

Walk and Stagger Through Review Process

Michael Ensminger
PAR3 Communications
(Email: mensming@ieee.org)

Abstract

Presents a two-phase review methodology. The first phase, the walk through, concentrates on the logical correctness of the functionality covered in the review item. The second phase, the stagger through, concentrates on error handling and unexpected flows through the review item. The method is particularly suited to teams without a lot of development experience. A case study from an internet company is described at the end of the paper.

Introduction

The use of peer reviews, walkthroughs and inspections as a cost-effective method of achieving higher quality software is well documented. [MYER79, BOEH87] These techniques are not widely used in many organizations, especially those who say they are operating in "internet time". I have used a modified, two-phase walkthrough process at two different internet startups. This process has allowed the team to achieve high-quality deliverables in "internet time" while other teams in the same environment were struggling with quality and schedule pressure.

The traditional definitions of walkthroughs and inspections are presented here for reference:

- Walkthrough – Loosely defined term where two or more team members review an item for the purpose of finding issues and improve the quality. (As opposed to trying to solve a known issue.) Usually, the author of the item leads the group through it in a lecture format. [MCCO96, FREE90]
- Inspections – Formal method of reviewing initially developed at IBM by Michael Fagan. In an inspection, each team member has a formal role such as moderator, scribe, etc. Usually, the focus of the inspection is narrowly focused to only one or two aspects. [FREE90]

The advantages and disadvantages of these methods have been studied thoroughly.

A small team may find it difficult to follow the rigor that an inspection requires. The leniency provided by the walkthrough may not provide the expected results. The following method works well with small teams, especially those with less professional experience. Non-development personnel may also fully participate in the process.

Walk Through and Stagger Through

In this paper, I present a two-phase review process. Phase 1 is labeled the walk through -- but differs from the traditional use of the term. Phase 2 is the stagger through -- to indicate that obstacles will be placed in the way and the team must stagger around them.

Phase 1: The Walk Through

The first phase, the walk through, is the systematic review of a use case, design, code, etc. as the final system will execute it with a focus on the "normal" -- that is, non-error -- processing. The walk through can be applied to any software artifact. The key is focusing on tracing through the artifact in the same order as will occur in the final system. Also, "normal" processing may include (and probably should) some common error cases. For example, a file not found error during a file open command is fairly common. How the system handles this condition should be examined in the walk through phase.

Overall the walk through process is as follows:

1. Select the focus of the walk and stagger through and determine the scope.
2. Determine the review team and any special assignments.
3. Pre-read.
4. During the actual walk through, trace through the logical flow of the artifact concentrating on normal processing.
5. Record any issues in the item(s) being reviewed. Note items that may be issues in items not covered in the current scope.
6. Give the team time to make corrections based on the discovered issues.
7. Follow-up before the stagger through.

Determine Focus and Scope

The focus and the scope of the walk and stagger through depend on many factors including time available for review, review item availability, item risk, etc. For a pilot program, a small items which can be covered in an hour or less should be selected. In no case, should an item be selected that cannot be covered in less than two hours.

For a small project, it is possible to follow the system from startup until termination. For most projects, this is unreasonable. Picking a unit whose invocation is well defined is a good starting point. This could be a feature, subfeature, method, routine, etc. When dividing up items for review, keep an eye out for items that do not lend themselves to this kind of examination. This may be the first sign of an issue for the item.

Multi-threading presents its own share of problems. Initially, examine thread execution in isolation. Then, expand the scope to include thread cooperation, concurrency issues, possible race conditions, other timing problems, etc.

Review Team and Special Assignments

The author of the item under review is one of the main participants. They will respond to many questions like "what will happen when..." or "where did this data come from?" Note that the role of the author differs from the traditional role of the author in a walkthrough. Instead of presenting the item in lecture style, the author is responding to questions from the team. When possible, the authors of items that will interact with the item under review will also be present. They should be especially aware of interface / integration issues as well as how errors will propagate through the system. A moderator should direct the session. The moderator will play the role of the users, the operating system, the data, etc. If the item under review contains a technology that is new or something that has been problematic in the past, one of the attendees should focus on that area. Quality assurance and test personnel can "virtually" execute the suite of test cases in this

environment. Operations personnel can determine the environmental needs of the system. Marketing can get a feel for various aspects of the final product and use that to craft their message. Junior personnel will gain knowledge of the system and the issues that must be addressed in a professional level deliverable. Management may also benefit by understanding the complexity that underlies seemingly simple functionality.

Pre-Read

Everyone's time will be better spent when all have done their homework. The pre-read is essential to the success of the walk through and stagger through. All participants should "actively" read the review material. By "actively", I mean reading with pen in hand and noting issues, questions and "gotcha" situations. The moderator should decide on the "flow" of the walk through, determining the starting point, order to take branches, which items to treat as black boxes, which items to descend into detail, etc. Other members of the team must achieve a general understanding of the item and drilling deeper based on their role.

It may be beneficial to submit issues and questions in advance. Especially if there is a large number, it will be time better spent for all to resolve these issues beforehand. Usually, the author and the moderator can determine whether revision should occur before continuing with the process.

Review Meeting

Once at the review meeting, the moderator instructs the author where in the material to begin - usually at some point that results from an operating system event or user action. The author then leads the group through the logical flow from that point. All participants should be insuring the logical correctness and making notes of what could go bad when not taking the non-error path. Whenever a piece of data is accessed, it should be clear where the data comes from. When the author has a choice to branch through the code, the moderator will decide which direction the review should follow.

When another function / method / unit is called or referenced, the moderator must decide whether to treat it as a black box or to drilldown into the details. This decision should not be treated lightly. Drilldown will require time and effort. The team will need to know whether to prepare for drilling down into material that may not be apparent as the subject of the review. Treating the item as a black box may result in issues not being discovered. My rules of thumb are:

Coverage Criteria

There are varying levels of coverage. Which is used depends on the risk level and the time available.

- Statement coverage - each item is reviewed at least once
- Branch coverage - For each possible branch, traverse each path
- Predicate coverage - Consider all possible combinations of true / false values in a logical function.
- Handling loops - Try to skip the loop, execute once, twice, a typical number and the maximum number of times.

Stricter levels of coverage are available. See the references below for more information. [BEIZ90, BEIZ95, KANE99]

Coverage and Various Design Artifacts

Applying the above criteria is easily applied during design and code reviews. It is more difficult for other software development artifacts. For those documents written in a natural language, the criteria can be used as guidelines when the following are encountered:

- The words 'and' and 'or' are encountered in the document
- Statement containing the word 'if' or 'while'
- Statement contain the words 'for each' or 'every'
- Any item that contains an alternate way to accomplish an action

- If the item is in the scope of the review and has not been reviewed before, drilldown.
- If the item is out of the scope of the review and has been reviewed in a prior review, treat it as a black box. If it appears that the current input types were not covered in the previous review, a mini-review to drilldown in this area will need to be scheduled.
- If the item is out of the scope of the review and has not been reviewed, treat it as a black box. Pay close attention to the interfaces. Note the inputs and outputs that the current review item expects. During the review of this item, pull out the notes from the current review to insure that the item behaves as expected.

Many times it will be necessary to review an item several times. This is due to the fact that we are taking one logical path through the item. We will need to retrace our steps to take other logical branches through the item. Use general coverage criteria (see sidebar above) to decide which branch to take, focusing on the "normal" path.

Other activities may also be taking place during the review meeting. A traceability analysis is readily undertaken when one team member is marking which upstream item is covered during the current review item. Technical writing staff may be reviewing their documentation covering the material in the review, making notes as to items not covered or misrepresented. The creation of test cases, online help, error documentation, etc. is aided by identifying key areas brought to the surface early in the development cycle during the review. Just as important as what is reviewed is what isn't. If a portion of the review item is never "executed" this may indicate unnecessary functionality, "dead" code, or an error (the item really should have been covered...).

All participants should question the assumptions of the item currently being "executed" in the review. They should actively participate by asking questions that were developed during the pre-read or have just occurred to them. When appropriate, any issues should be raised. However, this is not the time to raise the "gotcha" issues -- save these for the stagger through.

Record Issues / Make Changes

A successful walk through will identify numerous issues. There should be a recorder in the review meeting. In some peer review methodologies, it is suggested that there be a scribe to record the issues who is not actively reviewing the material. While this allows all actively reviewing team members to concentrate on the material being reviewed (and not busy recording the issues), I believe that something is lost. Writing down the issue helps to solidify it in the mind of the writer. If the issue is unclear, the recorder can raise it at that time. If the recorder is an impartial scribe, they may miss the nuances of the issue. Therefore, I believe the author of the item should be responsible for recording the issues with the review item. The author's notes should be readily visible to the entire team. All other team members should take copious notes to keep everyone honest and to resolve ambiguities later on. Each team member will also use these notes during the stagger through meeting.

After the meeting, everyone who had issues identified in their work should start working on corrections before the stagger through meeting. This "updated" work will be used as the basis of the stagger through. This allows the changes to be reviewed by the team and see what implications arise from the changes made. Even those who did not have items identified in their deliverables may need to make updates. The review should be personally successful to them if issues identified in the review item shed light on potential issues in their own work.

Sufficient time should be given to make the necessary changes. However, too much time between the walk through and the stagger through meetings reduce the effectiveness of the process.

Therefore, the schedule should be aggressive to get the changes in. If the changes are so extensive that they cannot be made in a day or two, it may indicate that a new walk through is needed before the stagger through.

Follow-up before the Stagger Through

Preparation for the stagger through is similar to the preparation for the walk through review meeting. However, since the scope and the team are already selected, the preparation concentrates on the pre-read. Before the revised documents are available, the team members should review their notes and the original documents to identify items they want to clarify in the stagger through. Once the revised documents are available, the updated items should be pre-read again, focusing on the changes.

Since the focus of the stagger through are error and non-normal paths through the review item, the team members should allow their more sinister side to show. At this point, team members should identify those circumstances where the review item may break. [SHUL00] These "gotchas" will flesh out the stagger through.

Phase 2: The Stagger Through

The second phase, the stagger through, is the systematic review of a use case, design, code, etc. as the final system will execute it with a focus on the "non-normal" -- that is error and uncommon execution paths -- processing. The stagger through process is very similar to the process used in the walk through -- only the focus has shifted. In general, the following occurs.

1. Prepare for the Stagger Through (pre-read, review changes, create "gotchas")
2. Stagger through review meeting
3. Record issues
4. Determine next steps
5. Make changes, etc.

Preparation for the Stagger Through

After the walk through, the team begins to prepare for the stagger through as detailed in the section "Follow-up before the Stagger Through". There may be some additional preparation beyond the pre-read. During the walk through it may have become apparent that the review team did not have all of the knowledge necessary to perform an adequate review. In this case, the moderator may wish to expand the team. The new member(s) will need to come up to speed on the review items and what occurred in the walk through.

During the walk through, questions may be raised about items peripheral to the review that cannot be answered within the team. All effort should be made to find these answers before the review meeting. This will allow the team to have all of the knowledge needed to make the review effective. It also lets each team member know their input is important and will be followed up on.

Finally, the moderator should follow up with each team member and see if anyone has any suggestions on how to improve the next phase. Many times, it will be a suggestion not to spend as much time in one section. It is a judgment call whether to heed this suggestion. As always, the moderator must determine whether the change will allow the team to effectively find additional

issues. The moderator should follow up on all suggestions and publicize any changes before the stagger through review meeting.

Stagger Through Review Meeting

The stagger through meeting follows the logical control flow through the review item as was done during the walk through. Normal processing which has not changed since the walk through can be covered quickly. It should not be skipped entirely since team members may have discovered new issues with understanding they gained during and after the walk through review meeting. Normal processing that has changed since the walk through session should be reviewed more thoroughly.

The effectiveness of the stagger through lies in the change of focus. Hopefully, the entire team is satisfied that the item under review "works". (If this is not the case, perhaps the walk through or the changes requested were not detailed enough. The moderator must take care of this situation, perhaps by convening a new walk through before the stagger through may continue.) The team now shifts to trying to "break" this review item that they are convinced "works" by concentrating on the non-normal control flow through the review item. Each team member has prepared a list of "gotchas" to spring on the review item at the appropriate time.

The first few errors will usually take longer to work through than subsequent ones. During the first few errors, the team will be examining the error handling philosophy and how well it deals with fatal, severe, routine and trivial errors. Care should be taken to examine error propagation and following the error to the response to the user. Once the team is satisfied with the general error handling philosophy, the team can determine whether the error will be handled (by the generic error mechanism or by some special case logic) and whether this behavior is appropriate. So what kind of errors should be examined? Here are a few:

Data

- Nonsensical user inputs
- Corrupt input files
- Database or file system errors
- Boundary cases
- Duplicated data (in unique situations)

Runtime Issues

- Failed system function calls
- Out of memory
- Out of disk space
- Multi-threading issues - lock and race conditions
- Recursion issues
- Services not available

Security

- User or system has inadequate permissions
- System under attack

Date Related

(Many business systems perform special processing based on the time of the year)

- First day of the year
- Last day of the year
- Last day of a leap year
- Leap day
- First day of the month
- Last day of the month
- First day of the quarter
- Last day of the quarter
- Transition to daylight savings time
- Transition to standard time

Interface Related

- Interface not available
- Interface returns an error
- Interface fails to return or timeouts

Infrastructure Related

- Network down
- Site down or unreachable
- Firewall / Proxy server issues
- Device offline or unreachable
- Normal maintenance activities

These are just a few of the types of errors that can be examined during a stagger through that are often missed during other types of reviews.

Record Issues

The record issues step is identical to the same step in the walk through process. If the walk through / stagger through is to be considered a formal review, a report to management will need to be created.

Determine Next Steps

The moderator and the team must decide what the next steps should be for the review items. If the stagger through identifies substantial issues that must be addressed, a new walk through / stagger through process will need to be scheduled. If the issues identified are more local in scope, another stagger through review meeting may be scheduled. If all of the issues are determined to be minor, another review session is probably not necessary. It is recommended that at least one team member read through the revised review item to make sure that all of the issues are adequately addressed.

Make Changes

The output of the stagger through will be a list of issues that need to be resolved. The author of the review item will need to address these issues in his deliverable. Even more so than the walk through, team members should have a list of items that need to be addressed in their own, yet-to-be-reviewed deliverables. Especially early in the development phase when many items have not been reviewed, stagger throughs will identify deficiencies in error handling and propagation. Consistency issues in how these items are dealt with will need to be addressed.

Case Study: Partes Corporate

The most successful implementation I have seen of this process was several years ago at Partes Corporation. Partes Corporation (since acquired by EDGAR Online) created software to enable the retrieval and analysis of SEC filing data in many different formats. I was recruited into the company as manager of quality. Besides the CTO and myself, most of the development team members had less than a year of professional development experience. The product was an internet enabled add-in to Excel to allow users to retrieve parsed SEC data from the Partes web accessible datastore and perform various analyses on the downloaded data.

A spiral development lifecycle was used with at least three iterations planned. The first iteration was to enable the selection and downloading of files into an Excel workbook. The subsequent iterations would add basic time series analysis functionality followed by more complex analysis. For the first cycle, the development plan called for about a third of the time spent in design, third in implementation and a third in testing. All of the staff had the theoretical knowledge to perform these tasks but perhaps not the practical experience to pull it off. We decided to reduce the risk by using the walk through / stagger through process on all design artifacts. The plan was to walk through the design in the same order as execution. For the first iteration, we decided on following logical flow through the application: Launch the Excel add-in, select a company, download some filings, save the Excel workbook and the load a saved analysis.

The first session involved explaining the process and starting with the initialization of the add-in. This particular session did not last long as the team had left out the mundane details of initialization. The existing design jumped right into the details of the functionality. The team's expectation of what was needed was reset and they were given a couple of days to make corrections. Once we got past the initialization review (which we would return to time and time again each time a data element was first referenced) the fun began.

In the session where we were tracing the flow of selecting a company we decided to drilldown to every element. This required many developers to be ready to have their items reviewed. The process of loading the company search dialog and constructing the request to the datastore took two sessions. Many interface issues were discovered along with some missing functionality. All in all, the walk through portion of selecting a company took nearly eight times longer than we expected. The team felt a large sense of accomplishment after completing the walk through and had a much better idea of what was required for professional application. After making the necessary corrections identified in the walk through, we were ready to start the stagger through.

Soon after introducing the first "non-normal" processing condition, it was apparent to all that a large portion of the design was missing. What error handling that was designed did not take into account the need to propagate the error up the calling chain (and eventually to the user

interface). There was also no clear agreement among the team which errors were fatal and which were recoverable. The team requested a week to rework the error handling strategy for the product and integrate it into the design. When the reviews reconvened, the increase in the quality of the design was immediately evident. What had not been taken into account was the sheer number of things that could go wrong in the application. All developers were seen immediately expanding their design to take into account new classes of errors that were exposed in the review meeting.

This process continued through the logical flow. We would often revisit reviewed items as questions arose. Sometimes these would result in revisions to the previously reviewed items. This was especially true of the initialization routine. The design process ended up taking nearly twice as long as we expected. The schedule had been adjusted using the same proportions mentioned above. We extended coding and testing by about twice as long. To our surprise, coding went incredibly fast. This was due primarily to the fact that most of the incongruities that would become apparent in coding has already been discussed and resolved during the review meeting. Testing went smoothly. There were few integration issues. Most of the changes introduced during testing were related to usability or special circumstances arising from the SEC data. In subsequent projects with this team, we did not drilldown to the same level of detail. This was due to the increase in experience. We did get burned a few times and wished we had gone into more detail.

Conclusion

The walk through / stagger through review methodology can be effectively used to verify the correctness and completeness of a development artifact. It is especially useful for teams with little development experience, experience in a technology or experience working together. The segregation between algorithmic correctness and error handling allows the team to focus on one at the exclusion of the other. This results in a more thorough review than trying to concentrate on both at the same time. Traditional review techniques can be supplemented with this approach based on team experience and risk analysis.

References

- BEIZ90 Beizer, Boris. Software Testing Techniques, 2nd Edition. New York: Van Nostrand Reinhold, 1990.
- BEIZ95 Beizer, Boris. Black-Box Testing, New York: John Wiley & Sons, Inc., 1995.
- BOEH87 Boehm, Barry. "Industrial Software Metrics Top 10 List." *IEEE Software*. (v4, n9, September 1987), pp. 84-85.
- FREE90 Freedman, Daniel P., and Weinberg, Gerald M. Handbook of Walkthroughs, Inspections, and Technical Reviews. New York: Dorset House Publishing Co. Inc., 1990.
- KANE99 Kaner, Cem, Falk, Jack, and Nguyen, Hung Quoc. Testing Computer Software, 2nd Edition. New York: John Wiley & Sons, Inc. 1999.

- MCCO96 McConnell, Steve. Rapid Development. Redmond, WA: Microsoft Press, 1996.
- MYER79 Myers, Glenford J. The Art of Software Testing. New York: John Wiley & Sons, 1979.
- SHUL00 Shull, Forrest, Rus, Ioana, and Basili, Victor. “How Perspective Based Reading Can Improve Requirements Inspections.” *IEEE Computer*. (v33, n7, July 2000), pp. 73-79.

About the Author

Michael Ensminger is Director of Quality Assurance at PAR3 Communications based in Seattle, WA. Prior experience (both management and practitioner of test and development teams) includes Internet, shrink-wrap and niche retail banking software. He holds a M.S. in Computer Science from University of Texas at Dallas.

© Copyright 2001 Michael Ensminger. All Rights Reserved.