



ソフトウェア開発 テスト技術チェックシート
Software Subcontract Test management and test case check list

No.	Category	Item	Check point	Answer and Evidence (Pre)	Comments				
TM-01	テスト管理 Test Management	実施しているテストの名称と定義を明確にする Clarify the name and definition of the test being conducted	<p>自社で使っているテストの名称と定義(目的、実施方法、主管部門等)を明確にする。例えば以下のようなテストの名称に対する定義を明確にする。</p> <p>1) 単体テスト(パス確認や単機能確認、コーダ実自身が実施、設計開発部門が主管等) 2) 結合テスト(結合状態での機能確認や性能確認や相互動作確認、選任のテスト担当者が実施、設計開発部門が主管等) 3) システムテスト(全機能の機能/性能/信頼性/耐久性/異常回復性/保守性/大規模環境動作の確認、QA部門テスト担当者が実施、QA部門が主管等) 4) 妥当性テスト(製品の妥当性確認。第三者検証機関で実施、QA部門が主管等)</p> <p>Define the name and definition (purpose, implementation method, department in charge, etc.) of the test used in-house. For example, the following definitions for test names are clarified.</p> <p>1) Tantaitestuto (pasu kakunin ya tan kinō kakunin, kōda jitsu jishin ga jissshi, sekkei kaihatu bumon ga shukan-tō) 2) ketsugō tesuto (ketsugō jōtai de no kinō kakunin ya seinō kakunin ya sōgo dōsa kakunin, sen'nin no tesuto tantōsha ga jissshi, sekkei kaihatu bumon ga shukan-tō) 3) shisutemu tesuto (zen kinō no kinō/ seinō/ shinrai-sei/ taikyū-sei/ ijō kaifuku-sei/ hoshu-sei/ ōkibo kankyō dōsa no kakunin, QA bumon tesuto tantōsha ga jissshi, QA bumon ga shukan-tō) 4) datōsei tesuto (seihin no datōsei kakunin. Daisansha kenshō kikan de jissshi, QA bumon ga shukan-tō)</p> <p>さらに表示 219/5000</p> <p>1) Unit test (pass check, single function check, coder's own conduct, design and development department take charge etc) 2) Bonding test (performed by the person in charge of testing by the person in charge of testing of the functional status and performance in the combined state, mutual operation check, appointment, etc.) 3) System test (all functions function / performance / reliability / durability / abnormal recoverability / maintainability / large scale operation check, QA department test person in charge, QA department in charge) 4) Relevance test (Product validation, conducted by a third party verification organization, led by QA department, etc.)</p>						
TM-02	テスト管理 Test Management	実施しているテストとテストの目的との対応を明確にする Clarify the correspondence between the test being conducted and the purpose of the test	<p>16種類のシステムテストカテゴリ(Test Category sheet)で定義されている目的を持ったテストを、TS-01項で記載したテストのどの名称のテストで実施しているのかの対応を確認する。(以下の表を完成させる、○はそのカテゴリの津とが実施される事を示す) 参考: Reference(Test Category)シートに16種類システムテストカテゴリについての詳細を記載してある。</p> <p>Check the correspondence between the tests with the purpose defined in the 16 system test categories (Test Category sheet) and the name of the test described in the section TS-01. (Complete the following table, ○ indicates that the category Tsu will be implemented) Reference: "The Reference (Test Category) "" sheet gives details about the 16 system test categories.</p> <table border="1" style="width: 100%; text-align: center;"> <tr> <td>Unit test</td> <td>Integration test</td> <td>Sytem test</td> <td>QA test</td> </tr> </table> <p>1) Facility test (Normal case) 1') Facility test (Abnormal case) 2) Configuration test 3) Security test 4) Documentation test 5) Usability test 6) Performance test 7) Storege test 8) Stress test 9) Volume test 10) Reliability test 11) Longrun test 12) Recovery test 13) Installability test 14) Compatibility test 15) Serviceability test 16) Procecedure test</p>	Unit test	Integration test	Sytem test	QA test		
Unit test	Integration test	Sytem test	QA test						
TM-03	テスト管理 Test Management	テストの量を確認する Check the amount of testing	<p>システムテストと妥当性テストの量を、テスト項目数や投入工数などの具体的な数値を使って確認する。確認したテストの量をソフトウェアの規模(概略のKLOC数)で除算してテスト密度を計算し、自社の類似製品の平均的なテスト密度と比較して大きく下回っていない事を確認する。</p> <p>Check the amount of system tests and validation tests using concrete numbers such as the number of test items and input man-hours. Calculate the test density by dividing the confirmed amount of test by the size of the software (general KLOC number), and make sure that it is not much lower than the average test density of your similar products.</p>						
TM-04	テスト管理 Test Management	テストの質を異常系テスト項目の比率で確認する Check test quality by the ratio of abnormal test items	<p>システムテストの全項目に対する、①正常系テスト項目と、②準正常系テスト項目と、③異常系テスト項目との比率を確認する。確認した準正常系と異常系のテスト項目の全体に対する比率が、自社の類似製品の平均的な比率の比較して大きく下回っていない事を確認する。 (備考: テストの定義)</p> <p>a) 正常系テスト項目: 正常な入力を与えた時に正常に処理が実行される事を確認するテスト項目 b) 準正常系テスト項目: 設計された異常状態が発生した時に、設計した通りの異常処理が実行される事を確認するテスト項目 (例: 通信プロトコルに存在する、予め定義されたエラー処理の動作を確認するようなテスト項目) c) 異常系テスト項目: 設計されていない異常状態が発生した時に、ハングやリセット等の異常状態にならずに稼働し続ける事を確認するテスト項目。 (例: ノイズによる異常動作や、壊れたデータの入力等に対する動作を確認する)</p> <p>Check the ratio of (1) normal test items, (2) semi-normal test items, and (3) abnormal test items to all system test items. Confirm that the ratio of the confirmed semi-normal and abnormal test items is not significantly lower than the average ratio of similar products of the company.</p> <p>(Note: Test definition) a) Normal test item: A test item to confirm that processing is executed normally when a normal input is given. b) Semi-normal test item: A test item that confirms that the designed abnormal process is executed when the designed abnormal condition occurs. (Example: A test item that exists in the communication protocol and confirms the predefined error handling operation) c) Abnormal test item: A test item that confirms that the product continues to operate without an abnormal state such as hang or reset when an undesigned abnormal state occurs. (Note: Check for abnormal operation due to noise, operation against input of broken data, etc.)</p>						

TM-05	テスト管理 Test Management	システムテストの質をテストカテゴリ毎のテスト項目の比率で確認する Check the quality of system tests by the ratio of test items for each test category	16種類のシステムテストカテゴリ(Test Category sheet) 毎に、システムテストのテスト項目数を数えて、テスト項目数毎の比率を計算する。計算した比率が、自社の類似製品の平均的なテストの比率の比べて大きく異なっていない事を確認する。 Count the number of system test items for each of the 16 system test categories (Test Category sheet) and calculate the ratio for each test item. Make sure that the calculated ratio is not significantly different from the average test ratio of similar products in your company. Test category ratio(system test) = number of test case in test category-A / total number of test case (in system test)		
TM-06	テスト管理 Test Management	ファームウェアのバージョンアップ機能のテストの質を確認する Check the quality of firmware upgrade function test	ファームウェアのバージョンアップ機能については、テストに用いる通信環境やサーバ環境が、実運用に近い環境でテストできているかを確認する。例えば以下のようなテスト環境が準備されているか確認する。 1) ノイズや切断の多い端末とサーバ間の通信回線 2) 遅延の大きい端末とサーバ間の通信回線 3) 応答速度が遅いサーバ For the firmware upgrade function, check whether the communication environment and server environment used for testing can be tested in an environment close to actual operation. For example, check whether the following test environment is prepared. 1) Communication line between a server and a server with a lot of noise and disconnection 2) Communication line between terminal and server with large delay 3) Server with slow response speed		
TM-07	テスト管理 Test Management	ファームウェアバージョンアップ機能のテストの量を確認する Make clear the quantity (or times) of "Software Version Up test"	ファームウェアのバージョンアップ機能については以下の回数のテストを計画しているか確認する。 1) 宅内でエンドユーザが利用する端末の場合:数百回~数千回 (エンドユーザの元でバージョンアップが行われる) 2) 局舎で動作する機器の場合:~数百回 (バージョンアップをシステムに習熟したオペレータが実施する) Make clear "software version up test" is planned to execute more than several hundred or several thousand times?		
TM-08	テスト管理 Test Management	無線機能のテストの質を確認する。(必要なら) Check the quality of wireless function tests. (If necessary)	無線機能が搭載されている場合は、各種の無線機能を確認するテストの手順の中に、無線の接続/切断の状態を起こしながらテストを実施するテスト手順が含まれている事を確認する。(無線の有効範囲からの出入り等の同等の操作でも良い) If the wireless function is installed, check that the test procedure for checking various wireless functions includes the test procedure for performing the test while causing the wireless connection / disconnection state. (Equivalent operation such as entering and leaving the wireless effective range may be used)		
TM-09	テスト管理 Test Management	テストの作業の進捗を管理する。 Manage the progress of the test work.	社内で実施するテストに関するテスト管理業務では以下のプロセスが実施されているか? a) テストの計画を作成する b) テストを実施しその結果を記録する c) テストの結果を報告書として整理する Are the following processes implemented in the test management work related to tests conducted in-house? a) Create a test plan b) Run the test and record the results c) Organize test results as a report		
TM-10	テスト管理 Test Management	テストの結果を評価する。 Evaluate test results.	社内で実施されたテストの結果は、計画していた品質指標を満足しているかどうかを確認する手順があるかを、確認する。 例えば、以下のような品質指標を用いている場合、計画していた目標値と実績値を比較して確認する。 a) テストの実行量やテスト密度 b) 検出不具合数や修正不具合数 The results of tests conducted in-house confirm whether there is a procedure for confirming whether the planned quality index is satisfied. For example, when the following quality index is used, the planned target value and the actual value are compared and confirmed. a) Test execution volume and test density b) Number of detected and corrected defects		
TC-01	テスト内容 Test Contents	単体テストの内容を明確にする。 Clarify the content of unit tests.	単体テストには以下のどれが含まれているのかを確認する。 1) 正常系の機能が設計/実装者の意図のとおりに行われる事の確認 (正常系テスト) 2) 異常系の機能が設計/実装者の意図のとおりに行われる事の確認 (準正常系テスト) 3) 設計/実装者の想定外の処理(実行されるはずの無いコード)から抜け出せる事の確認 (異常系テスト) 4) 全てのパスや条件分岐が設計/実装者の意図のとおりに行われる事の確認 (パステスト) Check which of the following is included in the unit test. 1) Confirm that normal system functions are executed as designed / implemented by the designer / implementer (normal system test) 2) Confirm that abnormal functions are executed as designed / implemented by the designer / implementer (Semi-normal test) 3) Confirm that it is possible to escape from unexpected processing (code that should not be executed) by the designer / implementer (abnormal system test) 4) Confirm that all paths and conditional branches are executed as designed / implemented by the designer (path test)		
TC-02	テスト内容 Test Contents	単体テストの実施のレベルを明確にする Clarify the level of unit testing	単体テストの実施方法として以下のような事が決まっているかを確認する。 1) 入力パラメータの選択の方法 (例えば 同値分割か境界値分析か) 2) パステストの網羅性の種類 (C0:命令網羅か、C1:分岐網羅か、C2:条件網羅か) 3) パステストの目標網羅率 (100% か、80% か、60% か) Check whether the following are decided as the unit test execution method. 1) Input parameter selection method(For example, equivalence partitioning or boundary value analysis) 2) Path test coverage type (C0: instruction coverage, C1: branch coverage, C2: condition coverage) 3) Pass test target coverage rate (100%, 80%, 60%)  		

TC-03	テスト内容 Test Contents	メモリーリークテストの有無を確認する Check if there is a memory leak test	メモリーリークの有無を確認するテストは実施しているか。そのテストはどんな方法で実施しているかを確認する。 例えば、以下のようなメモリーリークのテスト方法が具体的に書いてあるか 1) テスト開始前に空きメモリの容量を測定する 2) 長時間の稼働テストや各種の加速テストを実施する 3) テスト終了時に再度空きメモリの容量を測定し、テスト開始前から空きメモリの容量が変わっていない事を確認する。 Is there a test to check for memory leaks? Check how the test is being conducted. For example, the following memory leak test method is specifically written 1) Measure the amount of free memory before starting the test 2) Conduct long-term operation tests and various acceleration tests 3) Measure the amount of free memory again at the end of the test and confirm that the amount of free memory has not changed since the start of the test.		
TC-04	テスト内容 Test Contents	ソケットやメッセージのリークテストの有無を確認する Check for socket and message leak tests	ソケットやメッセージのリークの有無を確認するテストは実施しているか。そのテストはどんな方法で実施しているかを確認する。 Has a test been conducted to check for socket or message leaks? Check how the test is being conducted.		
TC-05	テスト内容 Test Contents	タイマーやカウンタのロールアップ時の動作をテストするテスト項目を確認する Check the test items to test the operation when the timer or counter rolls up	ラウンドオーバー(ラップアップ)するタイマーやカウンタの、ラウンドオーバー時の処理を確認するテストは実施しているか。そのテストはどんな方法で実施しているかを確認する。例えば以下のようなテスト方法が書いてあるか。 1) タイマーの値を タイムアップ-3 の値に設定する 2) タイマーの値が4 増加するまでシステムを動作させる 3) タイマーのラウンドアップ処理が実行された後もシステムが正常に動作している事を確認する Have you performed a test to check the round-over (wrap-up) timer or counter processing at the time of round-over? Check how the test is being conducted. For example, is the following test method written? 1) Set the timer value to the time-up-3 value 2) Run the system until the timer value increases by 4. 3) Confirm that the system is operating normally after the timer round-up process is executed.		
TC-06	テスト内容 Test Contents	49日問題、497日問題のテスト項目を確認する Check the test items for the 49th and 497th questions	タイマー問題の中でも有名な49日問題や497日問題についてのテスト方法を確認する。 (1m秒の32bitタイマーは 49日でラウンドオーバーする、10m秒の32bitタイマーは497日でラウンドオーバーする) 1) 初期値をラウンドオーバーに近い値に設定してソフトウェアを起動しラウンドオーバーを短時間で発生させる 2) デバッグツールで変数値をラウンドオーバーに近い値に設定し、ラウンドオーバーを発生させる 3) 特殊モードを組み込んで、変数値をラウンドオーバーに近い値に設定し、ラウンドオーバーを発生させる Check the test methods for the famous 49-day problem and 497-day problem among the timer problems. (A 1-msec 32-bit timer rounds over in 49 days, a 10-msec 32-bit timer runs over in 497 days) 1) Set the initial value to a value close to roundover, start the software, and generate roundover in a short time 2) Use the debug tool to set the variable value to a value close to roundover and generate a landover. 3) Incorporate special mode, set the variable value to a value close to roundover, and generate roundover		
TC-07	テスト内容 Test Contents	メモリ破壊の可能性のあるコードの確認方法を確認する Check how to check for possible memory corruption code	メモリ破壊問題を起こす可能性があるコードをコードレビュー等で確認する具体的な方法を確認する。 メモリ破壊問題とは、例えば下記のようなこと 1) メモリコピー時にコピーするバイト数を超過して設定しまい、コピー先の後続領域を破壊する 2) 配列変数のインデックスの超過/マイナス値などによる配列外のメモリ領域をアクセスする事で隣接するメモリ領域を破壊する Confirm a specific method of confirming the code that may cause the memory destruction problem by code review. Examples of memory corruption problems are as follows: 1) When the number of bytes to be copied is exceeded when setting memory copy, the subsequent area at the copy destination is destroyed. 2) Destroy the adjacent memory area by accessing the memory area outside the array due to the excess / minus value of the array variable index.		

TC-08	<p>リークや2重解放のバグを修正した時の二次不具合のチェック項目を確認する</p> <p>Check the check items for secondary problems when a leak or double release bug is fixed</p> <p>テスト内容 Test Contents</p>	<p>メモリや各種のOS資源について、リークや2重解放のバグがあってその場部を修正した時には、その実行条件に注意しないと2重解放やリークの新たなバグを埋め込む場合がある。</p> <p>コードレビューで使用するチェック項目に、2重開放やリークの修正による二次不具合の有無を確認する項目はあるか。例えば以下のような記述があるかを確認する。</p> <p>1) メモリリークのバグを修正した時には、メモリの2重解放のバグが新たに混入していないかを確認する 2) メモリの2重解放のバグを修正した時には、メモリリークの場部が新たに混入していないかを確認する</p> <p>For memory and various OS resources, when there is a leak or double release bug and the site is modified, if you do not pay attention to the execution conditions, you may embed a new double release or leak bug.</p> <p>Are there any check items used in the code review to check for secondary defects due to double opening or leak correction?</p> <p>For example, check whether there is the following description.</p> <p>1) When a memory leak bug is fixed, check whether a double memory release bug has been introduced. 2) When the memory double release bug is fixed, check if the memory leak field is newly mixed</p> <p>(example of secondary defects)</p> <pre>Sub_AAA() { get_resource(); execute some code; if error-X then { release_resource(); <-- 1st release <Bug1> if some_stateY then { retuen (ERROR); } } execute another code; release_resource(): <-- 2nd release ==> double release if error-X occured return(NO_ERROR); }</pre> <p>Sub_AAA() double release the resource if error-X occure and some_stateY not occure. And if corder remove 1st release code from Sub_AAA() Then Sub_AAA() leak resource if error-X occure and some_stateY occure.</p>		
-------	--	---	--	--