Code Walkthrough

Code Walkthrough is a form of peer review in which a programmer leads the review process and the other team members ask questions and spot possible errors against development standards and other issues.

**Participants**

At least two people are required for the code walkthrough:

* Developer - who wrote the code
* Reviewer - who reviews the code
* **Author**: explains the code from his perspective and Takes responsibility for fixing the defect found to improves the quality of the document
* **Scribe**: , who is not the author, marks the minutes of meeting and note down all the defects/issues so that it can be tracked to closure.
* **Reviewer**: Check material for defects and inspects

Procedure

A walkthrough is conducted by the author of the ‘document under review’ who takes the participants(reviewer) through the document and his or her thought processes, to achieve a common understanding and to gather feedback.

This is especially useful if people from outside the software discipline are present, who are not used to, or cannot easily understand software development documents.

The content of the document is explained step by step by the author, to reach consensus on changes or to gather information.

The participants are selected from different departments and backgrounds If the audience represents a broad section of skills and disciplines, it can give assurance that no major defects are ‘missed’ in the walk-through.(give example of java programmer developing a program from his perspective and takes the suggestion from other programmers like dotnet programmers who are present in audience during the walk through. They come to conclusion of a better solution to make the software more maintainable and no major defects are missed.)

Goals

The main purpose of walkthrough is to enable learning about the content of the document under review to help team members gain an understanding of the content of the document and also to find defects.

## Outcome of the Code Walkthrough

There can be three outcomes to the code walkthrough:

* Successful: all required checks and quality are present.  Software may be released into the approved build.
* Corrective Action Needed, No Further Review Needed:  a list of items to be corrected are presented.  Once these are performed, the code walkthrough is complete.
* Corrective Action Needed, Review Needed: a list of items to be corrected are presented.  One these are performed, another code walkthrough is warrented.

Desk Checking

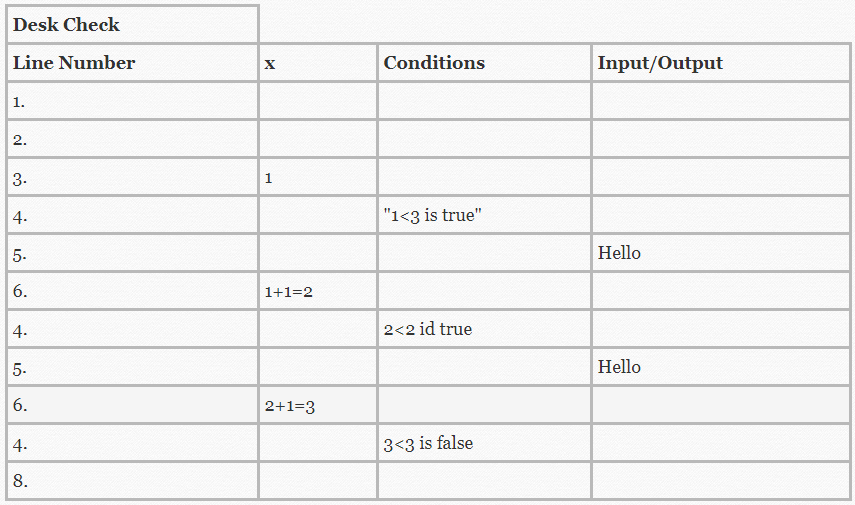
Desk checking is a method to verify the portions of the code for correctness. ¬

Normally done manually by the author of the code, this method is done before compiling and executing the code

Whenever errors are found, the author applies the corrections for errors on the spot. ¬ This method relies completely on the author’s thoroughness, diligence and skills. ¬ There is no process or structure that guarantees the effectiveness of this checking. ¬ This method is effective for correcting “obvious” coding errors but will not be effective in detecting errors that arise due to incorrect understanding of requirements. ¬

Procedure

Desk checking is a similar process to proofreading; in this exercise, the programmer runs through lines of code to identify errors and to check logic. Typically, the programmer will print out the code and go through it in a pencil and paper exercise. He may run a manual test on algorithms, checking that they work correctly and contain no coding errors. This usually involves creating a table with columns containing line numbers, variables, conditions, and inputs and outputs, depending on the checks he is making.



* *Line Number :*This is to check the number of lines to be executed.
* *One column per variable :*The variables are placed in an alphabetical order. As the algorithm proceeds new values are added to the consecutive columns. The final column is ideally for representing the output/result of the calculations.
* *Condition :*A condition column is used to show the evaluation of some condition. The result of the condition will will either be true or false.
* *Input/output column :*Input/output column shows the inputs by a user and the outputs generated by the system.

The process of desk checking typically involves the following steps :

* **Understand the given problem :**It is of course an important thing to understand the problem from every possible dimension, in order to devise the most appropriate solution for it.
* **Specify a high level solution :**Defining a solution means, after analysing the problem, one is required to determine the outputs, the process need to be followed in the process of attaining the right results, inputs required.
* **Develop the outline :**After gathering such data, we should be able to sketch an outline/design of the solution.
* **Develop the algorithm :**The simple thing here is to begin with writing the algorithm for the problem under consideration.
* **Test the algorithm :**Perform desk checking, validate that the logic that the algorithm depicts is worthy/suitable enough to generate the required output.

Lets see an example to understand each step of desk checking.

* Problem :

Calculate the discounted price of an item purchased by a consumer.

* Solution :

The problem requires calculation of price of an item. So the variables used will be - input\_price, discount\_percent, discount, discount\_price.

* Prepare Outline

|  |  |  |
| --- | --- | --- |
| **Input** | **Processing** | **Output** |
| item\_price | Calculating  the discount rate | Actual  discount value |

* Algorithm :
  1. calculate\_discountPrice()
  2. declare input\_price, discount\_percent
  3. discount = input\_price \* discount\_percent/100
  4. discount\_price = input\_price - discount
  5. Display discount\_price
  6. stop
* Test Data

This step is to check the output of the actual code.

*Inputs : input\_price, discount\_percent* *output : discount\_price*

## Advantages :

* Catch a bug even before it occurs. An experienced programmer also tends to make mistakes, so desk checking is a way to take precaution in that regard.
* It is an inexpensive and quick technique.
* As the programmer writes the code/algorithm on his own, he can check if there is any issue or not, and fix the same before the project moves on to the later stages.

## Disadvantages :

* Although the programmer is in a position to find out the errors in their own code, it may sometimes be otherwise. He may not be able to track the issue/bug in the code, which may result in erroneous outcomes.

## Conclusion :

Desk checking enables developers to proceed in a very systematic manner, in terms of coding. This technique eliminates the chances of erroneous code as proper review is done to analyse the problem and therefore offer the best solution for a given problem.