James O. Ebert. Migrating Publications: How do Technical Writers Bound an Uncertain Problem Space? A Master's paper for the M.S. in Information Science degree. April, 2000. 57 Pages. Advisor: Gary Marchionini

This paper describes writer activities to define and resolve information migration issues that retard throughput of new information into technical publications in a production-focused work environment. The paper also reports the results of a questionnaire administered to a convenience sample of technical writers using ISO 8879 Standard, Generalized Markup Language (SGML), examining their information needs and information-seeking activities.

Headings:

SGML

ISO 8879

Information needs

End-user searching

Migrating information

MIGRATING PUBLICATIONS: HOW DO TECHNICAL WRITERS BOUND AN UNCERTAIN PROBLEM SPACE?

By
James O. Ebert

A Master's paper submitted to the faculty of the School of Information and Library Science of the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Master of Science in Information Science.

Chapel Hill, North Carolina

April, 2000

Approved by:

_____

Advisor

DEDICATION:

To Canela, who woke me at 5 every morning to work on this paper, and left her toys on the floor to be sure I was awake when I sat down to write.

TABLE OF CONTENTS

INTRODUCTION

How do technical writers bound an uncertain space that contains problems and solutions when a publication is first migrated to a new authoring system? This paper describes writer activities that define and resolve issues[1] that impede throughput of information into technical publications. The paper also tabulates and analyzes a questionnaire on information-seeking activities of experienced technical writers who migrate one or more publications to a new authoring system.[2]

An example scenario starts when a writer opens a file in a What-You-See-Is-What-You-Get (WYSIWIG) editing application to the first chapter migrated to a new authoring system. The writer inserts one additional information item in a bulleted list, and marks the item with a change bar – a vertical line | in the margin – to indicate new material to the reader. The change bar appears in the margin.

However, the symbol representing a bullet next to the new item disappears. The writer adds several additional items, noting each new item gains a change bar and loses its bullet. Seeking information, the writer scans the current user's guide for the system, finding no solution. Subsequently, an outdated copy of a hints-and-tips file provides a non-intuitive answer: change the list item to itself, obtaining the bullet's appearance. After applying the fix, the writer finds the change bar can no longer be removed from the list item. Puzzled, the writer leaves the workstation and walks down the hall to ask advice of a colleague who earlier migrated a publication.

Key terms defined by this paper include a **user** who identifies and resolves a **problem space** in the **migration** of a **technical publication** to a **new authoring system**, demonstrating one or more **information-seeking** behaviors.

---

[1] While I have sought out problems to illustrate the gaps and solutions that occur during migration, the authoring system described in this paper represents a valuable, extended set of improvements, not an unusual number of problems. For more information on the authoring system, see "Using Standard Tools and Processes" on page 50.

[2] If the reader is not familiar with authoring publications that use Standard Generalized Markup Language (SGML), read "SGML, a View from The Trenches," on page 39 before continuing with the main body of this paper. For background on the general team and business practices context in which technical publishing occurs, also preview "An Interconnected Work Environment," on page 50.

User

A member of a population of active, experienced, critical users of information. The users gather and publish information about computer programs that are sold commercially.

Problem space

A collection of one or more repeatable difficulties in using some aspect of an authoring system. A problem space temporarily slows the normal flow of information about new or changed function into existing publications.

Migration

An interconnected series of changes to the coded structures in source files containing the information in a publication. Significant changes can also occur to the processes that edit these files and that transform the files to printable or online formats.

Technical publication

An organized description of planning, installing, configuring, and using a computer program. A technical publication, such as a user's guide, is published either as hardcopy or as an online, viewable or printable file.

New authoring system

A collection of programs that edit, display, and enable the publication of one or more source files containing information. At least one significant element, such as a WYSIWIG editing interface, has an aspect of novelty or significant change in a "new" authoring system. For a more complete definition of the products that comprise such a system, see “Using Standard Tools and Processes” on page 50.

Information seeking

Behavior that identifies and obtains information pertinent to a problem, enabling one or more viable solutions. An interval of uncertainty occurs during an information-seeking effort. A writer’s peers may also impose norms on information-seeking behavior such as:

- Did you search the known body of information first?
- Did you attempt to solve the problem first?
- Can you repeat the error, or was it a one-time event?

- Did you capture error messages, file locations, and other relevant information about the problem?

From the perspective of structuring information with standard, generalized markup (the ISO 8879 international standard), the significance of migrating to a new authoring system is a pinprick of effort in an overall architecture intended to maintain the value of a base of information. The effort described in this paper is a relatively simple demonstration that the data (the document content) remains portable while the tools to manipulate the data change (p. 28, Travis and Waldt).

From the point of view of a major change in government and industrial publication activities, migration is a significant cost issue. For example, an online source for SGML standards reports that "the US Library of Congress and several research level institutions have been engaged in the collaborative work of the EAD (Encoded Archival Description) initiative for several years…using the Standard Generalized Markup Language (SGML). The documents are viewable on the Internet … or in some cases, are translated into HTML on the fly" (Cover, at http:// www.oasis-open.org/cover/gov-apps.html).

The economic volume of work is significant. For example, one World Wide Web source describes the US Patent and Trademark Office (USPTO) as "…a non-commercial federal entity and one of 14 bureaus in the Department of Commerce. The office occupies a combined total of over 1,400,000 square feet, in numerous buildings in Arlington, Virginia. The office employs over 5,000 full time equivalent staff to support its major functions -- the examination and issuance of patents and the examination and registration of trademarks. As of November 1998, a collection of USPTO Web Patent Databases was available online" (Cover, at http:// www.oasis-open.org/cover/gov-apps.html).

As new authoring systems continue to evolve for storing and manipulating documents, the problem of migrating very large data volumes in technical documents will remain important and expensive. This paper addresses the problem at the grassroots level of the technical writer/planner, from the perspective of how the technical writer seeks and uses information during document migration.

LITERATURE REVIEW

This study describes technical writers who identify and solve problems during the migration of publications to a new authoring system at a version 1.0 level. The study group is expert knowledge users who encounter a knowledge deficit of their own in solving problems. The study population "is limited to those groups or classes of people who are active, experienced, and critical users of information. That is to say, they are aware of their problems; they know, at least in approximate terms, where they can find useful information; and they have a critical sensitivity to what constitutes a solution, or, better said, a resolution of a problem in their context" (Taylor, p. 219). The study population encounters a need to process documents in a reduced state of knowledge about both the structure and process of a new authoring system.

Writers undergo a series of learning and discovery activities to bound the extent of the migration problem space. Blended into other work activities, they enumerate and partition a new range of issues, determine which previous solutions are still valid, attach unsolved issues to valid solutions, and confirm the problem resolutions are robust. From the user's point of view, information seeking is a "holistic experience with thoughts, actions and feelings interwoven into a complex mosaic rather than as separate distinct entities" (Kuhlthau, p. 348). There are significant time pressures to exit an anomalous state of knowledge, a period of time in which the writer recognizes a need for critical information exists and attempts to express the need (Belkin, Oddy, and Brooks, page 62). Time pressures also typically prevent their comprehensive search for information (Johnson, p. 93). The information needs they experience represent a gap preventing movement toward a solution, and the technical writer as sense-maker uses whatever bridge is available to build across the gap (Dervin and Nilan, p. 21).

One study of solution-seeking behaviors shows that persons in a state of information need tend to seek out nearby colleagues for advice first, and somewhat later to seek expert help (Johnson, p. 96). The migration effort occurs in a work environment with norms and expected levels of performance in information-gathering behavior. A technical writer works in a closely-knit team of programmers who consider the technical writer a team member, as well as a closely-knit team of writers for similar publications. The writer additionally belongs to a collection of more loosely-knit groups that share information, sometimes across remote sites.

Office workers in general, and technical writers also, appear to bring a production bias to their efforts to learn and use computing applications. The learner's "paramount goal is throughput" (Carroll and Rosson, p. 80). Their investigation asks, "How can systems transit between stages so as to most facilitate transfer of old knowledge and incorporation of new knowledge?" (ibid, p. 90)

Search “activities imply active search resulting from an area of doubt or more specifically a recognized problem; useful implies ways of resolving a problem through clarification, alteration, or actual solution as a result of information gained” (Taylor, p. 221). A problem can be separated “into three parts: questions which specify, problems which connect, and sense making which orients…what is conjectured here is that a problem and its resolution cannot readily be separated” (Taylor, p 225).

"Early stages of information seeking commonly are fraught with uncertainty and confusion…a principle of uncertainty is indicated as an underlying conceptual framework for information retrieval and provision" (Kuhlthau, p. 344).

Kuhlthau’s model of an information search process includes:

- Initiation, when a person first becomes aware of a lack of knowledge
- Selection, when the person identifies and selects a general area to be investigated
- Exploration, a time of confusion, uncertainty, and doubt, as the person investigates information to extend personal understanding of the general problem
- Formulation, the turning point in the process, as a focus is formed on the information encountered
- Collection, when information pertinent to the problem is gathered
- Presentation, when the search completes and the problem is resolved

Taylor sees the first steps in information definition as a process that starts with a "visceral" stage, becomes a "compromised" need as the user attempts to translate the request into system terms, and finally a "negotiated" need, often using the help of an intermediate source, that can be understood by the system (Taylor, 1968).

Writers try to estimate in advance the time and effort needed for successful migration. Their efforts are influenced by their perception of the situation, as Dervin and Nilan describe:

> "Situations have been coded primarily in terms of how they are seen by users as constraining movement…Categories have included the nature of the stop, described in terms of such categories as decision (facing a road with two or more branches ahead), problematic (being dragged down a road not of your own choosing), or spin-out (having no road). Other situations categorizations have focused on judgments of perceptual embeddedness (how foggy is the road), situational embeddedness (how many intersections are on the road), social embeddedness (how many people are also traveling), and constraint (what stands in the way)" (Dervin and Nilan, p. 21).

Their search for information follows multiple, established "discourses," with some navigation behaviors preferred over others (Livonen and Sonnenwald, p. 315). Behaviors aimed at seeking solutions may depend on the seeker's available collection of prior or new scenario models, which Hasdogan describes as "…user models based on formal or informal story lines relating to users, usage, the usage environment and the usage circumstances of the product of interest" (Hasdogan, p. 23). He enumerates a variety of models used in design, including cognitive models "…which represent the human being's sensory and cerebral processing system, his characteristics and limitations related to the elements of that system, and the outcome of such processes" (Hasdogan, p. 23). Scenario-based models include "…complex electronic interfaces, where the user has to follow a series of actions with functions of the product to accomplish a task, the designer has to build certain scenarios to anticipate the different ways in which people might access those functions. This usually follows a formal or an informal 'task analysis' process, where the user's tasks, goals and activities, and the product's functions related to those tasks, are identified, and subsequently some of those functions are prioritized or systematically linked to each other in the design" (Hasdogan, p. 30). Scenario types are typically based on the user's level of experience, including the following scenarios:

- Least competent user "…is based on a user whose capabilities are at the poorest limits in the use of a particular product."
- Worst case "…involves all the worst possible events happening at the same time when a product is functioning."
- Evolutionary "…represent[s] the nature of the user's relation to the product in an evolutionary process where the product is designed to teach or lead the user

> to improve his relation with the product (e.g., transference from being a 'naive' user to an 'expert' user)" (Hasdogan, p. 32).

In an organized education effort, writers who are migrating publications attend one or two-day classes that contain a normal task sequence designed to navigate migration issues. Part of the class is a tutorial, an example of Kaplan's reconstructed logic, which is "not a description but rather an idealization of … practice" that attempts to transfer expert knowledge to the novice (Kaplan, p. 8). "Conversations with others who may have the information they need, conferences, workshops, and symposia are always listed as highly important sources of information in these studies…" (Marchionini, p. 46).

Information seeking happens. "The information-seeking process is both systematic and opportunistic. The degree to which a search exhibits algorithms, heuristics, and serendipity depends on the strategic decisions that the information seeker makes and how the information-seeking factors interact as the search progresses" (ibid, p. 49). Barry describes the information-seeking activity as processes "that are dependent upon the knowledge and perceptions of the user, and dynamic processes in which the user's information need is a changing and fluid situation" (Barry, p. 150).

Solutions to an array of problems employ a variety of individual behaviors, team activities on several levels, diagnostic tools, and the writer's previous knowledge. "A user is concerned with establishing some degree of clarity in an area of doubt (a) by recalling previous experience for analogy; (b) through new knowledge or by confirming knowledge that illuminates, resolves, or alters the problem; or (c) with the discovery that there may be no resolution" (Taylor, p. 225). To put Taylor's null outcome in everyday words: progress is the certainty that everything you tried so far is the wrong answer.

"Most respondents first used their personal stores of technical information; then, they asked coworkers within the organization and colleagues outside the organization" (Pinelli, p. 300). Efforts often stop short of obtaining authoritative information from a more formalized process. Pinelli described aircraft engineers whose initial information search was characterized by oral communication with peers and was influenced by goals of accessibility and technical quality (Pinelli, p. 278). In a new situation, it may not be clear what is and what is not authoritative or of adequate quality. In the case of "bugs" in a new system, it is very possible there is *no* authoritative information – only a group of

people tasked to discover root causes. However, "…the classic law of 'least effort' has been evoked to articulate why channels are chosen first that involve the least effort" (Johnson, p. 96). Johnson appears to ignore possible norms of information seeking in a group environment, which may require an individual's localized effort occur before more extended information sources are approachable. Information activities do appear to change "when the nature of the task required a departure from the comfortable and participants needed to design information and informing actions to bring order out of chaos" (Solomon, p. 155).

Information search activities have preferred sources, although they are not typically the first consulted. "Experts in a field of study have comprehensive vocabularies in the domain, know what types of sources are best applicable to problems, and are aware of alternative access points for finding information in the domain" (Marchionini, p. 27). Formal grammars of task analysis are sometimes used in research to determine whether, after learning one system, a user finds it difficult or easy to learn another system (Olson, p. 260). Olson explores schemes such as skill and task taxonomies that attempt to automate business office systems that correspond to human strengths and weaknesses.

AN INTERCONNECTED ENVIRONMENT

Writers work in a business environment with many interconnected groups, both loosely and closely-knit in their interactions. Basic definitions of group membership and work in the electronic environment include the following:

Group or community

One possible definition of a group or community is that it is "a social network whose ties are tightly bounded within a delimited set and are densely knit so that almost all network members are directly linked with each other" (Wellman, p. 180).

Densely knit

In a densely-knit group, almost all members have frequent interactions. Every member knows every other member well. Dense, bounded groups tend to be viewed as the desirable form of community and work.

Strong ties, weak ties

> Strong ties provide more emotional aid, goods and services, and other support. "Weak ties are not useless, because they tend to connect people who have dissimilar social worlds, providing new information" (Wellman, p. 196). The usefulness of advice from weakly-associated strangers can be substantial, but rests on motivation other than expectation of an immediate return of a collegial favor (Constant, Sproull, Liesler, 1997).

Leadership in the technical community also needs to support the new authoring tools. Beneficial change as a characteristic of a "learning organization" is defined as a "process in which members…detect error…and correct it by restructuring organization theory of action" (Clayton, p. 82). Whether the innovation succeeds or fails, the leadership in an organization is involved. New systems succeed when participatory management incorporates democratic styles of organizational leadership, according to studies by Kurt Lewin, Ronald Lippitt, and Ralph White, who described leadership qualities of successful, innovative leaders.

Changes in a work environment such as new authoring systems are often represented as innovations. The first users are classically labeled early adopters, those "who buy because they are in love with technology…or whose needs for the newly developed functions are so great that they are willing to put up with any other problems" (Norman, p. 25). Norman expands on Rogers' theme with a concern for usability, asserting that a new concentration is needed on work activities and user experience, rather than a focus on the computer technology behind it. Norman cites Everett Rogers' categories of a population encountering new technology: early adopters who gamble on the new technology because the benefits greatly outweigh costs. They are followed by the early majority, the late majority, and laggards (Rogers, p. 248). Attributes of an innovation include relative advantage, organizational compatibility, simplicity, trialability and reversibility, and by observability (Rogers, p. 15). Additional attributes of innovations include:

- Originating the innovation from within the organization, making it able to be owned by members in some manner.

- Making the project divisible to implement on a limited basis, and also reversible in its effects.
- Aligning the innovation with interrelated policies and with interests of multiple stakeholders.

Migration to an innovative new system may be only partially voluntary if the new system provides economic gains in converging a variety of previously unlike business practices or enabling efficiencies in translation. Widespread usage happens over an extended period of time. Neuman found that innovation in new electronic processes is probably characterized by gradualism. "The shift to reliance on new means of communication will be evolutionary rather than revolutionary" (Neuman, p. 165).

METHODOLOGY

The methodology on which this paper is based entails the direct involvement of the author in obtaining problem solutions in his work environment. I write and help design technical publications in the area of computer applications. While working on this paper, I participated in the beta test classes for a new authoring system, and subsequently migrated the source files for several of my own documents to the system, as well as helping solve critical migration problems for colleagues in my team and nearby teams. I also referred problems to a central tools group for solution. For approximately a year prior to using the new authoring system, I used its predecessor authoring systems and actively monitored its state of user readiness by means of occasional discussions with one of the authoring system designers.

To focus on information seeking and problem solving, the methodology mentions but largely excludes issues of usability and innovation. Both topics are often associated with using new computer systems. They are minimized to limit the scope of this paper. While I have sought out problems to illustrate the gaps and solutions that occur during migration, the authoring system described in this paper represents a valuable, extended set of improvements, not an unusual number of problems.

This is not a disinterested study. Responding to a business need to migrate to the new system, my effort represents what Everett Rogers would term an early adopter's focus on acquiring an innovation, a position willing to gamble on new technology on the belief the gains outweigh the costs. A number of the findings in this paper represent a process of introspection in defining cognitive models and scenarios that (I hope) are sufficiently abstracted from personal efforts. The problems are impersonal and real: given the same set of files and the new authoring system, another writer would need to identify and solve the same problem. The effort occurred over a five-month period, during which the documented product inserted a significant number of new function descriptions and changes. The publication changed to meet the comments in a major edit and completed its inspection schedule on time for production. First-time shipment to a translation group using the same new authoring system also marked this interval.

Choice of the moment to gather information is also part of the methodology, which gathers data from the very first implementation of version 1.0 of an application before the existence of a significant body of community usage and major fixes. This time

interval may offer the maximum opportunity to observe experienced workers bridging gaps in their knowledge during a critical effort. The interval occurs infrequently. These efforts are relatively brief episodes in the life of a publication or authoring system, because intense efforts soon resolve issues. I assert that the version 1.0 interval is of interest because it falls at least partially outside Johnson's assumed zone of comfort; it is a less comfortable time when problems have no authoritative answer, only persons tasked to resolve them. The interval is also interesting because domain experts, who may be defining or redefining major elements in the domain, are potentially stressed and access to their advice may be more limited than normal. There are also sub-domain experts to whom access may be restricted.

From a wider perspective, the migration occurred in a work environment capable of enabling or retarding effective solutions. My coworkers also migrated publications to the new authoring system. This paper provides a sample of some of their problem-solving activities. It also reports on the demographic characteristics of a small, convenience sample of technical writers. For an expanded view of team and operating practices that are germane to publications, but are not directly a part of problem identification and solution, see Appendix B: "An Interconnected Work Environment" on page 50.

DEVELOPING THE QUESTIONNAIRE

Development of the questionnaire began as earlier class exercises on adapting existing questions on innovation by Moore and Benbaset, and on user satisfaction by Doll and Torkzedah[3], both topics on a tangent to the final focus of the questionnaire in this paper, which examines user activities to close an information gap. My searches of existing literature located no existing questionnaires on publication migration and problems. I therefore used my own experience and observations about several writers with whom I work to generate a set of questions. Several other writers who have experienced previous authoring system migrations reviewed the questions and suggested changes. A preliminary version of the questionnaire was administered to two writers and subsequently revised.

[3] Studies included a survey on innovation by Gary Moore and Izak Benbaset published September, 1991, in *Information Systems Research*, and a survey on user satisfaction, "The Measurement of End-User Computing Satisfaction," by William J. Doll and Gholamreza Torkzadeh, p. 259 in *MIS Quarterly*, June, 1988.

Some questions were anticipated to provide common indicators of a construct such as readiness for migration. It was also possible to anticipate relationships to other responses, such as whether a writer who reported being ready for migration also found the migration difficult.

Conclusions anticipated for the questionnaire were not supported by actual data from the respondent population, using a value of .05 (a generally-accepted error level in social science research) on a Spearman's test of correlation. The total quantity of questionnaire responses was also not great enough to justify calculation of Cronbach's alpha for the presence of the following factors:

- A factor might exist between (1) Sufficient information was available in time… (2) I have enough time to solve migration problems. (3) My computer and its programs were ready in time…
- A factor might exist between (4) The first thing I normally do when I cannot solve a problem… (5) To solve problems, the most valuable information source is… (6) Talking to or consulting with someone near me… (7) To solve problems with someone else's advice…
- A factor might exist between (9) I can help most writers… (10) I reported my … significant problems…
- A factor might exist between (14) Migration for my publication was… (15) I am confident my next migration effort will be…

The following table is a summary of possible conclusions anticipated before administering the questionnaire.

Question Number	Measured Item	Compared to	Compared to	Compared to	Compared to	Possible Conclusion
1, 2, 3	Preliminary information available, writer had enough time, and computer was ready	Used runtime messages as primary information source (4, 5)	Used experts as primary information source	Rated migration significantly difficult		Examine whether pressured writers use their computer runtime messages, or turn to experts more frequently
1, 2, 3	Preliminary information available, writer had enough time, and computer was ready	Reported problems	Helped others	Rated migration significantly difficult	Consulted with a nearby person	Did pressured writers report problems, help others, or rate migration difficult?
4, 5, 6, 7	Writer may have preference for information type	Helped others or reported problems to the database	Rated migration significantly difficult	Said a particular structure or process problem was difficult	Had enough time to migrate	Do writers with plentiful expert advice also help others, or find migration significantly difficult?
8	Writer had problems with particular information type, such as user's guide	Writer rated migration significantly difficult				Is there a publication that is commonly found to be difficult to use by writers who have migration problems?
9, 10	Writer helps others, reports problems to database	Rated migration significantly difficult				Do persons with significant migration problems help others more, or report problems more often?
11	Writer's effort is persistent	Reported problems to the database				Do writers stop at a workaround solution?
12, 13	Selected a particular structure or process problem type	Remigrated publication	Asked for expert advice	Rated migration more difficult		Do writers with a specific structure or process problem act differently?
14, 15	Writer found migration easy and is confident.	Writer had structure or process difficulty of certain type	Writer helps others, reports problems to database	Preliminary information available, writer had enough time, and computer was ready	Writer may have preference for information type	For confident writers with very easy migration, what were the significant elements?

The following table lists possible conclusions based on demographic factors that influence a writer's solutions to migration issues.

Question Number	Measured Item	Compared to	Compared to	Compared to	Compared to	Possible Conclusion
A1	Respondent with more than 10 years experience or respondent who worked on a previous migration	Respondents who said they had enough time to migrate their documents	Respondents who either attended a class or said information was available before migration	Respondents who report errors to the problem database	Respondents who feel they can help others	Examine whether migration wise respondents took the class in advance, and reported adequate time to migrate.
A2	Respondent who holds a team lead role	Respondents who have greater than 10 years' experience writing	Respondents who use verbal contact with experts as a preferred problem solution	Respondents who report errors to the problem database	Respondents who either attended a class or said information was available before migration	Examine whether role plays a difference in helping others, having adequate time, or reporting errors.
A6	Respondent who changed a publication in advance of migration	Respondents who either attended a class or said information was available before migration	Respondents who worked on a very similar authoring system in advance of migration			Examine whether exposure to the new authoring system causes changes in advance planning for migration.
A7	Respondent with very large or 4 or more publications to migrate	Respondents who said they had enough time to migrate their documents	Respondents who did not take the class before migration, or said information was not available			Is there is a possible "work fog" factor?
A8	Respondent spent significant time	Respondent with team leader role	Attended class?			Does time spent migrating relate to role or class attendance?
A9	Respondent who re-migrated at least one document	Respondents who use verbal contact with experts as a preferred problem solution	Respondents who worked on a previous migration	Respondents who either attended a class or said information was available before migration	Respondents who said they had enough time to migrate their documents	Examine whether experience, work load, and preferred contact style are interrelated to early migration planning decisions.
A10	Respondent who attended class	Respondents who feel they can help others	Respondents who report errors to the problem database	Respondents who said they had enough time to migrate their documents		Is class a factor in helping others or reporting errors?

The sample population is heavily drawn from the Research Triangle Park, NC (RTP) site, which may strongly affect both years worked, contact with expert sources, and other data in the questionnaire returns. (For example, other sites may have fewer years worked; the expert sources reside at RTP.) The entire population of early adopters is small, affecting levels of confidence in drawing conclusions. In descending order of returns, the sample population was obtained from the following sources:

- Eleven persons in my group and nearby teams at RTP
- Two from a group of about 16 writers from other sites who submitted a problem report to the central tools database for the new authoring system
- Lists of class attendees for the new authoring system. Very few class attendees migrated publications in the timeframe in which this paper was written, providing a zero percent return.
- Two persons known to me at other sites or referred by persons at other sites

Questionnaire returns were entered in the SPSS statistical system and the following tests were run:

- Correlations of significance (Spearman's)
- Central tendency, returning the mean, median, mode, and standard deviation
- Factor analysis
- Frequencies

DISCUSSION AND RESULTS

Office workers in general, and technical writers also, appear to bring a production bias to their efforts to learn and use computing applications. Their primary interest is in throughput. Under pressure to complete their tasks, writers work in a state of partial knowledge, using potentially flawed methods. Their efforts start with known processes and authoring structures, and work outward toward a goal of publication, finding problems and providing workarounds that represent a satisfactory outcome (Carroll and Rosson, p. 80). An active view of life in publishing technical documents includes "bumping into the environment," (Marchionini, p. 27) but most writers attempt to reduce the pain in the collision.

The extent of the problem space at migration is unknown to the users. The writer discovers a need to enumerate and partition a new range of issues, determine which previous solutions are still valid, attach valid solutions to unsolved issues, and confirm that problem solutions are robust. Some problems occur only at the instant of document migration, while others occur in ongoing, normal use of a new authoring system. The writer must also determine when the entire problem space is known with certainty.

The writer becomes temporarily unaware of the operational steps to produce a desired authoring effect, and knowledge of a previous authoring system can disrupt using the new one. Experienced users have patterns of prior behavior and understanding that potentially interfere with new patterns (Carroll and Rosson, p. 94). Formal grammars of task analysis are sometimes used in research to determine whether, after learning one system, a user finds it difficult to learn another system (Olson, p. 260). While the designers of the new authoring system spend significant time in task analysis, the end user seldom has the leisure to abstract a collection of work activities in a similar manner. Actively running the error checking mechanisms in the authoring system itself consumes a very large component of the time writers spend on problem identification and resolution. When the process starts, the writer attempts to edit a document or transform it to a printed or online file. Subsequently, the system points at error locations in the publication files. The writer locates the error, investigates the nature of the error, arrives at a solution, changes the file, and re-runs the edit or transform.

To ensure the value of continued investment of effort in migrated publications, the writer attempts to optimize chances for success in balancing a workload. New

information-seeking behaviors occur to bring order to a new, sometimes chaotic problem set (Solomon, p. 155). For example, a table representing an estimate for overall migration project success might appear as follows:

Total Migration Problems	**Time Until Production**	**Publication Size**	**Non-Project Task Load**	**Available Expert Help**	**Estimates of Success/ Start Now?**
Low	Long	Large	Small	Abundant	Excellent, start now
Medium	Intermediate	Large	Significant	Uncertain	Average, measure available time carefully
High	Brief	Large	Significant	Low	Low, do not migrate until time allows

As writers make their estimates, they approximate the time and effort needed for successful migration, which can involve a significant variety of information-seeking situations (Dervin and Nilan, p. 21). Regardless of the amount of help and the richness of a supportive environment, the press of delivery deadlines and the appearance of new and complex problems at late stages in a project often generate writer behaviors commonly called "tunnel vision." Given significant delivery pressure and imminent deadlines, information-seeking activity can become highly focused on ensuring the viable triage of a core set of problems, deferring a search for more efficient methods. In less pressing circumstances, more efficient activity can include an element of gaming. For example, when time permits the user to observe that a new authoring system has fewer complex structures in common use, replacing existing complex structures (for example, structures that involve pointers or recursion) may reduce the subsequent migration problem space. The game is essentially a prediction: if I spend more time now reducing the complexity of my library with a known authoring system, I will spend less time later solving migration problems with a new authoring system.

Writers with an existing library of publications bring a complex inventory of structured information that tests the new authoring process for robustness. In this study, these structures are expressed as statements in SGML files.

For example, the inventory for a typical publication such as a user's guide[4] can include:

Type of Container	Publication 1 (314 pages)	Publication 2 (296 pages)	Average (305 pages)
Simple table	22	29	25
Complex table	8	13	10
Numbered, bulleted, or definition list	142	112	127
Message list items	-	147	73
Syntax structure	5	2	3
Screen capture	23	39	31
Cross reference	209	175	192
Index items (estimated)	450	900	675
Code examples	83	25	55

A typical inventory can also include running heads and running footers, special edition notices, footnotes, bleed tabs indicating the chapter title on the page margin, chapter numbers above a chapter head, page numbers in simple sequence or as folio by chapter, and an assortment of front and back cover pages. The inventory for one publication can contain special sets of text and file entities or variables commonly used to control proliferation of terminology. There can be a collection of "write-once-use-many-times" text containers in a separate file, as well as a document version control function that changes the actual text represented by a variable, depending on a value the writer selects. For more information on using these authoring constructs, see Appendix A, “SGML, A View from the Trenches” on page 39.

Migrating a publication inventory to a new authoring system generates a population of new problems. The inventory acquires potentially altered structures, and an additional collection of new processes and residual problems. Previously-trusted processes may be discarded, changed, or validated for use as-is, removing any uncertainty value caused by migration.

The allowed number of complex information structures may shrink. For example, complex information re-use and information linking structures such as indexes may exist more simply and be manipulated differently in a new authoring system. Alternatively, index functions may be fully equivalent, but the writer may choose to eliminate more

[4] Values are from *Tivoli® Manager for R/3 User’s Guide*, Version 2.0, and *Tivoli® Manager for MQSeries® User’s Guide*, Version 2.2.1, copyright 1999, by Tivoli Systems Inc., an International Business Machines (IBM®) company.

complex structures prior to migration to avoid the potential for problems. Once migrated, more complex structures may be re-inserted in the publication's source files. Basic assumptions about information re-use can be woven into core entity declarations and the overall file design of an entire library, requiring change throughout to meet a new authoring system's entry criteria.

The length of processing time also increases. Part of the increase is caused by sequential information search methods during the writer's newly impoverished grasp of problems and solutions. Not all the material to be searched is cross-linked or has an index, for example. Another part of the delay is potentially longer machine runtime processing caused by code complexity in the authoring system.

Diagnostic clues the writer was accustomed to monitoring may change their normal messages, change location, or become hidden. For example, log files that contain diagnostic information change the message content, the file names, and directory locations. Error messages cite new, unfamiliar conditions. Available runtime reporting dialogs default to a minimized appearance on the desktop and must be opened. In an information overload condition, the writer may not perceive the loss of essential problem diagnostic information.

Information overload occurs during migration and the associated cleanup effort. The writer attempts to assimilate and search for help in some or all of the following:

- An organized body of formal class materials including tutorial exercises and a user's guide
- Online collections of issues, solutions, and processes
- Notes from other writers and project experts
- Additional compilations of hints and workarounds
- Written and verbal accounts of local team experience
- Advice from traveling migration experts
- Verbal interactions that are both formal in the classroom, informal with peers in the hallway, and intermittent with traveling migration experts

Finding salient elements efficiently in the mass of this new information is also part of the problem space during migration to a new authoring system. As the user attempts to find advice and solutions within an array of information similar to the following, the most frequent feedback on current problems is often the error messages

provided by the components that edit source files or transform source to printable files or browsable online files.

An assessment of information sources might look like the following:

Information Source	Search Method	Frequency of Use	Completeness
Error messages from the authoring system	Sequential, for each job	High, used at every cycle of editing, generating a table of contents, printing, or transform to online files	Complete, but tightly focused on specific function
Class materials	Sequential page-by-page or by table of contents	Intermediate to low	Partial, focused on overview and first tasks
User's Guide	Index and table of contents	Intermediate	Complete, focused on normal operations
Hints and tips	Sequential	Intermediate	Incomplete, focused on special migration problems
Online requirements and errors lists by sub-component	Sequential, by problem type. Heads searchable by user-defined string.	Low	Incomplete, focused on expert problem identification and solution
Online help	Random access, indexed	Low	Complete, but tightly focused on specific function
Verbal interactions, team experiences	Conversations, memos, meetings	Intermediate	Incomplete, variable depending on participant or team

"Noise" levels rise temporarily in efforts to find the solution set. For example, extraneous visual information may need to be ignored in a new authoring system. The visual display of font size on a computer monitor during authoring work in a What-You-See-Is-What-You-Get (WYSIWYG) editing tool may differ in similar chapters, but printing a hardcopy of the font subsequently proves that the appearance on the monitor has no real effect on hardcopy output.

Ostensibly correct solutions may fail to produce a desired result. For example, an item in a bulleted list may fail to display the bullet if the migrated item was marked with a revision bar (an attribute which puts a visible mark in the margin of the page). Simple deletion of the offending list item and retyping the item may fail. Only "tricks" such as changing the item to itself may be the correct solution, not obvious to the writer.

Error reports may cite problems with attributes that do not exist in new structures. For example, the width of a definition list item may not be measured in "tsize," but the error message may cite the attribute, which is valid for a previous authoring system.

Graphic files known to be good may appear blurry or fuzzy in new processing. No error actually exists in any authoring system or graphic capture process, but the writer must discover the solution is to change the compression values in an associated program, such as the Adobe Acrobat® Distiller®, which makes portable document format (PDF) files.

User errors may occur in migration. For example, changes in making graphics files viewable, especially PostScript files, can damage their printed appearance. Accidentally mixing viewable and non-viewable files in a library can cause subsequent printing problems. Clues to the problem source may require a graphics consultant, and regeneration of the PostScript files from other file types. If those file types happen to be absent from the archive, and if the product environment is no longer available to provide new screen captures, stress levels can become increasingly higher.

DEFINING PROBLEMS, APPLYING SOLUTIONS

Focusing on a specific example of real work limits the problem-solution space, and allows mapping knowledge from known work methods or publication structures to new ones. There are other relatively constant structures that orient such work. For example, given the overall goal of creating valid, well-formed SGML structures that remain unchanging in their relationships, a writer who has previous experience in an SGML-based authoring system will mentally map structures from one appearance (an ASCII editor's display of SGML tagging, for example) to another (a WYSIWIG editor's display of SGML tagging, for example). The writer also maps a known series of normal task sequences in one editing program to their expression with menus in another editing product. The writer works outward from known to unknown boundaries. The cognitive journey may be thought of as an individual who revises old tables held in personal awareness and constructs new tables in personal awareness that contain at least two columns and many rows.

The first table column defines a problem, and the second asserts (or invents) a solution. For example:

Problem	**Valid Solution**
Previously-good graphics now print with a "fuzzy" appearance.	Change the compression settings on the portable document format distiller.
The edition notice spans a front and back page, but should be on only one page.	Reduce the scaling factor on the table containing the edition notice. If the notice is still too long, move its ending to back matter and reference the new location.
A bug exists in substituting the text "Cause" for the normally-provided "Explanation" in message lists. "Cause" prints OK, but the error log contains two error reports for every message in the list.	The bug is harmless, but limiting. Investigate and report the problem for a future fix, providing an interim workaround.

However, the writer in a work overload situation, or in one in which an incorrect solution is pursued, could also construct a less productive problem-solution table:

Problem	Pressured Solution
Previously-good graphics now print with a "fuzzy" appearance.	Recapture all the graphics and concurrently attempt to discover why the old graphics-generation process is flawed.
The edition notice spans a front and back page, but should be on only one page.	Ignore the problem. Focus on more critical issues, such as fuzzy graphics.
A bug exists in substituting the text "Cause" for the normally-provided "Explanation" in message lists.	Ignore the problem.

An anomalous state of knowledge (Belkin, p. 133) also has the simple label "bug" when applied to the collection of computer programs in a new authoring system. An element of triage occurs. Whether bugs get attention can depend on their significance in blocking the production of a given publication. Some problems reported by a given process, for example, have no available guidance and represent the true first instance of a bug that can afflict an entire population of users. To navigate to a solution for a bug, the writer might construct a problem-solution table that has a sequence of activities such as the following:

Problem	Solution Activity Sequence
Mapping a message item prefix (msgiprefix) element such as "Explanation" to substitute text such as "Cause" creates a recursive set of bogus, ignorable errors in the error file.	**1) Discover the error occurs:** A cross-reference in one chapter points to a message list in an appendix. Errors occur in the transform log for the id "Sii7" also point to the message list. Save the error log. **2) Identify the work practice:** A developer asked the writer to change the label "Explanation" to "Cause." You must tell the developer if this change is not made. **3) Find the coded location in a source file:** The error log points at the following tags: <msgitemdef classname="EXPLANATION"> <title id="Sii7">Cause</title></msgitemdef> **4) Identify the real problem and explore solutions:** Remove and reinsert all valid combinations of a cross reference to the messages appendix, discovering the cross reference is not the problem. After 3 hours' effort, refocus on the tags that substitute the text "Cause," which are also cross reference tag types that cause "Sii7" to be automatically generated. Remove the tags that substitute "Cause," re-run the transform of source files to an output file type, and observe the errors do not occur in the log. **5) Apply a workaround solution**: The workaround is to use the standard label "Explanation." Advise the developer that "Cause" cannot be mapped until a fix is provided in the authoring system. **6) Report the problem:** Collect the code examples, write up the diagnostic attempts, and report the problem to the appropriate source, with a severity estimate that generates a quicker or slower response. **7) Confirm the experts agree a problem exists:** Monitor returning messages that demonstrate that experts can replicate the problem and agree with the assigned severity estimate. **8) Track the fix and share the experience:** Talk about the problem and solution in the hall with a peer writer, and put the issue in a local chat space to reduce other persons' length of time addressing the wrong xref problem. Report the problem in a causal analysis meeting at the end of the project. Over time, track the problem's fix for its appearance in a "patch" or new point release. **9) Tally total time spent:** Tally the total time spent on this bug workaround at 6 hours.

Time spent in an anomalous state of knowledge (dealing with a "bug") is affected by the user's search strategies. For example, the cross-reference problem in the previous table took longer to solve because of user assumptions that the first hit of an error on a

target code structure is more valuable than cascaded hits on subsequent lines. The initial problem definition failed to recognize there were two cross-reference structures involved, and the second one was the problem area. After significant time was consumed eliminating all combinations that generate the first cross-reference, the search moved to the second cross-reference and the problem was solved immediately.

Locating the error can be a very significant part of problem definition when no specific error message points to a line of code. In some cases, trial and error, followed by a period of reflection can resolve a problem. Additionally, a brute force method of comparing and trying every possible combination of entries or actions can assist the solution.

For example, a publication's table of contents fails to show updates when headers change in various chapters. An error message indicates the Document Type Definition (DTD) is not found during the generate cycle of the table of contents. For more information on the function of a DTD, see Appendix A, "SGML, a View from the Trenches" on page 39. The only location information in the error message points to a "sandbox" directory name that normally holds discardable "play" files. Repeated attempts fail to insert a different table of contents or to apply the document styles to the table. Placing the correct style files in the sandbox directory fails to solve the problem.

A period of reflection indicates that the problem exists on the outermost "container" of the document, where the DTD is first called. The writer calls on previous knowledge of SGML structures and their relation to a DTD, which controls coded structures such as a table of contents. There are several types of DTDs for different corporate groups.

Brute force methods then compare the extensive list of attributes on the outermost container of the suspect document to an error-free document. One attribute ("doctype") is discovered to be complete in the good document, but not in the offending document. The problem is resolved by entering the correct style value for the doctype attribute on the outermost container and applying the document styles to the table of contents. The newly-generated table of contents then correctly acquires changes in headers throughout the document.

Depending on the previous authoring system, the writer's concepts of valid authoring statement hierarchies can change. For example, statements that produce

arbitrary levels of user-selected headlines may have been valid in a loosely-validated proprietary authoring system. The hierarchy is modified when a document migrates to SGML, which uses more rigorous parsing for valid and well-formed elements.

A common denominator effect occurs in the search for viable solutions if the publication's source files must be proofed in more than one output process. For example, if chapter titles overrun the bleeding tabs on a printed page, an alternative short title tag may provide workaround relief in one printing process, but appear as literal text at the start of the chapter in another process. Failure to meet both process criteria discards the workaround.

Changes occur to processing steps to create, modify, transfer, and archive special files such as graphics files, in distributed and in mainframe environments. Changes occur in steps to transform source files to print or online-viewable files.

Loss of local process control can occur. For example, an organization can request that the actual migration processing be isolated at a central site, adding significant time delays. If the local writer asserts the need to do the migration activity more quickly, the writer also acquires the need to solve previously-transparent issues the central site solved.

DELIVERING SOLUTIONS

Solutions to this array of problems come from a variety of individual behaviors, team activities on several levels, diagnostic tools, and the writer's previous knowledge (Taylor, p. 225). If it exists in the writer's experience, a previous knowledge of common structures in valid and well-formed SGML provides a basis for stability in a new authoring environment. Assuming prior exposure to SGML occurred, the writer knows from memory whether a particular list, table, or division structure is correctly formed. Additional visual aids that show the SGML structure of a document or a menu item that enables validation of the document tagging may also provide assistance. From exposure to predecessor authoring systems, experienced writers are able to describe equivalent solutions to the same need in multiple authoring systems.

The writer also absorbs critical documentation for the new system, sometimes after the initial migration attempts take place. A common experience is that information that is usually easy to find cannot be located during the height of the migration effort. In its place, verbal interaction occurs with the hallway community on a "this happened to me" basis, which can provide verbal reinforcement and guidance. Periodic ad hoc team

meetings occur to discuss the day's problems. Calls occur to persons at remote site who have skills in diagnosis and remedies.

The writer invents and populates solution "buckets" such as an ignorable error category, a tricks and workarounds category, a user-error-accidents category, a solved-in-the-next-patch category, and a looks-ugly-but-working-as-designed category. A bucket is a problem container with “fuzzy” boundaries. More generally, solutions begin to fall into several main categories: Familiar structures, familiar processes, known bugs, and known workarounds.

Significant time can pass before a workaround is no longer needed, based on the problem’s severity level, which usually has a formal definition similar to the following:

Severity 1 Users are unable to proceed with work because a crucial function is defective. A fix is provided in one working day if the site is ready to install the fix.

Severity 2 The problem is serious, but a circumvention exists. A fix is available in an install package within 60 days or in the next release.

Severity 3 The problem causes only a minimal reduction in function. A workaround exists or the problem is not serious, although no workaround exists.

Severity 4 The problem causes no immediate impact to test or cannot be reproduced. If the problem is accepted, a fix is provided within 365 days or in a future release.

A collection of workarounds raise long-term maintenance issues that surface, for example, when a fix is provided and a particular workaround is no longer needed, or conflicts in some way with the official fix.

In the process of populating the familiar process category, for example, the writer regains control of the steps to process documents, answering low-level process questions such as:

- Must I generate the book's table of contents and index before I transform its source to HTML?
- Where is the runtime log display window and the log file that shows me the error listings for the current task?

- Which ASCII editor do I use to examine graphics files for hidden characters that can prevent printing?
- Which steps shifted position in the processing sequence? For example, adding bleed tabs to the edge of chapter pages was previously a late step before production. The new authoring tool makes it an early step, and changes process from the use of a separate file to changing an item on a hidden master page in every chapter file.

Sorting out the problems into categories itself provides a significant level of solution. For example, a set of solution categories may include:

Is a Familiar Structure	**Is a Familiar Process**	**Is an Already-Reported Bug**	**Has a Workaround**	**Is a New, Undefined Problem**
SGML table tagging looks different in WYSIWIG editors.	Validating a document finds all the errors in transforms to online formats.	Substituting text labels in message lists causes bogus error messages in the log.	Fuzzy graphics need PDF distiller changes.	Change bars do not display on some list items.
Index markers use colons to indicate subordination.	Bleed tabs are manually changed on the master page, by chapter.	List items lose the graphic bullet.	Change list items to themselves to regain the graphic bullet.	Inserting a notice division after the preface causes the next division to be read-only.

Piggybacking on a solution that has already been validated against a similar problem is a familiar search strategy in solving problems in migrating technical publications. Taking the path of "least effort," the writer seeks out a colleague, whose information may be inferior (Johnson, p. 96). There is a preference for advice from an expert (Marchionini, p. 27), although that source may not be the first chosen.

Groups of writers attend formal classes that contain a normal sequence and tutorial designed to navigate migration issues. Writers seek out hints and tips, informal notes, as well as the formal documentation for the new authoring system. Typically, the very first classes available are targeted at senior writers and consultants who are expected to provide a level of expertise at their sites. Class content describes familiar structures and processes, known bugs, and recommended workarounds. Class activities occasionally demonstrate one or more new, undefined problems as the instructor or students attempt to use the system.

BACK TO BUSINESS AS USUAL

With most of the significant migration problems solved, the writer reads the new printed and online output for accuracy against trusted copies made with the previous

authoring system, and corrects errors, if any. The information-seeking process may evolve opportunistically as migration stress declines (Marchionini, p. 49). The subsequent problem-solution space starts to approach a business-as-usual perspective as the writer resolves the bulk of the migration problems. Given a smaller, familiar problem set, the writer can more clearly estimate the time to address a solution. There may be additional time for research into alternate solutions.

For example, smaller font sizes may change on table column headers, causing labels that previously fit to wrap or collide with the column rule. Relieved after migration from a state of tunnel vision, the writer may re-construct a more spacious cognitive problem-solution table similar to the following.

Problem	Negotiated Priority	Trusted Solution	Perspective
Font sizes cause column headers to overrun column rule	High	Simplify the column headers	Simple problem not normally requiring help for solution
Second numbered list has wrong starting number	High	Change the container tag to itself	Similar to solution in hints and tips
Index item overruns next index column	Medium	Embed spaces in index item to allow line break	Trial and error shows embedded spaces allow line breaks
Add missing change bars to text containers	Low	Change the tag to itself or re-enter the paragraph	Described in hints and tips file

QUESTIONNAIRE RESULTS

The following results were obtained from a questionnaire administered to technical writers who were currently migrating, or had recently completed migrating a technical publication. This section describes correlations of significance between question returns, followed by descriptions of the frequency of responses.

Correlations of Significance

Using the SPSS for Windows statistical package, questionnaire returns provide the following information with a degree of significance (.05 or less) on a Spearman's test of correlation:

- Responses on sufficient information availability are positively correlated to perceptions of the ability to help others, and also to publication size.
- Responses on an ability to help others are negatively correlated to confidence the next migration will be easy.

- Returns on having one's computer and its programs almost completely or completely ready are negatively correlated to responses on the total publications worked on in the last year.
- Responses assessing the ease of a current migration are positively correlated to the amount of time spent on the migration effort and also with confidence the next migration will be easy. Responses assessing the ease of a current migration correlate negatively with responses on years worked in technical publications.

Frequencies of Interest

The study population is generally experienced and contains a large number of team leaders. About 75 percent of the respondents reported 10 or more years of technical writing experience. Approximately 87 percent indicated they had worked in a previous migration effort. About half the study group held a team lead (writer) position.

Ease of migration was rated somewhat difficult by about half the population (53 percent), with another 40 percent judging migration reasonably easy. About seven percent reported migration significantly difficult. The population most frequently worked on multiple publications in the year before migration, often on large publications. About 60 percent reported working on more than four publications in the previous year. The largest group (40 percent) migrated publications greater than 350 pages in length.

The migration decision did not commonly depend on taking a class on the new authoring system, and the choice to migrate a publication was typically made with only partial knowledge. Two of three respondents reported they did not attend a formal class before the migration. More than 70 percent said they worked on a similar authoring system before migration to the new one. For 40 percent of the population surveyed, the information needed to make a migration decision was sufficient about half of the time. Another 20 percent believed the information was sufficient most of the time.

Computers and programs were not ready for migration for about 20 percent of the respondents. Approximately 26 percent said their equipment and software were somewhat ready. The largest group (40 percent) judged their computer almost completely ready, and about 1 in 10 reported their computer and its programs completely ready for migration activity.

Migration was not necessarily a one-time activity for a publication. Two of three respondents changed publications to fit a migration need, and the same ratio re-migrated a publication.

The most difficult authoring system structures to understand during migration were concentrated (60 percent) in front matter topics. The database of problem-solution items (27 percent) and the user's guide (20 percent) were regarded as the most difficult sources of information.

The very first problem solving efforts initially concentrated (47 percent) on reading the processing errors, changing the publication, and re-running the process. A person nearby was the next most frequent first source of help for about 20 percent of the group, followed by the authoring system user's guide for about 13 percent. Throughout the migration effort, a nearby writer was most frequently contacted for help in about a third of the problem cases. For 40 percent of respondents, obtaining others' advice solved the problem about half of the time. Another 20 percent reported that others' advice solved the problem most of the time. The most valued information source was expert advice (40 percent), followed by approximately equal attention to error logs and to a collection of hints and tips.

Unsolved problems were most frequently met with a workaround (67 percent). The next most frequent approach to unsolved problems (26 percent) was to determine if the problem severity blocked publication.

Duration of the migration experience was most typically between 5 to 20 hours (53 percent). Approximately equivalent numbers (13 percent) reported the activity was very easy (1 to 5 hours) or very difficult (more than 80 hours). Nearly half of the study group (47 percent) declared they report problems most of the time or almost always. Feedback to the new authoring system problem base happened "some of the time" for another 40 percent of the study population. Reports of the most significant process change were widely distributed, with document validation holding the largest percent (19 percent). An equivalent number of respondents found no difficulty in using the new process.

Reflecting on the Questionnaire and the Qualitative Experience

There appears to be qualitative value in an investigator's actual involvement in migrating publications, obtaining information on process task sequences, real problem-

solution instances, and other behaviors that a questionnaire of some length would find difficult to explore. For example, the instance of problem-solution triage for a "bug" seems an unlikely candidate for discovery by a questionnaire. The addition of a qualitative experience also seems justified by the small sample size of this early adopter population, which prevents extension of questionnaire findings to a more general population. Involvement in the actual migration experience also provides the basis to pose appropriate questions to the larger group.

Combining findings from the questionnaire with the qualitative experience also helps define the information gap that occurs within the larger framework of an organization using tools based on a standard such as SGML. The questionnaire provides a perspective on gap-bridging activities and their concentration in certain work activities, such as the frequent use of nearby writers' advice. The questionnaire exposes areas of vulnerability reported by the study group as a whole in its effort to build new conceptual "tables" of problems and solutions, forming the basis for recommendations that one individual's qualitative effort cannot as firmly support.

CONCLUSION AND RECOMMENDATIONS

Migrating publications to a new authoring system provides a unique window into the problem-solution definition work done by “active, experienced, and critical users of information” (Taylor, p. 219) who focus on expediting the normal flow of information for new function into existing publications. The population described in this paper can be characterized as pioneers, probably more experienced than the normal population. Their efforts support subsequent widespread usage by a larger population that happens over an extended period of time. Within the recent interval that saw SGML evolve as a standard, their efforts are part of an developmental sequence characterized by gradualism, one that is “evolutionary rather than revolutionary" (Neuman, p. 165).

Migration can cause uncertainty in both the authoring structures and the related processing of a document. Bumping into the need to resolve a gap, or anomalous state of knowledge, technical writers demonstrate behaviors for preferred sources of information. They revise previous cognitive “tables” of problems and solutions and build new ones. New problem–solution constructs occur to ensure the value of continued investment of effort in migrated publications.

Bounded by a finite amount of time in a project, the writer becomes temporarily unaware of the operational steps to produce a desired authoring effect. The full extent of the "problem space" at migration expands processing for the document. The writer discovers a need to enumerate and partition a new range of issues, determine which previous solutions are still valid, attach unsolved issues to valid solutions, and confirm the problem resolutions are robust. Information overload occurs when the writer approaches a new array of documentation.

An element of triage happens to a population of migration problems. Some are solved immediately with information at hand. Another set requires investigation and repeated cycles of processing. A more resistant collection requires expert help, highly-itemized delineation, and imposes possible delays to provide a “fix” for a “bug.”

Error checking mechanisms in the authoring system itself provide significant input into problem identification and resolution. A supportive work environment also provides migration experts, formal classes, a tools design group, and an abundance of documentation on normal processes and problem solutions. Senior writers and consultants who are expected to provide a level of expertise at their sites attend classes

that describe familiar structures and processes, known bugs, and recommended workarounds. Classes also occasionally demonstrate the volatility of a new system with the actual occurrence of a new, undefined problem as the instructor or students attempt to use the system. Behaviors in solution seeking demonstrate a preference for certain types of information sources, including experts who possess “comprehensive vocabularies in the domain, know what types of sources are best applicable to problems, and are aware of alternative access points for finding information in the domain” (Marchionini, p. 27). Teams provide norms for problem definition and resolution. Informal contacts in a “hallway culture” and ad hoc interest group meetings provide discussion arenas to share possible solutions. Extended access to useful contacts may be restricted by project milestone requirements. The appearance of new and complex problems at late stages in a project can generate tunnel vision.

The changes to structure and process can be significant. On the path to regaining control of authoring structures and processes, the writer invents and populates solution "buckets,” determining a new set of familiar structures, familiar processes, known bugs, and both their workarounds and longer-term correction. “Bucket sharing” occurs both between teams, and across sites, in attempts to improve productivity in using the new system and decrease the stress on new users. The information-seeking process may change after document migration, as work returns to business as usual. Look-ahead information search strategies occur for alternate solutions that affect the next cycle of information flow into publications.

Although heavily weighted toward one site, questionnaire results reported in this paper are consistent with the initial premise that the study group is expert knowledge users who encounter a knowledge deficit in solving problems. For example, a significant number reported encountering difficult structures in the front matter of a publication. They recognize a need for critical information exists and attempt to express the need. Their problem-solving efforts initially concentrate on reading errors, making publication changes, and re-running an authoring process, and on visiting a nearby person who might solve the problem, which is reported to succeed about half the time. Expert advice is the most valued information, typically consulted after initial error solution attempts and nearby peers fail to provide relief.

Interested in throughput to meet production deadlines, technical writers as sense-makers use whatever bridge is available to build across the gap. Those surveyed appear to solve the problem space in reasonable time intervals, commonly reporting that migration takes place in between 5 to 20 hours. Responses to a questionnaire demonstrate many in the study group have prior involvement in an available collection of scenario models, typically in similar authoring systems and previous migration efforts. The study group's efforts contain a look-ahead strategy, seeking to determine whether sufficient information is available to decide whether to migrate publications, and whether computers are ready for the task. In an opportunistic effort, the group members frequently change publications to fit migration needs. Problems are examined to determine if their severity blocks publication. Difficult problems are frequently resolved with a workaround.

RECOMMENDATIONS

Based on questionnaire data, this writer's recommendations include the following:

- Reducing opportunities for front matter problems to exist, which may provide significant savings in time spent by the population using the new authoring system. Do hints and tips documents address the extent of front matter problems with the same degree of focus that migration seems to bring to the topic?
- Ensuring that the "someone nearby" who is a common first source of help has a reasonable level of expertise. The person's task load should allow addressing problems locally and reporting unsolved issues to a central tools group.
- Providing greater ease of use in searching the central database of problems and requirements and understanding or participating in priorities of fixes in the next patch to the authoring system.
- Examining whether computer readiness is really satisfactory in the general user population, or whether the questionnaire conceals underlying issues with its concentration on team leaders and early adopters.
- Collecting a sample of workarounds to determine whether they represent a significant hazard to future publication maintenance.

- Collecting more granular responses on publication changes (changing a central index to a distributed index, for example) made prior to migration to determine if they represent solutions not cited in the migration readiness section of the authoring system's user's guide.
- Examining why remigration occurred with a significantly high frequency in the first migration attempts.

REFERENCES

Barry, C., “User-defined relevance criteria: an exploratory study” *in Journal of the American Society for Information Science*, 45(3), 1994

Belkin, Nicholas J., “Anomalous states of knowledge as a basis for information retrieval” in *Canadian Journal of Information Science*, vol. 5, 1980

Belkin, Nicholas J., Oddy, R. N., and Brooks, H. M, “Ask for Information Retrieval,” in *The Journal of Documentation*, 38(2), 1982

Carroll, John, and Rosson, Mary Beth, "Paradox of the Active User," in *Interfacing Thought*, John M. Carroll, ed., The MIT Press, Cambridge, Massachusetts, 1987

Clayton, Peter, *Implementation of Organizational Innovation*, Academic Press, San Diego, CA, 1997

Cover, Robin, *The SGML/XML Web Page: SGML/XML Applications: Government, Military, and Heavy Industry*, http://www.oasis-open.org/cover/gov-apps.html

Constant, David, Sproull, Lee, and Kiesler, Sara, "The Kindness of Strangers: On the Usefulness of Electronic Weak Ties for Technical Advice," in *Culture of the Internet*, Lawrence Erlbaum Associates, Inc., Mahwah, NJ, 1997

Dervin, Brenda, and Nilan, Michael, "Information Needs and Use," in *Annual Review of Information Science and Technology*, vol. 21, 1986, Martha Williams, ed., published for the American Society for Information Science (ASIS) by Knowledge Industry Publications, Inc.

Doll, William J., and Torkzadeh, Gholamreza, “The Measurement of End-User Computing Satisfaction,” in *MIS Quarterly*, June, 1988

Hasdogan, Gulay, “The role of user models in product design for assessment of user needs,” in *Design Studies* Vol. 17, January, 1996

Johnson, J. D., *Information Seeking: An Organizational Dilemma*, Quorum Books, Westport, CT, 1996

Kaplan, Abraham, *The Conduct of Inquiry*, Chandler Publishing company, San Francisco, CA , 1964

Kuhlthau, Carol, "A Principle of Uncertainty for Information Seeking," in *The Journal of Documentation*, vol. 49, number 4, Dec., 1993

Livonen, Mirja, and Sonnenwald, Diane H., "From Translation to Navigation of Different Discourses…" in *Journal of the American Society for Information Science*, 49, 1998, John Wiley & Sons, Inc.

Marchionini, Gary, "Information-seeking Perspective and Framework," in *Information Seeking in Electronic Environments*, Cambridge University Press, 1995

Moore, Gary, and Benbaset, Izak, "Development of an Instrument to Measure the Perceptions of Adopting an Information Technology Innovation," in *Information Systems Research*, September, 1991

Neuman, Russell W., *The Future of the Mass Audience*, Cambridge University Press, 1991

Norman, Donald A., *The Invisible Computer*, Donald A. Norman, The MIT Press, Cambridge, MA, 1998

Olson, Judith R., "Cognitive Analysis of People's Use of Software," in *Interfacing Thought*, John M. Carroll, ed., The MIT Press, Cambridge, Massachusetts, 1987

Pinelli, Thomas E., *Information-seeking and Communicating Behavior of Scientists and Engineers*, C. Steinke, ed., The Haworth Press, New York, NY

Rogers, Everett, *Diffusion of Innovations*, The Free Press, New York, NY, 1983

Solomon, Paul, "Information Mosaics," (in press), from proceedings of an international conference on research in information needs, August, 1998, Sheffield, England

Taylor, R. S. "Information Use Environments," in *Progress in Communication Sciences*, X, 1991

TivoliÒ Manager for R/3 User's Guide, Version 2.0, and *TivoliÒ Manager for MQSeriesÒ User's Guide, Version 2.2.1,* by Tivoli Systems Inc., an International Business Machines (IBM®) company, 1999

Travis, Brian E., and Waldt, Dale C., *The SGML Implementation Guide,* Springer-Verlag, New York, NY, 1995

Wellman, Barry, "An Electronic Group is Virtually a Social Network," in *Culture of the Internet*, Lawrence Erlbaum Associates, Inc., Mahwah, NJ, 1997

APPENDIX A: SGML, A VIEW FROM THE TRENCHES

To understand this paper, you need to have a general awareness of Standard Generalized Markup Language (SGML), editing tools, and transform engines used to create large-scale technical documentation. Taking a technical writer's point of view, this appendix focuses on understanding a document that is tagged in SGML language. For a sample product such as a user's guide, it describes how to organize files for ease in writing, common SGML tagging, and typical problems and solutions. After reading this information, the reader who has a beginning knowledge of SGML should be able to understand the reasons and the work context for use of certain tagging structures, and manipulate file and SGML elements used in a typical technical manual.

More information on SGML, editing tools, and transform engines is available at the following:

- http://www.pubsnet.com/adobe.html - An online resource for training in publishing using Adobe FrameMaker®+SGML
- http://www.adobe.com/products/framemaker/prodinfosgml.html - Key features of FrameMaker+SGML
- http://www.arbortext.com/ - Products and other information provided by Arbortext, Inc. for their SGML and XML editors
- http://www.omnimark.com - The OmniMark® Technology Corporation home page for free and purchased SGML tools, including transform engines to produce HTML and other output from SGML source

TERMS AND DEFINITIONS

Terms used in this appendix include the following:

Content outline

An outline of the major elements in a publication. Typically, these include a table of contents and a preface, an introduction, and descriptions of installation, configuration, using, and troubleshooting a program, as well as various appendices, such as an appendix listing log files, and back matter that contains an automatically-generated index.

Document type definition (DTD)

A part of SGML that defines the presentation of information – how text and figures appear when printed or viewed and which elements are contained in other elements. It is "..a collection of element, attribute definition list, entity, notation, short reference, and comment declarations" (p. 229, Travis and Waldt). For example, the DTD determines that a paragraph cannot contain another paragraph, but can contain one or more notes. The availability of column rules in a table, for example, is defined in the DTD.

Document version control

Using logical conditions to control the output of a variable or entity that represents a text string such as a product name. For example, you can invent several labels for the same information and use the information in separate, closely-related products.

Side file

A file that contains information containers you intend to change once and call many times from various locations in a publication's files. For example, a side file contains a paragraph with a unique ID, a paragraph block with a unique ID, or an entire division with a unique ID.

Single sourcing

Describing a function once and re-using the information many times, an efficient practice that prevents errors that occur when a concept is described several times in separate text.

Transform

A transform is the action to change SGML into another viewable or displayable coding format such as HTML.

Vocabulary control

Broadly put, the reason you create text entities and side files. Humans create variations in vocabulary. To control (and eliminate) vocabulary variants, you automate the coding of key terms and phrases in SGML structures called entities.

DOWN INTO THE TRENCHES

Temporarily put aside all the interesting information you may have read about Document Type Definitions, which are admittedly important for SGML documents. I

don't ask you to forget what the DTD functions are, but out of a group of 1,000 technical writers, typically one or two specialists ever get their hands on that file. Perhaps, if you're in a very small group, your chances of working on a DTD are better. So, come on down in the large-group trenches for a minute, and I'll give you a short tour of a way to think about SGML used in technical writing for computer applications.

The following code fragment at the top of your document master file is about as close as you ever get to the DTD:

```
<!DOCTYPE SOMEIDDOC PUBLIC "+//ISBN 0-933186::SOME//DTD
SOMEIDDoc//EN" [
<!--ArborText, Inc., 1988-1998, v.4002-->
```

Your job as technical writer is not to decide the presentation of information, but rather, its content. I repeat, content is important. Presentation is not. Page breaks don't matter. Font size doesn't matter. There's a truism in the business that people who have little or nothing to add to a document's real content will wear you out changing the format of the publication. SGML saves you as a writer from those difficulties. If you describe a variable, for example, you want it tagged as a variable. So, when you tag it <pv>sample_variable_name</pv>, your work is done. Someone else will decide whether its final appearance is *italic*, **boldface**, or some other display convention. Your job is to describe the function of an application or other invention, not to create fancy formats.

Content outlines are typically the major structure for your work. They are really hard work, amounting to "think first (i.e., know the application you're going to write about), code later." No SGML knowledge is needed at this stage. You work with developers to describe the prerequisites for an application, its installation, configuration, typical usage, and useful trouble-shooting activities. You debate with peer writers and with developers who know much more than you do about the product.

Later, at a more granular level, as you flesh out the content outline, you typically apply lower-level skills in using valid, well-formed SGML containers to label your information. Examples are probably already available as pre-structured files that a senior writer built to ensure similarities in library naming and file maintenance. A significant part of your bread and butter as a writer is competent understanding of frequently-used

SGML "containers." For example, you should be able to use entity declarations, understand and implement information reuse, structure a master file and its embedded chapter files, and provide subordinate elements within a division. You should be able to link between content in different SGML containers.

WHAT ARE "CONTAINERS?"

"Think containers" was the first advice from an expert showing me the basics of SGML tagging. Every SGML tag pair is a container, with a start and end tag. Like containers do not overlap. Each container must have both a beginning and ending tag. Only certain subordinate containers are valid inside others. The elements of valid, well-formed SGML apply with a terrible vengeance to technical writing, if you ignore them until the week before production.

In terms of frequency of use, there are relatively few containers. Almost all of them can have an ID you can reference, but typically, divisions, lists, tables, and figures have IDs. The following is not a comprehensive list:

Division A division is the typical largest container. It contains other subordinate divisions. Within a division, you can have smaller containers adjacent to each other, or containing each other. Smaller containers include:

- Subordinate divisions
- Paragraphs
- Lists of all kinds, such as simple bulleted lists, numbered lists, definition lists (a term, and a definition), and a list of related notes. Lists can contain other lists, paragraphs, and other containers.
- Figures
- Notes
- Special syntax structures used to label parameters, values, and other command elements
- Tables, perhaps the most complex of the structures you use

CAN I USE A REALLY SIMPLE EDITOR?

Tagging information in SGML without using a parsing editor is one of the more likely methods to wreck a good set of SGML files. The parser that proofs your SGML tagging is strict, far more strict than any HTML browser you've ever met. Using a

special edit program, such as the Adept editor sold by Arbortext, Inc. is an extremely wise use of your time.

For example, suppose you have 10,000 lines of tags and text in a file, and somewhere in these lines, you decide to use a simple editor such as pico to move and change the list item to a paragraph in the following:

```
<li>Install the &apprxsr; on a managed node that is a gateway
to &endpnt;s.</li>
```

The result (with an error - the first <p tag is missing the closing > delimiter) looks like:

```
<p Install the &apprxsr; on a managed node that is a gateway
to &endpnt;s.</p>
```

You later discover, after closing the file, that your SGML parser declares a code violation. Unfortunately, until the next version of the parser (six months in the future), it doesn't point to the offending line. You now get to read all 10,000 lines (or run a difference utility against a backup file) to locate the problem. Most people don't believe this advice, by the way, until they've lost several days' work several times. Try it, and good luck.

A partial solution to the problem is to enforce the use of source that is SGML-enabled, but is binary rather than ASCII. The FrameMaker+SGML 5.5+ product, for example, uses binary files. You simply cannot make some types of tagging mistakes, such as the example. FrameMaker does provide the freedom to break a number of the other SGML rules for well-formed code, however.

Having said all that, a person with a firm grasp on the rules of SGML parent and child tagging can be very efficient, and dangerous, using a simple ASCII editor. The work requires the discipline to backup files frequently and proof small segments of work frequently with a good parser. You do find very expert technical writers taking their chances with the utmost in simplicity in editing SGML. They get away with the practice until their publication is translated, which creates a translation memory that is vulnerable at subsequent releases to certain types of line break re-ordering caused by simple editors.

WHAT ARE ENTITY DECLARATIONS?

There are several basic types of entities. One type points to another file. For example, the next code fragment points at a file that contains entities:

```
<!ENTITY % m310uent SYSTEM "m310uent.ide">
%m310uent;
```

Another basic entity type declares a text string that is a likely candidate to change, that you intend to write in one location, and reference multiple times. Another basic type of entity names a file, such as a graphic drawing or a screen capture. The SGML parser insists the entity names be unique.

It's handy to point to another file, for example, when you want to build a file containing only entities. Grouping all entities in one file prevents you from creating an entity for a particular name, such as a product, in one chapter, and another entity for the exact same name in a different chapter.

Another common use of a text entity is to name a product that is expected to change its name, as it frequently does before general availability. You change the actual name in one entity declaration, and all the references are automatically changed throughout a publication. For example, the next entity declaration names an installable program likely to change its formal product name several times before you publish your work:

```
<!ENTITY apprxsr "Application Proxy server">
```

Inside a given chapter, you call the &apprxsr; entity in a line such as:

```
<li>Install the &apprxsr; on a managed node that is a gateway
to &endpnt;s.</li>
```

Entities that represent names provide another useful function - vocabulary control. Humans seem to love variation, and it shows in the invention of new terminology. Often, when a group of developers first start naming the parts and functions in their new application, several very similar variations are invented for the same function. But to the naive user, is the collection of synonyms really one function, or different but very closely-related functions? Text entities enforce the use of a standard vocabulary and reduce error and misunderstanding.

Allowing choice in entity declarations can cause problems, and your selection of an editing program can influence how you perceive the use of text entities. In FrameMaker+SGML 5.5.3, for example, variables are used as a substitute for text

entities. The problem at release 5.5.3 is that a variable can be declared local to a particular chapter file, becoming separated from the main body of entities, which are usually organized in the prolog (or setup) file and imported to all chapter files. Other products solve the problem of distributed entities by providing a menu only on the main editing panel to control all text entities, eliminating the possibility of a text entity declaration that is local only to an individual chapter file.

WHAT IS A MASTER FILE?

A master file could contain all the lines in your publication, but typically, it doesn't because breaking up content into chapter-level files helps organize your effort. A master file usually contains:

- A pointer to the DTD
- A reference to an entity file (although the entities could be at the top of the master file for simplicity sake)
- A variety of hidden information, such as the author, date, and revision levels, as well as publication numbers and other information
- Reusable information in what is called an object library or side file. The concept of writing information once and calling it in multiple places is the basis for such files, which are typically simple collections of information scattered throughout a publication.
- Container tags that mark the boundaries of the document. For example, parts, body, appendices, glossary, and index.
- References to chapter files
- Appendices
- Glossary
- Index

WHAT IS A SIDE FILE?

A side file collects the single instances of some bounded information. Each information container has an ID you can reference elsewhere. Suppose you have a command or parameter you need to describe several times throughout a book, or throughout a library of books. If you write out the meaning of the command each time it appears, you may find later that the meaning changes as developers rethink their product.

Publication errors occur if you forget to change every instance of the text in the book. How much simpler it is to write the explanation once and call it in different locations.

For example, the ID that is named stplb3 is an example of reused information:

```
<li id="stplb3">In the STEPLIB statements, replace
<xph>your.product.SCSQAUTH
</xph> and <xph>your.product.SCSQLOAD</xph> with the data set names for
your
actual product libraries.</li>
```

Of course, you don't have to use a side file. You can instead embed the commonly-repeated information at the top of the master file, in the object library (objlibbody).

WHAT ARE EMBEDDED CHAPTER FILES?

A chapter file provides the organization for some major element in your content outline, such as event handling. A chapter file typically starts with a fragment header:

```
<!-- Fragment document type declaration subset:
ArborText, Inc., 1988-1998, v.4001
<!DOCTYPE SOMEIDDOC PUBLIC "+//ISBN 0-933186::SOME//DTD
SOMEIDDoc//EN" [
<!ENTITY % m310uent SYSTEM "m310uent.ide">
%m310uent;
]>
-->
```

The remainder of the chapter is a simple sequence of paragraphs, lists, figures, and tables. A portion of such a chapter might look like the following:

```
<d id="introd" style="BKM:(topicsel=yes subjart=tivch1)">
<dprolog><titleblk>
<title>Introducing &ProductOnly;</title>
</titleblk></dprolog>
<dbody>
<p>The &ProductOnly; (&Module;) provides a centralized
system management tool for &appl; on the &company; platform.</p>
```

WHAT IS AN INDEX?

You've used an index many times, but when you create one for others, there are typically two types of index organization. Simpler is better, especially if you're trading content frequently with other writers, or suspect your writing tools might change, or you have translation requirements.

The simple type of organization puts the index declarations immediately adjacent to the information you want to index. For example, an entry would be:

```
<i1 id="cfgg"><idxterm>configuration</idxterm>
<i2 id="cfgmgt" refid="cfgg"><idxterm>management
</idxterm></i2>
</i1>
```

The second type of indexing, central indexing, is more complex. You organize all the index entries in a separate file, and then point to them from the actual location you want to index. Here's a sample entry in a separate, central file:

```
<i1 id="cfgg"><idxterm>configuration</idxterm>
<i2 id="cfgmgt"><idxterm>management
</idxterm></i2>
</i1>
```

And the reference (the "iref") in a particular chapter looks like this:

```
<iref refids="cfgmgt">
```

WHAT IS A TRANSFORM?

A transform is the action to change SGML into another viewable or displayable coding format. Very few SGML display tools exist for the general public. Technical writers use programs that transform SGML to usable output, such as HTML that a browser can display, or PostScript files, which can be subsequently printed, or modified to portable document format (PDF) files that are printer independent and are immediately displayable by a product such as the Adobe Acrobat Reader program.

Transforms are also handy to have because they provide additional syntax checking that you may not get from an SGML editor. Sometimes, you want to run both the print and the HTML transform to ensure you caught all the errors.

WHY USE DOCUMENT VERSION CONTROL?

Sometimes a functional unit of a program can be used interchangeably in several different products, or with very slight changes. It turns out to be efficient to use the same documentation too, and just change the product name or add or eliminate small changes based on which product is produced.

Here's why document version control is very useful. You declare a text entity that has several possible strings as its output, depending on the value of a variable, which

we'll call PRODUCT. The first string is something like "Darling Dark Chocolate" and the second string is "Creamy Milk Chocolate”.

```
<!ENTITY product "<ph props='DARK'>Darling Dark Chocolate</ph><ph
props='MILK'>Creamy Milk Chocolate</ph>">
```

Another file, usually called the VAL file, contains the value of DARK and MILK variables. VAL files are called at run time, when you generate a book or HTML file, to select which strings actually appear in the file. For example, to cause the value of PRODUCT to be “Darling Dark Chocolate” the file declares the following:

```
DARK:=#T
MILK:=#F
```

VAL files typically run in pairs: one file for one combination of truth conditions, the other for the reverse set of conditions. You don’t literally have to declare the Boolean value of both variables, but it helps the writer see all the possible combinations, and avoids mistakes by implying but not clearly stating all variables in the population. Being explicit also helps translation centers, because “throwing it over the wall” is a very typical working practice when you hand off publications to translation.

It’s common to use variable controls to work around problems in an authoring language or its transforms to HTML and print. For example, suppose a reserved entity called ✓ fails to show a check mark in HTML, but works OK in print. You can declare the following:

```
<!ENTITY ckm "<ph props='HTML'>X</ph><ph
props='PRINT'>&check;</ph>">
```

Another file contains the value of HTML and PRINT. For example, to cause an X to appear in HTML, the file contains:

```
PRINT:=#F
HTML:=#T
```

FILE EXAMPLES

A set of SGML files for a publication includes the following:

mt10u.idd master file, indicated by the *.idd filetype.

mt10ucpy.ide	copyright file, containing the legal boilerplate
mt10upre.ide	preface, providing a high-level abstract and pointers into the publication
mt10uint.ide	introductory chapter
mt10uins.ide	installation and configuration chapter for the product
mt10unxt.ide	the next chapter
mt10uapp.ide	sample appendix file to test the back matter
mt10uapb.ide	another appendix file to be sure a second appendix is generated
mt10uglo.ide	sample glossary file
mt10uprt.val	determines the value of a text entity in print
mt10uhtm.val	determines the value of a text entity in HTML

APPENDIX B: AN INTERCONNECTED WORK ENVIRONMENT

Strong expectations of business advantage often propel the adoption of a new authoring system. The migration effort occurs in the context of existing work practices, standard tools and processes, and a writer community composed of both closely and loosely-knit teams.

VECTORS SUPPORTING MIGRATION

Migration may occur because a new authoring system provides real gains in productivity, reducing the time to do a task such as indexing. The impetus may also start when acquisition of other companies with valuable applications brings together writers at a variety of remote sites. Their products already have documentation in a variety of authoring packages, but may need to provide a more standard appearance. Transfer of project development between sites causes a need to transfer the related documentation files, which are more efficiently changed at the next release if both groups use the same authoring tools. If the authoring system is a commercially-available product, it is easier to assess skills and hire contractors with previous experience that can be rapidly applied to existing tasks.

Anticipated efficiencies in translation can fuel migration to a new authoring system. Translation requires conserving the cost of generating and re-using translation memories from release to release. Common practices and publication tools are required for entry into the translation process.

Converging on a common, new authoring system also makes a centralized tools group more efficient in providing common process help and system upgrades to multiple sites. Costs of developing and extending an authoring package can be spread across many sites, reducing the cost per site.

USING STANDARD TOOLS AND PROCESSES

The writer community in general may have an authoring perspective that uses Standard Generalized Markup Language (SGML), which removes many, but not all, of the presentation aspects of a published work from the individual writer's domain. In general, the technical writer attempts to minimize uncertainty factors, using trusted:

- Publications programs
- Processes to generate, publish documents
- File transmission and packaging utilities

The technical writer applies standard tools and processes to bound the presentation of information. For example, a writer in an International Business Machines (IBM®) environment may use Frame+SGML®, produced by Adobe Systems Incorporated. The product provides a WYSIWIG editor for documents that conform to the SGML standard. The writer may alternatively use IBMIDDOC Workbench, or combine Workbench with an additional product called Frame2000. Or, a writer may use a legacy authoring system such as BookMaster®. In the migration described in this paper, the "new authoring system" includes Frame2000 support for the IBMIDDoc DTD. Frame2000 enables WYSIWIG editing of draft documents using Frame+SGML 5.5.6, with the final production processing provided by the Workbench. "The [Frame2000] solution will import and process valid IBMIDDoc data as well as export valid IBMIDDoc that fully conforms to the IBMIDDoc DTD and downstream tools and processes" (p. 2, *Frame2000 User's Guide*, Release 1.0, International Business Machines Corporation, 1999). Frame2000 is licensed to IBM by Softline International, Inc.

A partial list of additional benefits associated with using Frame2000 include:

- Integrated use of a tool to analyze English statements for translation, called the Easy English Analyzer
- An indexing wizard (Ixgen+SGML, Frank Stearns Associates) to speed indexing efforts
- Menus that integrate the use of IBMIDDoc Workbench to provide output file types for printing and online viewing

Similarities in conceptual treatment of material may span multiple products, using templates for information plans that contain similar content outlines. Across the entire organization, the template for information plans may be organized under the umbrella guidelines of a common development process. With the possibility of a compliance audit, teams apply standard procedures that meet ISO 9000 requirements, an industry standard that basically examines whether established processes are, in fact, followed in normal practice. Some publications may experience cross-site collaboration to provide common library templates for similar publications. Most publications make use of cross-site editing style guides.

Individual publications that are the work output typically share common content outlines, as well as similar file and lower-level source tagging schemes in an extended

library of similar publications. As part of describing a computer application, the writer installs and uses the application. As a member of a development group of programmers, the writer may also participate in designing user interface elements. The writer gathers information during the development cycle and periodically reviews the increment with test and development experts. Editor and peer reviews are also part of a normal publications cycle. Information deliverables are examined for content accuracy in an inspection process. Inspectors include application developers, customer service persons, human factors team members, and a variety of test and early customer involvement persons.

The writer typically simplifies complex language constructs to obtain a level of ease of translation. At the time of writing, there may also be a prevailing theme within the writing community, such as “minimalist information,” that also affects the compactness of an information component. At the production milestone, the writer's publications join the remainder of the application as a product.

WORKING IN LOOSELY AND CLOSELY-KNIT TEAMS

The writer is typically a member of both closely and loosely-knit groups in a technical community with ongoing channels of communication. Closely-knit groups have the ability to communicate to solve like problems more rapidly the second time a group member encounters them. The group may maintain a unique team practices document to unify solutions. Closely-knit teams are represented by one or more programming development groups that consider the technical writer a team member. Development teams share a core group of writers, the technical writer's other closely-knit group. More loosely-knit groups provide expertise beyond the parochial experience of the local group, which may fail to locate solutions to critical problems (Wellman, p. 180).

Writing teams range from one (a very small team) to perhaps ten writers. They typically have a team leader, who may also write one or more publications and coordinate team practices, schedules, and reporting. Teams focus on conserving their current library and estimating available team resources as migration approaches. On a look-ahead basis, they take steps to:

- Ensure readiness to use a new tool, typically by involving team members in classes or pre-availability trials.

- Evaluate approaching information flow from the products they support, and associated production milestones.
- Locate information, including class manuals and other information.
- Identify expected entry criteria, possibly changing certain library-wide authoring structures to reduce uncertainty in migration. For example, indexing may change from central indexing schemes to chapter-by-chapter indexing.

Teams may have a collective team outlook on new authoring systems, risking only a part of an entire library at a time, or with significant demand, all of a library. Pioneers who make the first attempts describe their experiences to others. Team members whose publications follow in later migration cycles participate in significant verbal discussions of current projects and their critical solution sets. In subsequent publication migrations, their staged learning can recall a valid solution, or find a team member who knows the answer without a sequential search of all available information.

The team expects to pass through an initial period of dependency on tools support, building a stable, standalone ability to troubleshoot most system issues without outside help. In this context, there are instances in which random browsing behavior by isolated individuals occurs to identify problem elements and locate solutions. Just as common in this context is a rapid, concurrent effort by several writers who set a time limit, assign a particular branch of random behavior to each team member, and then rejoin after the time limit to pool the results of their activities.

Team members vary widely in their range of experience. A team may have members who participated in previous migrations with earlier authoring systems, able to gather clues on the duration and loading effects expected in a new migration effort. From memory, migration-wise team members may relate current issues and solutions to valued information source types from an earlier effort.

Team members attend formal classes in using a new authoring system. An informal web of "heard-it-the-hallway" verbal exchanges, as well as informal channels of communication and friendships with critical information resources or experienced writers also generates clues and solutions to team problems with an authoring system. A variety of communication flows occur during business as usual in team settings. Members participate in intra-team and extended team information exchanges, including

communication across sites on topics of mutual interest. Special interest group meetings occur on a monthly or weekly basis, some of which provide requirements and usage feedback to the authoring system tools team.

Formal and informal ties may exist between some group members and knowledgeable authoring system designers, who are capable of providing strategic and timing advice for migration. A centralized team may design, implement, and guide the deployment of an authoring package.

In general, the team that designs and supports the authoring tool will complete a series of formal test exits and a series of evaluations before releasing a new authoring tool. On a more abstract level, an active writing community can be viewed as a pool into which the tool design team releases an agent such as a new authoring system that has a significant, ongoing wave effect. Feedback to the design team helps it detect dissonance on various moving objects (writing projects) in the pool, and also identify alternative and unauthorized ripple sources in the pool, such as the use of an obsolete editing system or a competing translation system.

Prior to general availability, beta testing and preliminary classes introduce members of the user community to new function in the tool. The authoring system team attempts to locate a significant sample of the existing publications for testing. As general availability approaches, estimates of formal classes for all users provide a calendar for education. Also generated are a variety of plans to handle anticipated user questions, error reports, and plans to distribute periodic fixes across the user population. Documentation is prepared, including training manuals, user's guides, instructor's guides, and hints and tips files. An open database allows users to track reported problems, fixes, and future requirements. Other tools-related efforts provide a communication forum for interested parties, including translation, external companies, and vendors that provide parts of the authoring tool package, such as an indexing wizard.

APPENDIX C: QUESTIONNAIRE ON PUBLICATIONS MIGRATION

Thanks for taking the time to answer a few questions. This questionnaire asks about your activities when you migrate a major publication, such as a user's guide. It includes a very short, confidential survey about yourself.

Migration: Please choose the response that best describes your activity:

1. Sufficient information was available in time to decide whether to migrate my publication.
   - ☐ Almost Never
   - ☐ Some of the Time
   - ☐ About Half of the Time
   - ☐ Most of the Time
   - ☐ Almost Always
   - ☐ Other: _____
2. I have enough time to solve migration problems.
   - ☐ Almost Never
   - ☐ Some of the Time
   - ☐ About Half of the Time
   - ☐ Most of the Time
   - ☐ Almost Always
3. My computer and its programs were ready in time to work on migration problems.
   - ☐ No
   - ☐ Somewhat
   - ☐ Almost completely
   - ☐ Completely
4. The very first thing I normally do when I cannot solve a problem in the publication itself is:
   - ☐ Read the errors, change the publication tagging, and re-run the process
   - ☐ Read the hints and tips
   - ☐ Search the authoring system's user's guide or other documentation
   - ☐ Check with a nearby person who might solve the problem
   - ☐ Look online in a problems and requirements database
   - ☐ Ask an expert
   - ☐ Other _____
5. To solve problems, the most valuable information source is:
   - ☐ Reading the error log, changing the publication tagging, and re-running the process
   - ☐ Reading the hints and tips
   - ☐ Searching the authoring system's user's guide
   - ☐ Checking with a nearby person who might solve the problem
   - ☐ Looking online in a problems and requirements database
   - ☐ Asking an expert
   - ☐ Other _____
6. Talking to or consulting with someone near me is the best way to solve a migration problem.
   - ☐ Almost Never
   - ☐ Some of the Time
   - ☐ About Half of the Time
   - ☐ Most of the Time
   - ☐ Almost Always
7. To solve problems with someone else's advice, my most frequent contact is:
   - ☐ Another writer in my group or nearby
   - ☐ A migration expert
   - ☐ The authoring system designer
   - ☐ My previous class instructor
   - ☐ A team leader
   - ☐ Other _____
8. When I searched for information, the most difficult source to use was:
   - ☐ Indexes to information
   - ☐ The authoring system's user's guide
   - ☐ Lotus Notes database problem-solution items
   - ☐ Online problems and requirements database
   - ☐ Hints and tips documents
   - ☐ Other _____

(more, next page)

9. I can help most writers when they migrate their publication.
   - ☐ Almost Never
   - ☐ Some of the Time
   - ☐ About Half of the Time
   - ☐ Most of the Time
   - ☐ Almost Always

10. I reported my publication's significant migration problems to the problems database.
   - ☐ Almost Never
   - ☐ Some of the Time
   - ☐ About Half of the Time
   - ☐ Most of the Time
   - ☐ Almost Always

11. If a problem remains unsolved after everyone's attempts to solve it, I usually (you may check more than one):
   - ☐ Put it in my unsolved problems bucket and use a workaround
   - ☐ Determine if its severity blocks publication
   - ☐ Re-document error information to expert sources
   - ☐ Other: _____

12. When I migrated my publication, the most difficult structure(s) to understand was:
   - ☐ Does not apply – structures were not difficult
   - ☐ Index
   - ☐ Lists or tables
   - ☐ Figures
   - ☐ Cross references and other links
   - ☐ Front matter, running footers, and related problems
   - ☐ Other _____

13. The most significant process change in migration was:
   - ☐ Does not apply – process was not difficult
   - ☐ Validating the document
   - ☐ Master page issues
   - ☐ Locating runtime error information
   - ☐ Adjusting graphics or screen captures
   - ☐ Solving variable or text entities
   - ☐ Other: _____

14. Migration for my publication was:
   - ☐ Reasonably easy
   - ☐ Somewhat difficult
   - ☐ Significantly difficult
   - ☐ Not possible

15. I am confident my next migration effort will be:
   - ☐ Reasonably easy
   - ☐ Somewhat difficult
   - ☐ Significantly difficult
   - ☐ Not possible

About Yourself: Please choose the response that best describes you:

A1) Years I have worked as technical writer are:

☐ 1-5 ☐ 5-10

☐ 10-15 ☐ 15-20

☐ more than 20

A2) My current work role is:

☐ writer ☐ team lead (and writer)

☐ team lead ☐ manager

☐ other _____

(more, next page)

A3) Total number of publications I wrote or significantly altered in the past year are:

☐ 1-2 ☐ 2-4

☐ more than 4

A4) I used a very similar authoring system before migrating my publication.

☐ Yes ☐ No

☐ Other: _____

A5) I worked on a previous migration effort with an earlier authoring system.

☐ Yes ☐ No

☐ Other: _____

A6) Before migration, I changed my publication to fit a migration need.

☐ Yes ☐ No

☐ Other: _____

A7) The size of the publication I have migrated or plan to migrate is approximately:

☐ 50-99 pages ☐ 100-150 pages

☐ 151-250 pages ☐ 250-350 pages

☐ larger than 350 pages

A8) After migration, the approximate time I worked to obtain an error-free (or ignorable errors) publication was approximately:

☐ 1 to 5 hours

☐ 5 to 20 hours

☐ 20 to 40 hours

☐ 40 to 80 hours

☐ more than 80 hours

A9) I re-migrated at least one publication.

☐ Yes ☐ No

☐ Other: _____

A10) I attended a formal class before migrating my publication.

☐ Yes

☐ No

A11) Additional comments:

_____

_____

_____

_____

You have my permission to tally my responses and publish this questionnaire in an anonymous manner.

Signed _____

Date _____