LEARNER MANUAL

PRODUCE DOCUMENTATION FOR A COMPUTER PROGRAM TO AGREED STANDARDS

US ID: 115388

NQF LEVEL: 5

CREDITS: 3

NOTIONAL HOURS: 30

Contents

This manual was compiled by SM Support on behalf of the training provider.

HOW TO USE THIS GUIDE

This workbook belongs to you. It is designed to serve as a guide for the duration of your training programme. It contains readings, activities, and application aids that will assist you in developing the knowledge and skills stipulated in the specific outcomes and assessment criteria. Follow along in the guide as the facilitator takes you through the material, and feel free to make notes and diagrams that will help you to clarify or retain information. Jot down things that work well or ideas that come from the group. Also, note any points you would like to explore further. Participate actively in the skill practice activities, as they will give you an opportunity to gain insights from other people's experiences and to practice the skills. Do not forget to share your own experiences so that others can learn from you too.

ICONS

HOW YOU WILL LEARN

The programme methodology includes facilitator presentations, readings, individual activities, group discussions, and skill application exercises.

PROGRAMME OVERVIEW

PURPOSE

Qualifying learners are able to:

- Plan documentation for a computer program to agreed standards
- Create documentation for a computer program to agreed standards
- Review documentation for a computer program for completeness

LEARNING ASSUMPTIONS

The credit value of this unit is based on a person having the prior knowledge and skills to:

- Apply the principles of Computer Programming
- Design, develop and test computer program segments to given specifications

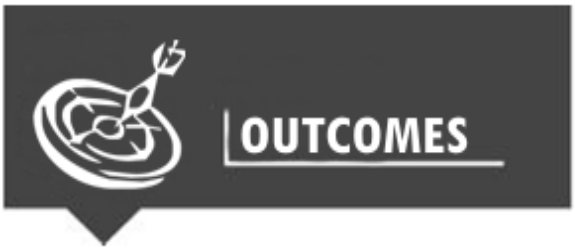
HOW YOU WILL LEARN

The programme methodology includes facilitator presentations, readings, individual activities, group discussions, and skill application exercises.

HOW YOU WILL BE ASSESSED

This programme has been aligned to registered unit standards. You will be assessed against the outcomes of the unit standards by completing a knowledge assignment that covers the essential embedded knowledge stipulated in the unit standards. When you are assessed as competent against the unit standards, you will receive a certificate of competence and be awarded 3 credits towards a National Qualification.

SESSION 1 PLAN AND DESIGN DOCUMENTATION



On completion of this section you will be able to Plan and design documentation for a computer program to agreed standards.

1. The documentation design covers the format of the documents in line with industry conventions.
2. The documentation plan covers full program documentation components.

1.1 INTRODUCTION

All large software development projects, irrespective of application, generate a large amount of associated documentation. For moderately sized systems, the documentation will probably fill several filing cabinets; for large systems, it may fill several rooms. A high proportion of software process costs is incurred in producing this documentation. Furthermore, documentation errors and omissions can lead to errors by end-users and consequent system failures with their associated costs and disruption.

Therefore, managers and software engineers should pay as much attention to documentation and its associated costs as to the development of the software itself. The documents associated with a software project and the system being developed have a number of associated requirements:

1. They should act as a communication medium between members of the development team.
2. They should be a system information repository to be used by maintenance engineers.
3. They should provide information for management to help them plan, budget and schedule the software development process.
4. Some of the documents should tell users how to use and administer the system.

Satisfying these requirements requires different types of document from informal working documents through to professionally produced user manuals. Software engineers are usually responsible for producing most of this documentation although professional technical writers may assist with the final polishing of externally released information.

1.2 TYPES OF DOCUMENTATION

We shall discuss the various types of program documentations;

PROCESS AND PRODUCT DOCUMENTATION

For large software projects, it is usually the case that documentation starts being generated well before the development process begins. A proposal to develop the system may be produced in response to a request for tenders by an external client or in response to other business strategy documents. For some types of system, a comprehensive requirements document may be produced which defines the features required and expected behaviour of the system. During the development process itself, all sorts of different documents may be produced – project plans, design specifications, test plans etc. It is not possible to define a specific document set that is required – this depends on the contract with the client for the system, the type of system being developed and its expected lifetime, the culture and size of the company developing the system and the development schedule that it expected.

However, we can generally say that the documentation produced falls into two classes:

1. Process documentation

   These documents record the process of development and maintenance. Plans, schedules, process quality documents and organizational and project standards are process documentation.
2. Product documentation

   This documentation describes the product that is being developed. System documentation describes the product from the point of view of the engineers developing and maintaining the system; user documentation provides a product description that is oriented towards system users.

Process documentation is produced so that the development of the system can be managed. Product documentation is used after the system is operational but is also essential for management of the system development. The creation of a document, such as a system specification, may represent an important milestone in the software development process.

Process documentation

Effective management requires the process being managed to be visible. Because software is intangible and the software process involves apparently similar cognitive tasks rather than obviously different physical tasks, the only way this visibility can be achieved is through the use of process documentation. Process documentation falls into a number of categories:

1. Plans, estimates and schedules. These are documents produced by managers which are used to predict and to control the software process.
2. Reports. These are documents which report how resources were used during the process of development.
3. Standards. These are documents which set out how the process is to be implemented. These may be developed from organizational, national or international standards.
4. Working papers. These are often the principal technical communication documents in a project. They record the ideas and thoughts of the engineers working on the project, are interim versions of product documentation, describe implementation strategies and set out problems which have been identified. They often, implicitly, record the rationale for design decisions.
5. Memos and electronic mail messages These record the details of everyday communications between managers and development engineers.

The major characteristic of process documentation is that most of it becomes outdated. Plans may be drawn up on a weekly, fortnightly or monthly basis. Progress will normally be reported weekly. Memos record thoughts, ideas and intentions which change. Although of interest to software historians, much of this process information is of little real use after it has gone out of date and there is not normally a need to preserve it after the system has been delivered. However, there are some process documents that can be useful as the software evolves in response to new requirements. For example, test schedules are of value during software evolution as they act as a basis for re-planning the validation of system changes. Working papers which explain the reasons behind design decisions (design rationale) are also potentially valuable as they discuss design options and choices made. Access to this information helps avoid making changes which conflict with these original decisions. Ideally, of course, the design rationale should be extracted from the working papers and separately maintained. Unfortunately this hardly ever happens.

PRODUCT DOCUMENTATION

Product documentation is concerned with describing the delivered software product. Unlike most process documentation, it has a relatively long life. It must evolve in step with the product which it describes. Product documentation includes user documentation which tells users how to use the software product and system documentation which is principally intended for maintenance engineers.

User Documentation

Users of a system are not all the same. The producer of documentation must structure it to cater for different user tasks and different levels of expertise and experience. It is particularly important to distinguish between end-users and system administrators:

1. End-users use the software to assist with some task. This may be flying an aircraft, managing insurance policies, writing a book, etc. They want to know how the software can help them. They are not interested in computer or administration details.
2. System administrators are responsible for managing the software used by end-users. This may involve acting as an operator if the system is a large mainframe system, as a network manager is the system involves a network of workstations or as a technical guru who fixes end-users software problems and who liaises between users and the software supplier.

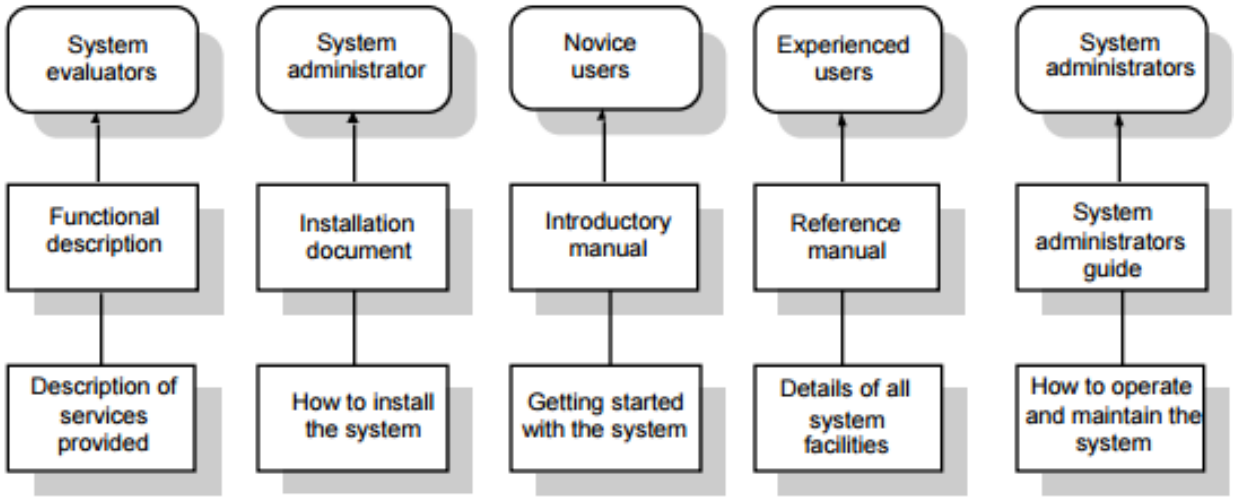


Figure 1: Different types of user documentation

To cater for these different classes of user and different levels of user expertise, there are at least 5 documents (or perhaps chapters in a single document) which should be delivered with the software system (Figure1). The functional description of the system outlines the system requirements and briefly describes the services provided. This document should provide an overview of the system. Users should be able to read this document with an introductory manual and decide if the system is what they need. The system installation document is intended for system administrators. It should provide details of how to install the system in a particular environment. It should contain a description of the files making up the system and the minimal hardware configuration required. The permanent files which must be established, how to start the system and the configuration dependent files which must be changed to tailor

the system to a particular host system should also be described. The use of automated installers for PC software has meant that some suppliers see this document as unnecessary. In fact, it is still required to help system managers discover and fix problems with the installation. The introductory manual should present an informal introduction to the system, describing its 'normal' usage. It should describe how to get started and how end-users might make use of the common system facilities. It should be liberally illustrated with examples. Inevitably beginners, whatever their background and experience, will make mistakes. Easily discovered information on how to recover from these mistakes and restart useful work should be an integral part of this document. The system reference manual should describe the system facilities and their usage, should provide a complete listing of error messages and should describe how to recover from detected errors. It should be complete. Formal descriptive techniques may be used. The style of the reference manual should not be unnecessarily pedantic and turgid, but completeness is more important than readability. A more general system administrator's guide should be provided for some types of system such as command and control systems. This should describe the messages generated when the system interacts with other systems and how to react to these messages. If system hardware is involved, it might also explain the operator's task in maintaining that hardware. For example, it might describe how to clear faults in the system console, how to connect new peripherals, etc. As well as manuals, other, easy-to-use documentation might be provided. A quick reference card listing available system facilities and how to use them is particularly convenient for experienced system users. On-line help systems, which contain brief information about the system, can save the user spending time in consultation of manuals although should not be seen as a replacement for more comprehensive documentation.

SYSTEM DOCUMENTATION

System documentation includes all of the documents describing the system itself from the requirements specification to the final acceptance test plan. Documents describing the design, implementation and testing of a system are essential if the program is to be understood and maintained. Like user documentation, it is important that system documentation is structured, with overviews leading the reader into more formal and detailed descriptions of each aspect of the system. For large systems that are developed to a customer's specification, the system documentation should include:

1. The requirements document and an associated rationale.
2. A document describing the system architecture.
3. For each program in the system, a description of the architecture of that program.
4. For each component in the system, a description of its functionality and interfaces.
5. Program source code listings. These should be commented where the comments should explain complex sections of code and provide a rationale for the coding

method used. If meaningful names are used and a good, structured programming style is used, much of the code should be self-documenting without the need for additional comments. This information is now normally maintained electronically rather than on paper with selected information printed on demand from readers.

6. Validation documents describing how each program is validated and how the validation information relates to the requirements.
7. A system maintenance guide which describes known problems with the system, describes which parts of the system are hardware and software dependent and which describes how evolution of the system has been taken into account in its design.

A common system maintenance problem is ensuring that all representations are kept in step when the system is changed. To help with this, the relationships and dependencies between documents and parts of documents should be recorded in a document management system as discussed in the final part of this paper. For smaller systems and systems that are developed as software products, system documentation is usually less comprehensive. This is not necessarily a good thing but schedule pressures on developers mean that documents are simply never written or, if written, are not kept up to date. These pressures are sometimes inevitable but, in my view, at the very least you should always try to maintain a specification of the system, an architectural design document and the program source code. Unfortunately, documentation maintenance is often neglected.

Documentation may become out of step with its associated software, causing problems for both users and maintainers of the system. The natural tendency is to meet a deadline by modifying code with the intention of modifying other documents later. Often, pressure of work means that this modification is continually set aside until finding what is to be changed becomes very difficult indeed. The best solution to this problem is to support document maintenance with software tools which record document relationships, remind software engineers when changes to one document affect another and record possible inconsistencies in the documentation. Such a system is described by Garg and Scacchi (1990).

1.2 DOCUMENTATION DESIGN

The documentation design must cover the following elements;

- Style
- Structure
- Content
- Format

1.2.1 STYLE

Every organisation has its own style when writing program documents. Standards and quality assessment are essential if good documentation is to be produced but document quality is fundamentally dependent on the writer's ability to construct clear and concise technical prose. In short, good documentation requires good writing. Writing documents well is neither easy nor is it a single stage process. Written work must be written, read, criticized and then rewritten until a satisfactory document is produced. Technical writing is a craft rather than a science but some broad guide-lines about how to write well are:

1. Use active rather than passive tenses It is better to say 'You should see a flashing cursor at the top left of the screen' rather than 'A flashing cursor should appear at the top left of the screen'.

2. Use grammatically correct constructs and correct spelling To boldly go on splitting infinitives (like this) and to misspell words (like mispell) irritates many readers and reduces the credibility of the writer in their eyes. Unfortunately, English spelling is not standardized and both British and American readers are sometimes irrational in their dislike of alternative spellings.

1. Do not use long sentences which present several different facts It is better to use a number of shorter sentences. Each sentence can then be assimilated on its own. The reader does not need to maintain several pieces of information at one time to understand the complete sentence.

2. Keep paragraphs short As a general rule, no paragraph should be made up of more than seven sentences. Our capacity for holding immediate information is limited. In short paragraphs, all of the concepts in the paragraph can be maintained in our short-term memory which can hold about 7 chunks of information.

3. Don’t be verbose If you can say something in a few words do so. A lengthy description is not necessarily more profound. Quality is more important then quantity.

4. Be precise and define the terms which you use Computing terminology is fluid and many terms have more than one meaning. If you use terms like module or process make sure that your definition is clear. Collect definitions in a glossary.

5. If a description is complex, repeat yourself It is often a good idea to present two or more differently phrased descriptions of the same thing. If readers fail to completely understand one description, they may benefit from having the same thing said in a different way.

6. Make use of headings and sub-headings. These break up a chapter into parts which may be read separately. Always ensure that a consistent numbering convention is used.

7. Itemize facts wherever possible. It is usually clearer to present facts in a list rather than in a sentence. Use textual highlighting (italics or underlining) for emphasis.

8. Do not refer to information by reference number alone Give the reference number and remind the reader what that reference covered. For example, rather than say ‘In section 1.3 …’ you should say ‘In section 1.3, which described management process models, …’

Documents should be inspected in the same way as programs. During a document inspection, the text is criticized, omissions pointed out and suggestions made on how to improve the document. In this latter respect, it differs from a code inspection which is an error finding rather than an error correction mechanism.

As well as personal criticism, you can also use grammar checkers which are incorporated in word processors. These checkers find ungrammatical or clumsy uses of words. They identify long sentences and paragraphs and the use of passive rather than active tenses. These checkers are not perfect and sometimes they use outmoded style rules or rules which are specific to one country. Nevertheless, because they often check style as you are typing, they can help identify phrases which could be improved.

1.2.2 STRUCTURE

Each organisation has a defined structure that it follows when creating program documents. The document structure is the way in which the material in the document is organized into

chapters and, within these chapters, into sections and subsections. The document structure is the way in which the material in the document is organized into chapters and, within these chapters, into sections and subsections. Document structure has a major impact on readability and usability and it is important to design this carefully when creating documentation.

Component	Description
Identification data	Data such as a title and identifier that uniquely identifies the document.
Table of contents	Chapter/section names and page numbers.
List of illustrations	Figure numbers and titles
Introduction	Defines the purpose of the document and a brief summary of the contents
Information for use of the documentation	Suggestions for different readers on how to use the documentation effectively.
Concept of operations	An explanation of the conceptual background to the use of the software.
Procedures	Directions on how to use the software to complete the tasks that it is designed to support.
Information on software commands	A description of each of the commands supported by the software.
Error messages and problem resolution	A description of the errors that can be reported and how to recover from these errors.
Glossary	Definitions of specialized terms used.
Related information sources	References or links to other documents that provide additional information
Navigational features	Features that allow readers to find their current location and move around the document.
Index	A list of key terms and the pages where these terms are referenced.
Search capability	In electronic documentation, a way of finding specific terms in the document.

As with software systems, you should design document structures so that the different parts are as independent as possible. This allows each part to be read as a single item and reduces problems of cross-referencing when changes have to be made. Structuring a document properly also allows readers to find information more easily. As well as document components such as contents lists and indexes, well-structured documents can be skim read so that readers can quickly locate sections or sub-sections that are of most interest to them.

The IEEE standard for user documentation (IEEE, 2001) proposes that the structure of a document should include the components shown in Figure above. The standard makes clear that these are desirable or essential features of a document but makes clear that the ways in which these components are provided depends on the designers of the documentation. Some (such as a table of contents) are clearly separate sections; other components such as navigational features will be found throughout the document.

CONTENT

Again every organisation specifies the kind of content to include in their program documents. The following are general tips when writing content;

1. Only include what's relevant: Do you still have the goal of your content fresh in your mind? Good. When you start writing, keep that goal in mind so that you only include information that supports your goal. Just because you know the whole alphabet about a subject doesn't mean all of it belongs in one piece of content.
2. Let yourself write: Stop me when this starts to sound familiar: You write a sentence. . . then you delete. You write three more and delete two. Then you get rid of a whole paragraph and pick at your title. Stop it! Writing and editing are two different stages of the content cycle, which means you shouldn't attempt to do them simultaneously. When you sit down to write, just write; don't self-edit. Focus on getting everything out that you want to say and putting it all down. Once it's written down, then you can edit and make it sound cohesive. But the more time you spend self-editing as you're writing, the longer and more fragmented your copy is going to sound.
3. Use short sentences: Short sentences are easier for writers to get out. They're also easier for readers to take in. Stick with them and stop confusing people with overly complicated writing. Like short sentences, it's that simple.8.
4. Use clear, direct titles: One of the best things you can do to improve your writing is learn to write killer titles. Direct titles aren't always the most fun to write (who doesn't love a good pun?), but they do the best job of telling readers and the search engines what your post is about. And that is your title's main goal – to set up your content and make someone want to read it. Avoid getting so clever with your titles that you make it impossible for readers to predict what's coming next or, even worse, set them up to be disappointed when your content isn't about what they hoped it was. When all else fails, say what you mean. It's true in life and in Web content.

Format

There are a range of formats for program documents. A design document must have the following headings;

- Statement of goals
- Functionality
- User interface
- Milestones
- conclusion

1.3 DOCUMENT PLANNING

One of the critical step before writing a program document is to plan. The document plan must include all program documentation components.

The following are a list of elements that must be included in the document plan;

- Design documents,
- program listings (including annotations),
- sample input/output,
- user manuals,
- technical manuals,
- Online help (all components needed as applicable to the type of program being documented).

SESSION 2: CREATING A PROGRAM DOCUMENTATION

On completion of this section you will be able to create documentation for a computer program to agreed standards

1. The documentation is created according to the design created in specific outcome 'Plan and design documentation for a computer program to agreed standards'.
2. The documentation components created are created according to the plan specified in specific outcome 'Plan and design documentation for a computer program to agreed standards'.
3. The documentation created is structured sensibly, defining how program specifications have been met.

2.1 CREATING OR PREPARING A PROGRAM DOCUMENT

Document preparation is the process of creating a document and formatting it for publication. Figure 3 shows the document preparation process as being split into 3 stages namely document creation, polishing and production. Modern word processing systems are now integrated packages of software tools that support all parts of this process.

However, it is still the case that for the highest-quality documents, it is best to use separate tools for some preparation processes rather than the built-in word processor functions. The three phases of preparation and associated support facilities are:

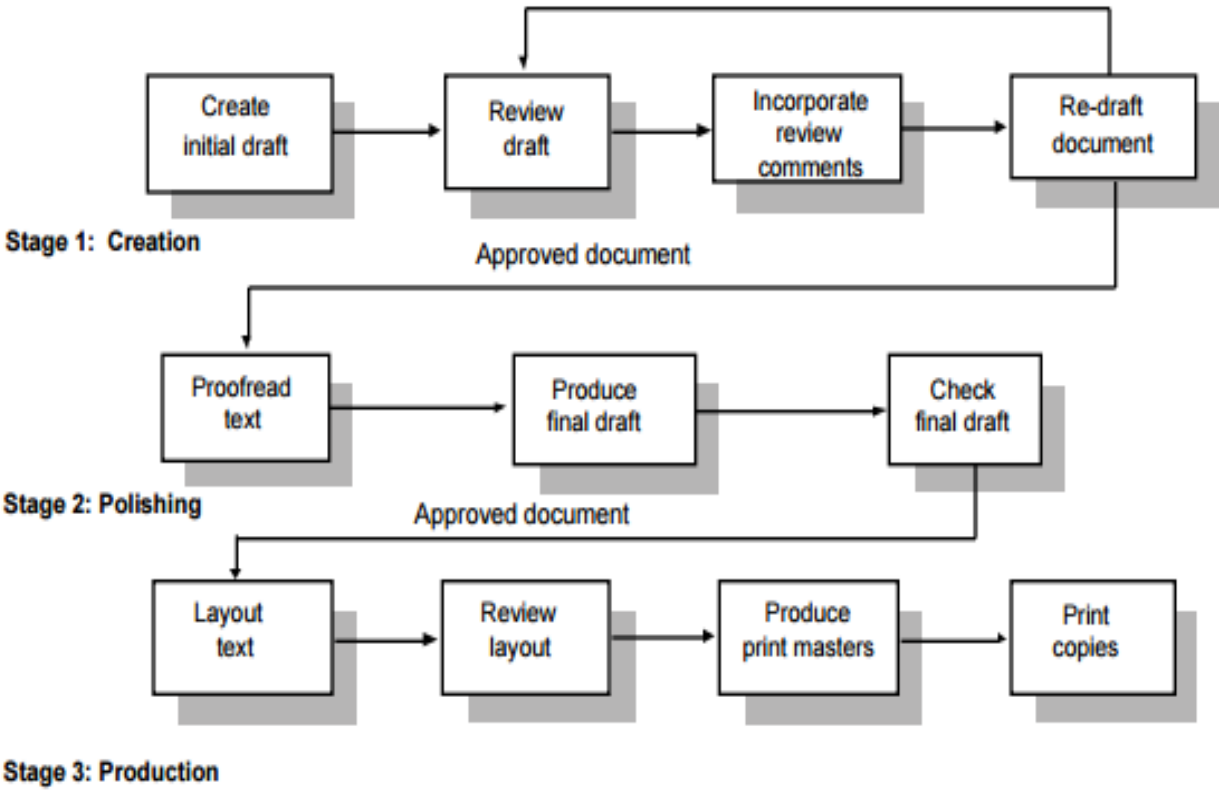
1. Document creation. The initial input of the information in the document. This is supported by word processors and text formatters, table and equation processors, drawing and art packages.
2. Document polishing. This process involves improving the writing and presentation of the document to make to make it more understandable and readable. This involves finding and removing spelling, punctuation and grammatical errors, detecting clumsy phrases and removing redundancy in the text. The process may be supported by tools such as on-line dictionaries, spelling checkers, grammar and style checkers and style checkers.
3. Document production This is the process of preparing the document for professional printing. It is supported by desktop-publishing packages, artwork packages and type styling programs.

As well as these tools to support the document production process, configuration management systems, information retrieval systems and hypertext systems may also be used to support document maintenance, retrieval and management.

Modern word processing systems are screen based and combine text editing and formatting. The image of the document on the user's terminal is, more or less, the same as the final form of the printed document. Finished layout is immediately obvious. Errors can be corrected and layout improved before printing the document. However, programmers who already use an editor for program preparation may sometimes prefer to use a separate editor and text formatting system.

Text formatting systems such as Latex interpret a layout program specified by the document writer. Layout commands (often chosen from a standard, definable command set) are interspersed with the text of the document. The text formatter processes these commands

and the associated text and lays the document out according to the programmer's instructions. The distinction between these systems and word processors is illustrated in Figure 4.



Text formatting systems can look ahead at the text to be laid out so can make better layout decisions than word processing systems whose working context is more restricted. Because the commands are really a programming language, programmers often prefer them to word processors but other, non-technical users usually find them more difficult to use.

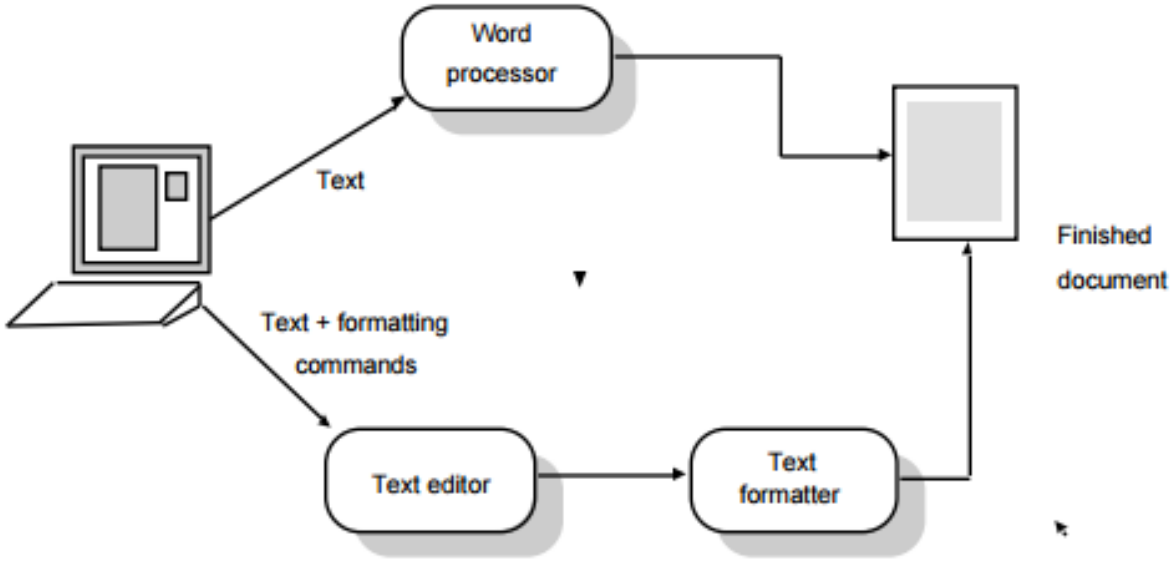


Figure 4: Word processors and text formatters

The major disadvantage of text processors, once their programming has been mastered, is that they do not provide an immediate display of the output they produce. The user must process the text (this may take several minutes) then display the output using a preview package. If an error is discovered, it cannot be fixed immediately. The original source must be modified and the preview process repeated. Thus, although they can result in higher quality documents, most users find text formatters more inconvenient than word processors.

The final stage of document production is a skilled task which, for documents with large print runs, should be left to professional printers. However, desktop publishing (DTP) systems and graphics systems that support scanning and processing photographs and artwork are now widely available. These have revolutionized document production. DTP systems partially automate the layout of text and graphics. They allow very fine-grain control over the layout and look of a document and can be used by engineers to produce finished system documentation. The advantage of using a desktop publishing system is that the cost of producing high-quality documents is reduced because some of the steps in the production process are eliminated. Even documents which are produced in small numbers can be produced to a high standard.

The disadvantage of using desktop publishing systems is that they do not automate the skills of the graphic designer. Their seductive ease-of-use means that they are accessible to unskilled users who may produce unattractive and badly designed documents. An enormous number of documents are produced in the course of a project and these need to be managed

so that the right version of the document is available when required. If a project is distributed, copies of documents will be produced and stored at different locations and it is very important to maintain a 'master file' of documents which contains the definitive versions of each document. This helps minimize a very common problem that arises when users of a document make mistakes because they are not working from the current version of a document.

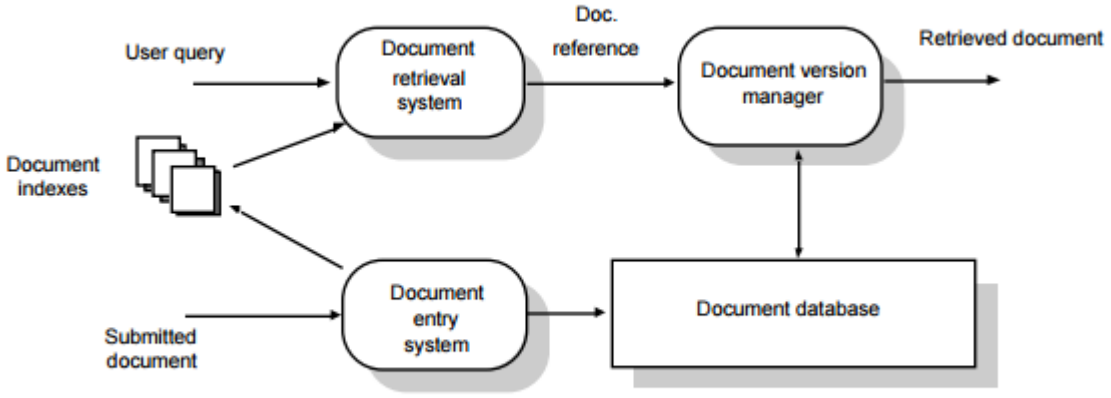


Figure 5: Document management

Each document should have a unique record and this can be used as a key in a document database record. However, retrieval by other fields such as the title and author should also be supported. The basic problem with managing documents using a file system to store the documents and a database management system to maintain document information is that users have to be disciplined in the way they use they system. They must ensure that they check out a copy of the document from the system each time they need it rather than use a local copy on their computer or the copy that they have printed. In practice, achieving this level of discipline is difficult and errors are always likely to arise. In very large projects, specialized document management systems may be used that integrate the storage of the documents and the maintenance of document information (Figure 5).

Document management software allows related documents to be linked, maintains records of who has checked out documents, may support the compression and de-compression of document text and provides indexing and information retrieval facilities so that documents can be found. Document management systems may also include version management facilities so that different document versions may be maintained.

Note

The documentation components include;

- Design documents,
- program listings (including annotations),
- sample input/output,
- user manuals,
- technical manuals,
- Online help (all components needed as applicable to the type of program being documented).

2.2 DOCUMENT STANDARDS

Documentation standards act as a basis for document quality assurance. Documents produced according to appropriate standards have a consistent appearance, structure and quality. However, it is not only standards that focus on documentation that are relevant. Other standards that may be used in the documentation process are:

1. Process standards: These standards define the process which should be followed for high-quality document production.
2. Product standards: These are standards which govern the documents themselves.
3. Interchange standards: It is increasingly important to exchange copies of documents via electronic mail and to store documents in databases. Interchange standards ensure that all electronic copies of documents are compatible

Standards are, by their nature, designed to cover all cases and, consequently, can sometimes seem unnecessarily restrictive. It is therefore important that, for each project, the appropriate standards are chosen and modified to suit that particular project. Small projects developing systems with a relatively short expected lifetime need different standards from large software projects where the software may have to be maintained for 10 or more years.

Process standards

Process standards define the approach to be taken in producing documents. This generally means defining the software tools which should be used for document production and defining the quality assurance procedures which ensure that high-quality documents are produced. Document process quality assurance standards must be flexible and must be able to cope with all types of document. In some cases, where documents are simply working papers or memos, no explicit quality checking is required. However, where documents are formal documents, that is, when their evolution is to be controlled by configuration management procedures, a formal quality process should be adopted. Figure 3 illustrates one possible process. Drafting, checking, revising and re-drafting is an iterative process which should be continued until a document of acceptable quality is produced. The acceptable quality level will depend on the document type and the potential readers of the document.

Product standards

Product standards apply to all documents produced in the course of the software development. Documents should have a consistent appearance and, documents of the same class should have a consistent structure. Document standards are project-specific but should

be based on more general organizational standards. Examples of product standards which should be developed are:

1. Document identification standards. As large projects typically produce thousands of documents, each document must be uniquely identified. For formal documents, this identifier may be the formal identifier defined by the configuration manager. For informal documents, the style of the document identifier should be defined by the project manager.
2. Document structure standards. As discussed in the previous section, there is an appropriate structure for each class of document produced during a software project. Structure standards should define this organization. They should also specify the conventions used for page numbering, page header and footer information, and section and subsection numbering.
3. Document presentation standards. Document presentation standards define a 'house style' for documents and they contribute significantly to document consistency. They include the definition of fonts and styles used in the document, the use of logos and company names, the use of colour to highlight document structure, etc.
4. Document update standards. As a document is changed to reflect changes in the system, a consistent way of indicating these changes should be used. These might include the use of different colours of cover to indicate a new document version and the use of change bars to indicate modified or deleted paragraphs.

Document standards should apply to all project documents and to the initial drafts of user documentation. In many cases, however, user documentation has to be presented in a form appropriate to the user rather than the project and it should be recast into that form during the production process.

Interchange standards Document interchange standards are important as more and more documents are produced in electronic format as well as or instead of on paper. For documentation that is delivered with a software system, the Adobe Portable Document Format (PDF) is now very commonly used. However, when documents are exchanged by the development team and drafts are circulated within an organization these are often in the format of whatever word processor is used (often Microsoft Word). Assuming that the use of a standard word processor and graphical editing system is mandated in the process standards, interchange standards define the conventions for using these tools. The use of interchange standards, allows documents to be transferred electronically and re-created in their original form. Interchange standards are more than simply an agreement to use a common version of a system for document production. Examples of interchange standards include the use of an

agreed standard macro set if a text formatting system is used for document production or the use of a standard style sheet for a word processor. Interchange standards may also limit the fonts and text styles used because of differing printer and display capabilities.

The IEEE standard for user documentation

The first IEEE standard for user documentation (IEEE, 1987) was produced in 1987 and, at the time of writing, a new draft of this standard is being prepared for publication (IEEE, 2001). Like all standards, this standard encapsulates wisdom and experience about software documentation and proposes a structure for user documentation. Using this structure as a basis, the standard discusses the content of software user documentation and proposes formatting standards for these documents. I have already covered the documentation structure proposed by the latest version of the standard. To illustrate the formatting advice in the standard, here are some quotations from the current draft standard of good practice:

The documentation should be provided in media and formats that allow its use by those with vision, hearing or other physical limitation. A description of how to print the electronic documentation should be included in both the electronic and the printed documentation. Because some users cannot distinguish between colours, documentation should provide text cues rather than using colours such as red and green as the only way to convey meaning. Warnings, cautions and notes shall be displayed in a consistent format that is readily distinguishable from ordinary text or instructional steps.

Documentation formats for user-entered commands or codes shall clearly distinguish between literals (to be input exactly as shown) and variables (to be selected by the user). Illustrations that accompany text should appear adjacent to their first reference in the text so that the associated text and illustration can be viewed simultaneously.

You can see from these that the standard is helpful without being proscriptive and therefore different conventions used by different companies and organizations can be accommodated. Like all standards, this documentation standard has to be adapted to the local situation where it is used. These should instantiate the advice in the standard to the local situation and define the specific structures and formats that should be used.

SESSION 3: REVIEW DOCUMENTATION FOR A COMPUTER PROGRAM FOR COMPLETENESS

On completion of this section you will be able to Review documentation for a computer program for completeness.

1. The review identifies if documentation was created according to the design created in specific outcome 'Plan and design documentation for a computer program to agreed standards'.
2. The review identifies if documentation was created consistent with the computer program being documented.

3.1 REVIEWING PROGRAM DOCUMENTS

The program document must be reviewed and edited to ensure that;

- Documentation was created according to the design created
- Documentation was created consistent with the computer program being documented.

By now, we all know that for a tester, documentation is an integral part of his daily life. There is an overload of testing artefacts that are created, reviewed, approved, used, maintained and distributed. We always have clear cut processes laid out for how to create a document, how to use it, who should it go to, etc.

Types of Reviews:

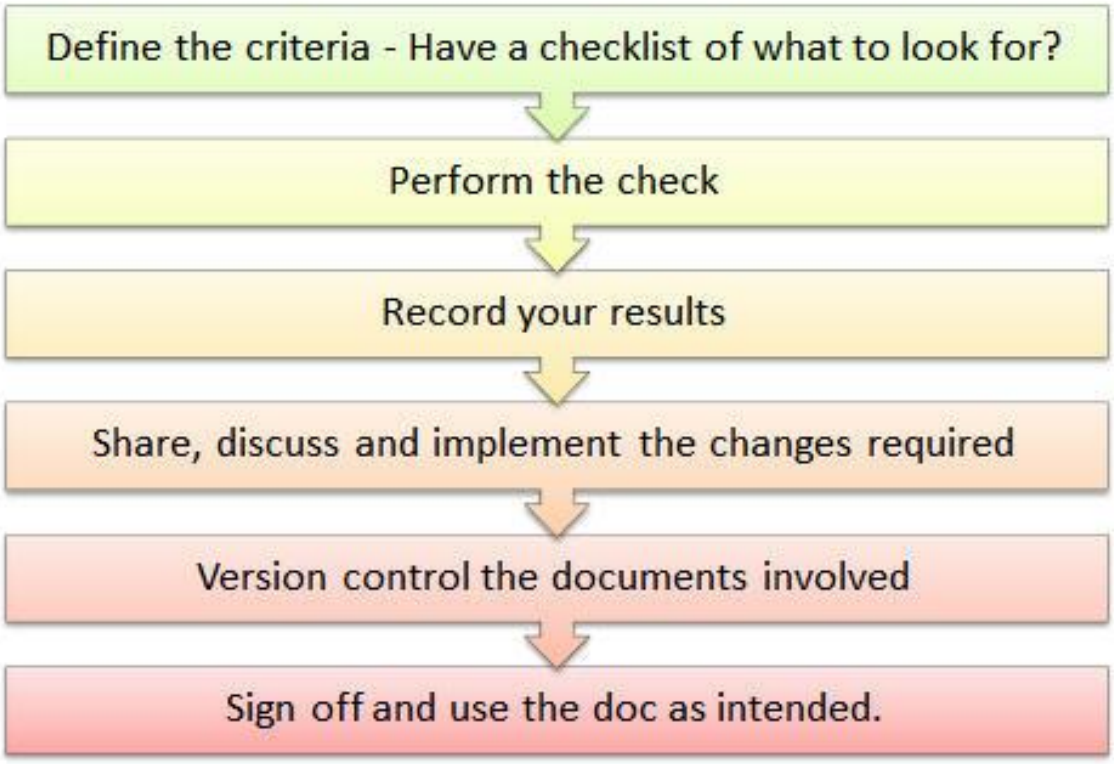
1. Reviewing your own work – Self Checking
2. Peer- review
3. Supervisory

If validation is one half of the testing practices then verification is the other, but often the guidelines are murky – So let's change that NOW. As it a general practice with the articles at STH, we will begin with the questions, What –Why-How.

What do we review? –

Everything created has to be reviewed. The following are some of the common artifacts reviewed:

1. Test plan
2. Test scenarios
3. Test templates
4. Test cases
5. Test data
6. Reports



Test Documents Review Process © Softwaretestinghelp.com

We will now discuss each step in the "How" section – in other words the process to perform it. (Most of us testers don't like the word process, isn't it? For us, it either means a lot more work or some high level managerial task that we have to do even if we don't want to – for the sake of some compliance that we have no idea about. But, trust me, when you come up with a process that works and is simple enough for us to understand why we have to do it, it can be fun! Just play along with me.)

The process for peer reviews and supervisory reviews is the same according to me, because a supervisor is also a peer despite the higher designation.

Step 1: Define the criteria.

a) What are you expecting to find? You can look for things like:

- Spelling mistakes (Sounds too silly? I don't think so, one time I wrote "Wed Object" instead of "Web Object" in one of my articles – Changes the meaning entirely. Almost makes it too silly to be taken seriously.)
- Format/template compliance
- Functionality coverage and correctness

- Ease of understanding
- Standards followed – naming conventions, consistent numbering …etc.

b) Make a checklist– Checklists are very versatile. It can be as complicated as a review checklist or as simple as a grocery list. All it takes is some time to make it and once you do, it is as simple as checking ON or OFF.

c) How to report the results? – Choose whatever is convenient, preferably a method that can be recorded and tracked.

- Sometimes this can be as simple as adding an extra column in the excel sheet with test cases and writing something in red when it is not what it is supposed to be.
- Can be word of mouth
- A list in an email

Step 2: Perform the check

a) Using the checklist you made earlier, verify the document and provide your feedback.

Step 3: Record your results

a) Again, using the method decided in the step 1, record and report your results.

b) When reporting your comments or suggestions for change, treat it no differently than reporting a defect. Don't overlook anything. Be detailed.

Step 4: Share, discuss and implement the changes required

a) Nobody likes to be told that their work is incorrect or incomplete. So keep in mind the following guidelines when you are providing negative feedback.

- Provide constructive criticism – Remember not to be critical of the person but point out flaws in his product
- Don't get competitive – just because he turned in 30 review comments on your test cases, don't try to beat it.
- Give reasons to back your comments

b) Obtain a sign off.

c) Have the changes made

Step 5: Version control the documents involved

a) Don't delete the older versions of any of the documents. Name them appropriately and keep them in a centralized project folder. After all, this is the evidence to all our work

Step 6: Sign off and use the doc as intended.

a) Once all the changes are incorporated, version saved, give the review process a sign off and move on to using the document for what it was created for.

b) Another question that comes up is – do we recheck after the changes are made? How many times is this process going to go on – work- review-fix-and then reviewed again? Until when?

No, review does not have to happen over and over again. It is a quality control activity that focuses on verifying if the testing aides are created right or not. As always, zero defect documents are impossible. So a reasonable level of review- one time by a peer is acceptable.

POINTS TO REMEMBER:

1. Every project does not have to follow this formalized method of review, but even if they have an informal method in place, these steps will help set the expectation and guide you along.
2. Test documentation time line estimates are typically based on time required for creating and reviewing the documents- so it is inbuilt into it even though we don't always recognize it.
3. Reviewing is not a process that is limited to manual testing teams. Automation teams also perform code walkthroughs, design reviews etc.

This is how a typical review comments document for test cases looks like. The comments are in red. Not necessarily real comments, but something to show how it's done.

Microsoft Excel - Book1

	G	H	I	J	K	L	M	N
1								
2	**Precondition**	**Steps**	**Test data**	**Expected result**	**Review comments**	**Post conditions**	**Status- Pass/Fail**	**Actual re**
3	1. the user should already have an account/profile information saved	1. Login into the KP site	user name and password	The user is logged in	Enter the user details in detail			
7		2. Click on the "View Profile" link(Button, link or image)		The profile info is displayed	Give the details of the profile in the expected result			
8		3. Update the "first name" field	swati change it to swatis		Expected result column is missing			
9		4. Click on Save		The changes should be saved	Details of what got saved needs to be displayed	When I login later, I should be able to see the first name as swatis.		