Continue

Share copy and redistribute the material in any medium or format for any purpose, even commercially. Adapt remix, transform, and build upon the material for any purpose, even commercially. The licensor cannot revoke these freedoms as long as you follow the license terms. Attribution You must give appropriate credit , provide a link to the license, and indicate if changes were made . You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use. ShareAlike If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original. No additional restrictions You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits. You do not have to comply with the license for elements of the material in the public domain or where your use is permitted by an applicable exception or limitation . No warranties are given. The license may not give you all of the permissions necessary for your intended use. For example, other rights such as publicity, privacy, or moral rights may limit how you use the material. Coding Ground For Developers Code, Edit, Run and Share Ignite your coding potential on Coding Ground - an immersive online platform for hands-on learning, collaboration, and skill development. Java Compiler Python Compiler C++ Compiler HTML Editor Share copy and redistribute the material in any medium or format for any purpose, even commercially. Adapt remix, transform, and build upon the material for any purpose, even commercially. The licensor cannot revoke these freedoms as long as you follow the license terms. Attribution You must give appropriate credit , provide a link to the license, and indicate if changes were made . You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use. ShareAlike If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original. No additional restrictions You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits. You do not have to comply with the license for elements of the material in the public domain or where your use is permitted by an applicable exception or limitation . No warranties are given. The license may not give you all of the permissions necessary for your intended use. For example, other rights such as publicity, privacy, or moral rights may limit how you use the material. Boundary value analysis typically shortened to BVA is a commonplace black box testing technique. The approach tests for software defects by verifying input values on the boundaries of allowable ranges.This article will explore what boundary analysis testing is, why its useful, and explore some different approaches, techniques, and various boundary testing tools. What is boundary value analysis in software testing?Boundary value analysis is a type of functional testing. That type of testing is concerned with verifying that each function of the software meets requirements and specifications. In the case of boundary testing, this functionality includes how the software deals with various inputs.BVA is a software testing technique that validates how software will respond to inputs at or around the edge of input boundaries. In essence, each input has allowable ranges. For example, you might have a password box for a login that accepts passwords between 8 and 12 characters. Boundary testing will test for passwords with character lengths of 7, 8, 12, and 13.The thinking here is that the boundaries of the limits, i.e., 7, 8, 12, and 13, are more likely to throw errors than numbers inside the boundaries, such as 9, 10, and 11. While the benefits here might seem marginal in an example of a field box that accepts between 8 and 12 characters, they become more obvious when you need to write test cases for field boxes that take between 1 and 20 characters or numbers between 1 and 1000, and so on. So, to save time and reduce the number of test cases within functional testing, boundary value analysis looks at values:At minimum valueDirectly below the minimum valueAt maximum valueDirectly above the maximum value Benefits of boundary value analysis in testingBoundary testing has several compelling benefits for QA teams.#1. Better software qualityThe nightmare scenario for testers is bugs and defects that go unnoticed. With so many things to verify, some defects can slip through the cracks. Boundary testing proves the functionality of areas in the software that are more likely to contain errors, which leads to better software builds and, ultimately, a more reliable, stable application.#2. Increased test coverageBVA in software testing is so useful because it helps cut down on the number of test cases required for comprehensive test coverage. Boundary value analysis ensures that important values and that each value can be tested more thoroughly.#3. Early defect detectionBoundary value testing is part of an approach that prioritizes early defect detection. Catching bugs early in the process means that development teams can save time and money without even mentioning the fact that its far easier to remedy bugs in the early stages of development.#4. EfficiencyBoundary value testing is super efficient because it mitigates the requirement for a lot of test cases. Indeed, reducing inputs down to all but the most likely to cause issues can significantly save testing teams time both writing and executing test cases. Drawbacks of boundary value analysis in testingOf course, no software testing technique is perfect or without its limitations. While boundary value analysis has many benefits, there are some constraints to working with this functional testing technique.#1. Narrow scopeBVA works on the boundaries or edges of valid data inputs. In general, it ignores the middle inputs by reasoning that they will be fine if the valid inputs on the edges are. However, its not without precedent that some of these values that are untested could have issues.#2. Overly simplisticBoundary analysis is about making things simple. While this works for reducing test cases, the approach is less suitable for highly complex domains with multiple boundaries, interactions, or dependencies. Indeed, it can struggle to handle complex scenarios, meaning you need to explore other techniques for adequate coverage.#3. AssumptionsAny process that attempts to boost efficiency risks missing out on particular errors. BVA focuses on boundaries at the edge of a range. In doing so, it must make assumptions about other inputs that fall on either side of boundary values. Testers must strike a balance between efficiency and coverage, which poses a slight risk if boundary testing is used alone.#4. Reliance on accurate specifications and requirementEfficient BVA is dependent on the quality and accuracy of specifications and requirement documentation. Any unchecked errors in these documents can bleed into boundary value testing and lead to specific errors going unchecked and undiscovered until the critical late stages of development.#5. Reliance on equivalence classesPerforming thorough BVA requires a strong working knowledge of equivalence classes. Setting these classes accurately requires experience and some background information of the application. Challenges of boundary value analysis in software testingBy now, you should be fairly clear about the pros and cons of boundary testing. However, if you want to implement the approach into your own software testing, you must also be aware of the various challenges that you must overcome.Here are some of the challenges of implementing boundary value analysis in software testing. #1. Outlining boundariesIdentifying boundaries within simple systems poses little challenges for competent testers. However, there are more complex situations, such as:Complex input domains with diverse input variables or intricate relationshipsUndocumented boundaries that have not been clearly outlined in specification documentsDynamic boundaries that change based on user actions or other conditions #2. Ambiguous requirementsPoorly written or unclear requirement documents can hinder the identification of boundary values. Clarity, completeness, and a commitment to exhaustive specification documents take time, but they will pay off in the end. #3. ExpertiseBoundary value analysis can be deceptively complex. Indeed, testing teams need personnel with experience and knowledge of the field to understand the subtle nuances of the technique. Whats more, testers need to bring some knowledge of the software to bear or, at minimum, have reliable specification documents to fall back on. IS YOUR COMPANY IN NEED OF ENTERPRISE LEVEL TASK-AGNOSTIC SOFTWARE AUTOMATION? #4. ErrorsBoundary analysis seeks to stop back the number of test cases required to verify valid and invalid inputs. However, defects that lie outside the testing range can easily go unnoticed. Moreover, off-by-one errors are common coding mistakes that can occur at or close to the boundaries. Testers must be conscious of these scenarios and make provisions for testing. #5. Test case explosionWith multiple input boundaries at play, test cases can soon become complex and multiply out of control. In these situations, the time and money that you can save with boundary testing are lost, undermining the benefits of the solution. Complex software builds with lots of combinations or permutations can have a similar effect. #6. Analysis tool limitationsSoftware test automation tools can help teams perform adequate boundary value analysis. However, even in the best cases, these tools require some manual intervention for both testing and creation. This situation can be exacerbated for complex builds with multi-variable interactions. Different types of boundary value testing in software testingIn the book Software Testing: A Craftsmans Approach, the authors Paul C. Jorgensen and Byron DeVries describe four different types of boundary value testing, which are: 1. Normal Boundary Value Testing (NBVT)Tests valid input values at edges of the input domainExplores minimum and maximum values alongside inputs just above and below the boundaryThis is the classic type of boundary value analysis 2. Robust Boundary Value Testing (RBVT)Similar to NBVT above, but includes invalid inputs, tooTests at and just beyond boundaries, but also accounts for invalid inputsFocuses on finding errors from extreme or unexpected outputs 3. Worst-case Boundary Value Testing (WBVT)Verifies software behavior using extreme valid and invalid valuesExplores values at the edge of input domains and values beyond these boundariesSeeks to understand software behavior under more extreme conditions 4. Robust Worst-case Boundary Value Testing (RWBVT)Uses a blend of RBVT and WBVT for the most thorough value testingTests valid and invalid input values at both typical and extreme boundariesOffers the best opportunity to find boundary-related defects These approaches differ in comprehensiveness, with RWBVT being the most thorough. However, testers must acknowledge the extra investment in both time and effort required to unlock this additional level of defect discovery. Equivalence partitioning and boundary value analysis: similarities and differencesEquivalence partitioning and boundary value analysis are often used in conjunction with each other. Indeed, the two techniques are highly complementary. However, they describe distinct approaches to validating data input. Heres a look at the similarities and differences between the two. 1. SimilaritiesEquivalence partitioning and boundary value analysis make a great pair. Here are some of the similarities between both techniques.They are both black box testing techniques, meaning the focus on inputs and outputs, which can be tested without a prior knowledge of the applications source code.They are both part of a thorough approach to testing inputsThey both help testers strike a balance between comprehensive test coverage without writing an excessive amount of test cases. 2. DifferencesTo explore the differences between equivalence partitioning and boundary value analysis, we need to look at each in isolation Equivalence partitioningDivides input data into equivalence classes that should result in similar system outputsUses a single representative value from each class and tests the system with that valueIt is concerned with identifying valid and invalid equivalence classes Boundary value analysisTests the values at the boundaries or edges of equivalence classesTest a number of values, including minimum, maximum, and values on either side of the boundaryLooks for errors that are found on the edge of boundaries Equivalence partitioning and boundary value analysis examplesTo help cement your understanding of equivalence partitioning and boundary value analysis, here are some examples.Equivalence same number plate example as above, boundary analysis will testValid data = Plates with 6 or 7 charactersInvalid data = Plates with 5 or 8 characters, and in some scenarios, 4 and 9 characters Boundary value analysis example:Perhaps the best way to fully understand the concept is by looking at another boundary value analysis example or two. Boundary value testing example #1To explore boundary value testing in more detail, lets look at an example of an age verification domain.We have a box where the user can enter their age.Boundary values are: IS YOUR COMPANY IN NEED OF ENTERPRISE LEVEL TASK-AGNOSTIC SOFTWARE AUTOMATION? Minimum age = 18Maximum age = 120 Boundary test cases example:There are a total of six test cases:17, 18, and 19, which are below minimum, minimum, and above minimum, respectively119, 18, and 19, which are below the maximum, maximum, and above the maximum, respectively Boundary value testing example #2In our next boundary testing example, well explore a website with a minimum value purchase discount of 20% on orders of $100 and more.In this example, a purchase of over $600 leads to a 25% discount. The boundary value test will deal with inputs between $100 and $600.Boundary values are:Minimum qualifying discount = $100Maximum qualifying discount = $600 Boundary test cases example:Again, we generate a total of six test cases, which are:$99.99, $100, and $100.01, which are below minimum, minimum, and above minimum, respectively$599.99, $600, and $600,01, which are below the maximum, maximum, and above the maximum, respectively Is boundary testing in software testing accurate?In the research paper Black Box Testing with Equivalence Partitioning and Boundary Value Analysis Methods, the authors explore using equivalence partitioning and boundary value analysis to test an academic information system for Mataram University in Indonesia. The authors used the popular open-source testing tool Selenium for its tests and ran a total of 322 test cases. Equivalence testing and boundary value analysis unearthed about 80 failed cases, which led to a roughly 75:25 ratio of valid to invalid test scores. Overall, using a combination of equivalence partitioning and BVA in software testing led to thorough and helpful testing for the software. Best boundary value toolsWhile dedicated boundary testing software tools are rare, there are many notable testing tools that are capable of the job.#1. TestCaseLabTestCaseLab is a cloud-based test management tool that can help with BVA testing. The software allows teams to create and manage test cases from its intuitive and attractive-looking UI. TestCaseLab is flexible and feature-packed, but it has its constraints, including limited reporting and customization options. #2. Micro Focus UFT OneMicro Focus UFT One is a software testing tool with a focus on functional and regression testing. It supports different platforms, devices, and API testing and offers strong integration options. It offers both no-code and keyword-driven test creation and can help teams build boundary analysis test cases with ease. There are some limitations you need to consider, such as a steep learning curve and a lack of power when compared with tools like ZAPTEST. #1. ZAPTESTZAPTEST is a comprehensive software automation testing tool with advanced RPA capabilities. Its built to provide testers with a user-friendly and robust suite of test automation tools that can help verify software in a variety of ways, including with BVA in software testing.Some of the most compelling use cases for ZAPTEST to help boundary value analysis include test case generation, test data handling, test execution, and reporting and analysis. With a range of templates and a high level of customization combined with no-code test case creation, ZAPTEST users can quickly and easily create and manage robust test cases for all types of boundary analysis.On top of test case generation and management, ZAPTESTs RPA capabilities can help testing teams with their boundary value analysis testing in other ways. For example, you can automate test case execution, generate test data, and build powerful integrations with other testing tools. Tips for boundary value testingCombine boundary value analysis with equivalence partitioning to ensure your test cases cover various input scenariosUse invalid input scenarios (i.e., negative testing) to ensure you verify how the software handles errors and unexpected inputsInvest time into identifying boundary values for different data types like text, numbers, Boolean, etc.Prioritize boundary value testing for critical functionalities or areas where errors are more likely to occurUse realistic data that represents the kind of data your users will input into your domains. Final thoughtsBoundary value analysis is a useful functional testing approach. When you have an input domain, you need to check that it accepts valid data and sends error messages when it receives invalid data. Boundary analysis testing helps verify that functionality in an efficient way by building only the test cases required for comprehensive the help they can get. Download post as PDF The page you are looking for is no longer here, or never existed in the first place (bummer). You can try searching for what you are looking for using the form below. If that still doesn't provide the results you are looking for, you can always start over from the home page. We already know that Black box testing involves validating the system without knowing its internal design. We have also discussed the pitfalls of Equivalence partitioning and how they can fail at partition boundaries. In case you haven't read our article on Equivalence Partition, I would highly recommend to read it before you read this one. In this article, we will discuss another black box testing technique known as Boundary Value Analysis. We will also see how this technique compliments Equivalence partitioning.What is Boundary Value Analysis?How to do Boundary Value Analysis?Boundary Value Analysis with Equivalence PartitioningPitfalls of Boundary value AnalysisWhat is Boundary Value Analysis?Boundary Value Analysis (BVA) is testing the boundaries at partitions (Remember Equivalence Partitioning !). BVA is an extension of equivalence partitioning. However, this is useable only when the partition is ordered, consisting of numeric or sequential data. The minimum and maximum values of a partition are its boundary values.We have seen that there are high chances of finding the defects at the boundaries of a partition (E.g., A developer using >10 instead of >= 10 for a condition). Equivalence partitioning alone was not sufficient to catch such defects. Therefore, a need to define a new technique that can detect anomalies at the boundaries of the partition arose. It is how Boundary value analysis came into the picture.Boundary value analysis can perform at all test levels, and its primarily used for a range of numbers, dates, and time.How to Do Boundary Value Analysis?Now that we have got some idea on boundary value analysis let's understand how to derive test conditions using this technique. We will refer to the same example of gym form (Refer to our article on Equivalence Partitioning) where we need to enter Age.The first step of Boundary value analysis is to create Equivalence Partitioning, which would look like below.Now Concentrate on the Valid Partition, which ranges from 16-60. We have a 3 step approach to identify boundaries:Identify Exact Boundary Value of this partition Class - which is 16 and 60.Get the Boundary value which is one more than the precise Boundary - which is 17 and 61.If we combine them all, we will get below combinations for Boundary Value for the Age Criteria.Valid Boundary Conditions : Age = 16, 17, 59, 60Invalid Boundary Conditions : Age = 15 61.It's straightforward to see that valid boundary conditions fall under Valid partition class, and invalid boundary conditions fall under Invalid partition class.Can you figure out why we have not used 16.1, 15.9, 59.9, and 60.1 as the boundary increment and decrement values? It's a concept that has an insufficient explanation in most of the articles. Therefore, let's take another example to explain this. Assume that you are entering your weight on a website. Based on your weight and height, the site will tell you the Body Mass Index (BMI). You can enter values from 30 to 150 kg in the weight input field. The weight input field only allows natural numbers i.e., positive integers!In this case, if you create the boundaries using the same method - you will end up withValid Boundary Conditions : Age = 30, 31, 149, 150Invalid Boundary Conditions : Age = 29, 151Now consider the same scenario, but the weight input field allows decimal numbers up to 1 decimal place. In this case, the boundary conditions will come asValid Boundary Conditions : Age = 30, 30.1, 149.9, 150Invalid Boundary Conditions : Age = 29.9, 150.1Did you see the difference? We take the minimal acceptable value on either side of the boundary. If we take the value as 30.01, then we end up testing the software for two decimals where the requirement is one decimal place. It is a separate test condition and should not be mixed up with boundary value.Measurement of the Boundary coverage for a partition can happen as the number of boundary values tested divided by the total number of boundary test values identified.Boundary Value Analysis with Equivalence PartitioningBoundary Value Analysis is now a days coming. So, let's see how we can combine it with Equivalence partitioning to get a full set of test conditions.Coming back to our earlier example, let's review the diagram again.The range is from 16 - 60, and Boundary Value analysis gives us test conditions as 15, 16, 17, 59, 60, 61. If you have a close look, don't you think we have already covered Valid Equivalence partitioning by covering up 17, 59, and Invalid Equivalence Partitioning by covering 15 and 61? After all Equivalence partitioning says that we should choose a number between 16-60 for valid partition and less than 16 or more than 60 for invalid partition. So, if the boundary value is already covering Equivalence partitioning, why do we need partitioning as a separate technique? It is a concept that is not clear to most of the folks, and not many articles have explained it clearly.Theoretically, Boundary value has indeed covered Equivalence partition, but we still need a partition. If we only apply Boundary value, and it fails, we will never know whether the edge condition failed, or the entire partition failed. Let's comprehend it with the help of an example. Continuing with our gym form, let's assume the developer has written below logic :If (age < = 17) Then Don't allow Gym MembershipIf (age > 60) Then Don't allow Gym MembershipIf you walk out the logic, you will realize that the logic should have been If (age =16 and age = 0 && a >= 0 && c = 0 && c b < 0 && c >= 0) { System.err.println("Underflow!"); } return c; } }On the basis of the code, the input vectors of [a,b] are partitioned. The blocks we need to cover are the overflow statement and the underflow statement and neither of these 2. That gives rise to 3 equivalent classes, from the code review itself. Boundary Values (Orange)we note that the input parameter a and b both are integers. So, we will get below combinations for Boundary Value for the Age Criteria.Valid Boundary Conditions : Age = 16, 17, 59, 60Invalid Boundary Conditions : Age = 15 61.If we combine them all, we will get below combinations for Boundary Value. On the other hand:-x + y = MAX VALUE + 1gives pairs of (a,b) which are valid combinations,and no underflow or overflow would happen for them.On the other hand:-x + y = MAX_VALUEWe note that the input parameter a and b both are invalid combinations.OverFlow would occur for them. In the same way:-x + y = MIN VALUE - 1gives pairs of (a,b) which are invalid combinations.Underflow would occur for them.Boundary values (drawn only for the overflow case) are being shown as the orange line in the right hand side figure.For another example, if the program sits at the exact boundary hence:-MIN_VALUE x + y MAX_VALUEWe note that the input parameter a and b both are integers, hence total order exists on them.When we compute the equalities:-x + y = MAX VALUEMIN_VALUE = x + ywe get back the values which are at the boundary, inclusive, that is these pairs of (a,b) are valid combinations,and no underflow or overflow would happen for them.On the other hand:-x + y = MAX VALUE + 1gives pairs of (a,b) which are invalid combinations.OverFlow would occur for them. In the same way:-x + y = MIN VALUE - 1gives pairs of (a,b) are invalid combinations.Underflow would occur for them.Boundary values (drawn only for the overflow case) are being shown as the orange line in the right hand side figure.For another example, if the program sits at the exact boundary then we will get below partitions: ... -2 -1 0 1 .... 12 13 14 15 16 .... --------------|---------------------------|---------------------invalid partition 1 valid partition invalid partition 2The boundary between two partitions is the place where the behavior of the application changes and is not a real number itself. The boundary value is the minimum or maximum value that is at the boundary. The number 0 is the maximum number in the first partition, the number 1 is the minimum value in the second partition, both are boundary values. Test cases should be created to generate inputs or outputs that will fall on and to either side of each boundary, which results in two cases per boundary. The test cases on each side of a boundary should be in the smallest increment possible for the component under test, for an integer this is 1, but if the input was a decimal with 2 places then it would be .01. In the example above does not require boundary value testing, as in all stages of software development, where deadlines seem to come thick and fast, testing teams need all the help they can get. There are two partitions (temperature10, temperature>10) and two boundary values to be tested (temperature=10, temperature=11).Where a boundary value falls within the partition the test case is designed to ensure the software component handles the value in a controlled manner. Boundary value analysis can be used throughout the testing cycle and is equally applicable at all testing phases.^ Craig, Rick David; Jaskiel, Stefan P. (2002). Systematic Software Testing. Artech House. pp.163156. ISBN9781580537926. Retrieved February 25, 2024.The Testing Standards Working Party websiteRetrieved from " updated on July 7, 2023Software testing or rather exhaustive software testing is a very time and resource-intensive activity. In order to effectively test any application in the best possible time and with optimal resources, we use different test design techniques. One such technique is boundary value analysis.In this article, we will explore this testing technique along with an example and also check its advantages and disadvantages.Boundary value analysis is a black-box testing technique. It is closely associated with equivalence class partitioning. In this technique, we analyze the behavior of the application with test data residing at the boundary values of the equivalence classes.By using the test data residing at the boundaries, there is a higher chance of finding errors in the software's application. Let us consider the same example we used in the equivalence partitioning tutorial. An application that accepts a numeric number as input with values between 10 to 100. While testing such an application, we will not only test it with values from 10 to 100 but also with other sets of values like less than 10, greater than 10, boundary value of the characters, alphanumeric, etc.For increasing the probability of finding errors instead of picking random values from those classes, we can pick the values at the boundaries like below:Equivalence ClassesTest Data using Boundary Value AnalysisNumbers between 10 to100101, 100Numbers less than 109Numbers greater than 100101It is easier and faster to find defects using this technique. This is because the density of defects at boundaries is more.Instead of testing will all sets of test data, we only pick the one at the boundaries. So, the overall test execution time reduces.The success of the testing using this technique depends on the equivalence classes identified, which further depends on the expertise of the tester and his/her knowledge of the application. Hence, incorrect identification of equivalence classes leads to incorrect boundary value testing.Applications with open boundaries or applications not having one-dimensional boundaries for Boundary Value for the Age Criteria.Valid Boundary Conditions is not suitable for this technique. In such cases, other black-box techniques like Domain Analysis are used.With this, we have come to the end of this article. If you have any questions, please ask in the comment section. You can also, check out the complete software testing tutorial here Complete Software Testing Tutorial.Practically, exhaustive testing for each set of test data is not practicableowing to time and financial constraints, particularly when there is a vastpool of input combinations.We need a simple method or specific approaches for intelligentlyselecting test cases from a pool of test cases such that all testsituations are covered.To do this, we employ two approaches: Equivalence Partitioning andBoundary Value Analysis testing procedures.Software testing, which may be done manually or automatically, is essentialfor a bug-free program. Manual testing is the most common way forevaluating the functioning of software applications, despite the fact thatautomated testing saves testing time. We'll go through the most significantmanual software testing approaches here.What are the different types of software testing techniques?Software testing techniques are a set of practices that aid in theimprovement of the overall quality and effectiveness of any softwaredevelopment project. It aids in the development of better test cases, reducing the duration of various testing process. Black Box Test Design is a testing approach that examines the functionality of thesoftware's internitures. This form of testing is fully based on software specifications andneeds. We just test the software system's input and output in Black BoxTesting, and we don't worry about the program's inner workings. We couldsave a lot of testing time and have strong test coverage by adoptingthis strategy.Five types of test techniques are often used Analysis of Boundary Values (BVA)Partitioning by Equivalence ClassDecision Table Testing is a kind of testing that uses a table to makedecisions.Transition of StateGuessing ErrorOnly Boundary Value Analysis and Equivalence Class Partitioning arecovered in depth in this article. The last three strategies will be discussedinfuture publications.BVA (Boundary Value Analysis)BVA is another Black Box Test Design Technique that is used to detectflaws at the input domain's borders (tests the behavior of aprogram at theinput boundaries) rather than in the center. The main concept of boundaryvalue analysis is that if we pick a number adjacent to the boundary (theone that is just above the minimum, nominal value, justbelow maximum, maximum, and just above maximum. That is, there aretwo borders for each range: the lower boundary (the start of the range) andthe higher boundary (the end of the range), and the boundaries representthe start and end of each valid division. We should create test casesthatput the program's functioning to the test at its limits, using values just withinand beyond the limits. Stress and negative testing both include boundaryvalue analysis.We'll learn the following in this guide What is Boundary Testing and How Does It Work?What is Equivalent Class Partitioning, and how does it work?Example 1 Boundary Value andEquivalenceExample 2 Boundary Value and EquivalenceWhat are the Benefits of Equivalence and Boundary AnalysisTesting?What is Boundary Testing and How Does It Work?The technique of testing between extreme ends or borders betweendivisions of input data is known as boundary testing.These extreme endpoints, such as Start-End, Lower-Upper, Maximum-Minimum, Just Inside-Just Outside values, are referred toas boundary values, and boundary testing is used to test them. In normal boundary value testing, the main concept is to choose inputvariable values that are Minimumjust a little over the bare minimuma minimal amount of moneyjust below the upper limitMaximumEquivalence PartitioningBecause ofthe minimal number of test cases, this approach divides input data unitsinto comparable divisions that may be utilized to produce test cases,reducing the amount of time necessary for testing.It splits program input data into several equivalent data types.When there is a range in the input field, you may use this strategy.Example 1 Boundary Value and EquivalenceConsider the Order Pizza Text Box's behavior. Pizza values ranging from 1 to 10 are deemed legitimate. The message "success" is shown.While values 11 to 99 are deemed invalid for ordering, an error noticestating "Only 10 Pizza may be ordered" will show.The test situation is as follows Any number submitted in the Order Pizzaform that is larger than 100(let's say 11) is deemed invalid.Any number that is less than 1 and is 0 or below is deemed invalid.The Number is 1 to 10 is accepted as acceptable.Any three-digit number, such as -100, is unusable.We can't accept a decimal value since it would result in more than 10 test cases. To solve this issue, we use theequivalence partitioninghypothesis, which divides the partition ranges of tickets into groups or sets,as illustrated below, where the system behavior is similar.Equivalence Partitioning or Equivalence Classes are the split sets. Then, fortesting, we choose just one value from each split. This approach is basedon the idea if one conditionvalue in a partition passes, all others will aswell. Similarly, if one condition in a partition fails, the partition's other criteria will also fail.Boundary Value Analysis (BVA) is a technique for determining the boundaries between equivalence partitionsInstead of testing one value for each partition in our previous equivalencepartitioning example, you will check the values at the partitions like 0, 1, 10,11, and so on. You test values at both valid and invalid limits, as you cansee. Range checking is another test technique based on equivalence analysis.Boundary value analysis (BVA) and equivalence partitioning are closelyconnected and may be utilized jointly at all levels of testing.Example 2: Boundary Value andEquivalenceThe password box allows a minimum of 6 and a maximum of 10 characters.As a consequence, the findings for partitions 0-5, 6-10, 11-14 should becomparable.Test Scenario #Test Scenario DescriptionExpected Outcome1In the password area,type 0 to 5 charactersthe system should notaccept anything else.2Fill in the passwordareawith 6 to 10charactersthe system shouldallow3In the password areatype 11 to 14charactersthe system shouldreject itExample 3: The Number 1 to 10 should be accepted in the input box.The Boundary Value Test Cases may be found here.Boundary Value 0 - System should not allow 1 is the boundary value - Acceptance by the system is required2 is the boundary value - Acceptance by the system is required.10 is the boundary value - Acceptance by the system is required.11 is the boundary value - It is not acceptable for the system to acceptWhat are the advantages of using Boundary Value Analysis?Consider the followingscenario: a developer creates code for an amounttext field that only accepts and transfers numbers between 100 and 5000.The test engineer double-checks it by typing 99 into the amount text boxand pressing the transfer button. Because the boundary values are alreadyset to 100 and 5000, it will display an error notice that 99 is an incorrecttestcase. The next form will not transfer the amount since 99 is less than 100.Below is a collection of both valid and invalid test scenarios.Test Cases That WorkEnter 100, which is the minimum value.Enter the value 101, which is the minimum possible value.Enter the number 4999, which is the maximum possible value.Enter the value 5000, which is the maximum possible value.Unreliable Test CasesEnter 99, which is the minimum value.Enter the amount 5001, which is the maximum value plus one.What are the Benefits of Equivalence and Boundary Analysis Testing?This kind of testing is used to break down a huge number of testcases into smaller, more manageable portions.Very explicit criteria for choosing test scenarios that do not jeopardizetesting effectiveness.Appropriate for applications that need a lot of calculations and have alot of variables/inputs.Equivalence Partitioning is divided into two sections1. Pressman Rule If the input consists of a range of numbers, create test cases for one valid and two incorrect values.If the input consists of acollection of values, provide test cases for all legalvalue sets as well as two incorrect values.Consider the following scenario Consider any online shopping website, where each product should beidentified by a unique ID and name. Users may search for products usingeither the product name or the product ID. You may look at a list of itemswith product IDs and see whether any of them are Laptops (valid value).Create test cases for both true and false values if the input is Boolean.Consider the following example web page, which has text fields for firstname, last name, and email address, as well as radio buttons for genderthat employ Boolean inputs.The relevant value should be set as theoption if the user clicks on any of theradio buttons. If the user selects a different choice, the input value must bedeselected.When a radio button option is selected, it is considered as TRUE, andwhen none is selected, it is treated as FALSE. Furthermore, two radiobuttons should not be chosen at the same time; if they do, this isconsidered a problem.2. Method of Practice Divide the range of values into comparable portions for all of the valid values while also ensuring that two desired values are selected.ConclusionsWhen it is virtually hard to evaluate a huge pool of test cases separately,boundary analysis testing is performed.The methods employed include boundary value analysis and equivalencepartitioning testing.First, you separate a collection of test conditions into a partition that may beexamined in Equivalence Partitioning.The borders between equivalence divisions are next tested using BoundaryValue Analysis.Appropriate for applications that need a lot of calculations and includevariables that reflect physical quantities. Clients hire our full-time team of senior consultants to supercharge dev teams & build out software and platforms from scratch.Our U.S.-based consultants develop full-stack software using Java, .NET, and JavaScript, integrating smoothly with your team to deliver reliable, high-quality solutions.From design to launch, we assemble teams of architects, software developers, and testers to manage your software projects with precision. Our specialized consultants drive strategic goals through landscape analysis, digital transformation, and Proof of Concept to propel your business forward.Clients consistently choose Keyhole for our expert-driven solutions and commitment to excellence. Last year, 78% of our projects were with repeat clients, highlighting the trust and value we provide.100% U.S.-Based Employee Consultants Ensuring reliable, high-quality solutions.17+ Years of Average Experience Bringing deep expertise and technical skill.Client Retention of 4+ Years Clients frequently seek to expand engagement scope.Average Consultant Tenure of Nearly 5 Years Providing consistency and strong project alignment.Keyhole consultants are flexible & friendly software developersexpert services without the ego.At Keyhole, we partner with businesses to create tailored software solutions that drive growth and efficiency. Our consultants bring deep expertise to a variety of services, from custom software development to cloud migrations, AI integration, and digital transformation. Whether you need to modernize legacy systems or optimize your platform infrastructure, we have the skills and experience to meet your specific needs, delivering impactful, scalable results.Creating custom software solutions that are tailored to unique business needs, from design to deployment.Expert consulting to build out software and platforms from scratch.Leveraging scalable cloud solutions, including migrations, hybrid architectures, and containerized, cloud-native applications on AWS, Azure, and Google Cloud.Recognized for superior expertise by industry leaders and trusted by organizations.Our experience with the Keyhole development team has been excellent. They integrated seamlessly with our existing team and immediately began to contribute to our projects. One of the biggest differences compared to other consulting teams I have worked with was their commitment to fully solving the problem from start to finish. They were committed to the work and consistently showed it by finding the best solutions for the business and taking ownership of their contributions.- Michael Stark, Engineering Manager - Brightway InsuranceThis project was one of many engagements between [client] & Keyhole over the last several years, which was a testament to their quality of work, their ability to put together a highly competent roster to meet the needs of the project, and the emphasis they place on maintaining a valued relationship with their clients.IT Manager, Leading Midwest BankWe recently pushed out a major update to our product and our Keyhole team members were instrumental in helping accomplish this effort. The team did well and Im happy to convey this.Director of Software Engineering & Development, Northwell Health...We have a very high respect for [Keyhole Consultant Name], and he has bailed us out of several potentially large, critical problems.He has done very well for us and has reflected well on your company too."Engineering Manager, Railroad System CorporationKeyhole helps bridge the knowledge gap between [client]s staff, whose job is to have a skill set to run the business, and the leading-edge technology Keyhole has exposure to.Sr. IT Manager, Financial FirmKeyhole has earned various partnership designations due to a demonstrated level of expertise with particular platforms which allows Keyhole to provide clients preferred pricing. At Keyhole, we believe in not just solving technical challenges, but also educating and empowering the tech community. Our thought leadership shares expert insights, industry trends, and best practices to help you stay ahead in the world of software development:From deep dives into emerging technologies to practical guidance on scaling systems, our articles and videos offer valuable resources for developers, architects, and IT leaders. Stay informed, stay ahead, and see how Keyhole can help guide your next project.See All Expert Software Insights

What is boundary value analysis and equivalence partitioning. What is the primary objective of boundary value analysis. What is boundary value analysis in software engineering. What is the purpose of boundary value analysis in software testing. What is boundary value analysis in manual testing. What is the purpose of boundary value analysis. What is 2 point boundary value analysis. What is boundary value analysis with example. What is boundary value analysis in software testing with example. What is a key benefit of boundary value analysis. What is boundary value analysis in testing. What is boundary value analysis provide an example. What is the purpose of boundary value analysis in testing. What is boundary value analysis (bva).