



Software Quality Processes at Digital Mesh

Digital Mesh Softech India (P) Limited

www.digitalmesh.com



1. Scope

The scope of this document is to define the following processes in the software development lifecycle.

- Review
- Version Control
- Defect Tracking
- Post-mortem Analysis
- Weekly Project Meeting
- Recording of Work
- Backup

All team members at Digital Mesh are required to clearly understand and regularly follow these processes for all CLIENT software development projects. This will help in improving the quality of the software products.

2. Introduction

The concept of **quality** can be summed up in the following three golden rules:

- Quality implies 100% conformance to requirements
- Required level of quality must be zero-defects
- Quality system should be prevention-based rather than appraisal-based

We will define some terms used in this document:

- **Procedure:**

A procedure represents the stepwise tasks that are necessary to produce the product(s) specified by the process.

- **Process:**

A process is an organized method for performing work. Effective processes require procedures to do work, and procedures to check work.

- **Reviews:**

A review is a quality control technique used in software development that relies on individuals other than the author(s) to evaluate the quality of a product.

3. Coding Standards

The Cold Fusion, ASP, PHP and VB coding standards are available on request and the HTML and the Database coding standards are also documented.



4. Software Development Lifecycle Stages

The following table lists out the various stages in the software development lifecycle and the deliverables to be produced at each stage along with the corresponding processes applicable to that stage:

Lifecycle Stage Deliverables Process

- Definition and Planning Product Specifications
- Software Development Plan
- Subcontract Plan
- Software Requirements Specification
- Design Stage Plan
- Risk Analysis and Management Plan
- Document Review
- Design Software Design
- Design Review Report
- Software System Design Verification Plan
- Implementation Stage Plan
- Implementation Source Code
- Code Review Reports
- Design Verification Stage Plan
- Code Review
- Design Verification Binaries
- Design Verification Report
- Defects Database
- Type Approval and Release Stage Plan
- Defect Tracking
- Type Approval Acceptance Test Report
- Release Release Guidelines
- Software Maintenance Document
- Post-mortem Post-mortem Report Post-mortem Analysis

NOTE: In addition, the version control and backup processes are applicable to each deliverable in every stage.



5. Daily Activities

The following activities should take place on a daily basis.

- Each engineer should maintain a separate register (work-book) for each project.
- Each engineer should record the amount of time spent on various activities. Please refer to section 12 (Recording of Work).
- Backups of the entire project tree on each engineer's client machine as well as the project archives on the server machine should be taken for every project.

6. Document Organization

This section gives an overview of the organization of the remaining sections in this document.

Section 7 describes the review process for both code and documents and also defines the formats for the review results.

Section 8 lays down the guidelines for effective use of a version control system.

Section 9 proposes a first-cut defect tracking system, which is likely to be modified subsequently after evaluating other such systems in use at DIGITAL MESH.

Section 10 lists the various activities that should be carried out during a post-mortem analysis.

Section 11 sets out the agenda for weekly project meetings. This section also gives some guidelines for preparing project schedules and reporting weekly project status using Gantt charts.

Section 12 describes the purpose of the work-recording activity and lists down the various heads under which work should be reported.

Section 13 describes the backup process.

7. Review Process

The following deliverable components are subject to the review process:

- Documents
- Code

The review process is intended to be an aid to achieve our quality objectives.

7.1 Purpose

The purpose of a review is different when reviewing documents or code. The review process is not meant to discuss specific solutions to any defects (such issues should be discussed in project meetings).



7.1.1 Document Review

The purpose of the document review process is to ensure the following:

- Document uses correct template
- Does not have punctuation or spelling errors
- Contents are correct and meaningful
- Guarantee conformance to requirements through a table which establishes a correspondence between the requirements and the different sections of the document

7.1.2 Code Review

The purpose of the code review process is to ensure the following:

- Coding standards are followed
- Code matches the design
- Logic does not have any defects
- Use of version control procedures
- Guarantee conformance to requirements through a table which establishes a correspondence between the requirements and the different modules/files

7.2 Who Co-ordinates

The Project Manager of the concerned project bears the responsibility of co-ordination of the review process.

7.3 Who Participates

The producer(s) of the document and the reviewers are required to participate in this process. It is recommended that for a given project, the same team participates in reviewing the documents as well as the code, but this is not mandatory.

7.4 Duration

The duration of a review meeting should be limited to 1.5 hours. If it requires more time, then the meeting should be split into two or more sessions.

7.5 When and How Often

It is recommended that documents be reviewed twice - first at the draft version stage and then at the final version stage.

For source code, it is recommended that a weekly review be done for the modules completed the previous week rather than holding a consolidated review for the entire source code at the end of the coding stage.

7.6 Planning for the Review

The review coordinator performs this step at least one week before the scheduled review. The review team is selected from among peers of the producer of the document. The average team size of the review team should be about 2-4 members. The following roles are assigned to individual members:



- **Moderator** - manages the review process and is accountable for its effectiveness
- **Recorder** - classifies errors during the review meeting and records them
- **Producer** - who is the author of the document, distributes material for review
- **Reviewer** - all participants, including those assigned specific roles listed above
- **Onlooker** - any interested person who does not play an active role in the review

Normally the QA engineer for the project should be one of the reviewers. Onlookers are optional members on the review team; they are primarily persons who are being trained for the review process.

The review coordinator should ensure that all potential reviewers have been properly trained in the objectives and processes of reviews.

Finally the review meeting is scheduled and a meeting notice is issued to all participants.

7.7 Distribution of Review Material

The producer distributes the document to be reviewed to all the reviewers 3 or more working days prior to the review meeting. This version must be checked out of the version control system.

The review package consists of printouts of the documents being reviewed as well as the input to the design/coding stage i.e. printout of the requirement/design document. The expected preparation time for the review is also to be mentioned.

The producer may, in advance, provide a broad overview of the document, and request the reviewers to concentrate on certain specific areas.

7.8 Perform Individual Reviews

Each of the reviewers performs an individual review of the given document. The individual review should be performed in one continuous time span.

The reviewers should pay particular attention to the following points.

For code:

- Does the code implement the design and the specification?
- Is the implementation clear and understandable?
- Are there logical errors?
- Is the structure sound?
- Is the commenting clear, complete and correct?
- Is the coding style up to standard?
- Are there maintenance pitfalls?
- How can the implementation be tested?
- Are there memory leaks?
- Are exceptional conditions handled?



For documents:

- Style and grammatical accuracy, particularly during reviews of final versions.
- Proper formatting used, particularly during reviews of final versions.
- Correctness of content, particularly during reviews of draft versions.

For test scripts and results:

- Test coverage and applicability.

Each reviewer records defects, questions and concerns to be addressed during the review meeting on the review material (i.e. printouts).

Each reviewer also records the total review time, including a breakdown by individual files.

7.9 Conduct Review Meeting

The moderator ensures that the review meeting starts at the scheduled time and that all reviewers are present. He/she also determines whether all reviewers have performed their preparatory work.

If not, the meeting should be re-scheduled so that the reviewers are ready for the meeting.

The review team, led by the moderator, goes through the document/source file being reviewed

section-wise or function-wise. In every section/function, the moderator encourages all reviewers

to point out and discuss any defects noted by them. The review team then discusses these defects or concerns.

The moderator tries to achieve a consensus to determine if the issue under discussion is a defect or not. The producer's view should also be taken for clarification. The moderator has the final say (in absence of a consensus) in determining whether the issue is a defect or not.

7.10 Recording of Defects

Once a defect has been determined, the recorder records the defect by location, narrative description and classification of the defect in the Review Defect List, whose format is given at the end of this chapter.

After all sections of the document have been reviewed, the moderator receives the Review Defect List from the recorder. The moderator then completes the Review Results document (as per the format given at the end of the chapter) which is then given to the producer(s) for rework. The recording of defects is different for document review and code review.



7.10.1 Classification of Defects in Code

The classification of defects in code review is done as follows:

- **Severity**

- Major: will cause observable product failure or departure from design
- Minor: will not cause failure of the product

- **Type**

- Interface: operations with unsupported library functions
- Capabilities: implementation of requirements not correct
- Descriptive: comments in code, naming of identifiers
- Performance: degradation of performance
- Test: difficult to prove correctness
- Maintenance: difficult to maintain
- Standards: does not follow standards
- Query: open issue, needs further exploration
- Porting: may not be portable across platforms

- **Category**

- Missing: material, which as per design should be in the code but isn't
- Extra: material, which should not be included in code as per design
- Wrong: material that is and should be present, contains an error

7.11 Analysis of Review Results\

The moderator should determine the number of defects for each type, category and severity of code defects. The findings of the review should also be compared with those of previous reviews.

7.12 Rework

Rework on the document/code, as per the Review Results is the responsibility of the producer(s). The author also marks the corresponding Status fields in the Review Defect List as "Done".

7.13 Follow up

When the rework has been completed, the moderator along with the QA person verifies and certifies that all defects have been corrected by marking the corresponding Status fields in the Review Defect List as "Verified".

7.14 Formats

This section defines the formats for recording the review results.



7.14.1 Format of the Code Review Results document

Section A
Project Name
Names of files reviewed
Moderator
Initials of reviewers
Total no. of lines in files
Total preparation time
Review duration

Section B

Review Defect List (XLS file)
Defect ID File Ver Line Type Class Severity Notes Status
Defect ID Serial no. of defect
File Name of the file
Ver Version of the file
Line Line number or range of lines
Type Classification as described in 7.10.1
Class Classification as described in 7.10.1
Severity Classification as described in 7.10.1
Notes Description of defect
Status Done - correction has been implemented
Verified - implementation of correction has been verified

7.14.2 Format of the Document Review Results document

Section A

Project Name
Names of files reviewed
Moderator
Initials of reviewers
Total no. of lines in files
Total preparation time
Review duration

Section B

Review Defect List (XLS file)
Defect ID File Ver Line Notes Status
Defect ID Serial no. of defect
File Name of the file
Ver Version of the file
Line Line number or range of lines
Notes Description of defect
Status Done - correction has been implemented
Verified - implementation of correction has been verified



8. Version Control System

Version control is the process of managing and maintaining multiple revisions of the same file in an archive. An archive contains information about each revision of the file, allowing members of the development team to retrieve, modify, and return any revision of a file in a safe, organized and consistent manner.

In any development environment, accurately tracking and recording changes to project files is essential. By tracking these changes with version control, one can create a copy of a file at any point in its development history, and get specific information about each change made to the file.

The risk of overwriting changes made by another developer or of losing data is minimized as all changes and information are archived and are easily accessible.

The PVCS Version Manager is the preferred version control system to be used for any project. This section gives the guidelines and policies to be followed for using any version control system.

8.1 User accounts

At the beginning of every software development project, a separate project must be created in the version control system. User accounts should be created for all team members of the project, so that they can access the project archives under the version control system with appropriate access rights. In order to access the version control system over the network, each team member must run the appropriate setup program on his/her workstation.

8.2 What to version control

All documents and files produced during the project should be version-controlled. Text files, including source code, should have appropriate keywords inserted in their contents so as to enable automatic tracking of versions by the version control system.

The following list shows the keywords that can be inserted in workfiles to obtain archive information and the kind of information the keywords reference.

- \$Archive\$ The full path name of the archive.
- \$Author\$ The user ID of the author of the revision.
- \$Date\$ The check-in date of the revision.
- \$Header\$ The archive name, revision number, revision date, and author ID.
- \$Log\$ Cumulative check-in messages.
- \$Modtime\$ The time of the last modification.
- \$Revision\$ The revision number.
- \$Workfile\$ The workfile name stored in the Version Manager archive.

8.3 Check-in, check-out policy

Every file should be checked into the version control system at the time of its creation with an initial version number. A team member who wishes to modify the file should check out the file from the version control system as writable with Lock. If the user wishes to only view the file but not modify it, it should be checked out as Read-only. When a user has finished modifying the file, it should be checked in with appropriate comments highlighting the change.



Once the initial draft version of any file has been completed, it should be checked in. Before checking in any document, it should be ensured that it has been checked for spelling and grammatical errors. Similarly error-free compilation of any source file should be ensured before checking it in. The rule of thumb for frequency/granularity of checking in further modifications is as follows:

If changes done can be described in couple of lines, the file should be checked in as a new version before making subsequent changes. It is also recommended that the versions of all files be frozen before any code review process or any deliverable to the customer.

8.4 Using Version Labels

A version label is a symbolic name that can be assigned to a group of files stored in the archives. Version labels are typically used for identifying a version, or specific release, of a software product. A version label is particularly useful when one wants to perform actions on multiple files simultaneously, such as checking files in or out, generating reports about revisions, or building an application.

Whenever a deliverable has to be shipped to the customer, all files (sources as well as documents) related to that deliverable should be assigned a common meaningful version label, which serves to identify the version of that particular deliverable.

8.5 Branching of Revisions

A branch is a separate line of development consisting of one or more revisions that diverge from a revision on the trunk, or another branch. Branching can be used to develop alternate variations of a file in parallel with other users who are working on the same file.

Typical reasons for creating a branch are:

- To try out changes that should not immediately affect the main line of development, such as a bug fix.
- Work on a new version of a software product has started, or files have to be ported to a new operating system, and the work on this revision is to be kept separate from the current line of development.
- A branch revision can be merged back into the trunk at any time. For example, if a temporary branch has been created to try out bug fixes, the branch tip revision can be merged back into the trunk after the bug fix is complete and verified.

8.6 Merging of Revisions

Merging is the process of combining two sets of changes to create a new file. Merging is useful for restoring a branch line of development to the main line.

Merging combines two files or revisions without losing data. If the version control system encounters conflicting differences, it marks the place in the merged file where the conflict occurs. The user should review the conflict and modify the output file before checking it in. For example, if the same line has been changed in both files, both are reported and it is upto the user to determine which version of the line to discard.



9. Defect Tracking System

In the design verification phase of a project, the software product is subjected to final testing as per the test plan. All defects that are discovered during this phase as well as any defect that is reported afterwards should be tracked using a proper defect tracking system. The aim of the defect tracking system is to ensure that the following steps are carried out for any defect:

- Defects are reported and recorded in a consistent manner
- A reported defect is investigated by the development team, which then either rejects it with an appropriate explanation or accepts it with appropriate classification
- Once accepted, a development team member is assigned the responsibility of identifying and implementing a fix for the defect
- The implementation of the fix is reviewed and then it is verified that the fix has actually solved the reported defect
- The fix is incorporated in a new version for release

It is proposed that we implement an electronic defect tracking system for maintaining a history of defects (and their corresponding status) related to a particular project. For each project, an Excel file will be maintained and information related to every defect reported for that project will be posted to the file in the following format:

ID Desc Origin Date Cause Class Action Pers Resp Status&Date

- ID A unique ID for the defect
- Desc Description of the defect
- Origin Person reporting the defect
- Date When the defect was first reported
- Cause Probable cause of the defect
- Class Classification of the defect as explained in the section below
- Action Recommended action for the defect
- Digital Mesh Software - Confidential 1 February, 2001 Page 17 of 22
- Pers Resp Initials of person who will fix the defect
- Status&Date Status of the defect as on a particular date
- If any document or source code file(s) have to be modified to take care of the defect, the defect
- ID should be mentioned in the PVCS header of the file.

9.1 Classification of Defects

The following criteria are used to classify defects in order to indicate their importance.

Class A

- the defect is a serious bug
- the product does not meet one of its major requirements
- the defect can cause the product to be rejected
- the defect needs to be fixed on an urgent basis



Class B

- the defect is not serious
- a minor requirement of the product is not met
- there exists a workaround for the defect
- the defect occurs rarely or causes only a minor irritation

Class C

- this is an enhancement not a bug; it is recorded so that next version may incorporate it

10. Post-mortem Analysis

After a software project is completed it enters the post-mortem phase. During this phase the project manager and the project team members should get together and carry out a post-mortem review.

The post-mortem review looks back at the project with the primary aim of providing feedback to future projects. The following is an indicative list of some of the important information that should come up in the post-mortem analysis:

- Comparison of original planned dates with actual completion dates
- Significant reasons for deviations between the planned and actual performances
- Original planned versus actual cost and effort analysis
- Analysis of recording of work to determine the ratios of times spent in various activities
- Analysis of the code review findings to determine the trend of defects with respect to time for each team member
- Analysis of the code review findings to indicate which type of defects are predominant, so that appropriate training can be given to the team members
- To come up with metrics such as defects found per thousand lines of code, number of lines coded per day, total number of lines of code, functions

11. Weekly Project Meeting

The Project Manager is responsible for holding a weekly project meeting for every project being managed by him/her. The day and time of the weekly project meeting should be decided at the beginning of every project and should be communicated to all team members.

The Project Manager should convene and chair the meeting on the scheduled day and time every week for the entire duration of the project. All team members of the project should attend the meeting.

The following is an indicative checklist that should be used as an agenda for the weekly project meeting. It is not mandatory for the proceedings of the meeting to be recorded in any prespecified format.



11.1 Agenda for Weekly Project Meeting

- Review Project Status:
 - activities scheduled for the previous week
 - activities actually completed
 - tasks which have slipped and reasons for slippage
 - plans for next week
 - any dependencies on other tasks affected by slipped tasks
 - update progress report
 - identify risks and recovery plan for risks
- Review of where project is headed in the long-term.
- Client Interaction: - weekly progress report (including next drop and long-term)
 - client dependencies
 - conference calls
- Project Logistics: - last version backup and daily backups
 - source code control
 - adherence to GUISE (GUIDelines for Software Engineering)
 - interaction with Software Services Group (SSG)
- Software Standards: - standards enforcement within group
 - presentations
- Project Staffing: - need to add/reduce engineers
 - leave (planned as well as unplanned)
- Project Training: - for engineers and QA
 - Resources: - Software
 - PCs and workstations
 - PC accessories
 - email related
- Admin issues impacting the engineers:
 - office facilities
 - equipment repair/replacement
 - policies
- Any other topics for discussion.

11.2 Weekly Status Report (Gantt Charts)

The schedule/plan for each project should be made in the form of a Gantt chart using MSProject. At the time of making the project file, a calendar should be defined for the project that takes into account the national holidays for the year. A list comprising of all the resources (persons) that will be involved in the project should also be prepared. A calendar for each resource should be maintained to take into account the planned absences (leaves) of that resource.

The key aspect while making a schedule for a project is to identify the various tasks comprising the project. For every task in the schedule, the Gantt chart should have the following columns:



- Task Name - maximum 50 characters with appropriate abbreviations
- Duration
- Start date
- Finish date
- Percentage completed
- Resource name - who is/are responsible for performing the task; use only initials for names

The various tasks should be grouped as per the different phases of a project, with the key tasks in each phase being broken down into subsidiary tasks. The breaking down of each task should be such that no sub-task has duration of greater than 2-3 days. Each task name should contain a numeric string (PhaseNo. KeyTaskNo. SubTaskNo format) which indicates its position in the hierarchy of tasks. Each task should be properly linked to its predecessor task. Milestone tasks should be specified separately.

Once a schedule has been made it should be base lined. If subsequently, some tasks need to be added to the original plan, only the tasks so added need to be saved in the baseline plan. The purpose of base lining is to provide a reference so that we can compare the actual time spent on various tasks/phases with the original estimated time.

As the project progresses, the plan should be tracked (by using the Tracking option of MSPProject) at the beginning of every week. Tasks that have been completed the previous week should be updated with the actual start and finish dates. Tasks that are ongoing should be updated with the percentage (estimated) of work completed in that task. This updated plan should be sent to the client as the weekly status report of the project.

The following is the list of standard bar styles to be used for depicting various types of information in the Gantt chart:

- Milestone Task
- Summary Task
- Baseline
- Progress

12. Recording of Work

Recording of work is an activity wherein all engineers in a project record the amount of time spent by them on different types of activities on a daily basis. The purpose of this recording is to have an actual real-life database, which over a period of time will give a true picture of the actual ratios of time spent on the various phases of the software development life cycle. This will facilitate the task of making better work/time estimates in future projects.

Hence it is essential that the recording of work be done conscientiously so as to reflect reality as closely as possible; otherwise the purpose of the activity is rendered worthless. It is emphasized that this activity is not intended to monitor the time of the employees. Hence it is not at all necessary that an employee's work activity report for a day should show a total of 8 hours of work. It may contain significantly less hours or perhaps even more than 8 hours; as long as the report correctly lists down the actual times spent on different activities, it's ok.



The work activity report for a particular day may be filled in either at the end of the day or at the start of the next working day. The exact mechanism (electronic/paper) for doing this is yet to be finalized. The activities should be recorded with a granularity of one-hour time spans. Every entry in the daily activity report should contain the project name, the stage of the software development lifecycle, the type of activity, and the time spent. The header of the report should contain the date of the report and the name of the concerned engineer.

The various categories of activities under which the recording of work should be done are described below:

- Documentation
- Reading
- Experimentation
- Design
- Coding
- Integration
- Testing
- Debugging
- Maintenance
- Rework
- Lifecycle (Project Management Related) – preparing weekly status reports, participating in weekly project meetings, review process.
- Supporting previously developed products/projects
- Other

The work activity reports from all team members will be collected for further analysis. The statistics generated during analysis will include actual person-hours spent for each category during all stages of the project.

13. Backup Process

13.1 What Is Backup

Backup is an activity wherein one archives, at various points of time during the lifecycle of a project, the documents/files related to the project. The purpose of backup is to insure oneself against loss/corruption of data due to circumstances such as failure of hard disk, computer breakdown etc. In case of such catastrophic events, one does not have to start from scratch; one can always fall back on the last set of backed up files.

13.2 What to Backup

The following are the types of backups:

- Daily - Monday to Friday
- Weekly - Week 1 to 4
- Monthly - Last 2 months
- Milestone - Each milestones of the project



13.2.1 Project Tree

The entire project tree, consisting of its subdirectories and files therein, should be backed up. If the project tree is split across more than one computer, all branches of the tree on the concerned computers should be backed up.

The files to be backed up should include the following:

- all documents produced during the various lifecycle stages
- all correspondence with the client, including weekly status reports
- sources
- make files
- batch files
- script files
- installation related files

13.2.2 Tools - Compiler/Linker versions

All the software tools required to generate binaries from the source code, which are supplied by the client and/or are not part of the company's software store, should be backed up. A backup of such tools should be taken for each version of the tool used in the project. Examples of such tools are compilers, linkers.

13.2.3 Third Party Tools

Any third party library being used in the project should be backed up.

13.3 Who Is Responsible

One of the project team members should be assigned the responsibility for taking daily, weekly, monthly and milestone backups for that project. In absence of this person, at least one more person should be identified as backup person.

13.4 Procedure for taking backups

The backup is taken on a tape device using a batch or script command.

Digital Mesh Softech India (P) Limited

43-A, E Block, 2nd Floor, Cochin Special Economic Zone, Kakkanad, Kochi – 682 037, Kerala, India .

Tel: +91-484-4060200, Fax: +91-484-4060201, e-mail : info@digitalmesh.com

Website: www.digitalmesh.co.in