

Developing Successful Software Documentation

By Steve Capri, [Writing Assistance, Inc.](#)

Introduction	2
Comparing Successful and Not-So-Successful Documentation	2
<i>Case Study</i>	3
<i>Unstructured vs. Structured</i>	4
Eight Phases to Successful Documentation	5
<i>Phase 1 – Analysis</i>	5
<i>Phase 2 – Design</i>	7
<i>Phase 3 – Development</i>	10
<i>Phase 4 – Validation</i>	10
<i>Phase 5 – Production</i>	10
<i>Phase 6 – Manufacturing</i>	11
<i>Phase 7 – Delivery</i>	11
<i>Phase 8 – Customer Satisfaction Assurance</i>	11
In Summary	11

Introduction

Whether software documentation is designed for a company's internal users or for a variety of end customers, one thing is for certain: Documentation that is well written, well structured, easily accessible, and thoroughly compliments the software it supports can play a significant role in a product's overall success. And it doesn't matter if the documentation stands alone or it is integrated with the product. As long as it is properly planned, developed, and configured, success is eminent.

Still, non-believers in many corporate settings are compelled to ask:

"Honestly....Why should anyone, especially software executives, care as much about building and deploying successful documentation as we do about building successful software? After all, who reads documentation anyway?"

My answer is always the same. Bottom line: Most often, your product's documentation is the first thing your customers see. Whether it's an online splash screen, installation instructions, day-to-day procedures, trouble-shooting tips, or a simple features summary printed along the side of your product's shrink-wrapped box – documentation is ANY user's first-line of support. As such, your documentation can make or break a new or potential user's first impression of your product and its usability.



Now think about it. Wouldn't you want your documentation – be it for a simple or complex product – to be the very best it can be? Would you dare risk building and distributing anything less? And if that doesn't convince you, ask yourself this: Would you risk building or distributing anything but a successful software product – documentation notwithstanding? With that said, let's proceed.

Comparing Successful and Not-So-Successful Documentation

Before we can build documentation that is truly successful, we must first understand what we mean by "successful." We must also understand what types of documentation – no matter how excellent – won't help your users and why.

Consider this: It's very unlikely that a talented, experienced writing staff (large or small) would ever develop bad or unusable documentation. It is, however, VERY likely that the same staff could develop too much "unnecessary" documentation that contributes little or nothing towards a product's success – especially without the proper planning. With that in mind, before we can even plan successful documentation, we must first identify which characteristics suggest success and which do not.

Some simplistic, yet common benchmark comparisons follow.

<p align="center">Successful Documentation...</p>	<p align="center">Not-So-Successful Documentation...</p>
<p align="center">  </p> <p> Makes information easily accessible. Provides a limited number of user entry points. Helps new users learn quickly. Simplifies the product. Helps cut support costs. </p>	<p align="center">  </p> <p> Makes information difficult to find. Provides too many places to hunt for information. Overwhelms users with detail. Tends to complicate the product. Can compromise support efforts. </p>

Look too simple? Sound too clear? Well, it's not always this easy and of course, every project is different. But by following a systematic approach to your documentation planning and development, you can at least feel confident that your end deliverables will not only be the right ones for your targeted audiences, but they will also enhance and ensure your product's success through its usability, marketability, and ease of support.

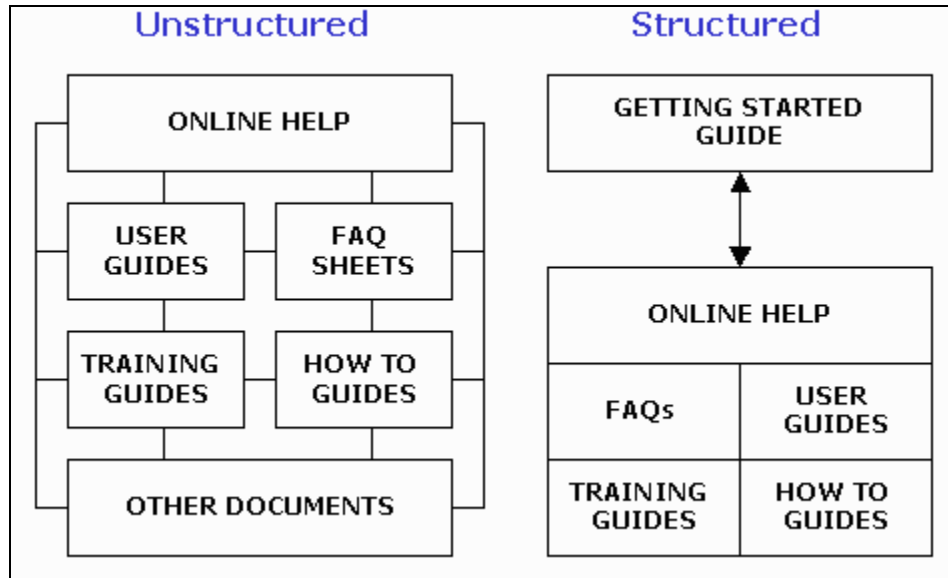
Case Study

There's an old adage that says writers can never provide users with too much information. While this might be true for some technical audiences, it's not necessarily always the best philosophy – especially for end users who need quick, up-to-date procedures to perform key support-related tasks. So let's be realistic. Just because an organization develops lots and lots of documentation for a product does not necessarily guarantee successful documentation.

Take SDL International (www.sdl.com) for example. Although the company had originally developed a vast array of professionally prepared documentation products for its *SDLX Translation Memory System*, the overall documentation was not presenting the right image. Although it seemed all the necessary pieces existed, they were so fragmented that users could not easily find them. To make matters worse, the documents were hard to use plus they made the system look far more complicated than it really was.

After hiring a professional technical-writing company to re-design the documentation, SDL reduced its number of documentation deliverables (not necessarily its content) to just a few key, integrated customer-centric items.

The following diagram illustrates this transition.



Unstructured vs. Structured

As illustrated above, the unstructured documentation gave the user no clear entry point upon which to grasp the fundamentals of the product. It doesn't mean the documentation wasn't good, it just means it wasn't configured or engineered properly for the targeted audiences.

After the documentation was re-engineered, it provided users with two clear entry points: A *Getting Started Guide* and comprehensive *Online Help*.

The *Getting Started* guide, which ended up half the size of the original user guides, explains the essential tasks that users perform. It includes an introductory chapter that explains typical scenarios in which users might work, as well as the software modules related to these working patterns.

The *Online Help* includes task- and reference-based information, FAQs, trouble-shooting detail, as well as lots of useful realistic diagrams and examples. For easy access, the content also points to the information in the *Getting Started* guide in lieu of repeating the information in two places.

Results from SDL's documentation re-design effort have reduced support calls and emails to SDL's help desk, and have been instrumental in generating new sales!

**** Case study and results provided by Techscribe, www.techscribe.co.uk/techw/cssdl.htm#sales**

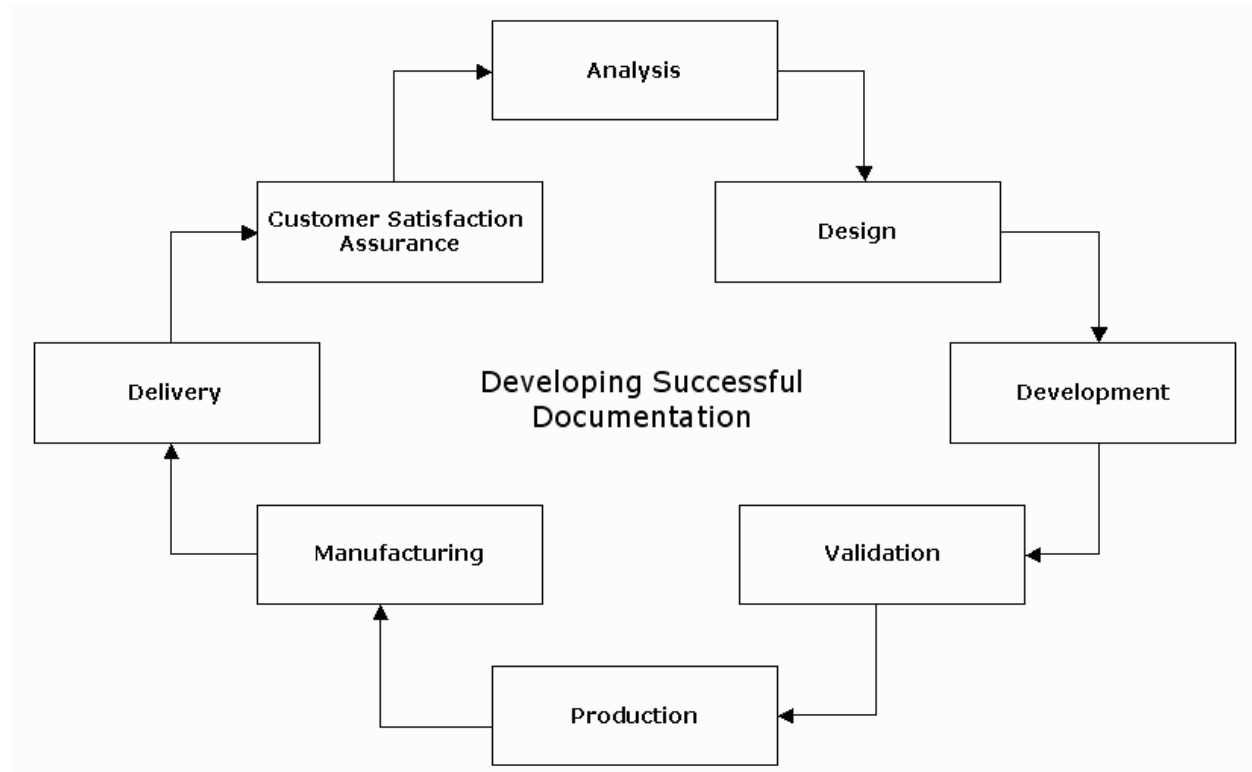
Eight Phases to Successful Documentation

OK. Now what? How do we develop the successful stuff? Essentially, this involves eight, closely related processes, including:

Phase 1 – Analysis
Phase 2 – Design
Phase 3 – Development
Phase 4 – Validation

Phase 5 – Production
Phase 6 – Manufacturing
Phase 7 – Delivery
Phase 8 – Customer Satisfaction Assurance

Following is a high-level illustration and discussion on each of these phases.



Phase 1 – Analysis

During the Analysis Phase, your primary goal is to identify who your various audiences are and the tasks they will perform while using your software. To get started, think about every possible audience that could potentially need documentation for the product you're supporting. It's best to approach this task in an unconstrained matter so that you don't overlook any important users.

To get started, imagine your documentation products as the center of attention. Ask yourself who might be looking at them from all angles. Consider the following diagram as a way to envision your target audiences.



You might be surprised how many audiences you identify for various software products. Also, be prepared periodically to interview all these people, to get feedback regarding their ongoing informational needs.

While this is an excellent, repeatable exercise, it is highly unlikely you'll have to address all audiences for all projects. After you have narrowed down the list of users for the software product you're currently documenting, you can start building a simple, yet unconstrained audience/task matrix as illustrated below.

	Development	Marketing	Support	Sales	Customers
Task 1			X		X
Task 2	X			X	X
Task 3	X	X	X		X
Task 4	X				X
Task 5		X	X	X	X

Across the top, identify your target audiences. Along the left side, list all the tasks that pertain to your various users. Next, associate specific tasks with the appropriate users by carefully placing **Xs** in the audience columns to intersect users with like tasks, and those with unique tasks.

After you've finished mapping tasks to audiences, you're ready to start identifying what documentation items (and on what mediums) you need to develop based on your various audience needs.

For example:

Since tasks 1-5 are applicable to end customers, you might want to develop an online *Getting Started User's Guide*.

Since tasks 1, 3, and 5, are applicable to customer-support personnel, perhaps you should develop a set of *Quick Reference Problem/Solution Placards*.

Since tasks 2-4 might address software developers, maybe you'll need a paper-based *Programmer's Handbook*.

And so on As a result, your "planned" deliverables list might look something like this.

Note: Multiple deliverables can be applicable to more than one user.

	Customers	Support	Developers
Getting Started Guide	X		X
Reference Cards		X	
Programmer's Guide			X
Help System		X	X

After you've identified all the information products for all your target audiences, you're ready for the Design Phase.

Phase 2 – Design

During this phase, your goal is to take all the documentation items you identified during the Analysis Phase and start designing/planning the content for each. As you address each item, typical activities should include preparing the following:

Outlines – This can include both substantive (full-sentence outlines) and structural outlines. Although not all writers work from stringent outlines, for medium-to-large projects, I recommend building outlines with at least some main topics and subtopics (usually called level-1, level-2, and level-3 headings). This will help you to more easily chunk and identify needed information elements for each user.

Unit Plans – Unit plans (commonly called chapter plans) take your outlines one step further. Typically, they include an introduction and enough skeletal content to step your team through the perceived flow of each document. For example, a full-blown, elaborate unit plan might include:

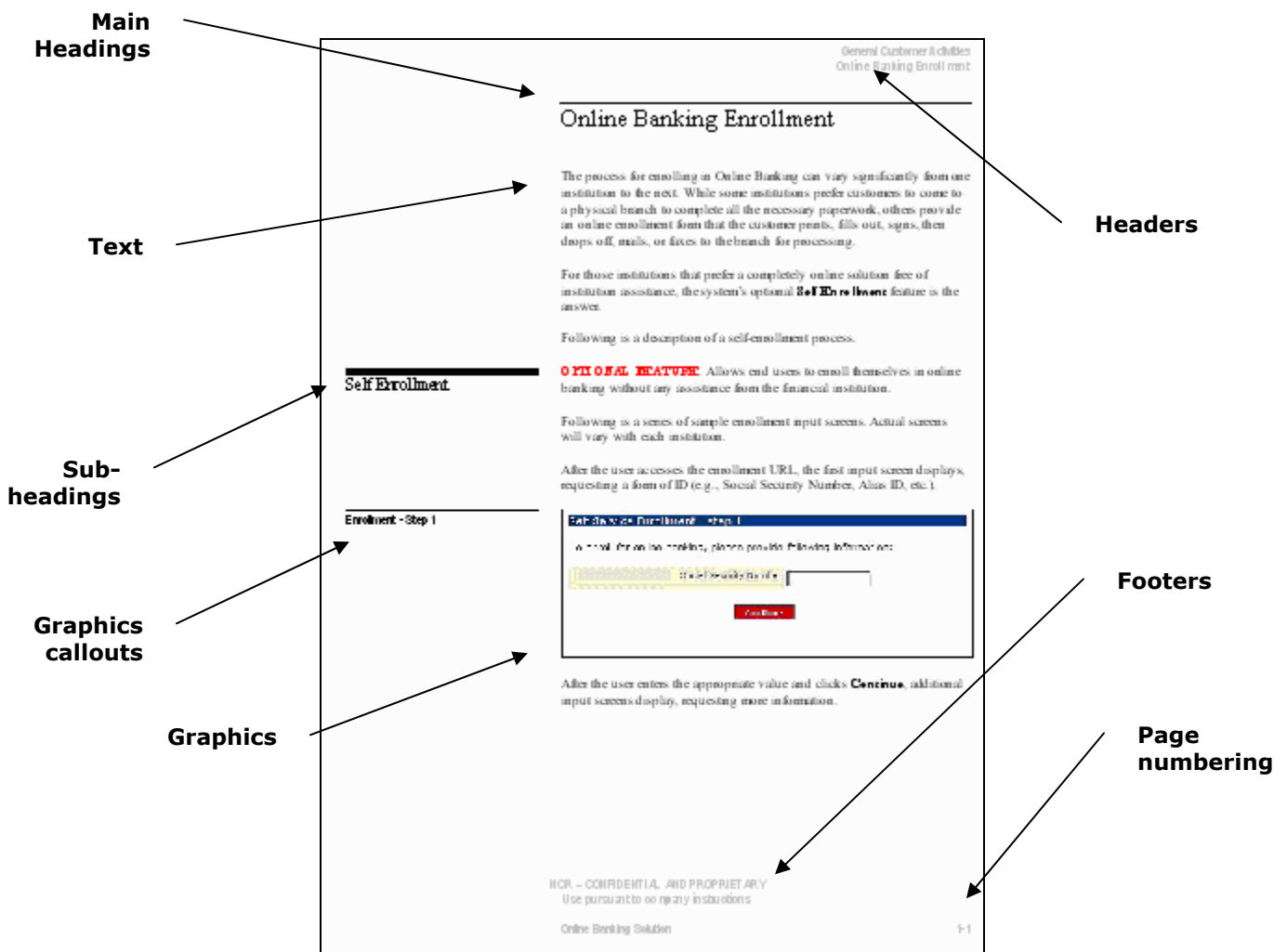
- Structured headings
- Summary of scope and content of the unit
 - Task and subtask objectives
 - Content description
 - Measurements
 - Student exercises
 - Unit prerequisites
- Links to and from this unit
- Sample exhibits or visual aids
- Note specifying anything the writer will need to complete the unit

Style Sheets and Formats – For all your documentation (both page-based and online), you will need to decide on two major issues to promote consistency:

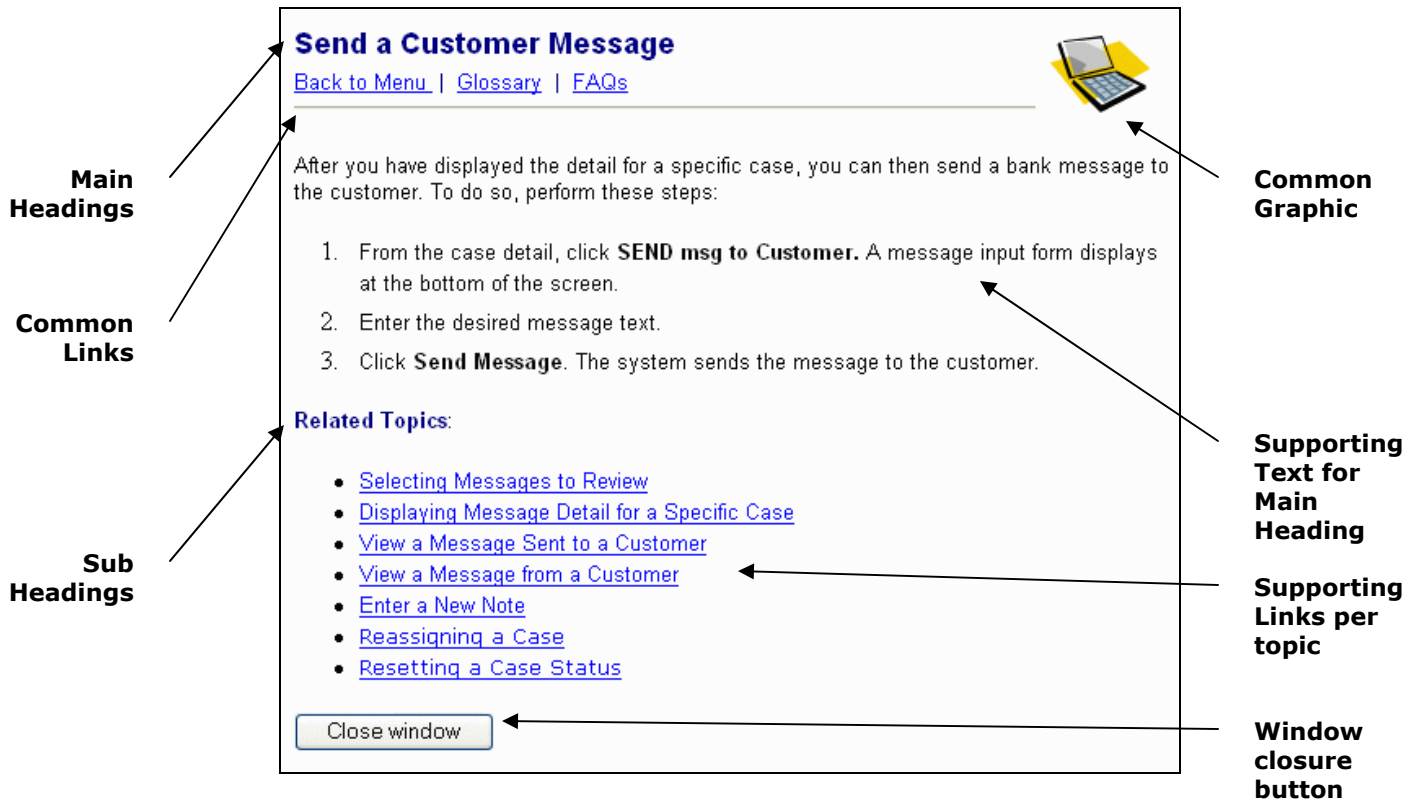
- Look and feel “templates” for headers, footers, fonts, illustration positioning, etc.
- Document “structure rules,” such as what goes on a cover page/initial screen, whether or not to include a preface, chapter-level tables of contents, a summary overview for each section, etc.

Some examples follow.

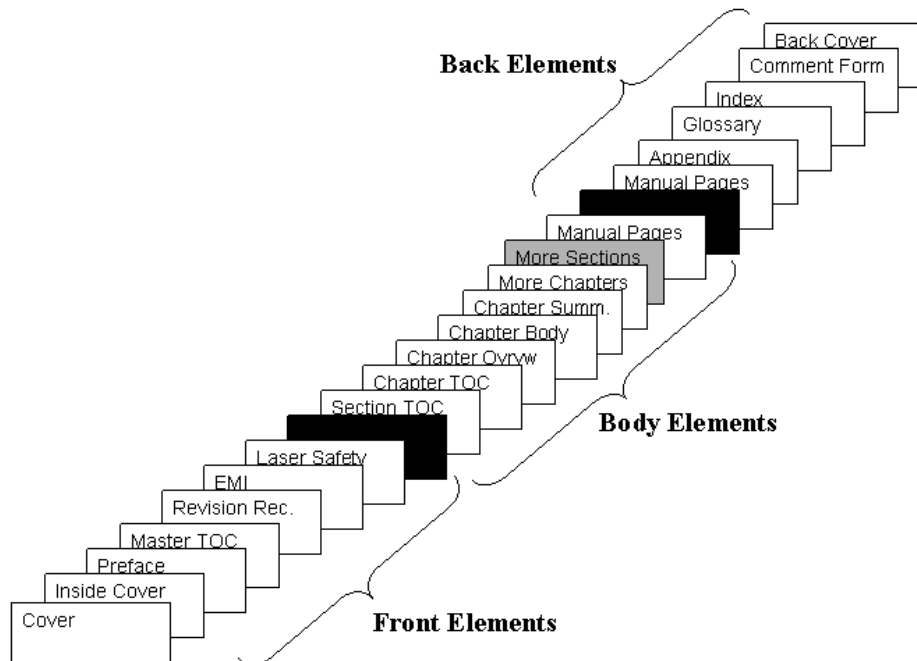
Sample Template for Paper-Based Document



Sample Template for Online Help



Sample Document Structure



Although the sample document structure is designed for paper-based deliverables such as user and training guides, for consistency across all your deliverables, you could invoke a similar structure for your online documentation.

Prototypes – Prototypes pertain to both printed and online material. So if you're developing user guides, you can use your unit plans to emulate your actual finished guides. For web sites and online help, don't spend a lot of time on formatting at this point. Instead, build prototypes that will cover the content and structural aspects, then implement the more finished look-and-feel aspects during the Development Phase.

Manufacturing and Delivery Plan – The closer you get to completing your deliverables, the more important it will be to have a plan in place for manufacturing and delivering your documentation in mass quantities. Chances are, you already have this sort of plan set up for your software delivery. To leverage those efforts, try integrating your documentation delivery into the same project plans.

Phase 3 – Development

The Development Phase is when your documentation visions really start to take shape. Up to this point, you've been planning. Now you're creating what you're actually going to deliver.

Using the templates (style sheets, page formats, screen designs, etc.) established during the Design Phase, activities at this time will include (but not necessarily be limited to):

- Acquiring your information from the appropriate Subject Matter Experts (SMEs).
- Developing content, graphics, and other illustrations.
- Testing your deliverables (guides, help, web sites, etc.) for usability and accuracy.
- Correcting typos, grammar, and other tactical errors.

As each documentation deliverable is completed, you can begin migrating it to the Validation Phase.

Phase 4 – Validation

During this Phase, activities include:

- Testing your documentation to ensure it meets its performance objectives, and meets the needs of its target audiences. To make this activity as productive as possible, it's best to use test subjects who most closely represent the target audiences and possess the pre-requisite skills.

- Revising and updating your information products as necessary after each validation test.

Phase 5 – Production

This Phase involves:

- Preparing reproducible masters for each documentation product. If you don't have the facilities for this, you will have to outsource this activity.
- Making arrangements with the necessary reproduction vendors to accommodate your eventual distribution and delivery.

Phase 6 – Manufacturing

The purpose of this phase is to produce high-quality finished goods (paper, videotape, audio, CD, online, etc.) to meet demand. This involves a variety of activities, such as:

- Establishing terms with your vendors/suppliers.
- Announcing software and documentation availability to existing and potentially new customers.

Phase 7 – Delivery

Once everything (software and documentation) is done, getting it all to your customers is the utmost focus. This involves:

- Processing orders in a timely and cost-effective manner.
- Packaging and shipping goods.
- Processing returns.
- Monitoring inventory.

Phase 8 – Customer Satisfaction Assurance

Even successful software and its documentation lends itself to improvement. That's why it's extremely important to follow-up with your customers as much as possible. Therefore, after your product rollout, be sure to:

- Collect ongoing customer-satisfaction information regarding the software AND its documentation.
- Notify customers of any corrective actions.
- Incorporate improvement plans for future releases.

In Summary

While there are no substantial, hard-core statistics on exactly how much ROI (Return on Investment) documentation can bring to a specific enterprise, there are many studies available that show how bad or unsuccessful documentation can compromise (if not destroy) a product's success. On those lines, it is also fitting to note that no documentation — no matter how good — can guarantee success for a poorly designed product.

As mentioned at the beginning of this article, well written, well structured, easily accessible documentation can definitely augment a product's success. That in itself defines successful documentation, which is an invaluable, worthwhile asset for any enterprise to achieve.

Suggested Reading

- Techscribe Case Studies — http://www.techscribe.co.uk/techw/case_studies.htm
- Society for Technical Communications (STC) — <http://www.stc.org>