STRUCTURED WALKTHROUGH

A structured walkthrough, a static testing technique performed in an organized manner between a group of peers to review and discuss the technical aspects of software development process.
The main objective in a structured walkthrough is to find defects inorder to improve the quality of the product.
Structured walkthroughs are usually NOT used for technical discussions or to discuss the solutions for the issues found. As explained, the aim is to detect error and not to correct errors. When the walkthrough is finished, the author of the output is responsible for fixing the issues.

- *It's a review of products at the end of stage in the development of a system by a group of relevant competence persons.*
- *Its purpose is to identify error, omissions or ambiguity in the system document produced during system development and initiates the necessary corrective action.*

The team must check that the model,

- Major objective: FIND ERRORS - Uncover errors in function, logic, or implementation for any representation of the software
- Meets system objectives.
- Is a correct representation of the system.
- Will do the job that it is supposed to do.
- Is easy to understand.
- Ensure that the software has been represented according to predefined standards
- Look for weaknesses or errors of style
- Verify that the software meets its requirements
- Make projects more manageable - Achieve software that is developed in a uniform manner

- These checks depend on the kind of model but must proceed in an orderly way. During the walkthrough, no actual design or system alteration takes place, problems are only noted for further action.

- The problems are noted in an action list which also specifies which members of the team are to be responsible for following up the problems.
- Walkthrough takes place throughout system development and there may be more than one walkthrough in each project phase and there are no set times for doing these.
- It is the time for walkthrough when you reach a point where you have done all you can on a model and you need to be sure that this correctly represents the system procedure followed in a walkthrough.

- Before the walkthrough begins, producer should ensure obvious problems have been eliminated from the model.
- A walkthrough team is brought together to apply the benefits of its combined knowledge to the whole system and to detect less obvious problems.
- Some people suggest that you should use a checklist on a structured model before you submit your documents for walkthrough.
- The checklist includes the common kinds of errors that often ocur in modelling and serves as a guide for detecting such errors.
- Once producers are satisfied with the model, it's time for a walkthrough.
- There are two possible outcomes of a walkthrough:

- No errors are found in the model and it's accepted.
- Errors are detected in the model so an action list is produced. The model is then ammended and later submitted to another walkthrough.

Preparing for the walkthrough

- Assemble walkthrough team.
- Assign roles to each member.
- Distribute relevant documentation to all team members. This should be done early to give members sufficient time to become familiar with the documentation.
- Call team members together for the walkthrough.

- The person who developed the model actually tracks through the documentation.
- Any ommissions, ambiguities or innaccuracies are noted in an action list during the walkthrough and followed up later. The members who have been assigned responsibility for correcting errors ammend the model. When this ammendment is completed, the models is ready for the next walkthrough.

Walkthrough team

The size of the walkthrough team depends on the material to be covered and upon skills and review experience of the potential participants. People with knowledge about the system under review should not be on the walkthrough team. The number should be between 5 to 7

They are selected to take the roles of

1. **Author:** The author of the work product is responsible for requesting the walkthrough when a meaningful portion of the product has been developed and is free of casual errors or defects. The author attends the walkthrough as an observer and answers reviewer's general questions.

2. **Presenter:** They usually develop the agenda for the walkthrough and present the work product that is being reviewed. The presenter should be familiar with the work product and should be a member of the project team.

3. **Moderator:** The main responsibility of the moderator is to facilitate the walkthrough session and ensure that the walkthrough agenda is followed. Moderators also encourage the participation of all reviewers. Moreover, the moderator can also be the scribe.

4. **Reviewers:** Evaluate the work product to determine if it is technically accurate. The reviewers also assess whether the project guidelines or standards are being followed, the project requirements are met, and whether the product is properly prepared.

5. **Scribe:** The scribe takes notes during the walkthrough. Their responsibility is to record the errors that are identified and any other technical comments, suggestions, and unresolved questions. Also, a scribe cannot be a reviewer.

Types of Walkthroughs

- Specification walkthroughs
  - System specification
  - Project planning
  - Requirements analysis
- Design walkthroughs
  - Preliminary design
  - Design
- Code walkthroughs
- Test walkthroughs
  - Test plan
  - Test procedure
- Maintenance reviews

Specification Walkthroughs

- Objective - Check the system specification for:
  - Problems
  - Inaccuracies
  - Ambiguities
  - Omissions
- Participants
  - User
  - Senior analyst
  - Project analysts
- Objects
  - DFDs, Data Dictionary, ERDs, ...

Design Walkthroughs

- Objective - Check the architecture of the design for:
  - Flaws
  - Weaknesses
  - Errors
  - Omissions
- Participants
  - User
  - Analyst
  - Senior designer
  - Project designers
- Objects
  - Structure charts, detailed design documents, ...

Code Walkthroughs

- Objective - Check the code for:
  - Errors
  - Standards violations
  - Lack of clarity
  - Inconsistency
- Participants
  - Author
  - Project programmers
  - Designer
  - Outside programmers
- Objects
  - Code listing, compiler listings, ...

Test Walkthroughs

- Objective - Check the testing documents for:

  - Inadequacy
  - Incompleteness
  - Lack of clarity
- Participants
  - Project programmers
  - Tester
  - Analyst
  - Designer
- Objects
  - Test plan, test procedures, sample test data, ...

Checklists

Checklist: System Specification

- Are major functions defined in a bounded and unambiguous fashion?
- Are interfaces between system elements defined?
- Have performance bounds been established for the system as a whole and for each element?
- Are design constraints established for each element?
- Has the best alternative been selected?
- Is the solution technologically feasible?
- Has a mechanism for system validation and verification been established?
- Is there consistency among all system elements?

Checklist: Requirements Analysis

- Is information domain analysis complete, consistent, and accurate?
- Is problem partitioning complete?
- Are external and internal interfaces properly defined?
- Does the data model properly reflect data objects, their attributes, and relationships?
- Are all requirements traceable to system level?
- Has prototyping been conducted for the user/customer?
- Is performance achievable within the constraints imposed by other system elements?

- Are requirements consistent with schedule, resources, and budget?
- Are validation criteria complete?

Checklist: Preliminary Design

- Are software requirements reflected in the software architecture?
- Is effective modularity achieved? Are modules functionally independent?
- Is the program architecture factored?
- Are interfaces defined for modules and external system elements?
- Is the data structure consistent with the information domain?
- Is the data structure consistent with software requirements?
- Has maintainability been considered?
- Have quality factors been explicitly assessed?

Checklist: Design

- Does the algorithm accomplish the desired function?
- Is the algorithm logically correct?
- Is the interface consistent with the architectural design?
- Is the logical complexity reasonable?
- Have error handling and "antibugging" been specified?
- Are local data structures properly defined?
- Are structured programming constructs used throughout?
- Is design detail amenable to implementation language?
- Which operating system or language-dependent features are used?
- Is compound or inverse logic used?
- Has maintainability been considered?

Checklist: Code

- Has the design properly been translated into code?
- Are there misspellings and typos?
- Does the code adhere to proper use of language conventions?
- Is there compliance with coding standards for language style, comments, prologues?

- Are there incorrect or ambiguous comments?
- Are data types and data declarations proper?
- Are physical constants correct?
- Have all the items on the design walkthrough checklist been reapplied as required?

Checklist: Test Plan

- Have major test phases properly been identified and sequenced?
- Has traceability to validation criteria and requirements been established?
- Are major functions demonstrated early? (top-down)
- Is the test plan consistent with the overall project plan?
- Has a test schedule been explicitly defined?
- Are test resources and tools identified and available?
- Has a test record-keeping mechanism been established?
- Have test stubs been identified and has work to develop them been scheduled?
- Has stress testing for the software been specified?
- Has a regression testing mechanism been established?

Checklist: Test Procedure

- Have both white and black box test been specified?
- Have all independent logic paths been tested?
- Have test cases been identified and listed with their expected results?
- Is error handling being tested?
- Are boundary values being tested?
- Are timing and performance being tested?
- Has an acceptable variation form the expected results been specified?

Checklist: Maintenance

- Have side effects associated with the change been considered?
- Has the request for change been documented, evaluated, and approved?
- Has the change, once made, been documented and reported to all interested parties?
- Have appropriate walkthroughs been conducted?

- Has a final acceptance review been conducted to ensure that all software has been properly updated, tested, and replaced?

Benefits

- Saves time and money as defects are found and rectified very early in the lifecycle.
- This provides value-added comments from reviewers with different technical backgrounds and experience.
- It notifies the project management team about the progress of the development process.
- It creates awareness about different development or maintenance methodologies which can provide a professional growth to participants.
- Validate and improve the related lifecycle work products.
- Provide professional growth to participants by giving them an opportunity to look at different development methodologies and approaches.

Problems associated with walkthrough

i. Inadequate preparation by reviewers.
ii. Too much time spent discussing solutions rather than identifying defects.
iii. Producers being defensive about his/her work.
iv. Tendency of people to forget the important points in favour of arguing over the inconsequent parts.
v. Tendency to fix blame on others and behave as prima donnas i.e upstaging everyone in an attempt to discredit the work of the other members.

Solutions to the problems

i. Reminding participants before the meeting of the purpose of the walkthrough and importance of preparation.
ii. Forcing a time limit (60 - 90 min).
iii. Ensuring walkthrough standards are adhered to.

iv. A checklist of specific topics for review should be available to keep them on track

v. Careful selection of team members.