Technical Communication & its Applications

Kelsey Caudle

First Edition, 2012

ISBN 978-81-323-1763-0

Published by:
Learning Press
4735/22 Prakashdeep Bldg,
Ansari Road, Darya Ganj,
Delhi - 110002
Email: info@wtbooks.com

Table of Contents

Chapter 1

Introduction to Technical Communication

Technical communication is a method of researching and creating information about technical processes or products directed to an audience through media. The information must be relevant to the intended audience. Technical communicators often work collaboratively to create products (deliverables) for various media, including paper, video, and the Internet. Deliverables include online help, user manuals, technical manuals, White papers, specifications, process and procedure manuals, industrial videos, reference cards, data sheets, journal articles, patents, training, business papers, and technical reports.

Technical domains can be of any kind, including the soft and hard sciences, high technology including computers and software, consumer electronics, and business processes and practices.

Technical communication jobs include the following:

- Technical writer
- Technical editor
- Technical illustrator
- Information architect
- Usability expert
- User interface designer
- User experience designer
- Technical trainer
- Technical translator
- API Writer

History

The origin of technical communication has been variously attributed to Ancient Greece, The Renaissance, and the mid 20th Century. However, a clear trend towards the professional field can be seen from the First World War on, growing out of the need for

technology-based documentation in the military, manufacturing, electronic and aerospace industries.

In the United States, two organizations concerned with improving the practice of technical communication were founded on the East Coast in 1953: the Society of Technical Writers, and the Association of Technical Writers and Editors. These organizations merged in 1957 to form the Society of Technical Writers and Editors, a predecessor of the current Society for Technical Communication (STC).

In the United Kingdom, the Institute of Scientific and Technical Communicators (ISTC) was formed in 1972 by the amalgamation of three existing associations: the Presentation of Technical Information Group (established in 1948), the Technical Publications Association (established in 1953, later the Institution of Technical Authors and Illustrators) and the Institute of Technical Publicity and Publications (established in 1963).

Content creation

Technical communication is sometimes considered a professional task for which organizations either hire specialized employees, or outsource their needs to communication firms. For example, a professional writer may work with a company to produce a user manual. Other times, technical communication is regarded as a responsibility that technical professionals employ on a daily basis as they work to convey technical information to coworkers and clients. For example, a computer scientist may need to provide software documentation to fellow programmers or clients.

The process of developing information products in technical communication begins by ensuring that the nature of the audience and their need for information is clearly identified. From there the technical communicator researches and structures the content into a framework that can guide the detailed development. As the information product is created, the paramount goal is ensuring that the content can be clearly understood by the intended audience and provides the information that the audience needs in the most appropriate format. This process, known as the 'Writing Process', has been a central focus of writing theory since the 1970s, and some contemporary textbook authors have applied it to technical communication.

Technical communication is important to engineers mainly for the purpose of being professional and accurate. These reports supply specific information in a concise manner and are very clear in their meaning if done correctly.

The technical writing process can be divided into five steps:

1. Determine purpose and audience
2. Collect information
3. Organize and outline information
4. Write the first draft

5. Revise and edit

Determining purpose and audience

All technical communication is done with a particular end in mind. The purpose is usually to facilitate the communication of ideas and concepts to the audience, but may sometimes be used to direct the audience in a particular course of action. The importance of the audience is in the notion that meaning is derived from the audience's interpretation of a piece of work. The purpose may be something as simple as having the audience understand the details of some technological system, or to take a particular action using that system. For example, if the workers in a bank were not properly posting deposits to accounts, someone would write the procedure so these workers might have the correct procedure. Similarly, a sales manager might wonder which of two sites would be a more appropriate choice for a new store, so he would ask someone to study the market and write a report with the recommendations. The sales manager would distribute the report to all parties involved in making that decision. In each of these instances, the person who is writing is transferring knowledge from the person who knows to the person who needs to know. This is the basic definition of technical communication.

The most commonly used form of technical communication is technical writing. Examples of technical writing include: project proposals, persuasive memos, technical manuals, and users' guides. Such materials should typically present an (informal) argument and be written diplomatically. A user's guide for an electronic device typically includes diagrams along with detailed textual explanations. The purpose should serve as a goal that the writer strives toward in writing.

The identification of the audience affects many aspects of communication, from word selection and graphics usage to style and organization. A non-technical audience might not understand, or worse, not even read a document that is heavy with jargon, while a technical audience might crave extra detail because it is critical for their work. Busy audiences do not have time to read an entire document, so content must be organized for ease of searching, for example by the frequent inclusion of headers, white space and other cues that guide attention. Other requirements vary on the needs of the particular audience.

Examples:

In Government:

Technical communication in the government is very particular and detailed. Depending on the particular segment of the government (and not to mention the particular country), the government component must follow distinct specifications. In the case of the US Army, the MIL-spec (Military specification) is used. It is updated continuously and technical communications (in the form of Technical Manuals, Interactive Electronic Technical Manuals, Technical Bulletins, etc.) must be updated as well.

The Department of Defense utilizes Technical Manuals regularly and is a core part of the agency's responsibilities. Although detail oriented in their requirements, the DoD has deficiencies in technical communication. The following paper discusses those deficiencies and identifies the major contributing factors.

Collecting information

The next step is to collect information needed for accomplishing the stated purpose. Information may be collected through primary research, where the technical communicator conducts research first-hand, and secondary research, where work published by another person is used as an information source. The technical communicator must acknowledge all sources used to produce his or her work. To ensure that this is done, the technical communicator should distinguish quotations, paraphrases, and summaries when taking notes.

Organizing and outlining information

Before writing the initial draft, all the ideas are organized in a way that will make the document flow nicely. A good way of doing this is to write all random thoughts down on a paper, and then circle all main sections, connect the main sections to supporting ideas with lines, and delete all irrelevant material.

Once each idea is organized, the writer can then organize the document as a whole. This can be accomplished in various ways:

- Chronological: This is used for documents that involve a linear process, such as a step-by-step guide describing how to accomplish something.
- Parts of an object: Used for documents which describe the parts of an object, such as a graphic showing the parts of a computer (keyboard, monitor, mouse, etc.)
- Simple to Complex (or vice versa): Starts with the easy-to-understand ideas, and gradually goes deeper into complex ideas.
- Specific to General: Starts with many ideas, and then organizes the ideas into sub-categories.
- General to Specific: Starts with a few categories of ideas, and then goes deeper.

Once the whole document is organized, it's a good idea to create a final outline, which will show all the ideas in an easy-to-understand document. Creating an outline makes the entire writing process much easier and will save the author time.

Writing the first draft

After the outline is completed, the next step is to write the first draft. The goal is to write down ideas from the outline as quickly as possible. Setting aside blocks of one hour or more, in a place free of distractions, will help the writer maintain a flow. Also, the writer should wait until the draft is complete to do any revising; stopping to revise at this stage

will break the writer's flow. The writer should start with the section that is easiest for them, and write the summary only after the body is drafted.

The ABC (Abstract, Body, and Conclusion) format can be used when writing a first draft. The Abstract describes the subject to be written about, so that the reader knows what he or she is going to be told in the document. The Body is the majority of the paper, in which the topics are covered in depth. Lastly, the Conclusion section restates the main topics of the paper.

The ABC format can also be applied to individual paragraphs, beginning with a topic sentence that clearly states the paragraph's topic. This is followed by the topic, and finally, the paragraph closes with a concluding sentence.

Revising and editing

Once the initial draft is laid out, editing and revising can be done to fine-tune the draft into a final copy. Four tasks transform the early draft into its final form, suggested by Pfeiffer and Boogard:

Adjusting and reorganizing content

During this step, the draft is revisited to 1) focus or elaborate on certain topics which deserve more attention, 2) shorten other sections, and 3) shift around certain paragraphs, sentences, or entire topics.

Editing for style

Good style makes the writing more interesting, appealing, or readable. In general the personal writing style of the writer will not be evident in technical writing. Some changes are made by choice, not for correctness, and may include:

- adding headings, lists, graphics
- changing passive-voice sentences to an active voice
- defining terminology
- rearranging paragraphs
- shortening paragraphs
- shortening sentences

Technical writing is a discipline that usually requires a technical writer to make particular use of a style guide. These guides may relate to a specific project, product, company or brand and in general they ensure that technical writing is devoid of a personal style.

Editing for grammar

At this point, the document can be checked for grammatical errors, such as comma usage and common word confusions (for example, there/their/they're).

Edit for context

Determining the necessary amount of context is important. There needs to be a balance between exuberance, which may lead the audience to take unintended additional meaning from the text, and terseness, which may leave the audience unable to interpret meaning because of missing words.

Chapter 2

Usability

Usability is the ease of use and learnability of a human-made object. The object of use can be a software application, website, book, tool, machine, process, or anything a human interacts with. A usability study may be conducted as a primary job function by a *usability analyst* or as a secondary job function by designers, technical writers, marketing personnel, and others. It is widely used in consumer electronics, communication, and knowledge transfer objects (such as a cookbook, a document or online help) and mechanical objects such as a door handle or a hammer.

Usability includes methods of measuring usability and the study of the principles behind an object's perceived efficiency or elegance. In human-computer interaction and computer science, usability studies the elegance and clarity with which the interaction with a computer program or a web site (web usability) is designed. Usability differs from user satisfaction insofar as the former also embraces usefulness.

Introduction

The primary notion of usability is that an object designed with a generalized users' psychology and physiology in mind is, for example:

- More efficient to use—takes less time to accomplish a particular task
- Easier to learn—operation can be learned by observing the object
- More satisfying to use

Complex computer systems find their way into everyday life, and at the same time the market is saturated with competing brands. This has made usability more popular and widely recognized in recent years, as companies see the benefits of researching and developing their products with user-oriented methods instead of technology-oriented methods. By understanding and researching the interaction between product and user, the *usability expert* can also provide insight that is unattainable by traditional company-oriented market research. For example, after observing and interviewing users, the usability expert may identify needed functionality or design flaws that were not anticipated. A method called *contextual inquiry* does this in the naturally occurring context of the users own environment.

In the user-centered design paradigm, the product is designed with its intended users in mind at all times. In the user-driven or participatory design paradigm, some of the users become actual or de facto members of the design team.

The term *user friendly* is often used as a synonym for *usable*, though it may also refer to accessibility. Usability describes the quality of user experience across websites, software, products, and environments.

There is no consensus about the relation of the terms ergonomics (or human factors) and usability. Some think of usability as the software specialization of the larger topic of ergonomics. Others view these topics as tangential, with ergonomics focusing on physiological matters (e.g., turning a door handle) and usability focusing on psychological matters (e.g., recognizing that a door can be opened by turning its handle).

Usability is also important in website development (web usability). According to Jakob Nielsen, "Studies of user behavior on the Web find a low tolerance for difficult designs or slow sites. People don't want to wait. And they don't want to learn how to use a home page. There's no such thing as a training class or a manual for a Web site. People have to be able to grasp the functioning of the site immediately after scanning the home page—for a few seconds at most." Otherwise, most casual users simply leave the site and browse or shop elsewhere.

Definition

ISO defines usability as "The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use." Usability is a qualitative attribute that assesses how easy user interfaces are to use. The word "usability" also refers to methods for improving ease-of-use during the design process. Usability consultant Jakob Nielsen and computer science professor Ben Shneiderman have written (separately) about a framework of system acceptability, where usability is a part of "usefulness" and is composed of:

- Learnability: How easy is it for users to accomplish basic tasks the first time they encounter the design?
- Efficiency: Once users have learned the design, how quickly can they perform tasks?
- Memorability: When users return to the design after a period of not using it, how easily can they re establish proficiency?
- Errors: How many errors do users make, how severe are these errors, and how easily can they recover from the errors?
- Satisfaction: How pleasant is it to use the design?

Usability is often associated with the functionalities of the product (cf. ISO definition, below), in addition to being solely a characteristic of the user interface (cf. framework of system acceptability, also below, which separates *usefulness* into *utility* and *usability*). For example, in the context of mainstream consumer products, an automobile lacking a

reverse gear could be considered *unusable* according to the former view, and *lacking in utility* according to the latter view.

When evaluating user interfaces for usability, the definition can be as simple as "the perception of a target user of the effectiveness (fit for purpose) and efficiency (work or time required to use) of the Interface". Each component may be measured subjectively against criteria, e.g., Principles of User Interface Design, to provide a metric, often expressed as a percentage. It is important to distinguish between usability testing and usability engineering. Usability testing is the measurement of ease of use of a product or piece of software. In contrast, usability engineering (UE) is the research and design process that ensures a product with good usability.

Usability is a non-functional requirement. As with other non-functional requirements, usability cannot be directly measured but must be quantified by means of indirect measures or attributes such as, for example, the number of reported problems with ease-of-use of a system.

Intuitive interfaces

The term intuitive is often listed as a desirable trait in usable interfaces, often used as a synonym for learnable. Some experts such as Jef Raskin have discouraged using this term in user interface design, claiming that easy to use interfaces are often easy because of the user's exposure to previous similar systems, thus the term 'familiar' should be preferred. As an example: Two vertical lines "||" on media player buttons do not intuitively mean "pause"—they do so by convention. Aiming for "intuitive" interfaces (based on reusing existing skills with interaction systems) could lead designers to discard a better design solution only because it would require a novel approach. This position is sometimes illustrated with the remark that "The only intuitive interface is the nipple; everything else is learned."

Bruce Tognazzini even denies the existence of "intuitive" interfaces, since such interfaces must be able to intuit, i.e., "perceive the patterns of the user's behavior and draw inferences." Instead, he advocates the term "intuitable," i.e., "that users could intuit the workings of an application by seeing it and using it." He continues, however, "But even that is a less than useful goal since only 25 percent of the population depends on intuition to perceive anything."

Investigation

The key principle for maximizing usability is to employ iterative design, which progressively refines the design through evaluation from the early stages of design. The evaluation steps enable the designers and developers to incorporate user and client feedback until the system reaches an acceptable level of usability.

The preferred method for ensuring usability is to test actual users on a working system. Although, there are many methods for studying usability, the most basic and useful is user testing, which has three components:

- Get some representative users.
- Ask the users to perform representative tasks with the design.
- Observe what the users do, where they succeed, and where they have difficulties with the user interface.

It's important to test users individually and let them solve any problems on their own. If you help them or direct their attention to any particular part of the screen, you will bias the test. Rather than running a big, expensive study, it's better to run many small tests and revise the design between each one so you can fix the usability flaws as you identify them. Iterative design is the best way to increase the quality of user experience. The more versions and interface ideas you test with users, the better.

Usability plays a role in each stage of the design process. The resulting need for multiple studies is one reason to make individual studies fast and cheap, and to perform usability testing early in the design process. Here are the main steps:

- Before starting the new design, test the old design to identify good parts you should keep or emphasize, and bad parts that give users trouble.
- Test competitors' designs to get data on a range of alternative designs.
- Conduct a field study to see how users behave in their natural habitat.
- Make mock-ups or paper prototypes of one or more new design ideas and test them. The less time you invest in these design ideas the better, because you'll need to change them based on the test results.
- Refine the design ideas that test best through multiple iterations, gradually moving from low-fidelity prototyping to high-fidelity representations that run on the computer. Test each iteration.
- Inspect the design relative to established usability guidelines, whether from your own earlier studies or published research.
- Once you decide on and implement the final design, test it again. Subtle usability problems always creep in during implementation.

Don't defer user testing until you have a fully implemented design. If you do, it will be impossible to fix the vast majority of the critical usability problems that the test uncovers. Many of these problems are likely to be structural, and fixing them would require major rearchitecting. The only way to a high-quality user experience is to start user testing early in the design process and to keep testing every step of the way.

ISO standards

ISO/TR 16982:2002

ISO/TR 16982:2002 ("Ergonomics of human-system interaction—Usability methods supporting human-centered design") is a standard that provides information on human-centred usability methods that can be used for design and evaluation. It details the advantages, disadvantages, and other factors relevant to using each usability method. It explains the implications of the stage of the life cycle and the individual project characteristics for the selection of usability methods and provides examples of usability methods in context.

The main users of ISO/TR 16982:2002 are project managers. It therefore addresses technical human factors and ergonomics issues only to the extent necessary to allow managers to understand their relevance and importance in the design process as a whole. The guidance in ISO/TR 16982:2002 can be tailored for specific design situations by using the lists of issues characterizing the context of use of the product to be delivered. Selection of appropriate usability methods should also take account of the relevant life-cycle process. ISO/TR 16982:2002 is restricted to methods that are widely used by usability specialists and project managers. It does *not* specify the details of how to implement or carry out the usability methods described.

ISO 9241

ISO 9241 is a multi-part standard that covers a number of aspects of people working with computers. Although originally titled *Ergonomic requirements for office work with visual display terminals (VDTs)*, it has been retitled to the more generic *Ergonomics of Human System Interaction*.

As part of this change, ISO is renumbering the standard so that it can include many more topics. The first part to be renumbered was part 10 (now renumbered to part 110). Part 1 is a general introduction to the rest of the standard. Part 2 addresses task design for working with computer systems. Parts 3–9 deal with physical characteristics of computer equipment. Parts 110 and parts 11–19 deal with usability aspects of software, including Part 110 (a general set of usability heuristics for the design of different types of dialogue) and Part 11 (general guidance on the specification and measurement of usability).

Usability considerations

Usability includes considerations such as:

- Who are the users, what do they know, what can they learn?
- What do users want or need to do?
- What is the users' general background?
- What is the users' context for working?
- What must be left to the machine?

Answers to these are obtained through user and task analysis at the start of the project.

Other considerations

- Can users easily accomplish intended tasks at their desired speed?
- How much training do users need?
- What documentation or other supporting materials are available to help the user? Can users find solutions in these materials?
- What and how many errors do users make when they interact with the product?
- Can the user recover from errors? What do users have to do to recover from errors? Does the product help users recover from errors? For example, does software present comprehensible, informative, non-threatening error messages?
- Does the product meet the special needs of disabled users? (Is it (accessible?)
- Are there substantial differences between the cognitive approaches of various users that affect the design, or does a one-size-fits-all approach work?

Ways to answer these and other questions include user-focused requirements analysis, building user profiles, and usability testing.

Discoverability

Even if software is usable as per the above considerations, it may still be hard to *learn* to use. Other questions that must be asked are:

- Is the user ever expected to do something that is not obvious? (e.g., Are important features only accessible by right-clicking on a menu header, on a text box, or on an unusual GUI element?)
- Are there hints and tips and shortcuts that appear as the user is using the software?
- Should there be instructions in the manual that actually belong as contextual tips shown in the program?
- Is the user at a disadvantage if they don't know certain keyboard shortcuts? A user has the right to know all major and minor keyboard shortcuts and features of an application.
- Is the learning curve (of hints and tips) skewed towards point-and-click users rather than keyboard users?
- Are there any "hidden" or undocumented keyboard shortcuts, that would better be revealed in a "Keyboard shortcuts" Help-Menu item? A strategy to prevent this "undocumented feature disconnect" is to automatically generate a list of keyboard shortcuts from their definitions in the code.

Lund, 1997 usability maxims

When evaluating the design and usability of a website, consider the following:

- Know the user, and YOU are not the user.
- Things that look the same should act the same.

- The information for the decision must be there when the decision is needed.
- Error messages should actually mean something to the user and tell the user how to fix the problem.
- Every action should have a reaction.
- Everyone makes mistakes, so every mistake should be fixable.
- Don't overwhelm the user.
- Consistency, consistency, consistency.
- Minimize the need for a mighty memory.
- Keep it simple.
- The user should always know what is happening.
- The more you do something, the easier it should be to do.
- The user should control the system. The system should not control the user. The user is the boss and the system should show it.
- Eliminate unnecessary decisions and illuminate the rest.
- The best journey has the fewest steps. Shorten the distance between the user and the goal.
- User should be able to do what they want.
- Alert users to an error before things get worse.
- Users should always know how to find out what to do next.
- Strive to empower the user, not speed up the system.
- Things that look different should act different.

These are presented in a descending order determined by their mean rating of importance.

Designing for usability

Any system designed for people should be easy to use, easy to learn, easy to remember, and helpful to users. John Gould and Clayton Lewis recommend that designers striving for usability follow these three design principles

- Early focus on users and tasks
- Empirical measurement
- Iterative design

Early focus on users and tasks

The design team should be user driven and in direct contact with potential users. Several evaluation methods, including personas, cognitive modeling, inspection, inquiry, prototyping, and testing methods may contribute to understanding potential users.

Usability considerations such as who the users are and their experience with similar systems must be examined. As part of understanding users, this knowledge must "...be played against the tasks that the users will be expected to perform." This includes the analysis of what tasks the users will perform, which are most important, and what decisions the users will make while using your system. Designers must understand how cognitive and emotional characteristics of users will relate to a proposed system.

One way to stress the importance of these issues in the designers' minds is to use personas, which are made-up representative users.

Empirical measurement

Test the system early on, and test the system on real users using behavioral measurements. This includes testing the system for both learnability and usability. It is important in this stage to use quantitative usability specifications such as time and errors to complete tasks and number of users to test, as well as examine performance and attitudes of the users testing the system. Finally, "reviewing or demonstrating" a system before the user tests it can result in misleading results. The emphasis of empirical measurement is on measurement, both informal and formal, which can be carried out through a variety of evaluation methods.

Iterative design

Iterative design is a design methodology based on a cyclic process of prototyping, testing, analyzing, and refining a product or process. Based on the results of testing the most recent iteration of a design, changes and refinements are made. This process is intended to ultimately improve the quality and functionality of a design. In iterative design, interaction with the designed system is used as a form of research for informing and evolving a project, as successive versions, or iterations of a design are implemented. The key requirements for Iterative Design are: identification of required changes, an ability to make changes, and a willingness to make changes. When a problem is encountered, there is no set method to determine the correct solution. Rather, there are empirical methods that can be used during system development or after the system is delivered, usually a more inopportune time. Ultimately, iterative design works towards meeting goals such as making the system user friendly, easy to use, easy to operate, simple, etc.

Evaluation methods

There are a variety of usability evaluation methods. Certain methods use data from users, while others rely on usability experts. There are usability evaluation methods for all stages of design and development, from product definition to final design modifications. When choosing a method, consider cost, time constraints, and appropriateness.

Cognitive modeling methods

Cognitive modeling involves creating a computational model to estimate how long it takes people to perform a given task. Models are based on psychological principles and experimental studies to determine times for cognitive processing and motor movements. Cognitive models can be used to improve user interfaces or predict problem errors and pitfalls during the design process. A few examples of cognitive models include:

Parallel Design

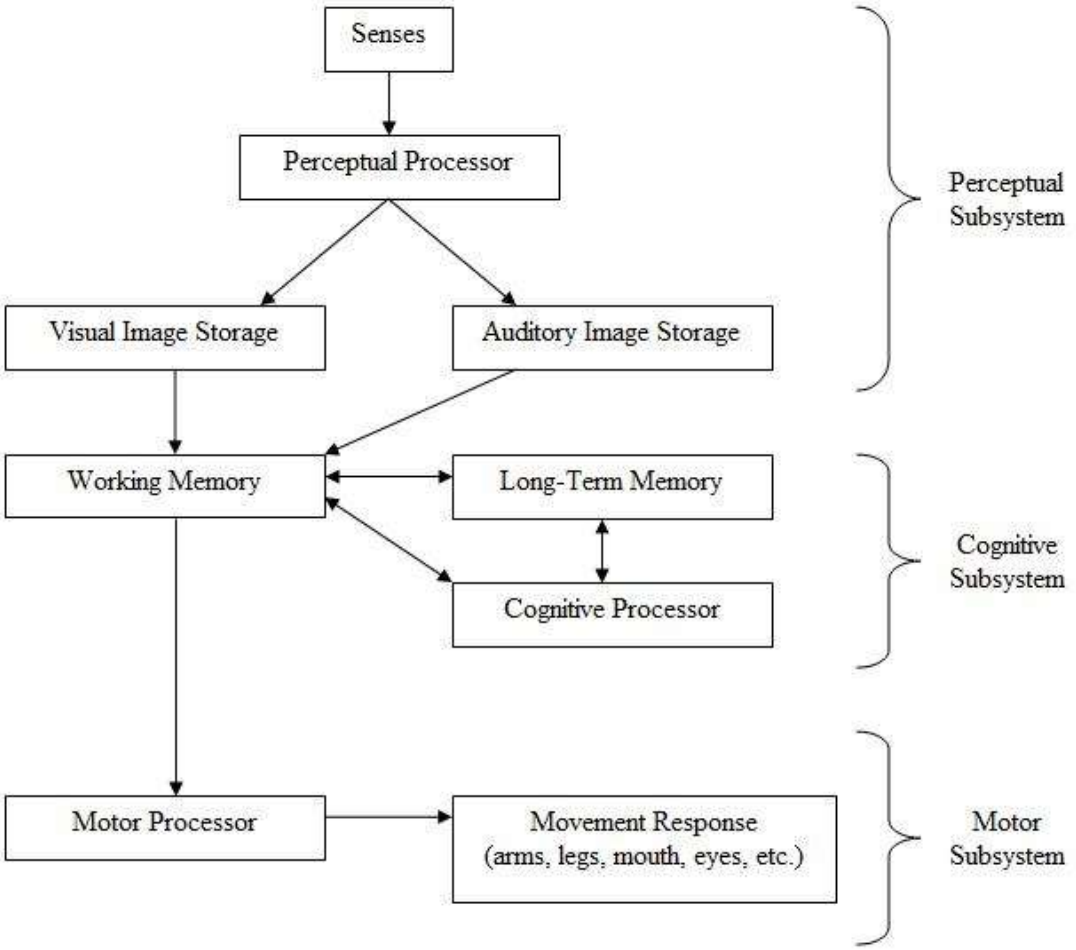
With parallel design, several people create an initial design from the same set of requirements. Each person works independently, and when finished, shares concepts with the group. The design team considers each solution, and each designer uses the best ideas to further improve their own solution. This process helps generate many different, diverse ideas, and ensures that the best ideas from each design are integrated into the final concept. This process can be repeated several times until the team is satisfied with the final concept.

GOMS

GOMS stands for *goals, operator, methods, and selection rules*. It is a family of techniques that analyzes the user complexity of interactive systems. Goals are what the user must accomplish. An operator is an action performed in pursuit of a goal. A method is a sequence of operators that accomplish a goal. Selection rules specify which method satisfies a given goal, based on context.

Human Processor Model

Sometimes it is useful to break a task down and analyze each individual aspect separately. This helps the tester locate specific areas for improvement. To do this, it is necessary to understand how the human brain processes information. A model of the human processor is shown below.



Many studies have been done to estimate the cycle times, decay times, and capacities of each of these processors. Variables that affect these can include subject age, aptitudes, ability, and the surrounding environment. For a younger adult, reasonable estimates are:

Parameter	Mean	Range
Eye movement time	230 ms	70-700 ms
Decay half-life of visual image storage	200 ms	90-1000 ms
Perceptual processor cycle time	100 ms	50-200 ms
Cognitive processor cycle time	70 ms	25-170 ms
Motor processor cycle time	70 ms	30-100 ms
Effective working memory capacity	7 items	5-9 items

Long-term memory is believed to have an infinite capacity and decay time.

Keystroke level modeling

Keystroke level modeling is essentially a less comprehensive version of GOMS that makes simplifying assumptions in order to reduce calculation time and complexity.

Inspection methods

These usability evaluation methods involve observation of users by an experimenter, or the testing and evaluation of a program by an expert reviewer. They provide more quantitative data as tasks can be timed and recorded.

Card sorts

Card sorting is a way to involve users in grouping information for a website's usability review. Participants in a card sorting session are asked to organize the content from a Web site in a way that makes sense to them. Participants review items from a Web site and then group these items into categories. Card sorting helps to learn how users think about the content and how they would organize the information on the Web site. Card sorting helps to build the structure for a Web site, decide what to put on the home page, and label the home page categories. It also helps to ensure that information is organized on the site in a way that is logical to users.

Tree tests

Tree testing is a way to evaluate the effectiveness of a website's top-down organization. Participants are given "find it" tasks, then asked to drill down through successive text lists of topics and subtopics to find a suitable answer. Tree testing evaluates the findability and labeling of topics in a site, separate from its navigation controls or visual design.

Ethnography

Ethnographic analysis is derived from anthropology. Field observations are taken at a site of a possible user, which track the artifacts of work such as Post-It notes, items on desktop, shortcuts, and items in trash bins. These observations also gather the sequence of work and interruptions that determine the user’s typical day.

Heuristic Evaluation

Heuristic evaluation is a usability engineering method for finding and assessing usability problems in a user interface design as part of an iterative design process. It involves having a small set of evaluators examining the interface and using recognized usability principles (the "heuristics"). It is the most popular of the usability inspection methods, as it is quick, cheap, and easy.

Heuristic evaluation was developed to aid in the design of computer user-interface design. It relies on expert reviewers to discover usability problems and then categorize and rate them by a set of principles (heuristics.) It is widely used based on its speed and

cost-effectiveness. Jakob Nielsen's list of ten heuristics is the most commonly used in industry. These are ten general principles for user interface design. They are called "heuristics" because they are more in the nature of rules of thumb than specific usability guidelines.

- *Visibility of system status*: The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.
- *Match between system and the real world*: The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.
- *User control and freedom*: Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.
- *Consistency and standards*: Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.
- *Error prevention*: Even better than good error messages is a careful design that prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.
- *Recognition rather than recall*: Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.
- *Flexibility and efficiency of use*: Accelerators—unseen by the novice user—may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.
- *Aesthetic and minimalist design*: Dialogues should not contain information that is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.
- *Help users recognize, diagnose, and recover from errors*: Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.
- *Help and documentation*: Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

Thus, by determining which guidelines are violated, the usability of a device can be determined.

Usability Inspection

Usability inspection is a review of a system based on a set of guidelines. The review is conducted by a group of experts who are deeply familiar with the concepts of usability in

design. The experts focus on a list of areas in design that have been shown to be troublesome for users.

Pluralistic Inspection

Pluralistic Inspections are meetings where users, developers, and human factors people meet together to discuss and evaluate step by step of a task scenario. As more people inspect the scenario for problems, the higher the probability to find problems. In addition, the more interaction in the team, the faster the usability issues are resolved.

Consistency Inspection

In consistency inspection, expert designers review products or projects to ensure consistency across multiple products to look if it does things in the same way as their own designs.

Activity Analysis

Activity analysis is a usability method used in preliminary stages of development to get a sense of situation. It involves an investigator observing users as they work in the field. Also referred to as user observation, it is useful for specifying user requirements and studying currently used tasks and subtasks. The data collected is qualitative and useful for defining the problem. It should be used when you wish to frame what is needed, or "What do we want to know?"

Inquiry methods

The following usability evaluation methods involve collecting qualitative data from users. Although the data collected is subjective, it provides valuable information on what the user wants.

Task Analysis

Task analysis means learning about users' goals and users' ways of working. Task analysis can also mean figuring out what more specific tasks users must do to meet those goals and what steps they must take to accomplish those tasks. Along with user and task analysis, we often do a third analysis: understanding users' environments (physical, social, cultural, and technological environments).

Focus Groups

A focus group is a focused discussion where a moderator leads a group of participants through a set of questions on a particular topic. Although typically used as a marketing tool, Focus Groups are sometimes used to evaluate usability. Used in the product definition stage, a group of 6 to 10 users are gathered to discuss what they desire in a product. An experienced focus group facilitator is hired to guide the discussion to areas

of interest for the developers. Focus groups are typically videotaped to help get verbatim quotes, and clips are often used to summarize opinions. The data gathered is not usually quantitative, but can help get an idea of a target group's opinion.

Questionnaires/Surveys

Surveys have the advantages of being inexpensive, require no testing equipment, and results reflect the users' opinions. When written carefully and given to actual users who have experience with the product and knowledge of design, surveys provide useful feedback on the strong and weak areas of the usability of a design. This is a very common method and often does not appear to be a survey, but just a warranty card.

Prototyping methods

Rapid Prototyping

Rapid prototyping is a method used in early stages of development to validate and refine the usability of a system. It can be used to quickly and cheaply evaluate user-interface designs without the need for an expensive working model. This can help remove hesitation to change the design, since it is implemented before any real programming begins. One such method of rapid prototyping is paper prototyping.

Testing methods

These usability evaluation methods involve testing of subjects for the most quantitative data. Usually recorded on video, they provide task completion time and allow for observation of attitude.

Remote usability testing

Remote usability testing (also known as unmoderated or asynchronous usability testing) involves the use of a specially modified online survey, allowing the quantification of user testing studies by providing the ability to generate large sample sizes. Additionally, this style of user testing also provides an opportunity to segment feedback by demographic, attitudinal and behavioural type. The tests are carried out in the user's own environment (rather than labs) helping further simulate real-life scenario testing. This approach also provides a vehicle to easily solicit feedback from users in remote areas.

Thinking Aloud

The Think aloud protocol is a method of gathering data that is used in both usability and psychology studies. It involves getting a user to verbalize their thought processes as they perform a task or set of tasks. Often an instructor is present to prompt the user into being more vocal as they work. Similar to the Subjects-in-Tandem method, it is useful in pinpointing problems and is relatively simple to set up. Additionally, it can provide insight into the user's attitude, which can not usually be discerned from a survey or questionnaire.

RITE Method

Rapid Iterative Testing and Evaluation (RITE) is an iterative usability method similar to traditional "discount" usability testing. The tester and team must define a target population for testing, schedule participants to come in to the lab, decide on how the users behaviors will be measured, construct a test script and have participants engage in a verbal protocol (e.g. think aloud). However it differs from these methods in that it advocates that changes to the user interface are made as soon as a problem is identified and a solution is clear. Sometimes this can occur after observing as few as 1 participant. Once the data for a participant has been collected the usability engineer and team decide if they will be making any changes to the prototype prior to the next participant. The changed interface is then tested with the remaining users.

Subjects-in-Tandem

Subjects-in-tandem is pairing of subjects in a usability test to gather important information on the ease of use of a product. Subjects tend to think out loud and through their verbalized thoughts designers learn where the problem areas of a design are. Subjects very often provide solutions to the problem areas to make the product easier to use.

Component-based usability testing

Component-based usability testing is an approach which aims to test the usability of elementary units of an interaction system, referred to as interaction components. The approach includes component-specific quantitative measures based on user interaction recorded in log files, and component-based usability questionnaires.

Other methods

Cognitive walkthrough

Cognitive walkthrough is a method of evaluating the user interaction of a working prototype or final product. It is used to evaluate the system’s ease of learning. Cognitive walkthrough is useful to understand the user’s thought processes and decision making when interacting with a system, specially for first-time or infrequent users.

Benchmarking

Benchmarking creates standardized test materials for a specific type of design. Four key characteristics are considered when establishing a benchmark: time to do the core task, time to fix errors, time to learn applications, and the functionality of the system. Once there is a benchmark, other designs can be compared to it to determine the usability of the system. Many of the common objectives of usability studies, such as trying to understand user behavior or exploring alternative designs, must be put aside. Unlike many other usability methods or types of labs studies, benchmark studies more closely resemble true

experimental psychology lab studies, with greater attention to detail on methodology, study protocol and data analysis.

Meta-Analysis

Meta-Analysis is a statistical procedure to combine results across studies to integrate the findings. This phrase was coined in 1976 as a quantitative literature review. This type of evaluation is very powerful for determining the usability of a device because it combines multiple studies to provide very accurate quantitative support.

Persona

Personas are fictitious characters created to represent a site or product's different user types and their associated demographics and technographics. Alan Cooper introduced the concept of using personas as a part of interactive design in 1998 in his book *The Inmates Are Running the Asylum*, but had used this concept since as early as 1975.

Personas are a usability evaluation method that can be used at various design stages. The most typical time to create personas is at the beginning of designing so that designers have a tangible idea of who the users of their product will be. Personas are the archetypes that represent actual groups of users and their needs, which can be a general description of person, context, or usage scenario. This technique turns marketing data on target user population into a few physical concepts of users to create empathy among the design team, with the final aim of tailoring a product more closely to how the personas will use it.

To gather the marketing data that personas require, several tools can be used, including online surveys, web analytics, customer feedback forms, and usability tests, and interviews with customer-service representatives.

Evaluating with tests and metrics

Regardless to how carefully a system is designed, all theories must be tested using usability tests. Usability tests involve typical users using the system (or product) in a realistic environment. Observation of the user’s behavior, emotions, and difficulties while performing different tasks, often identify areas of improvement for the system.

Prototypes

It is often very difficult for designers to conduct usability tests with the exact system being designed. Cost constraints, size, and design constraints usually lead the designer to creating a prototype of the system. Instead of creating the complete final system, the designer may test different sections of the system, thus making several small models of each component of the system. The types of usability prototypes may vary from using paper models, index cards, hand drawn models, or storyboards.

Prototypes are able to be modified quickly, often are faster and easier to create with less time invested by designers and are more apt to change design; although sometimes are not an adequate representation of the whole system, are often not durable and testing results may not be parallel to those of the actual system.

Metrics

While conducting usability tests, designers must use usability metrics to identify what it is they are going to measure, or the usability metrics. These metrics are often variable, and change in conjunction with the scope and goals of the project. The number of subjects being tested can also affect usability metrics, as it is often easier to focus on specific demographics. Qualitative design phases, such as general usability (can the task be accomplished?), and user satisfaction are also typically done with smaller groups of subjects. Using inexpensive prototypes on small user groups provides more detailed information, because of the more interactive atmosphere, and the designer's ability to focus more on the individual user.

As the designs become more complex, the testing must become more formalized. Testing equipment will become more sophisticated and testing metrics become more quantitative. With a more refined prototype, designers often test effectiveness, efficiency, and subjective satisfaction, by asking the user to complete various tasks. These categories are measured by the percent that complete the task, how long it takes to complete the tasks, ratios of success to failure to complete the task, time spent on errors, the number of errors, rating scale of satisfactions, number of times user seems frustrated, etc. Additional observations of the users give designers insight on navigation difficulties, controls, conceptual models, etc. The ultimate goal of analyzing these metrics is to find/create a prototype design that users like and use to successfully perform given tasks.

After conducting usability tests, it is important for a designer to record what was observed, in addition to why such behavior occurred and modify the model according to the results. Often it is quite difficult to distinguish the source of the design errors, and what the user did wrong. However, effective usability tests will not generate a solution to the problems, but provide modified design guidelines for continued testing.

Benefits of usability

The key benefits of usability are:

- Higher revenues through increased sales
- Increased user efficiency and satisfaction
- Reduced development costs
- Reduced support costs

Corporate integration

An increase in usability generally positively affects several facets of a company's output quality. In particular, the benefits fall into several common areas:

- Increased productivity
- Decreased training and support costs
- Increased sales and revenues
- Reduced development time and costs
- Reduced maintenance costs
- Increased customer satisfaction

Increased usability in the workplace fosters several responses from employees. Along with any positive feedback, "workers who enjoy their work do it better, stay longer in the face of temptation, and contribute ideas and enthusiasm to the evolution of enhanced productivity." In order to create standards, companies often implement experimental design techniques that create baseline levels. Areas of concern in an office environment include (though are not necessarily limited to):

- Working Posture
- Design of Workstation Furniture
- Screen Displays
- Input Devices
- Organizational Issues
- Office Environment
- Software Interface

By working to improve said factors, corporations can achieve their goals of increased output at lower costs, while potentially creating optimal levels of customer satisfaction. There are numerous reasons why each of these factors correlates to overall improvement. For example, making a piece of software's user interface easier to understand would reduce the need for extensive training. The improved interface would also tend to lower the time needed to perform necessary tasks, and so would both raise the productivity levels for employees and reduce development time (and thus costs). It is important to note that each of the aforementioned factors are not mutually exclusive, rather should be understood to work in conjunction to form the overall workplace environment.

Conclusion

Usability is now recognized as an important software quality attribute, earning its place among more traditional attributes such as performance and robustness. Various academic programs focus on usability. Several usability consultancy companies have emerged, and traditional consultancy and design firms offer similar services.

Chapter 3

Datasheet

A **datasheet**, **data sheet**, or **spec sheet** is a document summarizing the performance and other technical characteristics of a product, machine, component (e.g. an electronic component), material, a subsystem (e.g. a power supply) or software in sufficient detail to be used by a design engineer to integrate the component into a system. Typically, a datasheet is created by the component/subsystem/software manufacturer and begins with an introductory page describing the rest of the document, followed by listings of specific characteristics, with further information on the connectivity of the devices. In cases where there is relevant source code to include, it is usually attached near the end of the document or separated into another file.

Depending on the specific purpose, a data sheet may offer an average value, a typical value, a typical range, engineering tolerances or a nominal value. The type and source of data are usually stated on the data sheet.

A data sheet is usually used for technical communication to describe technical characteristics of an item or product. It can be published by the manufacturer to help people choose products or to help use the products. By contrast, a technical specification is an explicit set of requirements to be satisfied by a material, product, or service.

Typical electronics datasheet information

Examples include:

Electronics component

A typical datasheet for an electronic component contains most of the following information:

- manufacturer's name
- product number and name
- list of available package formats (with images) and ordering codes

- notable device properties
- short functional description
- pin connection diagram
- absolute minimum, maximum ratings (supply voltage, power consumption, input currents, temperatures for storage, operating, soldering, etc)
- recommended operating conditions (as absolute minimum, maximum ratings)
- DC specifications (various temperatures, supply voltages, input currents, etc.)
- AC specifications (various temperatures, supply voltages, frequencies, etc.)
- input/output wave shape diagram
- timing diagram
- physical details showing minimum/typical/maximum dimensions, contact locations and sizes
- test circuit
- ordering codes for differing packages and performance criteria
- liability disclaimer regarding device use in certain environments such as nuclear power stations and life-support systems
- application recommendations, such as required filter capacitors, circuit board layout, etc.
- errata, often published prior to subsequent correction and relevant datasheet revision

Personal computer

- Number of ports
  - IEEE 1394, SPDIF, PS/2, Serial, USB, Parallel, LAN
- Expansion bays
  - 5.25 inch bays
  - 3.5 inch bays
- Mainboard
  - CPU socket
    - Front-side bus (FSB)
    - Back-side bus (BSB)
  - Chipset
    - North bridge
    - South bridge
  - Random access memory (RAM) slots
  - Peripheral Component Interconnect (PCI) and PCI Express (PCIe) buses and slots
  - Floppy, ATA (IDE) and SATA interfaces
  - Fans and temperature monitoring
  - Integrated graphics controllers
  - Integrated LAN interfaces
  - BIOS
  - Form factor
- Graphics card
  - AGP type

    - Memory
- Audio card
- Hardware compatibility requirements and basic setup details for computer device drivers
- operating system and other installed Software (if included)

A datasheet may include a "typical use" circuit diagram, as well as programming examples in the case of programmable devices. However, this sort of information is often published in a separate **application note**, with a high level of detail.

Historically, datasheets were typically available in a **databook** containing many data sheets, usually grouped by manufacturer or general type. Today, they are also available through the Internet in table form or via downloadable (usually PDF) documents.

Application notes

An application note is a document that gives more specific details on using a component in a specific application, or relating to a particular process, e.g. the physical assembly of a product containing the component. Application notes are especially useful for giving guidance on more unusual uses of a particular component, which would be irrelevant to many readers of the more widely read datasheet.

Application notes may either be appended to a datasheet, or presented as a separate document.

Chemical data sheets

Material Safety Data Sheets

A Material Safety Data Sheet, or MSDS, is a technical data sheet summarizing information about material identification; hazardous ingredients; health, physical, and fire hazards; first aid; chemical reactivities and incompatibilities; spill, leak, and disposal procedures; and protective measures required for safe handling and storage. These are required by agencies such as OSHA in its Hazard Communication Standard, 29 C.F.R. 1910.1200.

Chemical data

Data sheets and pages are available for specific properties of chemicals in Chemical elements data references: example, Melting points of the elements (data page). Specific materials have techncial data in individual sheets such as Ethanol (data page): this includes subjects such as structure and properties, thermodynamic properties, spectral data, vapor pressure, etc. Other chemical data sheets are available from individual producers of chemicals, often on their web pages.

Data sheets for automobiles

Data sheets for automobiles may be described under several names such as features, specs, engineering data, technical summary, etc. They help communicate the technical information about a car to potential buyers and are useful for comparisons with similar cars. They might include: critical inside and outside dimensions, weight, fuel efficiency, engine and drive train, towing capability, safety features and options, warranty, etc.

Other data sheets

Data sheets are common with a wide range of materials, components, machinery, structures, systems, etc.

Chapter 4

Darwin Information Typing Architecture

The **Darwin Information Typing Architecture** (**DITA**) is an XML-based architecture for authoring, producing, and delivering information. Although its main applications have so far been in technical publications, DITA is also used for other types of documents such as policies, procedures, and training.

Origin and name

The DITA architecture and a related DTD and XML Schema were originally developed by IBM. The architecture incorporates ideas in XML architecture, such as modular information architecture, various features for content reuse, and *specialization*, that had been developed over previous decades. DITA is now an OASIS standard.

The first word in the name "Darwin Information Typing Architecture" is a reference to the naturalist Charles Darwin. The key concept of "specialization" in DITA is in some ways analogous to Darwin's concept of evolutionary adaptation, with a specialized element inheriting the properties of the base element from which it is specialized.

Features and limitations

Topic orientation

DITA content is written as modular *topics*, as opposed to long "book-oriented" files. A DITA *map* contains links to topics, organized in the sequence (which may be hierarchical) in which they are intended to appear in finished documents. A DITA map defines the table of contents for deliverables. *Relationship tables* in DITA maps can also specify which topics link to each other.

Modular topics can be easily reused in different deliverables. However, the strict topic-orientation of DITA makes it an awkward fit for content that contains lengthy narratives that do not lend themselves to being broken into small, standalone chunks. Experts stress the importance of content analysis in the early stages of implementing structured authoring.

Content references

Fragments of content within topics (or less commonly, the topics themselves) can be reused through the use of content references (*conref*), a transclusion mechanism.

Conditional text

Conditional text allows filtering or styling content based on attributes for audience, platform, product, and other properties.

Metadata

DITA includes extensive metadata elements and attributes, which make topics easier to find.

Information typing

DITA specifies three basic topic types: *Task*, *Concept* and *Reference*. Each of the three basic topic types is a specialization of a generic *Topic* type, which contains a title element, a prolog element for metadata, and a body element. The body element contains paragraph, table, and list elements, similar to HTML.

1. A *Task* topic is intended for a procedure that describes how to accomplish a task. A Task topic lists a series of steps that users follow to produce an intended outcome. The steps are contained in a taskbody element, which is a specialization of the generic body element. The steps element is a specialization of an ordered list element.
2. *Concept* information is more objective, containing definitions, rules, and guidelines.
3. A *Reference* topic is for topics that describe command syntax, programming instructions, and other reference material, and usually contains detailed, factual material.

Specialization

DITA allows adding new elements and attributes through *specialization* of base DITA elements and attributes. Through specialization, DITA can accommodate new topic types, element types, and attributes as needed for specific industries or companies. Specializations of DITA for specific industries, such as the semiconductor industry, are standardized through OASIS technical committees or subcommittees. A significant percentage of organizations using DITA also develop their own specializations.

The extensibility of DITA permits organizations to specialize DITA by defining specific information structures and still use standard tools to work with them. The ability to define company-specific information architectures enables companies to use DITA to enrich

content with metadata that is meaningful to them, and to enforce company-specific rules on document structure.

Compatibility with non-DITA content

The element types and structures in DITA topics are similar to popular languages such as HTML. For example, a bulleted or numbered list can be copied and pasted directly from HTML to DITA.

DITA maps can include both DITA topics and non-DITA documents (such as HTML files and Microsoft Word documents) in document hierarchies. However, processors are generally limited in their ability to merge DITA and non-DITA content into consolidated printed documents.

Creating content in DITA

DITA map and topic documents are XML files. As with HTML, any images, video files, or other files which need to appear in output are inserted via reference. Any XML editor can therefore be used to write DITA content, with the exception of editors that support only a limited set of XML schemas (such as XHTML editors). Various editing tools have been developed that provide specific features to support DITA, such as visualization of conrefs.

Publishing content written in DITA

DITA is conceived as an end-to-end architecture. In addition to indicating what elements, attributes, and rules are part of the DITA language, the DITA specification includes rules for publishing DITA content in print, HTML, online Help, and other formats.

For example, the DITA specification indicates that if the *conref* attribute of element *A* contains a path to element *B*, the contents of element *B* will be displayed in the location of element *A*. DITA-compliant publishing solutions, known as DITA processors, must handle the *conref* attribute according to the specified behaviour. Rules also exist for processing other rich features such as conditional text, index markers, and topic-to-topic links. Applications that transform DITA content into other formats, and meet the DITA specification's requirements for interpreting DITA markup, are known as *DITA processors*.

DITA Open Toolkit

When DITA was released as a public XML standard in 2001, IBM contributed the **DITA Open Toolkit** (DITA OT) to the wider community. The DITA OT was therefore the first DITA processor, and continues to be the foundation of most publishing of DITA content. It is currently an active open-source project, with contributions from several companies.

Out of the box, the DITA OT handles all valid DITA specializations and produces several output formats, including:

- PDF, through XSL-FO
- XHTML
- Microsoft Compiled HTML Help
- Eclipse Help
- Java Help
- Oracle Help
- Rich Text Format

The DITA OT can also be extended to produce other (arbitrary) output formats. The raw DITA OT can be run from the command line. Some DITA authoring tools and content management systems now integrate the DITA OT, or parts of it, into their own publishing workflows. Standalone tools have also been developed to run the DITA OT via a graphical user interface instead of the command line.

The DITA OT includes customizable stylesheets that control the formatting and layout of human-readable deliverables.

Brief history

- March 2001 Introduction by IBM
- May 2002 Domain specialization added to topic specialization
- April 2004 OASIS Technical Committee for DITA formed
- February 2005 SourceForge begins DITA Open Toolkit support
- June 2005 DITA v1.0 approved as an OASIS standard
- August 2005 DITA Open Toolkit v1.1 is released
- March 2006 OASIS launches DITA.XML.org
- August 2007 DITA V1.1 is approved by OASIS, including Bookmap specialization
- December 2010 DITA V1.2 is approved by OASIS, includes:
    - Indirect linking with keys
    - New content reuse features
    - Enhanced glossary support, including acronyms
    - New industry specializations (Training, Machinery)
    - New support for controlled values / taxonomies (Subject Scheme specialization)

Chapter 5

DocBook

DocBook	
Filename extension	`.dbk, .xml`
Internet media type	`application/docbook+xml`
Developed by	OASIS
Type of format	markup language
Extended from	SGML, XML
Standard(s)	4.5 (June 2006), 5.0 (November 2009)

DocBook is a semantic markup language for technical documentation. It was originally intended for writing technical documents related to computer hardware and software but it can be used for any other sort of documentation.

As a semantic language, DocBook enables its users to create document content in a presentation-neutral form that captures the logical structure of the content; that content can then be published in a variety of formats, including HTML, XHTML, EPUB, PDF, man pages and HTML Help, without requiring users to make any changes to the source.

Overview

DocBook is an XML language. In its current version (5.0), DocBook's language is formally defined by a RELAX NG schema with integrated Schematron rules. (There are also W3C XML Schema+Schematron and Document Type Definition (DTD) versions of the schema available, but these are considered non-standard.)

As a semantic language, DocBook documents do not describe what their contents "look like," but rather the *meaning* of those contents. For example, rather than explaining how the abstract for an article might be visually formatted, DocBook simply says that a

particular section *is* an abstract. It is up to an external processing tool or application to decide where on a page the abstract should go and what it should look like. (And, indeed, to decide whether or not it should be included in the final output at all.)

DocBook provides a vast number of semantic element tags. They are divided into three broad categories: structural, block-level, and inline.

Structural tags specify broad characteristics of their contents. The `book` element, for example, specifies that its child elements represent the parts of a book. This includes a title, chapters, glossaries, appendices, and so on. DocBook's structural tags include, but are not limited to:

- `set`: a titled collection of one or more `book`s. Sets can be nested with other sets.
- `book`: a titled collection of `chapter`s, `article`s, and/or `part`s, with optional glossaries, appendices, and so forth.
- `part`: a titled collection of one or more `chapter`s. Parts can be nested with other parts. May have special introductory text.
- `article`: a titled, unnumbered collection of block-level elements.
- `chapter`: a titled, numbered collection of block-level elements. DocBook does not actually require that chapters be explicitly given numbers; it is understood by the semantics that the number of a chapter is the number of previous chapter elements in the XML document plus 1.
- `appendix`: the contained text represents an appendix.
- `dedication`: the text represents the dedication of the contained structural element.

Structural elements can contain other structural elements. Structural elements are the only permitted top-level elements in a DocBook document.

Block-level tags are elements like paragraph, lists, and so forth. Not all of these elements can contain actual text directly. Sequential block-level elements are expected to be rendered one "after" another. After, in this case, can differ depending on the language. In most Western languages, "after" means below: text paragraphs are printed down the page. Other languages' writing systems can have different directionality; for example, in Japanese, text is often printed in columns, with paragraphs running from right to left, so "after" in that case would be to the left. DocBook semantics are entirely neutral to these kinds of language-based concepts.

Inline-level tags are elements like emphasis, hyperlinks, and so forth. They wrap text within a block-level element. These elements do not cause the text to break when rendered in a paragraph format, but typically they cause the document processor to apply some kind of distinct typographical treatment to the enclosed text, by changing the font, size, or similar attributes. (The DocBook specification *does* say that it expects different typographical treatment, but it does not offer specific requirements as to what this treatment may be.) That is, it is not required that a DocBook processor transform an

`emphasis` tag into "italics." A reader-based DocBook processor could increase the volume of the words. Or, a text-based processor could use bold instead of italics.

Sample document

```
 <?xml version="1.0" encoding="UTF-8"?>
 <book xml:id="simple_book" xmlns="http://docbook.org/ns/docbook"
version="5.0">
   <title>Very simple book</title>
   <chapter xml:id="chapter_1">
     <title>Chapter 1</title>
     <para>Hello world!</para>
     <para>I hope that your day is proceeding
<emphasis>splendidly</emphasis>!</para>
   </chapter>
   <chapter xml:id="chapter_2">
     <title>Chapter 2</title>
     <para>Hello again, world!</para>
   </chapter>
 </book>
```

Semantically, this document is a "book," with a "title," that contains two "chapters" each with their own "titles." Those "chapters" contain "paragraphs" that have text in them. The markup is fairly readable in English.

In more detail, the root element of the document is `book`. All DocBook elements are in an XML Namespace, so the root element has an *xmlns* attribute to set the current namespace. Also, the root element of a DocBook document must have a *version* that specifies the version of the format that the document is built on.

(XML documents can include elements from multiple namespaces at once. For simplicity, the example does not illustrate this.)

A `book` element must contain a `title`, or an `info` element containing a `title`. This must be before any child structural elements. Following the title are the structural children, in this case, two `chapter` elements. Each of these must have a `title`. They contain `para` block elements which can contain free text and other inline elements like the `emphasis` in the second paragraph of the first chapter.

Schemas and validation

Rules such as the ones alluded to in the preceding paragraph (*"a `book` element must contain a `title`, or an `info` element containing a `title`,"* etc.) are formally defined in the DocBook *schema.* Appropriate programming tools can be used to *validate* an XML document (DocBook or otherwise), against its corresponding schema, in order to determine if (and if so, where) the document fails to conform to that schema. XML editing tools can also use schema information to avoid creating non-conforming documents in the first place.

DocBook authoring

Because DocBook is XML, documents can be created and edited with any text editor. A dedicated XML Editor is likewise a functional DocBook editor. DocBook provides schema files for popular XML schema languages, so any XML Editor that can provide content completion based on a schema can do so for DocBook. Many graphical or WYSIWYG XML editors come with the ability to edit DocBook like a Word Processor.

DocBook processing

Because DocBook is an XML format, conforming to a well-defined schema, documents can be validated and processed using any tool or programming language which includes XML support.

DocBook files are used to prepare output files in a wide variety of formats. Nearly always, this is accomplished using DocBook XSL stylesheets. These are XSLT stylesheets that transform DocBook documents into a number of formats (HTML, XSL-FO for later conversion into PDF, etc). These stylesheets can be sophisticated enough to generate tables of contents, glossaries, and indexes. They can oversee the selection of particular designated portions of a master document to produce different versions of the same document (such as a "tutorial" or a "quick-reference guide," where both of these consist of a subset of the material).

Because the standard DocBook XSL stylesheets *are* well-formed XSL stylesheets, and DocBook *is* well-formed XML, users can write their own customized stylesheets or even a full-fledged program to process the DocBook into an appropriate output format as their needs dictate.

History

DocBook began in 1991 as a joint project of HAL Computer Systems and O'Reilly & Associates and eventually spawned its own maintenance organization (the Davenport Group) before moving in 1998 to the *SGML Open* consortium, which subsequently became OASIS. DocBook is currently maintained by the *DocBook Technical Committee* at OASIS.

DocBook is available in both SGML and XML forms, as a DTD. RELAX NG and W3C XML Schema forms of the XML version are available. Starting with DocBook 5, the RELAX NG version is the "normative" form from which the other formats are generated.

DocBook originally started out as an SGML application, but an equivalent XML application was developed and has now replaced the SGML one for most uses. (Starting with version 4 of the SGML DTD, the XML DTD continued with this version numbering scheme.) Initially, a key group of software companies used DocBook since their representatives were involved in its initial design. Eventually, however, DocBook was adopted by the open source community where it has become a standard for creating

documentation for many projects, including FreeBSD, KDE, GNOME desktop documentation, the GTK+ API references, the Linux kernel documentation, and the work of the Linux Documentation Project.

Norman Walsh and the DocBook Project development team maintain the key application for producing output from DocBook source documents: A set of XSL stylesheets (as well as a legacy set of DSSSL stylesheets) that can generate high-quality HTML and print (FO/PDF) output, as well as output in other formats, including RTF, man pages and HTML Help.

Walsh is also the principal author of the book *DocBook: The Definitive Guide*, the official documentation of DocBook. This book is available online under the GFDL, and also as a print publication.

Pre DocBook v5.0

The current version of DocBook, 5.0, is fairly recent. Prior versions have been and still are in widespread use, so this section provides an overview of the changes to the older 4.x formats.

Until DocBook 5, DocBook was defined normatively by a Document Type Definition (DTD). Since DocBook was built originally as an application of SGML, the DTD was the only available schema language. DocBook 4.x formats can be SGML or XML, but the XML version does not have its own namespace.

As an outgrowth of being defined by a DTD, DocBook 4.x formats were required to live within the restrictions of being defined by a DTD. The most significant for the language being that an element name uniquely defines its possible contents. That is, an element named `info` must contain the same information no matter where it is in the DocBook file. As such, there are many kinds of info elements in DocBook 4.x: `bookinfo`, `chapterinfo`, etc. Each of them has a slightly different content model, but they do share some of their content model. Additionally, they repeat context information. The book's `info` element is that because it is a direct child of the book; it does not need to be named specially for a human reader. However, because the format was defined by a DTD, it did have to be named as such.

The root element does not have or need a *version*, as the version is built into the DTD declaration at the top of a pre-DocBook 5 document.

DocBook 4.x documents are not compatible with DocBook 5, but they can be converted into DocBook 5 documents through the use of an XSLT stylesheet. One is provided as part of the distribution of the DocBook 5 schema and specification package.

Simplified DocBook

DocBook offers a large number of features that may be overwhelming to a new user. For those who want the convenience of DocBook without a large learning curve, **Simplified DocBook** was designed. It is a small subset of DocBook designed for single documents such as articles or white papers (i.e., "books" are not supported). The Simplified DocBook DTD is currently at version 1.1.

Chapter 6

Virtual Community

A **virtual community** is a social network of individuals who interact through specific media, potentially crossing geographical and political boundaries in order to pursue mutual interests or goals. One of the most pervasive types of virtual community include social networking services, which consist of various online communities.

The term *virtual community* is attributed to the book of the same title by Howard Rheingold, published in 1993. The book, which could be considered a social enquiry, putting the research in the social sciences, discussed his adventures on The WELL and onward into a range of computer-mediated communication and social groups, broadening it to information science. The technologies included Usenet, MUDs (Multi-User Dungeon) and their derivatives MUSHes and MOOs, Internet Relay Chat (IRC), chat rooms and electronic mailing lists; the World Wide Web as we know it today was not yet used by many people. Rheingold pointed out the potential benefits for personal psychological well-being, as well as for society at large, of belonging to such a group.

These virtual communities all encourage interaction, sometimes focusing around a particular interest, or sometimes just to communicate. Quality virtual communities do both. They allow users to interact over a shared passion, whether it be through message boards, chat rooms, social networking sites, or virtual worlds.

Introduction

The traditional definition of a community is of a geographically circumscribed entity (neighborhoods, villages, etc). Virtual communities, of course, are usually dispersed geographically, and therefore are not communities under the original definition. Some offline communities are linked geographically, and are known as community websites. However, if one considers communities to simply possess boundaries of some sort between their members and non-members, then a virtual community is certainly a community. Virtual communities resemble real life communities in the sense that they both provide support, information, friendship and acceptance between strangers.

Early research into the existence of media-based communities was concerned with the nature of reality, whether communities actually could exist through the media, which could place virtual community research into the social sciences definition of ontology. In the 17th-century, scholars associated with the Royal Society of London formed a

community through the exchange of letters. "Community without propinquity", coined by urban planner Melvin Webber in 1963 and "community liberated," analyzed by Barry Wellman in 1979 began the modern era of thinking about non-local community. As well, Benedict Anderson's *Imagined Communities* in 1983, described how different technologies, such as national newspapers, contributed to the development of national and regional consciousness among early nation-states.

Purpose of virtual communities

Virtual communities are used for a variety of social and professional groups. It does not necessarily mean that there is a strong bond among the members, although Howard Rheingold, author of the book of the same name, mentions that virtual communities form "when people carry on public discussions long enough, with *sufficient human feeling,* to form webs of personal relationships". An email distribution list may have hundreds of members and the communication which takes place may be merely informational (questions and answers are posted), but members may remain relative strangers and the membership turnover rate could be high. This is in line with the liberal use of the term *community*.

Internet-based virtual communities

The explosive diffusion of the Internet since the mid-1990s has also fostered the proliferation of virtual communities taking the form of social networking services and online communities. The nature of those communities is diverse, and the benefits that Rheingold envisioned are not necessarily realized, or pursued, by many. At the same time, it is rather commonplace to see anecdotes of someone in need of special help or in search of a community benefiting from the use of the Internet.

Virtual communities may synthesize Web 2.0 technologies with the community, and therefore have been described as Community 2.0, although strong community bonds have been forged online since the early 1970s on timeshare systems like PLATO and later on USENET. Online communities depend upon social interaction and exchange between users online. This emphasizes the reciprocity element of the unwritten social contract between community members.

The embedding of virtual community in the experiences of everyday life and its reflection of and influence on the communication practices and patterns of identity formation make online community a colossal research enterprise which requires continuous investigation and theorizing.

Impacts of virtual communities

On health

It's argued that online relations are not as valuable as offline ones because there is less socialization. Concerns with this kind of interaction also include verbal aggression and

inhibitions, promotion of suicide and issues with privacy. Studies regarding the health effects of these communities did not show any negative effects, but that doesn't mean that there is no harm done. There was a high drop out rate of participants in the study that if continued could have led to a negative net effect. The health related effects are not clear because of the lack of thoroughness and the variation in studies done on the subject.

On civic participation

Online communities seem to have a direct impact on civic participation. 20.3% of members do something in real life at least once a year to support a cause related to their online community. 65% of members have started involvement in civic causes since they connected to the Internet. 43.7% are more involved with social activism since connecting with their online communities. Over half of virtual community members sign into their respective communities every day and 70% interact with other members daily.

Types of virtual communities

Internet message boards

An online message board is a forum where people can discuss thoughts or ideas on various topics. Online message centers allow users to choose which thread, or board of discussion, users would like to read or contribute to. A user will start a discussion by making a post on a thread. Other users who choose to respond can follow the discussion by adding their own post to that thread. Message boards are not conversation based because user responses do not have to take place right away. Whenever the user revisits the message board, he/she can make a response. Unlike a conversation, message boards do not have an instantaneous response and require that users actively go to the site to check for responses.

Anyone can register to participate in an online message board. A message board is unique because people can choose to participate and be apart of the virtual community, even if they choose not to contribute their thoughts and ideas. Registered users can simply view the various threads or contribute if they choose to. Message boards can also accommodate an almost infinite number of users, something a chat room is limited to.

Internet users' urges to talk to and reach out to strangers online opposes real-life encounters where people are hesitant and often unwilling to step in to help strangers. Studies have shown that people are more likely to intervene if they are the only one in the situation. With Internet message boards, a user sitting at his or her computer is the only one present in their online experience, which might have to do with why they are more willing to reach out. Another possible reason for this is that people can withdraw from a situation much easier online. They can simply click exit or log off, whereas they would have to find a physical exit and deal with the repercussions of trying to leave a situation in real life. The lack of status that is presented with an online identity also might encourage people because if you choose to keep it private, there is no label of gender, age, ethnicity, race or lifestyle associated with yourself

Online chat rooms

Shortly after the rise of interest in message boards and forums, people started to want a way of communicating with their "communities" in real time. The downside to message boards was that people would have to wait until another user replied to their posting, which, with people all around the world in different time frames, could take awhile. The development of online chat rooms allowed people to talk to whoever was online at the same time they were. This way, messages were sent and online users could immediately respond back.

The original development by CompuServe CB hosted forty channels in which users could talk to one another in real time. The idea of forty different channels led to the idea of chat rooms that were specific to different topics. Users could choose to join an already existent chat room they found interesting, or start a new "room" if they found nothing to their liking. Real time chatting was also brought into virtual games, where people could play against one another and also talk to one another through text. Now, chat rooms can be found on all sorts of topics, so that people can talk with others who share similar interests. Chat rooms are now provided by Internet Relay Chat (IRC) and other individual websites such as Yahoo, MSN, and AOL.

Chat room users communicate through text based messaging. Most chat room providers are similar and include an input box, a message window, and a participant list. The input box is where users can type their text based message to be sent to the providing server. The server will then transmit the message to the computers of anyone in the chat room so that it can be displayed in the message window. The message window allows the conversation to be tracked and usually places a time stamp once the message is posted. There is usually a list of the users who are currently in the room, so that people can see who is in their virtual community.

Users can communicate as if they are speaking to one another in real life. This "like reality" attribute makes it easy for users to form a virtual community, because chat rooms allow users to get to know one another as if they were meeting in real life. The individual "room" feature also makes it more likely that the people within a chat room share a similar interest; an interest that allows them to bond with one another and be willing to form a friendship.

Virtual worlds

Virtual worlds are the most interactive of all virtual community forms. In this type of virtual community, people are connected by living as an avatar in a computer-based world. Users create their own avatar character (from choosing the avatar's outfits to designing the avatar's house)and control their character's life and interactions with other characters in the 3-D virtual world. It is similar to a computer game, however there is no objective for the players. A virtual world simply gives users the opportunity to build and operate a fantasy life in the virtual realm. Characters within the world can talk to one

another and have almost the same interactions people would have in reality. For example, characters can socialize with one another and hold intimate relationships online.

This type of virtual community allows for people to not only hold conversations with others in real time, but also to engage and interact with others. The avatars that users create are like humans. Users can choose to make avatars like themselves, or take on an entirely different personality than them. When characters interact with other characters, they can get to know one another not only through text based talking, but also by virtual experience(such as having avatars go on a date in the virtual world). A chat room form of a virtual community may give real time conversations, but people can only talk to one another. In a virtual world, characters can do activities together, just like friends could do in reality. Communities in virtual worlds are most similar to real life communities because the characters are physically in the same place, even if the users who are operating the characters are not. It is close to reality, except that the characters are digital. Second Life is one of the most popular virtual worlds on the Internet.

Social Network Services

Social networking services are the most prominent type of virtual community. They are either a website or software platform that focuses on creating and maintaining relationships. Facebook, Twitter, and Myspace are all virtual communities. With these sites, one often creates a profile or account, and adds friends or follow friends. This allows people to connect and look for support using the social networking service as a gathering place. These websites often allow for people to keep up to date with their friends and acquaintances' activities without making much of an effort. On Facebook, for example, one can upload photos and videos, chat, make friends, reconnect with old ones, and join groups or causes. All of these functions encourage people to form a community, large or small, on the Internet.

Howard Rheingold's Study

Howard Rheingold's *Virtual Community* could be compared with Mark Granovetter's ground-breaking "strength of weak ties" article published twenty years earlier in the *American Journal of Sociology*. Rheingold translated, practiced and published Granovetter's conjectures about strong and weak ties in the online world. His comment on the first page even illustrates the social networks in the virtual society: "My seven year old daughter knows that her father congregates with a family of invisible friends who seem to gather in his computer. Sometimes he talks to them, even if nobody else can see them. And she knows that these invisible friends sometimes show up in the flesh, materializing from the next block or the other side of the world." (page 1). Indeed, in his revised version of *Virtual Community*, Rheingold goes so far to say that had he read Barry Wellman's work earlier, he would have called his book "online social networks".

Rheingold's definition contains the terms "social aggregation and personal relationships" (pp3). Lipnack & Stamps (1997) and Mowshowitz (1997) point out how virtual communities can work across space, time and organizational boundaries; Lipnack &

Stamps (1997) mention a common purpose; and Lee, Eom, Jung and Kim (2004) introduce "desocialization" which means that there is less frequent interaction with humans in traditional settings, eg. an increase in virtual socialization. Calhoun (1991) presents a dystopia argument, asserting the impersonality of virtual networks. He argues that IT has a negative influence on offline interaction between individuals because virtual life takes over our lives. He believes that it also creates different personalities in people which can cause frictions in offline and online communities and groups and in personal contacts. (Wellman & Haythornthwaite, 2002). Recently, Mitch Parsell (2008) has suggested that virtual communities, particularly those that leverage Web 2.0 resources, can be pernicious by leading to attitude polarization, increased prejudices and enabling sick individuals to deliberately indulging in their diseases.

Advantages to Internet Communities

Internet communities offer the advantage of instant information exchange that is not possible in a real-life community. This allows people to engage in many activities from their home, such as: shopping, paying bills, and searching for specific information. Users of online communities also have access to thousands of specific discussion groups where they can form specialized relationships and access information in such categories as: politics, technical assistance, social activities, and recreational pleasures. Virtual communities provide an ideal medium for these types of relationships because information can easily be posted and response times can be very fast. Another benefit is that these types of communities can give users a feeling of membership and belonging. Users can give and receive support, and it is simple and cheap to use.

Economically, virtual communities can be commercially successful, making money through membership fees, subscriptions, usage fees, and advertising commission. Consumers generally feel very comfortable making transactions online as long as the seller has a good reputation throughout the community. Virtual communities also provide the advantage of disintermediation in commercial transactions, which eliminates vendors and connects buyers directly to suppliers. This eliminates pricey mark-ups and allows for a more direct line of contact between the consumer and the manufacturer.

Disadvantages to Internet Communities

While instant communication means fast access, it also mean that information is posted without out being reviewed for correctness. It is difficult to choose reliable sources because there is no editor that reviews each post and makes sure it is up to a certain degree of quality. Everything comes from the writer with no filter in between.

Identities can be kept anonymous online so it is common for people to use the virtual community to live out a fantasy as another type of person. Users should be wary of where information is coming from online and be careful to double check facts with professionals.

Information online is different from information discussed in a real-life community because it is permanently online. As a result, users must be careful what information they disclose about themselves to ensure they are not easily identifiable, for safety reasons.

Chapter 7

Interaction Design and Technical Writer

Interaction design

Interaction design (abbreviated as **IxD**) defines the structure and behavior of interactive systems. Interaction Designers strive to create meaningful relationships between people and the products and services that they use, from computers to mobile devices to appliances and beyond. The practice typically centers on "embedding information technology into the ambient social complexities of the physical world."

Interactivity, however, is not limited to technological systems. It can also apply to other types of non-electronic products and services, and even organizations. Also, people have been interacting with each other as long as humans have been a species. Therefore, interaction design can be applied to the development of all solutions (or offerings), such as services and events. Those who design these offerings have, typically, performed interaction design inherently without naming it as such.

Related disciplines

Industrial Design

The core principles of Industrial Design overlap with those of interaction design, and vice versa.These include Physical form of an object, Aesthetics, Human perception & desire, and usability.

Human factors & Ergonomics

Certain basic principles of Ergonomics provide grounding for interaction design. These include anthropometry,biomechanics, kinesiology, physiology and psychology as they relate to human behavior in the built environment.

Cognitive psychology

Certain basic principles of cognitive psychology provide grounding for interaction design. These include mental models, mapping, interface metaphors, and affordances.

Many of these are laid out in Donald Norman's influential book *The Design of Everyday Things.*

Human computer interaction

Academic research in human-computer interaction (HCI) includes methods for describing and testing the usability of interacting with an interface, such as cognitive dimensions and the cognitive walkthrough.

Design research

Interaction designers are typically informed through iterative cycles of user research. User research is used to identify the needs, motivations and behaviour of end users. They design with an emphasis on user goals and experience, and evaluate designs in terms of usability and affective influence.

Architecture

As interaction designers increasingly deal with ubiquitous computing and urban computing, the architects' ability to make, place, and create context becomes a point of contact between the disciplines.

User interface design

Like User Interface design and Experience design, Interaction Design is often associated with the design of system interfaces in a variety of media but concentrates on the aspects of the interface that define and present its behavior over time, with a focus on developing the system to respond to the user's experience and not the other way around.

Methodologies

Interaction designers often follow similar processes to create a solution (not *the* solution) to a known interface design problem. Designers build rapid prototypes and test them with the users to validate or rebut the idea.

There are six major steps in interaction design. Based on user feedback, several iteration cycles of any set of steps may occur.

1. Design research

Using design research techniques (observations, interviews, questionnaires, and related activities), designers investigate users and their environment in order to learn more about them and thus be better able to design for them.

2. Research analysis and concept generation

Drawing on a combination of user research, technological possibilities, and business opportunities, designers create concepts for new software, products, services, or systems. This process may involve multiple rounds of brainstorming, discussion, and refinement.

To help designers realize user requirements, they may use tools such as personas or user profiles that are reflective of their targeted user group. From these personae, and the patterns of behavior observed in the research, designers create scenarios (or user stories) or storyboards, which imagine a future work flow the users will go through using the product or service.

After thorough analysis using various tools and models, designers create a high level summary spanning across all levels of user requirements. This includes a vision statement regarding the current and future goals of a project.

3. Alternative design and evaluation

Once a clear view of the problem domain exists, designers develop alternative solutions with crude prototypes to help convey concepts and ideas. Proposed solutions are evaluated and, perhaps, merged. The end result should be a design that solves as many of the user requirements as possible.

Among the tools that may be used for this process are wireframing and flow diagrams. The features and functionality of a product or service are often outlined in a document known as a wireframe ("schematics" is an alternate term). Wireframes are a page-by-page or screen-by-screen detail of the system, which include notes ("annotations") describing how the system will operate. Flow Diagrams outline the logic and steps of the system or an individual feature.

The cognitive dimensions framework provides a specialized vocabulary to evaluate particular design solutions, and aid in the creation of new designs from existing ones through *design manoeuvres*.

4. Prototyping and usability testing

Interaction designers use a variety of prototyping techniques to test aspects of design ideas. These can be roughly divided into three classes: those that test the **role** of an artifact, those that test its **look and feel** and those that test its **implementation**. Sometimes, these are called **experience prototypes** to emphasize their interactive nature. Prototypes can be physical or digital, high- or low-fidelity.

5. Implementation

Interaction designers need to be involved during the development of the product or service to ensure that what was designed is implemented correctly. Often, changes need

to be made during the building process, and interaction designers should be involved with any of the on-the-fly modifications to the design.

6. System testing

Once the system is built, often another round of testing, for both usability and errors ("bug catching") is performed. Ideally, the designer will be involved here as well, to make any modifications to the system that are required.

Aspects of interaction design

Social interaction design

Social interaction design (SxD) is emerging because many of our computing devices have become networked and have begun to integrate communication capabilities. Phones, digital assistants and the myriad connected devices from computers to games facilitate talk and social interaction. Social interaction design accounts for interactions among users as well as between users and their devices. The dynamics of interpersonal communication, speech and writing, the pragmatics of talk and interaction—these now become critical factors in the use of social technologies. And they are factors described less by an approach steeped in the rational choice approach taken by cognitive science than that by sociology, psychology, and anthropology.

Affective interaction design

Throughout the process of interaction design, designers must be aware of key aspects in their designs that influence emotional responses in target users. The need for products to convey positive emotions and avoid negative ones is critical to product success. These aspects include positive, negative, motivational, learning, creative, social and persuasive influences to name a few. One method that can help convey such aspects is the use of expressive interfaces. In software, for example, the use of dynamic icons, animations and sound can help communicate a state of operation, creating a sense of interactivity and feedback. Interface aspects such as fonts, color pallet, and graphical layouts can also influence an interface's perceived effectiveness. Studies have shown that affective aspects can affect a user's perception of usability.

Emotional and pleasure theories exist to explain peoples responses to the use of interactive products. These include Don Norman's emotional design model, Patrick Jordan's pleasure model, and McCarthy and Wright's Technology as Experience framework.

Interaction design domains

Interaction designers work in many areas, including software interfaces, (business) information systems, internet, physical products, environments, services, and systems

which may combine many of these. Each area requires its own skills and approaches, but there are aspects of interaction design common to all.

Interaction designers often work in interdisciplinary teams as their work requires expertise in many different domains, including graphic design, programming, psychology, user testing, product design, etc. Thus, they need to understand enough of these fields to work effectively with specialists.

History

The term *interaction design* was first proposed by Bill Moggridge and Bill Verplank in the late 1980s. To Verplank, it was an adaptation of the computer science term *user interface design* to the industrial design profession. To Moggridge, it was an improvement over *soft-face*, which he had coined in 1984 to refer to the application of industrial design to products containing software (Moggridge 2006).

In 1990, Gillian Crampton-Smith established an interaction design MA at the Royal College of Art (RCA) in London (originally entitled "computer-related design" and now known as Design Interactions). In 2001, she helped found the Interaction Design Institute Ivrea, a small institute in Northern Italy dedicated solely to interaction design; the institute moved to Milan in October 2005 and merged courses with Domus Academy. In 2007, some of the people originally involved with IDII have now set up the Copenhagen Institute of Interaction Design (CIID).

Today, interaction design is taught in many schools worldwide.

Relationship with instructional design

Instructional design is a goal-oriented, user-centric approach to creating training and education software or written materials. Interaction design and instructional design both rely on cognitive psychology theories to focus on how users will interact with software. They both take an in-depth approach to analyzing the user's needs and goals. A needs analysis is often performed in both disciplines. Both approach the design from the user's perspective. Both involve gathering feedback from users and making revisions until the product or service has been found to be effective. (Summative / formative evaluations) In many ways, instructional design can be considered a precursor to interaction design.

Technical writer

A **technical writer** (also called a **technical communicator**) is a professional writer who designs, creates, and maintains technical documentation. This documentation includes

online help, user guides, white papers, design specifications, system manuals, and other documents.

Engineers, scientists, and other professionals may also produce technical writing, usually handing their work to a professional technical writer for proofreading, editing, and formatting. A technical writer produces technical documentation for technical, business, and consumer audiences.

Skill set

In addition to solid research, language, and writing skills, a technical writer may have skills in:

- Information design
- Information architecture
- Training material development
- Illustration
- Typography

Technical writing may be on any subject that requires explanation to a particular audience. A technical writer is not usually a subject matter expert (SME), but possesses and applies expertise to interview SMEs and conduct research necessary to produce accurate, comprehensive documents. Companies, governments, and other institutions typically hire technical writers not for expertise in a particular subject, but for expertise in technical writing, i.e., their ability to gather information, analyze subject and audience and produce clear documentation.

A good technical writer creates documentation that is accurate, complete, unambiguous, and as concise as possible. Technical writers create documentation in many forms: printed, web-based or other electronic documentation, training materials, and industrial film scripts.

Qualifications

Technical writers work under many job titles, including *Technical Communicator, Information Developer*, *Data Development Engineer*, and *Technical Documentation Specialist*. In the United Kingdom and some other countries, a technical writer is often called a *technical author* or *knowledge author*.

Technical writers normally possess a mixture of technical and language abilities. They may have a degree or certificate in technical communications. Many technical writers switch from another technical field such as engineering or science, often after learning the required skills through technical communications classes.

A good technical writer has the ability to create, assimilate, and convey technical material in a concise and effective manner. A technical writer may specialize in a particular area.

For example, API writers mostly work on API documents, while other technical writers specialize in electronic commerce, manufacturing, scientific, or medical material.

Methodology

To create a technical document, a technical writer gathers information by studying existing material and interviewing SMEs. The technical writer also studies the audience to learn their needs and technical level.

Well-formed technical documents follow common publishing guidelines. Technical documentation comes in many styles and formats, depending on the medium. Printed and online documentation differ in necessary ways but adhere to largely identical guidelines for prose, information structure, and layout. Usually, technical writers follow formatting conventions described in a standard style guide. In the US, technical writers typically use the Chicago Manual of Style (CMS). Many companies have internal corporate style guides that cover specific corporate issues such as logo use, branding, and other aspects of corporate style. The Microsoft Manual of Style for Technical Publications is typical of these.

Engineering projects, particularly defense or aerospace related projects, often follow national and international documentation standards—such as ATA100 for civil aircraft or S1000D for defense platforms.

Environment

Technical writers often work as part of a writing or project development team. Typically, the writer finishes a draft and passes it to one or more SMEs who conduct a *technical edit* to review accuracy and completeness. In some cases the writer or others test the document on audience members. On a project team, a technical writer develops the overall project documentation as other members of the team develop other areas of the project. For example, as engineers design and integrate a system, the technical writer generates manuals that go with the system.

Career growth

A technical writer has no standard career path, but technical writers may move into project management over other writers. A writer may advance to a senior technical writer position, handling complex projects or a small team of writers and editors. In larger groups, a documentation manager might handle multiple projects and teams.

Technical writers may also gain expertise in a particular technical domain and branch into related forms, such as software quality analysis or business analysis. A technical writer who becomes a subject matter expert in a field may transition from technical writing to work in that field.

Senior writers in some software documentation departments are increasingly called *Individual Contributor* (IC). In API/software documentation, ICs typically work with a team of developers or testers across many physical locations. In such software development in "software research organizations," an IC plays an important role in the delivery of API/Software documentation.

Chapter 8

User Interface and Information Architecture

User interface

In the industrial design field of human-machine interaction, the user interface is (a place) where interaction between humans and machines occurs. The goal of interaction between a human and a machine at the user interface is effective operation and control of the machine, and feedback from the machine which aids the operator in making operational decisions. Examples of this broad concept of user interfaces include the interactive aspects of computer operating systems, hand tools, heavy machinery operator controls. and process controls. The design considerations applicable when creating user interfaces are related to or involve such disciplines as ergonomics and psychology.

A **user interface** is the system by which people (users) interact with a machine. The user interface includes hardware (physical) and software (logical) components. User interfaces exist for various systems, and provide a means of:

- Input, allowing the users to manipulate a system, and/or
- Output, allowing the system to indicate the effects of the users' manipulation.

Generally, the goal of human-machine interaction engineering is to produce a user interface which makes it easy, efficient, and enjoyable to operate a machine in the way which produces the desired result. This generally means that the operator needs to provide minimal input to achieve the desired output, and also that the machine minimizes undesired outputs to the human.

Ever since the increased use of personal computers and the relative decline in societal awareness of heavy machinery, the term user interface has taken on overtones of the (graphical) user interface, while industrial control panel and machinery control design discussions more commonly refer to human-machine interfaces.

Other terms for **user interface** include ***human-computer interface*** (HCI) and ***man-machine interface*** (MMI).

Introduction

To work with a system, users have to be able to control and assess the state of the system. For example, when driving an automobile, the driver uses the steering wheel to control the direction of the vehicle, and the accelerator pedal, brake pedal and gearstick to control the speed of the vehicle. The driver perceives the position of the vehicle by looking through the windshield and exact speed of the vehicle by reading the speedometer. The *user interface of the automobile* is on the whole composed of the instruments the driver can use to accomplish the tasks of driving and maintaining the automobile.

Terminology

There is a distinct difference between **User Interface** versus **Operator Interface** or **Human Machine Interface (HMI)**.

- The term ***user interface*** is often used in the context of (personal) **computer systems** and electronic devices
    - Where a network of equipment or computers are interlinked through an MES (Manufacturing Execution System)-or Host.
    - An HMI is typically local to one machine or piece of equipment, and is the interface method between the human and the equipment/machine. An Operator interface is the interface method by which multiple equipment that are linked by a host control system is accessed or controlled.
    - The system may expose several user interfaces to serve different kinds of users. For example, a computerized library database might provide two user interfaces, one for library patrons (limited set of functions, optimized for ease of use) and the other for library personnel (wide set of functions, optimized for efficiency).
- The user interface of a **mechanical system, a vehicle or an industrial installation** is sometimes referred to as the **human-machine interface** (**HMI**). HMI is a modification of the original term MMI (man-machine interface). In practice, the abbreviation MMI is still frequently used although some may claim that MMI stands for something different now. Another abbreviation is HCI, but is more commonly used for human-computer *interaction*. Other terms used are operator interface console (OIC) and operator interface terminal (OIT). However it is abbreviated, the terms refer to the 'layer' that separates a human that is operating a machine from the machine itself.

In science fiction, HMI is sometimes used to refer to what is better described as **direct neural interface**. However, this latter usage is seeing increasing application in the real-life use of (medical) prostheses—the artificial extension that replaces a missing body part (e.g., cochlear implants).

In some circumstance computers might observe the user, and react according to their actions without specific commands. A means of tracking parts of the body is required,

and sensors noting the position of the head, direction of gaze and so on have been used experimentally. This is particularly relevant to **immersive interfaces**.

Usability

User interfaces are considered by some authors to be a prime ingredient of Computer user satisfaction.

The design of a user interface affects the amount of effort the user must expend to provide input for the system and to interpret the output of the system, and how much effort it takes to learn how to do this. Usability is the degree to which the design of a particular user interface takes into account the human psychology and physiology of the users, and makes the process of using the system effective, efficient and satisfying.

Usability is mainly a characteristic of the user interface, but is also associated with the functionalities of the product and the process to design it. It describes how well a product can be used for its intended purpose by its target users with efficiency, effectiveness, and satisfaction, also taking into account the requirements from its context of use.

User interfaces in computing

In computer science and human-computer interaction, the *user interface (of a computer program)* refers to the graphical, textual and auditory information the program presents to the user, and the control sequences (such as keystrokes with the computer keyboard, movements of the computer mouse, and selections with the touchscreen) the user employs to control the program.

Types

Currently (as of 2009) the following types of user interface are the most common:

- **Graphical user interfaces** (GUI) accept input via devices such as computer keyboard and mouse and provide articulated graphical output on the computer monitor. There are at least two different principles widely used in GUI design: Object-oriented user interfaces (OOUIs) and application oriented interfaces.
- **Web-based user interfaces** or **web user interfaces** (WUI) are a subclass of GUIs that accept input and provide output by generating web pages which are transmitted via the Internet and viewed by the user using a web browser program. Newer implementations utilize Java, AJAX, Adobe Flex, Microsoft .NET, or similar technologies to provide real-time control in a separate program, eliminating the need to refresh a traditional HTML based web browser. Administrative web interfaces for web-servers, servers and networked computers are often called Control panels.
- **Touchscreens** are displays that accept input by touch of fingers or a stylus. Used in a growing amount of mobile devices and many types of point of sale, industrial processes and machines, self-service machines etc.

User interfaces that are common in various fields outside desktop computing:

- **Command line interfaces**, where the user provides the input by typing a command string with the computer keyboard and the system provides output by printing text on the computer monitor. Used by programmers and system administrators, in engineering and scientific environments, and by technically advanced personal computer users.

- **Touch user interface** are graphical user interfaces using a touchpad or touchscreen display as a combined input and output device. They supplement or replace other forms of output with haptic feedback methods. Used in computerized simulators etc.

Other types of user interfaces:

- **Attentive user interfaces** manage the user attention deciding when to interrupt the user, the kind of warnings, and the level of detail of the messages presented to the user.
- **Batch interfaces** are non-interactive user interfaces, where the user specifies all the details of the *batch job* in advance to batch processing, and receives the output when all the processing is done. The computer does not prompt for further input after the processing has started.
- **Conversational Interface Agents** attempt to personify the computer interface in the form of an animated person, robot, or other character (such as Microsoft's Clippy the paperclip), and present interactions in a conversational form.
- **Crossing-based interfaces** are graphical user interfaces in which the primary task consists in crossing boundaries instead of pointing.
- **Gesture interface** are graphical user interfaces which accept input in a form of hand gestures, or mouse gestures sketched with a computer mouse or a stylus.
- **Intelligent user interfaces** are human-machine interfaces that aim to improve the efficiency, effectiveness, and naturalness of human-machine interaction by representing, reasoning, and acting on models of the user, domain, task, discourse, and media (e.g., graphics, natural language, gesture).
- **Motion tracking interfaces** monitor the user's body motions and translate them into commands, currently being developed by Apple
- **Multi-screen interfaces**, employ multiple displays to provide a more flexible interaction. This is often employed in computer game interaction in both the commercial arcades and more recently the handheld markets.
- **Noncommand user interfaces**, which observe the user to infer his / her needs and intentions, without requiring that he / she formulate explicit commands.
- **Object-oriented user interface (OOUI)**
- **Reflexive user interfaces** where the users control and redefine the entire system via the user interface alone, for instance to change its command verbs. Typically this is only possible with very rich graphic user interfaces.
- **Tangible user interfaces**, which place a greater emphasis on touch and physical environment or its element.

- **Task-Focused Interfaces** are user interfaces which address the information overload problem of the desktop metaphor by making tasks, not files, the primary unit of interaction
- **Text user interfaces** are user interfaces which output text, but accept other form of input in addition to or in place of typed command strings.
- **Voice user interfaces**, which accept input and provide output by generating voice prompts. The user input is made by pressing keys or buttons, or responding verbally to the interface.
- **Natural-Language interfaces -** Used for search engines and on webpages. User types in a question and waits for a response.
- **Zero-Input interfaces** get inputs from a set of sensors instead of querying the user with input dialogs.
- **Zooming user interfaces** are graphical user interfaces in which information objects are represented at different levels of scale and detail, and where the user can change the scale of the viewed area in order to show more detail.

History

The history of user interfaces can be divided into the following phases according to the dominant type of user interface:

- Batch interface, 1945–1968
- Command-line user interface, 1969 to present
- Graphical user interface, 1981 to present

Consistency

A key property of a good user interface is consistency. Good user interface design is about getting a user to have a consistent set of expectations, and then meeting those expectations.

There are three important aspects to consistency.

First, the controls for different features should be presented in a consistent manner so that users can find the controls easily. For example, users find it difficult to use software when some commands are available through menus, some through icons, some through right-clicks, some under a separate button at one corner of a screen, some grouped by function, some grouped by “common,” some grouped by “advanced.” A user looking for a command should have a consistent search strategy for finding it. The more search strategies a user has to use, the more frustrating the search will be. The more consistent the grouping, the easier the search.

Second, the "principle of least astonishment" is crucial. Various features should work in similar ways. For example, some features in Adobe Acrobat are "select tool, then select text to which apply." Others are "select text, then apply action to selection." . Commands should work the same way in all contexts.

Third, consistency counsels against user interface changes version-to-version. Change should be minimized, and forward-compatibility should be maintained. For example, the change from the menu bars of Microsoft Office 2003 to the ribbon toolbar of Microsoft Office 2007 caused mixed reactions to the redesign, intended to enhance access to the most used functions. It was said to cause "anger and frustration," and "major efforts in time, training and costs." Power users said it "takes too much time and patience to learn" the new interface. An online survey by an Excel user group reports that about 80% of respondents had a negative opinion of the change, and within that 80%, the self-estimated reduction in productivity was "about 35%".

Consistency is one quality to trade off in user interface design—not the only thing, but one of the most important. In some cases, a violation of consistency principles can provide sufficiently clear advantages that a wise and careful user interface designer may choose to violate consistency to achieve some other important goal. Generally, less mature software has fewer users who are entrenched in the status quo. Older, more broadly used software must more carefully hew to the status quo to avoid disruptive costs. The most experienced users, and the ones who derive most value from a program, are the users who tend to bear the greatest costs from change. However, of those trade-offs, consistency is one of the most important core principles, and it should be violated least often. Bad user interface design, and poorly implemented changes to an existing user interface, can impose staggering costs on users.

Modalities and modes

A **modality** is a path of communication employed by the user interface to carry input and output. Examples of modalities:

- Input — computer keyboard allows the user to enter typed text, digitizing tablet allows the user to create free-form drawing
- Output — computer monitor allows the system to display text and graphics (*vision modality*), loudspeaker allows the system to produce sound (*auditory modality*)

The user interface may employ several redundant input modalities and output modalities, allowing the user to choose which ones to use for interaction.

A **mode** is a distinct method of operation within a computer program, in which the same input can produce different perceived results depending of the state of the computer program. For example, caps lock sets an input mode in which typed letters are uppercase by default; the same typing produces lowercase letters when not in caps lock mode. Heavy use of modes often reduces the usability of a user interface, as the user must expend effort to remember current mode states, and switch between mode states as necessary.

Information architecture

Information architecture (**IA**) is the art of expressing a model or concept of information used in activities that require explicit details of complex systems. Among these activities are library systems, Content Management Systems, web development, user interactions, database development, programming, technical writing, enterprise architecture, and critical system software design. Information architecture has somewhat different meanings in these different branches of IS or IT architecture. Most definitions have common qualities: a structural design of shared environments, methods of organizing and labeling websites, intranets, and online communities, and ways of bringing the principles of design and architecture to the digital landscape.

Historically the term "information architect" is attributed to Richard Saul Wurman. Wurman sees architecture as "used in the words architect of foreign policy. I mean architect as in the creating of systemic, structural, and orderly principles to make something work--the thoughtful making of either artifact, or idea, or policy that informs because it is clear."

Definitions

Information architecture is a specialized skill set that interprets information and expresses distinctions between signs and systems of signs. It originates, to some degree, in the library sciences. Many schools with library and information science departments teach information architecture.

Information architecture is the categorization of information into a coherent structure, preferably one that most people can understand quickly, if not inherently. It's usually hierarchical, but can have other structures, such as concentric or even chaotic. It has nothing to do with philosophy or semiotics.

In the context of information systems design, information architecture refers to the analysis and design of the data stored by information systems, concentrating on entities, their attributes, and their interrelationships. It refers to the modeling of data for an individual database and to the corporate data models an enterprise uses to coordinate the definition of data in several (perhaps scores or hundreds) of distinct databases. The "canonical data model" is applied to integration technologies as a definition for specific data passed between the systems of an enterprise. At a higher level of abstraction it may also refer to the definition of data stores.

I.A.I. definition

Information architecture is defined by the Information Architecture Institute as:

1. The structural design of shared information environments.

2. The art and science of organizing and labeling web sites, intranets, online communities, and software to support findability and usability.
3. An emerging community of practice focused on bringing principles of design and architecture to the digital landscape.

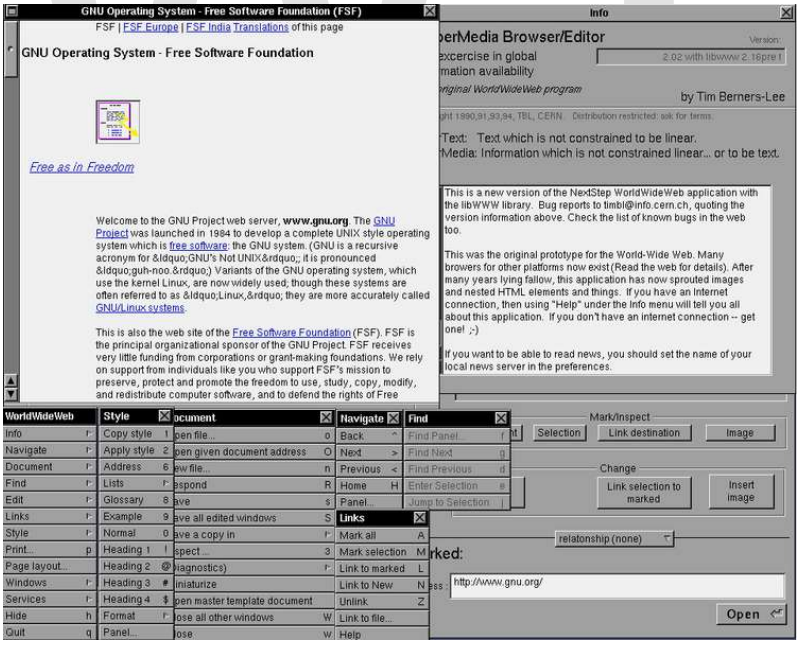
Chapter 9

Web Browser

A **web browser** or **Internet browser** is a software application for retrieving, presenting, and traversing information resources on the World Wide Web. An *information resource* is identified by a Uniform Resource Identifier (URI) and may be a web page, image, video, or other piece of content. Hyperlinks present in resources enable users to easily navigate their browsers to related resources.

Although browsers are primarily intended to access the World Wide Web, they can also be used to access information provided by Web servers in private networks or files in file systems. Some browsers can also be used to save information resources to file systems.

History



WorldWideWeb for NeXT, released in 1991, was the first Web browser.

The history of the Web browser dates back in to the late 1980s, when a variety of technologies laid the foundation for the first Web browser, WorldWideWeb, by Tim

Berners-Lee in 1991. That browser brought together a variety of existing and new software and hardware technologies.

Ted Nelson and Douglas Engelbart developed the concept of hypertext long before Berners-Lee and CERN. It became the core of the World Wide Web. Berners-Lee does acknowledge Engelbart's contribution.

The introduction of the NCSA Mosaic Web browser in 1993 – one of the first graphical Web browsers – led to an explosion in Web use. Marc Andreessen, the leader of the Mosaic team at NCSA, soon started his own company, named Netscape, and released the Mosaic-influenced Netscape Navigator in 1994, which quickly became the world's most popular browser, accounting for 90% of all Web use at its peak.

Microsoft responded with its browser Internet Explorer in 1995 (also heavily influenced by Mosaic), initiating the industry's first browser war. By bundling Internet Explorer with Windows, Microsoft was able to leverage its dominance in the operating system market to take over the Web browser market; Internet Explorer usage share peaked at over 95% by 2002. Internet Explorer has 60% browser usage share as of September 2010 according to Net Applications, and it continues to show a negative trend.

Opera first appeared in 1996; although it has never achieved widespread use, with a browser usage share that is stable around 2.4% as of September 2010, it has a substantial share of the fast-growing mobile phone Web browser market, being preinstalled on over 40 million phones. It is also available on several other embedded systems, including Nintendo's Wii video game console.

In 1998, Netscape launched what was to become the Mozilla Foundation in an attempt to produce a competitive browser using the open source software model. That browser would eventually evolve into Firefox, which developed a respectable following while still in the beta stage of development; shortly after the release of Firefox 1.0 in late 2004, Firefox (all versions) accounted for 7.4% of browser use. As of September 2010, Firefox has a 23% usage share.

Apple's Safari had its first beta release in January 2003; it has a dominant share of Apple-based Web browsing, accounting for 5.3% of the entire browser market as of September 2010 and is slowly gaining. Its rendering engine, called WebKit, is also running in the standard browsers of several mobile phone platforms, including Apple iOS, Google Android, Nokia S60 and Palm webOS.

The most recent major entrant to the browser market is Google's WebKit-based Chrome, first released in September 2008. Its market share has quickly risen; as of September 2010, it has an 8% usage share and appears to be gaining further in the coming months.

Function

The primary purpose of a web browser is to bring information resources to the user. This process begins when the user inputs a Uniform Resource Identifier (URI. The prefix of the URI determines how the URI will be interpreted. The most commonly used kind of URI starts with *http:* and identifies a resource to be retrieved over the Hypertext Transfer Protocol (HTTP). Many browsers also support a variety of other prefixes, such as *https:* for HTTPS, *ftp:* for the File Transfer Protocol, and *file:* for local files. Prefixes that the web browser cannot directly handle are often handed off to another application entirely. For example, *mailto:* URIs are usually passed to the user's default e-mail application and *news:* URIs are passed to the user's default newsgroup reader.

In the case of *http*, *https*, *file*, and others, once the resource has been retrieved the web browser will display it. HTML is passed to the browser's layout engine to be transformed from markup to an interactive document. Aside from HTML, web browsers can generally display any kind of content that can be part of a web page. Most browsers can display images, audio, video, and XML files, and often have plug-ins to support Flash applications and Java applets. Upon encountering a file of an unsupported type or a file that is set up to be downloaded rather than displayed, the browser prompts the user to save the file to disk.

Interactivity in a web page can also be supplied by JavaScript, which usually does not require a plugin. JavaScript can be used along with other technologies to allow "live" interaction with the web page's server via AJAX.

Information resources may contain hyperlinks to other information resources. Each link contains the URI of a resource to go to. When a link is clicked, the browser navigates to the resource indicated by the link's target URI, and the process of bringing content to the user begins again.

Features

Available web browsers range in features from minimal, text-based user interfaces with bare-bones support for HTML to rich user interfaces supporting a wide variety of file formats and protocols. Browsers which include additional components to support e-mail, Usenet news, and Internet Relay Chat (IRC), are sometimes referred to as "Internet suites" rather than merely "web browsers".

All major web browsers allow the user to open multiple information resources at the same time, either in different browser windows or in different tabs of the same window. Major browsers also include pop-up blockers to prevent unwanted windows from "popping up" without the user's consent.

Most web browsers can display a list of web pages that the user has *bookmarked* so that the user can quickly return to them. Bookmarks are also called "Favorites" in Internet Explorer. In addition, all major web browsers have some form of built-in web feed

aggregator. In Mozilla Firefox, web feeds are formatted as "live bookmarks" and behave like a folder of bookmarks corresponding to recent entries in the feed. In Opera, a more traditional feed reader is included which stores and displays the contents of the feed.

Furthermore, most browsers can be extended via plug-ins, downloadable components that provide additional features.

User interface

Most major web browsers have these user interface elements in common:

- *Back* and *forward* buttons to go back to the previous resource and forward again.
- A history list, showing resources previously visited in a list (typically, the list is not visible all the time and has to be summoned)
- A *refresh* or *reload* button to reload the current resource.
- A *stop* button to cancel loading the resource. In some browsers, the stop button is merged with the reload button.
- A *home* button to return to the user's home page
- An address bar to input the Uniform Resource Identifier (URI) of the desired resource and display it.
- A search bar to input terms into a search engine
- A status bar to display progress in loading the resource and also the URI of links when the cursor hovers over them, and page zooming capability.

Major browsers also possess incremental find features to search within a web page.

Privacy and security

Most browsers support HTTP Secure and offer quick and easy ways to delete the web cache, cookies, and browsing history.

Standards support

Early web browsers supported only a very simple version of HTML. The rapid development of web browsers led to the development of non-standard dialects of HTML, leading to problems with interoperability. Modern web browsers support a combination of standards-based and *de facto* HTML and XHTML, which should be rendered in the same way by all browsers.

Chapter 10

Technical Writing and Online Help

Technical writing

Technical writing, a form of technical communication, is a style of writing used in fields as diverse as computer hardware and software, chemistry, the aerospace industry, robotics, finance, consumer electronics, and biotechnology. Technical writers explain technology and related ideas to technical and nontechnical audiences. This could mean, for example, telling a programmer how to use a software library or telling a consumer how to operate a television remote control.

Technical writers gather information from existing documentation and from subject matter experts. A subject matter expert (SME) is any expert on the topic that the writer is working on. Technical writers are often not SMEs themselves (unless they are writing about creating good technical documentation). Workers at many levels, and in many different fields, have a role in producing technical communications. A good technical writer needs strong language and teaching skills and must understand the many conventions of modern technical communications.

Technical writing teams or departments are often referred to as *Information Development*, *User Assistance*, *Technical Documentation*, or *Technical Publications*. Technical writers themselves may be called *API Writers*, *information developers*, *documentation specialists*, *documentation engineers*, or *technical content developers*. Advanced technical writers often move into specialized areas such as API writing, information architecture or document management.

Example

For technical documents to be useful, readers must understand and act on them without having to decode wordy and ambiguous prose. Good technical writing clarifies technical jargon; that is, it presents useful information that is clear and easy to understand for the intended audience. Poor technical writing may increase confusion by creating unnecessary technical jargon, or failing to explain unavoidable technical terms that reader would not be expected to be familiar with.

Consider a technical writer writing a cake recipe:

- **Audience:** Is the audience composed of people in home kitchens or highly trained chefs in professional kitchens?
- **Source:** Is there existing documentation—a rough draft? Who is the subject matter expert (SME)?
- **Deliverable:** Is the deliverable simple text for inclusion in a book, or formatted to final form? Is the target a paper, a Web page, or something else?

The technical writer determines that the recipe is written down on the back of a napkin but is partially indecipherable, so he must also interview a SME—the chef who created it. He is told that the audience consists of people in their own kitchens, so the writer must adjust the style accordingly and replace or explain words in the source material like "beurre mixer" or "springform pan." The chef reviews a draft of the recipe (a *technical edit*) and marks in needed technical corrections (*bake at 350 degrees*, not *325 degrees*). The writer prepares a final draft and the document goes into English edit to ensure that all instructions are grammatically correct. The document owner and any other stakeholders perform a final review and approve the recipe before it is sent to the printer.It is also an act of giving instructions.

Communicating with the audience

Technical writing is a communication to convey a particular piece of information to a particular audience for a particular purpose. It is often exposition about scientific subjects and technical subjects associated with sciences.

Technical writing translates complex technical concepts and instructions into simpler language in order to enable users to perform a specific task in a specific way. To present appropriate information, writers must understand the audience and their goals. Audience analysis is a key feature of all technical writing.

History

The origins of technical writing have been variously attributed to Ancient civilizations such as Indian, Greece, the Renaissance, and the mid-19th century. However, a clear trend towards the discipline can be seen from the First World War on, growing out of the need for technology-based documentation in the military, manufacturing, electronics, and aerospace industries. In 1953, two organizations concerned with improving the practice of technical communication were founded on the East Coast the Society of Technical Writers, and the Association of Technical Writers and Editors. These organizations merged in 1957 to form the Society of Technical Writers and Editors, a predecessor of the current Society for Technical Communication (STC).

Deliverables

Technical writing is often associated with online help and user manuals. However, technical writers create many other forms of technical content. These include product release notes, product troubleshooting guides, product user guides, tutorials, software installation guides, API programmers' guides, legal disclaimers, policies and procedures, business proposals, and white papers.

Associations

- IEEE Professional Communication Society
- International Council for Technical Communication
- Society for Technical Communication
- Institute of Scientific and Technical Communicators (UK-based)
- Association for Business Communication
- Technical Communicator's Association of New Zealand (NZ-based)
- Usability Professional's Association of New Zealand (NZ-based)
- Czech Society for Technical Communication (Cz-based)
- Tekom Professional organization for technical communication, Germany
- Elephant Professional organization for Technical Writers, Israel
- Association of Teachers of Technical Writing

Online help

Online help is topic-oriented, procedural or reference information delivered through computer software. It is a form of user assistance. Most online help is designed to give assistance in the use of a software application or operating system, but can also be used to present information on a broad range of subjects. When online help is linked to the state of the application (what the user is doing), it is called context-sensitive help.

Microsoft products such as Microsoft Visual Studio now refer to locally installed help as **Offline help** or **Local help**, while help installed on their web server is referred to as **Online help**. However the general term **Online help** has always referred to the electronic documentation associated with an application (local or web based).

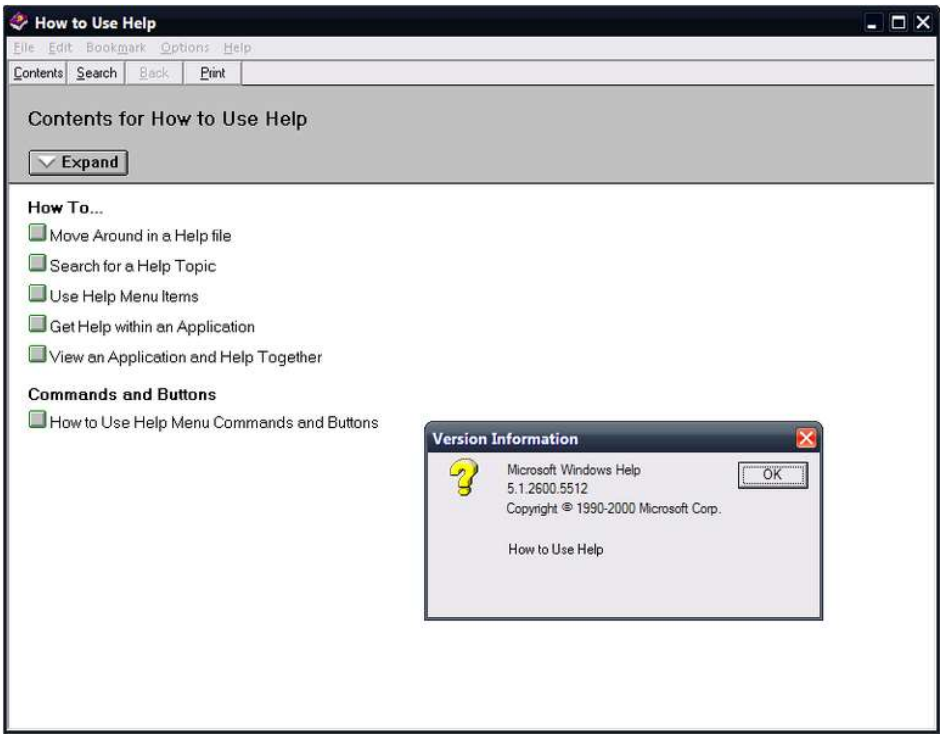
Online help is created using help authoring tools. It is delivered in a wide variety of formats, some proprietary and some open-standard, including:

- Hypertext Markup Language (HTML), which includes HTML Help, HTML-based Help, JavaHelp, and Oracle Help
- Adobe Portable Document Format (PDF)

Microsoft help platforms

1. WinHelp

Microsoft WinHelp



Help File

Filename extension	`.hlp`
Developed by	Microsoft
Initial release	1990
Container for	RTF
Standard(s)	No

Microsoft WinHelp is a proprietary format for online help files that can be displayed by the Microsoft Help browser *winhelp.exe* or *winhlp32.exe*. The file format is based on Rich Text Format (RTF). It remained a popular Help platform from Windows 3.0 platform through Windows XP. WinHelp was removed in Windows Vista to discourage software developers from using the obsolete format and encourage use of newer help formats.

History

- 1990 - WinHelp 1.0 shipped with Windows 3.0.
- 1995 - WinHelp 4.0 shipped with Windows 95 / Windows NT.
- 2006 - Microsoft announced its intentions to phase out WinHelp as a supported platform. WinHelp will not be a part of Windows Vista out-of-the-box. WinHelp files come in 16 bit and 32 bit types. Vista treats these two files types differently. When starting an application which uses the 32 bit .hlp format, Windows displays a warning saying the format is not supported any more. A downloadable viewer for viewing 32 bit .hlp files is available from the Microsoft Download Center. The 16 bit WinHelp files continue to display in Windows Vista without the viewer download.
- October 14, 2009 - Microsoft announced the downloadability of Windows Help program (WinHlp32.exe) for Windows 7 at the Microsoft Download Center.

File format

A WinHelp file has a ".hlp" suffix. It can be accompanied by an optional table of contents (.cnt) file if the help developer created one. When Windows opens a WinHelp file, it creates a .gid file in the same directory, containing information about the .hlp file such as the window size and location. If the user clicks the "Find" tab and enables keyword indexing, Windows creates an index file with a .fts (full text search) extension.

A WinHelp file can also be decompiled, providing copies of its source documents (HPJ, CNT, RTF, BMP, SHG), using a number of software tools. An HPJ file is the project file that is created and edited in the Help Workshop (or a third party help authoring tool). The HPJ contains information about what RTF files to compile into the help, the MAP IDs and Aliases that provide links from a calling application to the help file, and help file appearance (window size, default buttons, color schemes, etc.). The CNT file provides the Table of Contents for the help file. An SHG file is a "SHED" graphics file that essentially creates an image map of help calls for a graphic file (e.g., a BMP).

There are number of tools which can read and explore these files.

Source files and compilation

The source files required to compile a .hlp file consist of one or more documents with the extension .rtf and a help project file with the extension .hpj, along with any image files (.bmp, .wmf, or .shg) that are used within the Help file. An optional Table of Contents file with the extension .cnt can also be created for use with the .hlp file.

Within the .rtf files, topics are separated by page breaks. Each topic will have a series of footnotes which contain information for the help compiler: # footnotes contain the topic ID (used to create links to that topic); $ footnotes contain the topic name as it will be displayed in the table of contents, index, and other locations; K footnotes contain keywords for the index; A footnotes contain; * footnotes contain build tags; + footnotes

contain browse sequence information; and ! footnotes contain topic entry macros. Only the # footnote is required; all others are optional.

The text within each topic can contain limited types of formatting, including bold text, italics, and colors, etc. (Superscript and subscript are not allowed.) Jumps between topics in the same Help file usually appear in the source document as double-underlined text (green by default, although this can be overridden) followed by a topic ID in hidden text. Popup links appear in the source document as text with a single underline (also green by default) followed by a topic ID in hidden text. (In the .hlp file, the jumps will show up as green text with a single underline, and popups will show up as green text with a dotted underline.)

Images can be added using codes such as {bmc image.bmp}. Supported image formats include .bmp, .wmf, and .shg (used for image maps, which can contain jumps or popups that are triggered by clicking on specific parts of the image).

After the source files have been created, the help file can be compiled using a WinHelp compiler such as HCW.exe or by using a commercial software program such as RoboHelp or HelpBreeze, most of which (included the two cited here) also use hcw.exe as the backend compiler.

WinHelp appearance and features

Depending on how it has been launched and what settings the Help author has chosen, a WinHelp file will open either to its default topic, its Table of Contents, or its Index.

A topic in a WinHelp file opens in a separate window, the size and initial position of which can be chosen by the Help author. Users can resize or reposition the window if they choose. The Help author can control whether the Help file will remember the user's settings between sessions or whether it will always open with its default size and position.

When a topic is open, a title bar at the top of the Help window displays the title that the Help author has given to the file. Below that is a row of menus (**File**, **Edit**, **Bookmark**, **Options**, and **Help**) which can be used to control various aspects of the file. A row of buttons usually appears below the menus. The Help author controls which buttons, if any, appear. Typical buttons include **Contents**, **Index**, **Back**, and **Print**, along with << and >> buttons to browse through the file. Help authors can also create custom buttons to jump to specific topics or perform other actions.

Below the buttons is the main text area of the window. Typically, the text begins with a heading, often bold or in a larger font than the rest of the text. This heading may sometimes be in what's called a non-scrolling region - an area of the window which will not move up or down if the user moves the scrollbar at the side of the window. Non-scrolling regions can only be used at the beginning of a topic. If a non-scrolling region is used, the Help author can control the size and background color of it.

Help authors can also control the background color of the main text area, where the actual text of the topic appears. This text can be formatted and arranged in many ways. Within the text, jumps will appear as green text with a single underline. Single-clicking on a jump will open a different topic. Some jumps may open secondary Help windows to display their information. Popups appear in the text as green text with a dotted underline. Single-clicking on a popup will open a small window with no menus, buttons, or scrollbars, sized to fit the text it contains. Often, popups are used to provide short definitions of key terms or other supplemental information about the main text. The popup will automatically disappear the next time the user clicks anywhere or presses any key.

Most Help files also contain a Table of Contents and an Index to help users locate the information they need. These feature appear in a separate, tabbed window. Clicking on the **Contents** tab will open the Table of Contents, in which users can click on headings to see the topics available for them. Often the headings are marked with icons that look like small books, while the topics have icons that look like pages. Double-clicking on a topic (or clicking on a topic then clicking **Display**) will open that topic. Clicking on the **Index** tab will open the index, which has a typing field and an alphabetical list of keywords. Typing in the typing field will automatically scroll the list of keywords to the closest match for what you typed. Double-clicking on a keyword (or clicking on a keyword then clicking **Display**) will display the topic associated with that keyword (if there is only one) or bring up a list of all the topics associated with it (if there is more than one). The index is a very important tool in helping users locate the information they need. Sometimes Help files also have a **Find** tab, which allows the user to search for any word used in the text of the file, not just for keywords.

WinHelp also supports a feature known as context-sensitive help. Context-sensitive help is assistance that is appropriate to where the user is in the software application, and what they are trying to do.

A rather security critical feature is that one can also include a DLL file containing custom code and associating it with WinHelp topics. Effectively this makes .HLP files equivalent to executables.

End of support

At the 2006 WritersUA conference, Microsoft announced its intentions to phase out WinHelp as a supported platform. Ted Dworkin (Partner Director of WinHelp Experience) stated, "WinHelp does not meet the code standards established for Vista. These standards include security, reliability, and performance. WinHelp is architected in such a way that we would have to rewrite it from the ground up to meet the Vista code standards. And that approach doesn't make sense given that we have two other Help systems in Vista."

The updated licensing agreement prohibits application developers from packaging the WinHelp libraries with their installers. What this means is that WinHelp manuals for

legacy applications will never be readable on a clean Windows Vista PC—in order to read them, the end-user must obtain the 32-bit WinHelp viewer from Microsoft's website and manually install it (which requires WGA "activation" and an account with administrative rights).

Update

In 2008 Microsoft relented and released a download of WinHlp32.exe for Windows Vista and Windows Server 2008.

Windows 7 support

Microsoft published Windows Help program (WinHlp32.exe) for Windows 7 on October 14, 2009, a week before the scheduled October 22 deadline. Microsoft has recently announced that separate downloads for Windows 7 and Windows Server 2008 R2 will be offered after the RTM versions of these products are available to public.

Other Documentation File Formats

Although documentation *can* be maintained entirely in a vendor-specific presentation format such as WinHelp, it is more often the case that documentation will need to be published in several different presentation formats at once: Microsoft Compiled HTML Help (CHM), WinHelp, HTML pages, Java Help, PDF, *etc.* It would be very expensive and error-prone to maintain each format separately.

For this reason, documentation is often maintained in an industry-standard, vendor-neutral authoring format, such as DocBook, from which several different presentation formats (including WinHelp) can be *generated.* All of the various presentation files thus produced (WinHelp or otherwise) will always be consistent with one another because all of them were generated from the same sources.

2. Microsoft Compiled HTML Help

Microsoft Compiled HTML Help

Filename extension	`.chm`
Internet media type	`application/x-chm`
Developed by	Microsoft
Initial release	1997
Extended to	.lit

Standard(s) No

Microsoft Compiled HTML Help is a proprietary format for online help files, developed by Microsoft and first released in 1997 as a successor to the Microsoft WinHelp format. It was first introduced with the release of Windows 98, and is still supported and distributed through Windows XP, Vista and Windows 7 platforms.

HTML Help files are made with Help authoring tools. Microsoft ships the Help Workshop with supported versions of Microsoft Windows and makes the tool available as a free download. There are also a number of third-party Help authoring tools available.

The Microsoft Reader .LIT file format is basically a modification of the HTML Help CHM format. CHM files are sometimes used for e-books.

In 2002, Microsoft announced some security risks associated with the .CHM format, as well as some security bulletins and patches. They have since announced their intentions not to develop the .CHM format further, and will be moving to a new generation of Windows Help called Microsoft Assistance Markup Language in the Windows Vista operating system.

History

Month	Year	Description
February	1996	Microsoft announces plans to stop development of WinHelp and start development on HTML Help.
August	1997	HTML Help 1.0 (HH 1.0) is released with Internet Explorer 4.
February	1998	HTML Help 1.1a ships with Windows 98.
January	2000	HTML Help 1.3 ships with Windows 2000.
July		HTML Help 1.32 releases with Internet Explorer 5.5 and Windows Me.
October	2001	HTML Help 1.33 releases with Internet Explorer 6 and Windows XP.
March		At the WritersUA (formerly WinWriters) conference, Microsoft announces plans for a new help platform, Help 2, which is also HTML based.
January	2003	Microsoft decides not to release Microsoft Help 2 as a general Help platform.

Format

A CHM Help file name has a ".chm" extension. It has a set of web pages written in a subset of HTML and a hyperlinked table of contents. CHM format is optimized for reading, as files are heavily indexed. All files are compressed together with LZX compression. Most CHM browsers have the capability to display a table of contents outside of the body text of the Help file.

The file starts with bytes "ITSF" (in ASCII), for "Info-Tech Storage Format". The format has been reverse-engineered by Matthew Russotto with assistance from Peter Ferrie and Paul Wise. Russotto's documentation is freely available.

On Windows computers, this Help file can be compiled using hhc.exe, part of the HTML Help Workshop. There are some open source tools which can read and explore these files, but they lack various features of the Microsoft Windows tools, most importantly write support.

A CHM file can contain links to other CHM files. When opening such a CHM file for the first time, the HTML Help viewer creates an index file with the extension ".ch*w*". The CHW file contains all the index terms of the master and linked CHM files, and enables faster searching for indexed terms.

Design

The format has the following advantages over other systems:

- File size smaller than plain HTML
- Range of formatting options that HTML gives for text presentation
- Ability to search the full text
- Ability to assemble several CHM files into one file with common TOC, index and search
- Ability to Generate TOC and Topic Folders containing international characters. Standard HTML Help will not generate these correctly.

The .chm file format's ability to contain and execute arbitrary code is a potential security threat. As such, the ability to view the file is often restricted by network security settings, and is affected by patches to the Windows operating system. HTML Help also does not fully support Unicode.

Applications

This format was originally intended only for encoding Help files, but other uses have since been found. It is very handy for packing saved HTML pages in one compact and browsable archive and for creating compact e-books. Some people use it to keep personal notes, because it can organize them in an ordered hierarchical table and allows quick text searching.

3. Microsoft Help 2

Microsoft Help 2.x is a proprietary format for online help files, developed by Microsoft and first released in 2001 as a help system for Visual Studio .NET (2002) and MSDN Library.

Microsoft Help 2.x is the help engine used in Microsoft Visual Studio 2002/2003/2005/2008 and Office 2007. Help files are made with the Help 2.0 Workshop (VSHIK), a help authoring tool. The default viewer for Help 2.x files is Microsoft Document Explorer, and there are several third-party viewers available such as H2Viewer and Help Explorer Viewer.

Visual Studio 2010 uses a new help engine, Microsoft Help System.

History

- March 2001 - Microsoft announced Microsoft Help 2.x at WritersUA (formerly WinWriters) conference.
- January 2003 - Microsoft decided not to release Microsoft Help 2 as a general Help platform. Help 2 remained a Visual Studio Help integration tool.
- August 2003 - Borland have released C# Builder. Documentation is all in Microsoft Help 2 format and displayed in Microsoft Document Explorer.
- December 2005 - Microsoft continues support of Help 2 by releasing the Help Integration Wizard which is Visual Studio 2005 compatible.
- December 2006 - Office 2007 is released and uses Microsoft Help 2. The Office help viewer is a custom viewer that can only view Office 2007 help.
- April 2009 - Microsoft announced at 2009 WritersUA conference that Microsoft Help System 1.x (development name was MS Help 3) will ship with Visual Studio 2010.

File format

A Microsoft Help 2.x file has a ".hxs" extension. A compressed .HxS help file (help title) is compiled from a set of topic pages written in a subset of HTML, a .HxC main project file, an .HxF include file, a .HxT table of contents, a .HxA attribute definition file, and a number of .HxK indexes (keyword Index, NamedURL index, optional associated and context links indexes).

Applications

This format was originally intended only for the help system used by Visual Studio .NET 2002 help, MSDN Library and TechNet, but now it is used in Office 2007 and some IDEs, such as Borland Developer Studio.

4. Microsoft Assistance Markup Language

Microsoft Assistance Markup Language (**Microsoft AML**, generally referred to as **MAML**) is an XML-based markup language developed by the Microsoft User Assistance Platform team to provide user assistance ("online help") for the Microsoft Windows Vista operating system. It is somewhat of a departure from all previous types of user assistance for Windows operating systems. Some of its features have been available in .NET Framework 2, but more options shipped with the release of .NET Framework 3.

The current type of user assistance for Windows operating systems uses files created with a command line compiler (hhc.exe) that compiles a .hhp (project) file, .hhc (table of contents) file, .hhk (index) file, and a collection of HTML topic files and the related resources (CSS, JavaScript, and image files) into one .chm file. The most significant aspect of MAML is that it shifts the production of user assistance to the concept of structured authoring (somewhat similar to DITA or DocBook). Documents and their constituent elements are defined by their context.

The emphasis is on content and the tasks a user performs with a computer, not the features of the software. Presentation is managed as part of the rendering engine when a user requests a topic.

The MAML authoring structure is divided into segments related to a type of content: conceptual, FAQ, glossary, procedure, reference, reusable content, task, troubleshooting, and tutorial.

Three levels of transformation occur when a topic is displayed: structure, presentation, and rendering.

The structural transformation contains reusable content and applies conditional logic to determine the structure that content should take when it is displayed, and the content of the text itself.

The presentation transformation enables content authored in MAML to use many different formats, including DHTML, XAML, RTF, and printed material.

5. Microsoft Help Viewer

Microsoft Help Viewer 1.x is the offline help system (local help) developed by Microsoft that ships with Visual Studio 2010 and its associated MSDN Library.

Microsoft Help Viewer 1.x supersedes Microsoft Help 2 which is the help system used by Microsoft Visual Studio 2002/2003/2005/2008 and Office 2007.

This is a new product and does not use any of the old help 2 code base. During development it was referred to as **MS Help 3.x**. With the growing need for a general Unicode based help system, it has the potential of becoming the next general help system for Windows.

History

- Jan 2008 - April Reagan [MS PM] blogs that Microsoft will replace Microsoft Help 2.
- Apr 2009 - At WritersUA 2009 conference April Reagan and Anand Raman announced Microsoft Help 3 will ship with Visual Studio 2010.
- Nov 2009 - Preview of new offline help ships with the VS 2010 Beta 2.
- Jan 2010 - Formal name changed from Microsoft Help 3.0 to Microsoft Help Viewer 1.0
- 12th April 2010 - Microsoft Help Viewer 1.0 is RTM (Release to Manufacturing) as part of the Visual Studio 2010 release.

File Format

Help file has a ".mshc" (Microsoft Help Container) file extension and is simply a standard Zip file renamed. It contains no proprietary files, just the author's content files.

A compiler (Workshop) is not required. Instead help files are ripped (Indexed) at installation time.

Topics files are written in XHTML 1.x compatible HTML. Standard HTML Meta Tags are used to define various topic attributes including the Table of Contents (TOC), Visible Index and F1 Keyword list.

User Experience

The user experience for Microsoft Help Viewer 1.x is that topics can be viewed in any installed web browser -- a separate application, such as the Microsoft Document Explorer included with Microsoft Help 2, is not necessary. The browser-based model is meant to provide a more lightweight navigation, downloading, and reading experience than earlier help-viewer models.

Visual Studio 2010 includes a taskbar applet in the Windows notification area (system tray) that arbitrates between viewing offline help and online help in the browser when F1 is pressed, and resolves help topic URIs to the proper topic page. It also includes a "library manager" application to manage the download, installation and uninstallation of help topics on the system, as well as whether to prefer online help when connected to the Internet.

Other platforms

- Apple Help (.HELP) - Apple Computer's proprietary help platform for the Mac OS 8.5+ operating system.
- Sun JavaHelp (.js) - A platform-independent help system written in Java programming language by Sun Microsystems. It runs on any platform and browser that supports the Java Runtime Environment (JRE).
- Oracle Help - Two formats developed by the Oracle Corporation: Oracle Help for Java (OHJ) and Oracle Help for Web (OHW).
- DotNetHelp - A new Windows help format, designed to replace the .chm format, that also supports .NET applications.
- The Texinfo program and format (also known as the "info" system) - the official documentation system for the GNU project.
- Unix man pages - the standard method used to document Unix commands.
- Information Presentation Facility (IPF) - the help system used by IBM's OS/2 system.

Chapter 11

HTML : Web Language

HTML

(HyperText Markup Language)

Filename extension	`.html, .htm`
Internet media type	`text/html`
Type code	TEXT
Uniform Type Identifier	public.html
Developed by	World Wide Web Consortium & WHATWG
Type of format	Markup language
Extended from	SGML

Extended to	XHTML
Standard(s)	ISO/IEC 15445 W3C HTML 4.01 W3C HTML 5 (draft)

HTML, which stands for **HyperText Markup Language**, is the predominant markup language for web pages. A markup language is a set of markup tags, and HTML uses markup tags to describe web pages.

HTML is written in the form of HTML elements consisting of "tags" surrounded by angle brackets (like <html>) within the web page content. HTML tags normally come in pairs like and . The first tag in a pair is the ***start tag***, the second tag is the ***end tag*** (they are also called ***opening tags*** and ***closing tags***).

The purpose of a web browser is to read HTML documents and display them as web pages. The browser does not display the HTML tags, but uses the tags to interpret the content of the page.

HTML elements form the building blocks of all websites. HTML allows images and objects to be embedded and can be used to create interactive forms. It provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. It can embed scripts in languages such as JavaScript which affect the behavior of HTML webpages.

HTML can also be used to include Cascading Style Sheets (CSS) to define the appearance and layout of text and other material. The W3C, maintainer of both HTML and CSS standards, encourages the use of CSS over explicit presentational markup.

History

The historic logo made by the W3C.

Origins

Tim Berners-Lee

In 1980, physicist Tim Berners-Lee, who was a contractor at CERN, proposed and prototyped ENQUIRE, a system for CERN researchers to use and share documents. In 1989, Berners-Lee wrote a memo proposing an Internet-based hypertext system. Berners-Lee specified HTML and wrote the browser and server software in the last part of 1990. In that year, Berners-Lee and CERN data systems engineer Robert Cailliau collaborated on a joint request for funding, but the project was not formally adopted by CERN. In his personal notes from 1990 he lists "*some of the many areas in which hypertext is used*" and puts an encyclopedia first.

First specifications

The first publicly available description of HTML was a document called *HTML Tags*, first mentioned on the Internet by Berners-Lee in late 1991. It describes 20 elements comprising the initial, relatively simple design of HTML. Except for the hyperlink tag, these were strongly influenced by SGMLguid, an in-house SGML based documentation format at CERN. Thirteen of these elements still exist in HTML 4.

HTML is a text and image formatting language used by web browsers to dynamically format web pages. Many of the text elements are found in the 1988 ISO technical report TR 9537 *Techniques for using SGML*, which in turn covers the features of early text formatting languages such as that used by the RUNOFF command developed in the early 1960s for the CTSS (Compatible Time-Sharing System) operating system: these formatting commands were derived from the commands used by typesetters to manually format documents. However, the SGML concept of generalized markup is based on elements (nested annotated ranges with attributes) rather than merely print effects, with also the separation of structure and processing; HTML has been progressively moved in this direction with CSS.

Berners-Lee considered HTML to be an application of SGML. It was formally defined as such by the Internet Engineering Task Force (IETF) with the mid-1993 publication of the first proposal for an HTML specification: "Hypertext Markup Language (HTML)" Internet-Draft by Berners-Lee and Dan Connolly, which included an SGML Document Type Definition to define the grammar. The draft expired after six months, but was notable for its acknowledgment of the NCSA Mosaic browser's custom tag for embedding in-line images, reflecting the IETF's philosophy of basing standards on successful prototypes. Similarly, Dave Raggett's competing Internet-Draft, "HTML+ (Hypertext Markup Format)", from late 1993, suggested standardizing already-implemented features like tables and fill-out forms.

After the HTML and HTML+ drafts expired in early 1994, the IETF created an HTML Working Group, which in 1995 completed "HTML 2.0", the first HTML specification intended to be treated as a standard against which future implementations should be based. Published as Request for Comments 1866, HTML 2.0 included ideas from the HTML and HTML+ drafts. The 2.0 designation was intended to distinguish the new edition from previous drafts.

Further development under the auspices of the IETF was stalled by competing interests. Since 1996, the HTML specifications have been maintained, with input from commercial software vendors, by the World Wide Web Consortium (W3C). However, in 2000, HTML also became an international standard (ISO/IEC 15445:2000). The last HTML specification published by the W3C is the HTML 4.01 Recommendation, published in late 1999. Its issues and errors were last acknowledged by errata published in 2001.

Version history of the standard

HTML version timeline

November 24, 1995

HTML 2.0 was published as IETF RFC 1866. Supplemental RFCs added capabilities:

- November 25, 1995: RFC 1867 (form-based file upload)
- May 1996: RFC 1942 (tables)
- August 1996: RFC 1980 (client-side image maps)
- January 1997: RFC 2070 (internationalization)

In June 2000, all of these were declared obsolete/historic by RFC 2854.

January 1997

HTML 3.2 was published as a W3C Recommendation. It was the first version developed and standardized exclusively by the W3C, as the IETF had closed its HTML Working Group in September 1996.
HTML 3.2 dropped math formulas entirely, reconciled overlap among various proprietary extensions and adopted most of Netscape's visual markup tags. Netscape's blink element and Microsoft's marquee element were omitted due to a mutual agreement between the two companies. A markup for mathematical formulas similar to that in HTML wasn't standardized until 14 months later in MathML.

December 1997

HTML 4.0 was published as a W3C Recommendation. It offers three variations:

- Strict, in which deprecated elements are forbidden,
- Transitional, in which deprecated elements are allowed,
- Frameset, in which mostly only frame related elements are allowed;

Initially code-named "Cougar", HTML 4.0 adopted many browser-specific element types and attributes, but at the same time sought to phase out Netscape's visual markup features by marking them as deprecated in favor of style sheets. HTML 4 is an SGML application conforming to ISO 8879 - SGML.

April 1998

HTML 4.0 was reissued with minor edits without incrementing the version number.

December 1999

HTML 4.01 was published as a W3C Recommendation. It offers the same three variations as HTML 4.0 and its last errata were published May 12, 2001.

May 2000

ISO/IEC 15445:2000 ("ISO HTML", based on HTML 4.01 Strict) was published as an ISO/IEC international standard. In the ISO this standard falls in the domain of the ISO/IEC JTC1/SC34 (ISO/IEC Joint Technical Committee 1, Subcommittee 34 - Document description and processing languages).

As of mid-2008, HTML 4.01 and ISO/IEC 15445:2000 are the most recent versions of HTML. Development of the parallel, XML-based language XHTML occupied the W3C's HTML Working Group through the early and mid-2000s.

HTML draft version timeline

October 1991

HTML Tags, an informal CERN document listing twelve HTML tags, was first mentioned in public.

June 1992

First informal draft of the HTML DTD, with seven subsequent revisions (July 15, August 6, August 18, November 17, November 19, November 20, November 22)

November 1992

HTML DTD 1.1 (the first with a version number, based on RCS revisions, which start with 1.1 rather than 1.0), an informal draft

June 1993

Hypertext Markup Language was published by the IETF IIIR Working Group as an Internet-Draft (a rough proposal for a standard). It was replaced by a second version one month later, followed by six further drafts published by IETF itself that finally led to HTML 2.0 in RFC1866

November 1993

HTML+ was published by the IETF as an Internet-Draft and was a competing proposal to the Hypertext Markup Language draft. It expired in May 1994.

April 1995 (authored March 1995)

HTML 3.0 was proposed as a standard to the IETF, but the proposal expired five months later without further action. It included many of the capabilities that were in Raggett's HTML+ proposal, such as support for tables, text flow around figures and the display of complex mathematical formulas.

W3C began development of its own Arena browser as a test bed for HTML 3 and Cascading Style Sheets, but HTML 3.0 did not succeed for several reasons. The draft was considered very large at 150 pages and the pace of browser development, as well as the number of interested parties, had outstripped the resources of the IETF. Browser vendors, including Microsoft and Netscape at the time, chose to implement different subsets of HTML 3's draft features as well as to introduce their own extensions to it. These included extensions to control stylistic aspects of documents, contrary to the "belief [of the academic engineering community] that such things as text color, background texture, font size and font face were definitely outside the scope of a language when their only intent was to specify how a document would be organized." Dave Raggett, who has been a W3C Fellow for many years has commented for example, "To a certain extent, Microsoft built its business on the Web by extending HTML features."

January 2008

HTML 5 was published as a Working Draft (link) by the W3C.

Although its syntax closely resembles that of SGML, HTML 5 has abandoned any attempt to be an SGML application and has explicitly defined its own "html" serialization, in addition to an alternative XML-based XHTML 5 serialization.

XHTML versions

XHTML is a separate language that began as a reformulation of HTML 4.01 using XML 1.0. It continues to be developed:

- XHTML 1.0, published January 26, 2000 as a W3C Recommendation, later revised and republished August 1, 2002. It offers the same three variations as HTML 4.0 and 4.01, reformulated in XML, with minor restrictions.
- XHTML 1.1, published May 31, 2001 as a W3C Recommendation. It is based on XHTML 1.0 Strict, but includes minor changes, can be customized, is reformulated using modules from Modularization of XHTML, which was published April 10, 2001 as a W3C Recommendation.
- XHTML 2.0,. There is no XHTML 2.0 standard. XHTML 2.0 is incompatible with XHTML 1.x and, therefore, would be more accurate to characterize as an XHTML-inspired new language than an update to XHTML 1.x.
- XHTML 5, which is an update to XHTML 1.x, is being defined alongside HTML 5 in the HTML 5 draft.

Markup

HTML markup consists of several key components, including *elements* (and their *attributes*), character-based *data types*, *character references* and *entity references*. Another important component is the *document type declaration*, which specifies the Document Type Definition. As of HTML 5, no Document Type Definition will need to be specified and will only determine the layout mode.

The Hello world program, a common computer program employed for comparing programming languages, scripting languages and markup languages is made of 9 lines of code although in HTML newlines are optional:

```
<!doctype html>
<html>
  <head>
    <title>Hello HTML</title>
  </head>
  <body>
    <p>Hello World!</p>
  </body>
</html>
```

(The text between <html> and </html> describes the web page, and The text between <body> and </body> is the visible page content.)

This Document Type Declaration is for HTML 5. If the `<!doctype html>` declaration is not included, Windows Internet Explorer will render using "quirks mode".

Elements

HTML documents are composed entirely of **HTML elements** that, in their most general form have three components: a pair of element ***tags***, a "start tag" and "end tag"; some element ***attributes*** within the start tag; and finally, any textual and graphical *content* between the start and end tags. The **HTML element** is everything between and including the tags. Each **tag** is enclosed in angle brackets.

The general form of an **HTML element** is therefore: `<tag attribute1="value1" attribute2="value2">content to be rendered</tag>` The name of the HTML element is also the name of the tag. Note that the end tag's name is preceded by a slash character, "/". If attributes are not assigned, default values are used.

Element examples

Header of the HTML document:<head>...</head>. Usually the title should be included in the head, for example:

```
<head>
  <title>The title</title>
</head>
```

Headings: HTML headings are defined with the <h1> to <h6> tags:

```
<h1>Heading1</h1>
<h2>Heading2</h2>
<h3>Heading3</h3>
<h4>Heading4</h4>
<h5>Heading5</h5>
<h6>Heading6</h6>
```

Paragraphs:

```
<p>Paragraph 1</p>  <p>Paragraph 2</p>
```

Line breaks:
. The difference between <br> and <p> is that 'br' breaks a line without altering the semantic structure of the page, whereas 'p' sections the page into paragraphs. Note also that 'br' is an *empty element* in that, while it may have attributes, it can take no content or end tag.

```
<code><p>This <br> is a paragraph <br> with <br> line breaks</p></code>
```

Comments:

```
<!-- Explanation here -->
```

Comments can help understanding of the markup and do not display in the webpage.

There are several types of markup elements used in HTML.

- **Structural** markup describes the purpose of text. For example, `<h2>Golf</h2>` establishes "Golf" as a second-level heading, which would be rendered in a browser in a manner similar to the "HTML markup" title at the start of this section. Structural markup does not denote any specific rendering, but most web browsers have default styles for element formatting. Text may be further styled with Cascading Style Sheets (CSS).
- **Presentational** markup describes the appearance of the text, regardless of its purpose. For example `<b>boldface</b>` indicates that visual output devices should render "boldface" in bold text, but gives little indication what devices which are unable to do this (such as aural devices that read the text aloud) should do. In the case of both `<b>bold</b>` and `<i>italic</i>`, there are other elements that may have equivalent visual renderings but which are more semantic in nature, such as `<strong>strong emphasis</strong>` and `<em>emphasis</em>` respectively. It is easier to see how an aural user agent should interpret the latter two elements. However, they are not equivalent to their presentational counterparts: it would be undesirable for a screen-reader to emphasize the name of a book, for instance, but on a screen such a name would be italicized. Most presentational markup elements have become deprecated under the HTML 4.0 specification, in favor of CSS based styling.
- **Hypertext** markup makes parts of a document into links to other documents. An anchor element creates a hyperlink in the document with the `href` attribute set to the link URL.

Attributes

Most of the attributes of an element are name-value pairs, separated by "=" and written within the start tag of an element after the element's name. The value may be enclosed in single or double quotes, although values consisting of certain characters can be left unquoted in HTML (but not XHTML). Leaving attribute values unquoted is considered unsafe. In contrast with name-value pair attributes, there are some attributes that affect the element simply by their presence in the start tag of the element, like the `ismap` attribute for the `img` element.

There are several common attributes that may appear in many elements:

- The `id` attribute provides a document-wide unique identifier for an element. This is used to identify the element so that stylesheets can alter its presentational properties, and scripts may alter, animate or delete its contents or presentation. Appended to the URL of the page, it provides a globally unique identifier for the element, typically a sub-section of the page.
- The `class` attribute provides a way of classifying similar elements. This can be used for semantic or presentation purposes. For example, an HTML document might semantically use the designation `class="notation"` to indicate that all elements with this class value are subordinate to the main text of the document. In

presentation, such elements might be gathered together and presented as footnotes on a page instead of appearing in the place where they occur in the HTML source. Class attributes are used semantically in microformats. Multiple class values may be specified; for example `class="notation important"` puts the element into both the 'notation' and the 'important' classes.

- An author may use the `style` attribute to assign presentational properties to a particular element. It is considered better practice to use an element's `id` or `class` attributes to select the element from within a stylesheet, though sometimes this can be too cumbersome for a simple, specific, or ad hoc styling.
- The `title` attribute is used to attach subtextual explanation to an element. In most browsers this attribute is displayed as a tooltip.
- The `lang` attribute identifies the natural language of the element's contents, which may be different from that of the rest of the document. For example, in an English-language document:
- `<p>Oh well, <span lang="fr">c'est la vie</span>, as they say in France.</p>`

The abbreviation element, `abbr`, can be used to demonstrate some of these attributes:

```
<abbr id="anId" class="jargon" style="color:purple;" title="Hypertext
Markup Language">HTML</abbr>
```

This example displays as HTML; in most browsers, pointing the cursor at the abbreviation should display the title text "Hypertext Markup Language."

Most elements also take the language-related attribute `dir` to specify text direction, such as with "rtl" for right-to-left text in, for example, Arabic, Persian or Hebrew.

Character and entity references

As of version 4.0, HTML defines a set of 252 character entity references and a set of 1,114,050 numeric character references, both of which allow individual characters to be written via simple markup, rather than literally. A literal character and its markup counterpart are considered equivalent and are rendered identically.

The ability to "escape" characters in this way allows for the characters < and `&` (when written as `<` and `&`, respectively) to be interpreted as character data, rather than markup. For example, a literal < normally indicates the start of a tag, and `&` normally indicates the start of a character entity reference or numeric character reference; writing it as `&` or `&#x26;` or `&#38;` allows `&` to be included in the content of an element or in the value of an attribute. The double-quote character (`"`), when used to quote an attribute value, must also be escaped as `&quot;` or `&#x22;` or `&#34;` when it appears within the attribute value itself. Equivalently, the single-quote character (`'`), when used to quote an attribute value, must also be escaped as `&#x27;` or `&#39;` (not as `&apos;` except in XHTML documents) when it appears within the attribute value itself. If document authors overlook the need to escape such characters, some browsers can be very forgiving

and try to use context to guess their intent. The result is still invalid markup, which makes the document less accessible to other browsers and to other user agents that may try to parse the document for search and indexing purposes for example.

Escaping also allows for characters that are not easily typed, or that are not available in the document's character encoding, to be represented within element and attribute content. For example, the acute-accented `e` (é), a character typically found only on Western European keyboards, can be written in any HTML document as the entity reference `&eacute;` or as the numeric references `&#233;` or `&#xE9;`, using characters that are available on all keyboards and are supported in all character encodings. Unicode character encodings such as UTF-8 are compatible with all modern browsers and allow direct access to almost all the characters of the world's writing systems.

Data types

HTML defines several data types for element content, such as script data and stylesheet data, and a plethora of types for attribute values, including IDs, names, URIs, numbers, units of length, languages, media descriptors, colors, character encodings, dates and times, and so on. All of these data types are specializations of character data.

Document type declaration

HTML documents are required to start with a Document Type Declaration (informally, a "doctype"). In browsers, the doctype helps to define the rendering mode—particularly whether to use quirks mode.

The original purpose of the doctype was to enable parsing and validation of HTML documents by SGML tools based on the Document Type Definition (DTD). The DTD to which the DOCTYPE refers contains a machine-readable grammar specifying the permitted and prohibited content for a document conforming to such a DTD. Browsers, on the other hand, do not implement HTML as an application of SGML and by consequence do not read the DTD. HTML 5 does not define a DTD, because of the technology's inherent limitations, so in HTML 5 the doctype declaration, `<!doctype html>`, does not refer to a DTD.

An example of an HTML 4 doctype is

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
```

This declaration references the DTD for the 'strict' version of HTML 4.01. SGML-based validators read the DTD in order to properly parse the document and to perform validation. In modern browsers, a valid doctype activates standards mode as opposed to quirks mode.

In addition, HTML 4.01 provides Transitional and Frameset DTDs, as explained below.

Semantic HTML

Semantic HTML is a way of writing HTML that emphasizes the meaning of the encoded information over its presentation (look). HTML has included semantic markup from its inception, but has also included presentational markup such as `<font>`, `<i>` and `<center>` tags. There are also the semantically neutral span and div tags. Since the late 1990s when Cascading Style Sheets were beginning to work in most browsers, web authors have been encouraged to avoid the use of presentational HTML markup with a view to the separation of presentation and content.

In a 2001 discussion of the Semantic Web, Tim Berners-Lee and others gave examples of ways in which intelligent software 'agents' may one day automatically trawl the Web and find, filter and correlate previously unrelated, published facts for the benefit of human users. Such agents are not commonplace even now, but some of the ideas of Web 2.0, mashups and price comparison websites may be coming close. The main difference between these web application hybrids and Berners-Lee's semantic agents lies in the fact that the current aggregation and hybridisation of information is usually designed in by web developers, who already know the web locations and the API semantics of the specific data they wish to mash, compare and combine.

An important type of web agent that does trawl and read web pages automatically, without prior knowledge of what it might find, is the Web crawler or search-engine spider. These software agents are dependent on the semantic clarity of web pages they find as they use various techniques and algorithms to read and index millions of web pages a day and provide web users with search facilities without which the World Wide Web would be only a fraction of its current usefulness.

In order for search-engine spiders to be able to rate the significance of pieces of text they find in HTML documents, and also for those creating mashups and other hybrids as well as for more automated agents as they are developed, the semantic structures that exist in HTML need to be widely and uniformly applied to bring out the meaning of published text.

Presentational markup tags are deprecated in current HTML and XHTML recommendations and are illegal in HTML 5.

Good semantic HTML also improves the accessibility of web documents. For example, when a screen reader or audio browser can correctly ascertain the structure of a document, it will not waste the visually impaired user's time by reading out repeated or irrelevant information when it has been marked up correctly.

Delivery

HTML documents can be delivered by the same means as any other computer file. However, they are most often delivered either by HTTP from a web server or by email.

HTTP

The World Wide Web is composed primarily of HTML documents transmitted from web servers to web browsers using the Hypertext Transfer Protocol (HTTP). However, HTTP is used to serve images, sound, and other content, in addition to HTML. To allow the Web browser to know how to handle each document it receives, other information is transmitted along with the document. This meta data usually includes the MIME type (e.g. `text/html` or `application/xhtml+xml`) and the character encoding.

In modern browsers, the MIME type that is sent with the HTML document may affect how the document is initially interpreted. A document sent with the XHTML MIME type is expected to be well-formed XML; syntax errors may cause the browser to fail to render it. The same document sent with the HTML MIME type might be displayed successfully, since some browsers are more lenient with HTML.

The W3C recommendations state that XHTML 1.0 documents that follow guidelines set forth in the recommendation's Appendix C may be labeled with either MIME Type. The current XHTML 1.1 Working Draft also states that XHTML 1.1 documents should be labeled with either MIME type.

HTML e-mail

Most graphical email clients allow the use of a subset of HTML (often ill-defined) to provide formatting and semantic markup not available with plain text. This may include typographic information like coloured headings, emphasized and quoted text, inline images and diagrams. Many such clients include both a GUI editor for composing HTML e-mail messages and a rendering engine for displaying them. Use of HTML in e-mail is controversial because of compatibility issues, because it can help disguise phishing attacks, because it can confuse spam filters and because the message size is larger than plain text.

Naming conventions

The most common filename extension for files containing HTML is `.html`. A common abbreviation of this is `.htm`, which originated because some early operating systems and file systems, such as DOS and FAT, limited file extensions to three letters.

HTML Application

An HTML Application (HTA; file extension ".hta") is a Microsoft Windows application that uses HTML and Dynamic HTML in a browser to provide the application's graphical interface. A regular HTML file is confined to the security model of the web browser, communicating only to web servers and manipulating only webpage objects and site cookies. An HTA runs as a fully trusted application and therefore has more privileges, like creation/editing/removal of files and Windows Registry entries. Because they operate

outside the browser's security model, HTAs cannot be executed via HTTP, but must be downloaded (just like an EXE file) and executed from local file system.

Current variations

> HTML is precisely what we were trying to PREVENT— ever-breaking links, links going outward only, quotes you can't follow to their origins, no version management, no rights management.
>
> Ted Nelson

Since its inception, HTML and its associated protocols gained acceptance relatively quickly. However, no clear standards existed in the early years of the language. Though its creators originally conceived of HTML as a semantic language devoid of presentation details, practical uses pushed many presentational elements and attributes into the language, driven largely by the various browser vendors. The latest standards surrounding HTML reflect efforts to overcome the sometimes chaotic development of the language and to create a rational foundation for building both meaningful and well-presented documents. To return HTML to its role as a semantic language, the W3C has developed style languages such as CSS and XSL to shoulder the burden of presentation. In conjunction, the HTML specification has slowly reined in the presentational elements.

There are two axes differentiating various variations of HTML as currently specified: SGML-based HTML versus XML-based HTML (referred to as XHTML) on one axis, and strict versus transitional (loose) versus frameset on the other axis.

SGML-based versus XML-based HTML

One difference in the latest HTML specifications lies in the distinction between the SGML-based specification and the XML-based specification. The XML-based specification is usually called XHTML to distinguish it clearly from the more traditional definition. However, the root element name continues to be 'html' even in the XHTML-specified HTML. The W3C intended XHTML 1.0 to be identical to HTML 4.01 except where limitations of XML over the more complex SGML require workarounds. Because XHTML and HTML are closely related, they are sometimes documented in parallel. In such circumstances, some authors conflate the two names as (X)HTML or X(HTML).

Like HTML 4.01, XHTML 1.0 has three sub-specifications: strict, loose and frameset.

Aside from the different opening declarations for a document, the differences between an HTML 4.01 and XHTML 1.0 document—in each of the corresponding DTDs—are largely syntactic. The underlying syntax of HTML allows many shortcuts that XHTML does not, such as elements with optional opening or closing tags, and even EMPTY elements which must not have an end tag. By contrast, XHTML requires all elements to have an opening tag and a closing tag. XHTML, however, also introduces a new shortcut: an XHTML tag may be opened and closed within the same tag, by including a slash before the end of the tag like this: `<br/>`. The introduction of this shorthand, which is not

used in the SGML declaration for HTML 4.01, may confuse earlier software unfamiliar with this new convention. A fix for this is to include a space before closing the tag, as such: `<br />`.

To understand the subtle differences between HTML and XHTML, consider the transformation of a valid and well-formed XHTML 1.0 document that adheres to Appendix C (see below) into a valid HTML 4.01 document. To make this translation requires the following steps:

1. **The language for an element should be specified with a `lang` attribute rather than the XHTML `xml:lang` attribute.** XHTML uses XML's built in language-defining functionality attribute.
2. **Remove the XML namespace (`xmlns=URI`).** HTML has no facilities for namespaces.
3. **Change the document type declaration** from XHTML 1.0 to HTML 4.01.
4. If present, **remove the XML declaration.** (Typically this is: `<?xml version="1.0" encoding="utf-8"?>`).
5. **Ensure that the document's MIME type is set to `text/html`.** For both HTML and XHTML, this comes from the HTTP `Content-Type` header sent by the server.
6. **Change the XML empty-element syntax to an HTML style empty element** (`<br/>` to `<br>`).

Those are the main changes necessary to translate a document from XHTML 1.0 to HTML 4.01. To translate from HTML to XHTML would also require the addition of any omitted opening or closing tags. Whether coding in HTML or XHTML it may just be best to always include the optional tags within an HTML document rather than remembering which tags can be omitted.

A well-formed XHTML document adheres to all the syntax requirements of XML. A valid document adheres to the content specification for XHTML, which describes the document structure.

The W3C recommends several conventions to ensure an easy migration between HTML and XHTML. The following steps can be applied to XHTML 1.0 documents only:

- Include both `xml:lang` and `lang` attributes on any elements assigning language.
- Use the empty-element syntax only for elements specified as empty in HTML.
- Include an extra space in empty-element tags: for example `<br />` instead of `<br/>`.
- Include explicit close tags for elements that permit content but are left empty (for example, `<div></div>`, not `<div />`).
- Omit the XML declaration.

By carefully following the W3C's compatibility guidelines, a user agent should be able to interpret the document equally as HTML or XHTML. For documents that are XHTML

1.0 and have been made compatible in this way, the W3C permits them to be served either as HTML (with a `text/html` MIME type), or as XHTML (with an `application/xhtml+xml` or `application/xml` MIME type). When delivered as XHTML, browsers should use an XML parser, which adheres strictly to the XML specifications for parsing the document's contents.

Transitional versus strict

HTML 4 defined three different versions of the language: Strict, Transitional (once called Loose) and Frameset. The Strict version is intended for new documents and is considered best practice, while the Transitional and Frameset versions were developed to make it easier to transition documents that conformed to older HTML specification or didn't conform to any specification to a version of HTML 4. The Transitional and Frameset versions allow for presentational markup, which is omitted in the Strict version. Instead, cascading style sheets are encouraged to improve the presentation of HTML documents. Because XHTML 1 only defines an XML syntax for the language defined by HTML 4, the same differences apply to XHTML 1 as well. The Transitional version allows the following parts of the vocabulary, which are not included in the Strict version:

- **A looser content model**
  - Inline elements and plain text are allowed directly in: `body`, `blockquote`, `form`, `noscript` and `noframes`
- **Presentation related elements**
  - underline (`u`)
  - strike-through (`s`)
  - `center`
  - `font`
  - `basefont`
- **Presentation related attributes**
  - `background` and `bgcolor` attributes for `body` element.
  - `align` attribute on `div`, `form`, paragraph (`p`) and heading (`h1`...`h6`) elements
  - `align`, `noshade`, `size` and `width` attributes on `hr` element
  - `align`, `border`, `vspace` and `hspace` attributes on `img` and `object` elements
  - `align` attribute on `legend` and `caption` elements
  - `align` and `bgcolor` on `table` element
  - `nowrap`, `bgcolor`, `width`, `height` on `td` and `th` elements
  - `bgcolor` attribute on `tr` element
  - `clear` attribute on `br` element
  - `compact` attribute on `dl`, `dir` and `menu` elements
  - `type`, `compact` and `start` attributes on `ol` and `ul` elements
  - `type` and `value` attributes on `li` element
  - `width` attribute on `pre` element
- **Additional elements in Transitional specification**
  - `menu` list (no substitute, though unordered list is recommended)

    - `dir` list (no substitute, though unordered list is recommended)
    - `isindex` (element requires server-side support and is typically added to documents server-side, `form` and `input` elements can be used as a substitute)
    - `applet` (deprecated in favor of object element)
- **The `language` attribute on script element** (redundant with the `type` attribute).
- **Frame related entities**
    - `iframe`
    - `noframes`
    - `target` attribute on `anchor`, client-side image-map (`imagemap`), `link`, `form` and `base` elements

The Frameset version includes everything in the Transitional version, as well as the `frameset` element (used instead of `body`) and the `frame` element.

Frameset versus transitional

In addition to the above transitional differences, the frameset specifications (whether XHTML 1.0 or HTML 4.01) specifies a different content model, with `frameset` replacing `body`, that contains either `frame` elements, or optionally `noframes` with a `body`.

Summary of specification versions

As this list demonstrates, the loose versions of the specification are maintained for legacy support. However, contrary to popular misconceptions, the move to XHTML does not imply a removal of this legacy support. Rather the X in XML stands for extensible and the W3C is modularizing the entire specification and opening it up to independent extensions. The primary achievement in the move from XHTML 1.0 to XHTML 1.1 is the modularization of the entire specification. The strict version of HTML is deployed in XHTML 1.1 through a set of modular extensions to the base XHTML 1.1 specification. Likewise, someone looking for the loose (transitional) or frameset specifications will find similar extended XHTML 1.1 support (much of it is contained in the legacy or frame modules). The modularization also allows for separate features to develop on their own timetable. So for example, XHTML 1.1 will allow quicker migration to emerging XML standards such as MathML (a presentational and semantic math language based on XML) and XForms—a new highly advanced web-form technology to replace the existing HTML forms.

In summary, the HTML 4.01 specification primarily reined in all the various HTML implementations into a single clearly written specification based on SGML. XHTML 1.0, ported this specification, as is, to the new XML defined specification. Next, XHTML 1.1 takes advantage of the extensible nature of XML and modularizes the whole specification. XHTML 2.0 will be the first step in adding new features to the specification in a standards-body-based approach.

Hypertext features not in HTML

HTML lacks some of the features found in earlier hypertext systems, such as typed links, source tracking, fat links and others. Even some hypertext features that were in early versions of HTML have been ignored by most popular web browsers until recently, such as the link element and in-browser Web page editing.

Sometimes Web services or browser manufacturers remedy these shortcomings.

WYSIWYG Editors

There are some WYSIWYG editors (What You See Is What You Get), in which the user lays out everything as it is to appear in the HTML document using a graphical user interface, where the editor renders this as an HTML document, no longer requiring the author to have extensive knowledge of HTML.

The WYSIWYG editing model has been criticized, primarily because of the low quality of the generated code; there are voices advocating a change to the WYSIWYM model.

WYSIWYG editors remains a controversial topic because of their perceived flaws such as:

- Relying mainly on layout as opposed to meaning, often using markup that does not convey the intended meaning but simply copies the layout.
- Often producing extremely verbose and redundant code that fails to make use of the cascading nature of HTML and CSS.
- Often producing ungrammatical markup often called tag soup.
- As a great deal of information of HTML documents is not in the layout, the model has been criticized for its 'what you see is all you get'-nature.

Nevertheless, since WYSIWYG editors offer convenience over hand-coded pages as well as not requiring the author to know the finer details of HTML, they still dominate web authoring.