own projects (e.g., taken from historical data) are generally the best source to use in the estimation process. These standard ratios can then be used to estimate the test effort for the new project. For example, if in the previous project the development-to-test effort ratio was 3:2, and in the current project the developmenteffortis expected tobe 600person-days, thetest effort canbeestimated to be 400 person-days.

Extrapolation. In this metrics-based technique, measurements are made as early as possible in the current projecttogatherthe data. Havingenoughobservations,theeffortrequiredforthe remainingwork canbeapproximatedbyextrapolatingthisdata (usuallybyapplyinga mathematicalmodel). Thismethod is very suitable in iterative SDLCs. For example, the team may extrapolate the test effort in the forthcoming iteration as the averaged effort from the last three iterations.

WidebandDelphi.Inthis iterative,expert-basedtechnique,expertsmakeexperience-basedestimations. Eachexpert,inisolation,estimatestheeffort. The results arecollectedand ifthereardeviationsthatare out of range of the agreed upon boundaries, the experts discuss their current estimates. Each expert is thenaskedto makea newestimation based onthatfeedback,againinisolation.Thisprocessisrepeated until a consensus is reached. Planning Poker is a variant of Wideband Delphi, commonly used in Agile softwaredevelopment.InPlanningPoker,estimatesareusuallymadeusingcardswithnumbersthat represent the effort size.

Three-point estimation. In this expert-based technique, three estimations are made by the experts: the mostoptimisticestimation(a),themostlikelyestimation(m)and the mostpessimisticestimation (b). The final estimate (E) is their weighted arithmetic mean. In the most popular version of this technique, the estimate is calculated as E = (a + 4*m + b) / 6. The advantage of this technique is that it allows the experts to calculate the measurement error: SD = (b – a) / 6. For example, if the estimates (in person-hours) are: a=6, m=9 and b=18, then the final estimation is 10±2 person-hours (i.e., between 8 and 12 person-hours), because E = (6 + 4*9 + 18) / 6 = 10 and SD = (18 – 6) / 6 = 2.

See(Kan2003,Koomen2006,Westfall2009)fortheseandmanyothertestestimationtechniques.

5.1.5. TestCase Prioritization

Oncethe testcasesand testprocedures arespecifiedandassembledinto testsuites,thesetestsuites can be arranged in a test execution schedule that defines the order in which they are to be run. When prioritizing test cases, different factors can be taken into account. The most commonly used test case prioritization strategies are as follows:

- Risk-basedprioritization,wheretheorderoftest executionisbased ontheresultsof riskanalysis (see section 5.2.3). Test cases covering the most important risks are executed first.
- Coverage-based prioritization, where the order of test execution is based on coverage (e.g., statement coverage). Test cases achieving the highest coverage are executed first. In another variant,calledadditionalcoverageprioritization,thetestcaseachievingthe highest coverageis executed first; each subsequent test case is the one that achieves the highest additional coverage.
- Requirements-based prioritization, where the order of test execution is based on the priorities of therequirementstracedbacktothecorrespondingtest cases.Requirement prioritiesaredefined by stakeholders. Test cases related to the most important requirements are executed first.

Ideally, test cases would be ordered to run based on their priority levels, using, for example, one of the above-mentioned prioritization strategies. However, this practice may not work if the test cases or the featuresbeingtestedhave dependencies.Ifa testcasewithahigherpriorityisdependent onatestcase with a lower priority, the lower priority test case must be executed first.

Theorder oftest executionmustalsotakeinto accounttheavailabilityof resources. Forexample,the requiredtest tools,testenvironmentsorpeoplethatmayonlybeavailableforaspecifictimewindow.

5.1.6. TestPyramid

The test pyramid is a model showing that different tests may have different granularity. The test pyramid modelsupports theteamin testautomationandintesteffortallocationbyshowingthatdifferentgoals are supportedby differentlevels oftest automation. Thepyramidlayers represent groupsoftests. Thehigher the layer, the lower the test granularity, test isolation and test execution time. Tests in the bottom layer are small, isolated, fast, and check a small piece of functionality, so usually a lot of them are needed to achieve a reasonable coverage. The top layer represents complex, high-level, end-to-end tests. These high-level tests are generally slower than the tests from the lower layers, and they typically check a large piece of functionality, so usually just a few of them are needed to achieve a reasonable coverage. The number and naming of the layers may differ. For example, the original test pyramid model (Cohn 2009) defines three layers: “unit tests”, “service tests” and “UI tests”. Another popular model defines unit (component)tests,integration (componentintegration)tests,andend-to-endtests. Othertestlevels(see section 2.2.1) can also be used.

5.1.7. Testing Quadrants

Thetestingquadrants,definedbyBrianMarick (Marick 2003,Crispin 2008),groupthetestlevelswiththe appropriate test types, activities, test techniques and work products in the Agile software development. The model supports test management in visualizing these to ensure that all appropriate test types and test levels are included in the SDLC and in understanding that some test types are more relevant to certaintestlevelsthanothers. This modelalsoprovidesa waytodifferentiateanddescribethetypes of tests to all stakeholders, including developers, testers, and business representatives.

In this model, tests can be business facing or technology facing. Tests can also support the team (i.e., guidethedevelopment)orcritiquetheproduct(i.e.,measureitsbehavior againstthe expectations). The combination of these two viewpoints determines the four quadrants:

- Quadrant Q1 (technology facing, support the team). This quadrant contains component and componentintegrationtests.Thesetests shouldbeautomatedandincludedintheCI process.
- Quadrant Q2 (business facing, support the team). This quadrant contains functional tests, examples,userstorytests,userexperienceprototypes, APItesting,andsimulations.Thesetests check the acceptance criteria and can be manual or automated.
- QuadrantQ3(businessfacing,critiquetheproduct).Thisquadrantcontainsexploratorytesting, usability testing, user acceptance testing. These tests are user-oriented and often manual.
- QuadrantQ4 (technologyfacing,critiquetheproduct).Thisquadrantcontainssmoketestsand non-functional tests (except usability tests). These tests are often automated.

5.2. RiskManagement

Organizationsfacemany internalandexternalfactorsthatmakeit uncertainwhetherandwhentheywill achieve their objectives (ISO 31000). Risk management allows the organizations to increase the likelihood of achieving objectives, improve the quality of their products and increase the stakeholders’ confidence and trust.

Themainriskmanagementactivities are:

- Riskanalysis(consistingof riskidentificationandriskassessment;seesection5.2.3)
- Riskcontrol(consisting ofriskmitigationandriskmonitoring;seesection5.2.4)

Thetestapproach,inwhichtestactivities areselected, prioritized,andmanagedbasedon riskanalysis and risk control, is called risk-based testing.

5.2.1. RiskDefinition andRiskAttributes

Riskisapotentialevent,hazard,threat,orsituationwhoseoccurrencecausesanadverseeffect.Arisk can be characterized by two factors:

- Risklikelihood–theprobabilityoftheriskoccurrence(greaterthanzeroandlessthan one)
- Riskimpact(harm)–theconsequencesofthisoccurrence

Thesetwofactorsexpresstherisklevel, whichisa measureforthe risk.Thehighertherisklevel,the more important is its treatment.

5.2.2. ProjectRisksandProductRisks

Insoftwaretestingone isgenerallyconcernedwithtwo typesofrisks: projectrisksandproductrisks.

Projectrisksarerelated tothemanagement andcontrol oftheproject.Projectrisksinclude:

- Organizationalissues(e.g.,delaysinworkproductsdeliveries,inaccurateestimates,cost-cutting)
- Peopleissues(e.g.,insufficientskills,conflicts,communicationproblems,shortageofstaff)
- Technicalissues(e.g.,scopecreep,poortoolsupport)
- Supplierissues (e.g.,third-partydeliveryfailure,bankruptcyofthesupportingcompany)

Projectrisks,whenthey occur, mayhaveanimpact ontheproject schedule,budget orscope,which affects the project's ability to achieve its objectives.

Product risks are related to the product quality characteristics (e.g., described in the ISO 25010 quality model). Examples of product risks include: missing or wrong functionality, incorrect calculations, runtime errors,poorarchitecture,inefficientalgorithms,inadequateresponsetime,pooruserexperience,security vulnerabilities. Product risks, when they occur, may result in various negative consequences, including:

- Userdissatisfaction
- Lossofrevenue,trust,reputation
- Damagetothirdparties
- Highmaintenancecosts,overloadofthehelpdesk
- Criminalpenalties
- Inextremecases,physical damage,injuriesoreven death

5.2.3. ProductRiskAnalysis

Fromatestingperspective,thegoalof productrisk analysisistoprovideanawarenessof productriskin order tofocusthe testing effortinaway that minimizestheresidual level of productrisk.Ideally, product risk analysis begins early in the SDLC.

Product risk analysis consists of risk identification and risk assessment. Risk identification is about generatinga comprehensivelistof risks.Stakeholderscanidentify risksbyusingvarioustechniques and tools, e.g., brainstorming, workshops, interviews, or cause-effect diagrams. Risk assessment involves: categorization of identified risks, determining their risk likelihood, risk impact and level, prioritizing, and proposing ways to handle them. Categorization helps in assigning mitigation actions, because usually risks falling into the same category can be mitigated using a similar approach.

Riskassessmentcanuseaquantitativeorqualitative approach, oramixofthem.Inthe quantitative approach the risk level is calculated as the multiplication of risk likelihood and risk impact. In the qualitative approach the risk level can be determined using a risk matrix.

Productrisk analysismayinfluencethethoroughnessandscopeoftesting.Itsresultsareusedto:

- Determinethescopeoftestingtobecarriedout
- Determinetheparticulartest levelsandpropose testtypestobeperformed
- Determinethetesttechniquestobe employedandthecoveragetobe achieved
- Estimatethe testeffort requiredforeachtask
- Prioritizetestinginan attemptto findthecritical defectsasearlyaspossible
- Determinewhetheranyactivitiesinadditiontotestingcouldbeemployed toreducerisk

5.2.4. ProductRiskControl

Productriskcontrolcomprisesallmeasuresthat are takeninresponsetoidentifiedandassessedproduct risks. Product risk control consists of risk mitigation and risk monitoring. Risk mitigation involves implementingtheactions proposedinriskassessment to reduce therisklevel.Theaim ofrisk monitoring is to ensure that the mitigation actions are effective, to obtain further information to improve risk assessment, and to identify emerging risks.

Withrespect toproduct riskcontrol,onceariskhasbeen analyzed,several responseoptionstoriskare possible,e.g.,risk mitigation bytesting, risk acceptance,risktransfer, or contingencyplan(Veenendaal 2012). Actions that can be taken to mitigate the product risks by testing are as follows:

- Selectthetesterswith therightlevelofexperienceandskills,suitableforagivenrisktype
- Applyanappropriatelevelofindependenceoftesting
- Conductreviewsandperformstaticanalysis
- Applytheappropriatetesttechniquesandcoveragelevels
- Applytheappropriatetesttypesaddressing theaffectedqualitycharacteristics
- Performdynamictesting,includingregressiontesting

5.3. TestMonitoring,Test ControlandTest Completion

Test monitoring is concerned with gathering information about testing. This information is used to assess testprogress andtomeasurewhetherthe testexitcriteria orthe testtasks associatedwiththe exitcriteria are satisfied, such as meeting the targets for coverage of product risks, requirements, or acceptance criteria.

Testcontrolusestheinformation from testmonitoringto provide,in aformofthecontrol directives, guidance and the necessary corrective actions to achieve the most effective and efficient testing. Examples of control directives include:

- Reprioritizingtestswhenanidentifiedriskbecomes anissue
- Re-evaluatingwhetheratestitemmeetsentrycriteriaorexitcriteriaduetorework
- Adjustingthetestscheduletoaddressadelayin thedeliveryofthetestenvironment
- Addingnewresourceswhenandwhere needed

Test completion collects data from completed test activities to consolidate experience, testware, and any other relevantinformation.Testcompletion activitiesoccuratprojectmilestonessuchaswhena testlevel is completed,an agileiterationis finished, atest projectis completed(orcancelled), a softwaresystemis released, or a maintenance release is completed.

5.3.1. MetricsusedinTesting

Testmetricsaregatheredtoshowprogress againstthe plannedschedule and budget,thecurrent quality of the test object, and the effectiveness of the test activities with respect to the objectives or an iteration goal. Test monitoring gathers a variety of metrics to support the test control and test completion.

Commontestmetricsinclude:

- Projectprogressmetrics(e.g.,taskcompletion,resourceusage,testeffort)
- Testprogressmetrics(e.g.,testcaseimplementationprogress,testenvironmentpreparation progress, number of test cases run/not run, passed/failed, test execution time)
- Productqualitymetrics(e.g.,availability,responsetime,meantimetofailure)
- Defectmetrics(e.g.,numberandprioritiesofdefects found/fixed,defectdensity,defectdetection percentage)
- Riskmetrics(e.g.,residual risklevel)
- Coveragemetrics(e.g.,requirementscoverage,codecoverage)
- Costmetrics (e.g.,costoftesting,organizationalcostofquality)

5.3.2. Purpose,ContentandAudienceforTestReports

Test reporting summarizes and communicates test information during and after testing. Test progress reports support the ongoing control of the testing and must provide enough information to make modifications to the test schedule, resources, or test plan, when such changes are needed due to deviationfromtheplanorchangedcircumstances.Testcompletion reportssummarizeaspecificstageof testing (e.g., test level, test cycle, iteration) and can give information for subsequent testing.

Duringtest monitoring andcontrol, thetestteamgeneratestestprogress reportsforstakeholderstokeep them informed. Test progress reports are usually generated on a regular basis (e.g., daily, weekly, etc.) and include:

- Testperiod
- Testprogress(e.g.,aheadorbehindschedule),includinganynotabledeviations
- Impedimentsfortesting,andtheirworkarounds
- Testmetrics(seesection5.3.1for examples)
- Newandchangedriskswithintestingperiod
- Testingplannedforthenextperiod

A test completion report is prepared during test completion, when a project, test level, or test type is complete andwhen,ideally,itsexitcriteria havebeen met.Thisreportusestestprogress reportsand other data. Typical test completion reports include:

- Testsummary
- Testingandproductqualityevaluationbasedon theoriginaltestplan(i.e.,testobjectives andexit criteria)
- Deviationsfromthetestplan(e.g.,differencesfromtheplannedschedule,duration,andeffort).
- Testingimpedimentsandworkarounds
- Testmetricsbasedon testprogressreports
- Unmitigatedrisks,defectsnotfixed
- Lessonslearnedthatarerelevanttothetesting

Different audiencesrequire differentinformationinthereports,andinfluencethe degree offormalityand the frequency of reporting. Reporting on test progress to others in the same team is often frequent and informal, while reporting on testing for a completed project follows a set template and occurs only once.

TheISO/IEC/IEEE29119-3standardincludestemplates andexamplesfortest progress reports(called test status reports) and test completion reports.

5.3.3. CommunicatingtheStatusofTesting

The best means of communicating test status varies, depending on test management concerns, organizationalteststrategies,regulatorystandards,or,inthecaseofself-organizingteams(seesection 1.5.2), on the team itself. The options include:

- Verbalcommunicationwithteammembersandotherstakeholders
- Dashboards(e.g.,CI/CDdashboards,taskboards,andburn-downcharts)
- Electroniccommunicationchannels(e.g.,email,chat)
- Onlinedocumentation
- Formaltestreports(seesection 5.3.2)

One or more of these options can be used. More formal communication may be more appropriate for distributedteamswheredirectface-to-facecommunicationisnotalways possible duetogeographical distance or time differences. Typically, different stakeholders are interested in different types of information, so communication should be tailored accordingly.

5.4. ConfigurationManagement

In testing, configuration management (CM) provides a discipline for identifying, controlling, and tracking workproductssuchastestplans, teststrategies,testconditions,testcases,testscripts, testresults,test logs, and test reports as configuration items.

For a complex configuration item (e.g., a test environment), CM records the items it consists of, their relationships,andversions.Iftheconfigurationitemisapproved fortesting,it becomesabaseline and can only be changed through a formal change control process.

Configurationmanagementkeeps arecordofchangedconfigurationitems whenanewbaselineis created. It is possible to revert to a previous baseline to reproduce previous test results.

Toproperlysupporttesting,CMensuresthefollowing:

- All configuration items, including test items (individual parts of the test object), are uniquely identified,versioncontrolled,trackedforchanges, andrelatedtootherconfigurationitems sothat traceability can be maintained throughout the test process
- Allidentifieddocumentation andsoftwareitemsarereferencedunambiguouslyintest documentation

Continuous integration, continuous delivery, continuous deployment and the associated testing are typicallyimplemented aspartof anautomatedDevOps pipeline(seesection2.1.4),inwhichautomated CM is normally included.

5.5.DefectManagement

Since one of the major test objectives is to find defects, an established defect management process is essential. Although we refer to "defects" here, the reported anomalies may turn out to be real defects or something else (e.g., false positive, change request) - this is resolved during the process of dealing with the defect reports. Anomalies may be reported during any phase of the SDLC and the form depends on the SDLC. At a minimum, the defect management process includes a workflow for handling individual anomalies from their discovery to their closure and rules for their classification. The workflow typically comprises activities to log the reported anomalies, analyze and classify them, decide on a suitable response such as to fix or keep it as it is and finally to close the defect report. The process must be followedbyallinvolvedstakeholders.It isadvisabletohandledefectsfromstatic testing (especiallystatic analysis) in a similar way.

Typicaldefectreports havethefollowingobjectives:

- Providethoseresponsibleforhandlingandresolvingreporteddefectswithsufficientinformation to resolve the issue
- Provideameansoftrackingthequalityofthe workproduct
- Provideideasforimprovement ofthedevelopmentandtestprocess A

defect report logged during dynamic testing typically includes:

- Uniqueidentifier
- Titlewithashortsummaryoftheanomalybeingreported
- Datewhentheanomalywasobserved,issuingorganization,and author,includingtheirrole
- Identificationofthetest objectandtestenvironment
- Contextof thedefect (e.g., testcasebeingrun,test activitybeingperformed,SDLCphase,and other relevant information such as the test technique, checklist or test data being used)

- Descriptionofthefailure toenablereproductionand resolutionincludingthe stepsthatdetected the anomaly, and any relevant test logs, database dumps, screenshots, or recordings
- Expectedresultsandactualresults
- Severityofthedefect(degreeofimpact)ontheinterestsofstakeholdersorrequirements
- Prioritytofix
- Statusofthedefect(e.g.,open, deferred, duplicate,waitingtobefixed, awaitingconfirmation testing, re-opened, closed, rejected)
- References(e.g.,tothetestcase)

Some of this data may be automatically included when using defect management tools (e.g., identifier, date,author andinitialstatus).Document templatesforadefectreportand exampledefectreports canbe found in the ISO/IEC/IEEE 29119-3 standard, which refers to defect reports as incident reports.