On the Power of Domain-Specific Hypertext Environments[1]

Walt Scacchi
ATRIUM Laboratory
Information and Operations Management Dept.
University of Southern California
Los Angeles, CA 90089-1421
213-740-4782, 213-740-8494 (fax)

Abstract

What is the potential power of hypertext technology? This article examines this question and outlines the answer by focussing attention to a domaim-specific view of hypertext environments. I first define what domain-specific hypertext environments (DSHE) represent. Next, I examine DSHE for the domains of encyclopedic and classical studies, creative writing and interactive fiction, journal and book publishing, insurance policy management, and software engineering. Then I describe in more detail the structure of information to evolve within a DSHE for software engineering in terms of document products, processing tasks and mechanisms, and workplace attributes. In turn, this examination provides the basis for identifying seven dimensions along which the power of DSHE can be defined, experienced, and accumulated. I also address the organizational costs that may be borne to realize this power. I conclude with observations as to the source of DSHE power as well as identifying topics for further investigation.[2]

---

[1]This is a revised and expanded version of the paper appearing in the *J. Amer. Soc. Info. Sci.*, Vol. 40(3), pp. 183-191, (May 1989).

[2]Acknowledgements: This work is supported as part of the System Factory Project at USC. Sponsors for this work included Bell Communications Research, Eastman Kodak Inc., and Hughes Aircraft, Radar Systems Group. However, no endorsements are implied. Pankaj Garg played a key role as a designer and implementor of the CASE hypertext mechanisms referenced in this article. In addition, a number of people contributed to the ideas or system development work on which this report is based including Salah Bendifallah, Sean Burke, Joe Chen, Song Choi, George Drapeau, David Faucette, Abdulaziz Jazzar, Peiwei Mi, and John Noll. All of these contributions are greatly appreciated.

1. Introduction

What is the power of hypertext[3]? Power in other contexts usually refers to the ability of some entity or agent to affect the behavior of another, or to achieve advantage over another in ways that the other cannot avoid. Following this, we might expect that as a *technology*, hypertext systems offer relative advantage over alternatives to automated text/document information systems including conventional word processing systems, file systems, and data base management systems. In this way, hypertext systems offer a degree of information processing power that enables new kinds of applications, much like the advent of so-called expert system shells[4] enabled the creation of expert applications for domains such as computer configuration and pathological diagnosis. Similarly, as a *medium for writing and reading electronic documents* hypertext systems allow the redefinition of the structure and content of documents which alter the constraints and opportunities for conveying information in contrast to the linear print medium. Thus, the power of hypertext in whatever form it exists is subtle and incremental: it is potential rather than kinetic. However, such power can accumulate in different application domains in the long run if effectively mobilized and integrated with domain-specific tasks, processing mechanisms and workflow.

The power of hypertext technology in different application domains can be realized in a number of ways. In particular, technology that supports a unified view of the hypertext documents, production processes, processing mechanisms, and settings of use appears to offer the greatest power at this time. To see this, I first define what a domain-specific hypertext environment (DSHE) is in the following section. Next, I examine five application domains for hypertext environments in order to identify the supporting tools and document production/consumption techniques associated with the processing of information content within each domain. The focus then shifts to describe the structure of information to evolve within a DSHE in terms of product, task, and setting information for the domain of computer-aided software engineering (CASE) as used in the System Factory Project at USC [36]. Together, these sections then set the final stage for a discussion of the power of DSHE. In particular, I identify seven ways the power of DSHE can be realized, followed by an examination of the costs that are borne in order to realize this power.

2. Domain-Specific Hypertext Environments (DSHE)

Until about 1985, the number of automated hypertext systems could be easily counted. Since then, interest in the development and use of hypertext systems has exploded. Conklin reviews most of what might be called the first generation of hypertext systems in his survey published in 1987 [7]. He identifies a dozen features that characterize the 18 hypertext systems he surveys. The *Hypertext '87, Computer-Supported Cooperative Work, Office Information Systems*, and *Hypermedia '88* conferences have been showcases for the more recent progress in the research and development of hypertext systems. However, few of the new hypertext systems provide support for all of the features Conklin identifies, as we see below.

---

[3]I use the term "hypertext" to be synonymous with "hypermedia" and other similar terms that denote the non-linear representation of interrelated textual, graphic, filmic, or auditory information.

[4]In other places, such shells are considered "domain-independent", but when instantiated with rules about patterns among application domain data, the expert system application is considered "domain-specific."

2.1. Domain-Specific Hypertext

Many of the published descriptions of hypertext systems make casual reference to the contents or non-linear structures of their hypertext bases. Instead, the focus is usually on the computational mechanisms through which a subject hypertext is processed. This is in most cases reasonable, since the research publications seek to describe innovative technologies that might be applied to many different hypertext application domains. However, our focus here is to examine on more of an equal footing how the content and structure of application domains goes along with the use of different processing mechanisms and technologies. This is a way of saying that pre-college students, humanities scholars, technical professionals, periodical publishers, insurance actuaries, manufactured product designers, and software engineers all process different kinds of textual information, in different settings, with different processing needs and infrastructural support. Alternatively, this is another way of saying what has already been realized elsewhere: choose information structures and processing mechanisms that fit the users' skills, tasks, and resources in the application domain. This choice is preferred since if the processing mechanisms are inappropriate for the application domain, they will fit poorly causing the system to be more troublesome to use, more costly to maintain, more likely to be resisted, and most likely to fail [20, 16, 3]. Thus, what makes a hypertext domain-specific is the codification and configuration of the application domain's agents, objects, attributes, relations, constraints, tasks/processes, transaction event rules, mechanisms, and resources into the information structures that are woven together to form a hypertext.

2.2. Hypertext Environments

Environments represent another new direction for investigating the information processing capabilities of hypertext systems. Specifically, *hypertext environments* represent the integration of hypertext features such as those described in Conklin's survey (hierarchical structures, non-hierarchical links, typed links, procedural attachment, concurrent multi-users, graphic browsing, etc.) with automated tools, processing tasks, and application workplaces. These environments--coupled ensembles of automated systems and techniques for their use by people with skills and interests appropriate for their application domain--essentially utilize the hypertext systems to organize or manage large data bases of non-linearly structured text. As we see next, the combination of domain-specific hypertext information structures with application processing environments, leads us to a more *infrastructural technology* for different kinds of application domains which I call *domain-specific hypertext environments*.

3. Application Domains for Hypertext Environments

Our strategy is to examine five application domains for hypertext environments and identify salient processing mechanisms and information structures appropriate for each content domain. The application domains to be covered include encyclopedic and classical studies, creative writing and interactive fiction, journal publication, insurance policy processing, and CASE. This examination serves to identify the various kinds of processing mechanisms for different tasks and settings being used in domain-specific hypertext environments.[5] This in turn proceeds and set the stage for a more detailed discussion of the structure of information employed in a hypertext environment for the domain of CASE in Section 4.

[5]These mechanisms are highlighted in italics.

3.1. Encyclopedic and Classical Studies

Hypertext systems that support electronic encyclopedia can primarily be treated as information retrieval systems. Here the encyclopedic hypertext already exists, perhaps even with links already "wired". The user is primarily a reader who queries and browses the hypertext in search of encyclopedic information. The user does not act as a writer, linker, or editor of these hypertexts. A number of example applications have appeared, and a sample set is described next.

Weyer was among the first to describe to development of a prototype electronic encyclopedia for historical studies [40]. He employed a version of Grolier's Electronic Encyclopedia[6] to demonstrate the basic concepts, mechanisms, and potential of an electronic encyclopedia. In another effort, elementary school students using the Hyperties system from the University of Maryland [25], searched the Grolier's hypertextual encyclopedia to locate information with queries formed with interest words and Boolean connectives (AND, OR, NOT). An accompanying empirical study demonstrated the children's tendency to use low cognitive load browsing strategies [26]; that is, the students did not (or were not able to) put much effort into forming succinct queries for finding the information they sought. But user query retrievals in Hyperties are staged as *alphabetically sorted lists of article titles* within the proximity of the query fragment. Thus, the staged *proximal retrievals to weakly formed queries* compensated for some of the formal search inadequacies of these children.

Elsewhere, the ongoing revision and update of the tremendous volume of medical knowledge is another area where hypertext encyclopediae are being researched and developed. Given the substantial volume of evolving medical knowledge, health care providers often seek to find information pertaining to the latest research findings regarding selected illnesses, diagnosis techniques, and therapeutic interventions. Frisse describes an effort at Washington University to utilize information about the structure of a dynamic medical therapeutics textbook to propagate weights to neighboring texts to produce a ranked list of potential retrieval starting points for browsing [12]. Similarly, Rada and Martin seek to connect heterogeneous information from different medical information systems where terminological differences pervade the user community [32]. Accordingly, they discuss their experience in connecting diagnosis and treatment information, patient records, and current medical bibliographies through *augmenting and merging the concept thesauri* used in each system for more effective medical information search and retrieval.

At University of Texas in Austin, Simmons has taken the three-volume Handbook of Articifical Intelligence research survey, and augmented it with semantic networks that identify linked relations between concepts appearing in the text [38]. At MCC in Austin, the CYC project headed by Doug Lenat seeks to develop a large-scale intelligent system with encyclopedic knowledge [22]. Their plan is to carefully represent approximately 400 articles from a one-volume desk encyclopedia, as well as a number of hueristics for interpreting this knowledge within the same knowledge hypertext base. Their objective is to *represent the conceptual contents of their encyclopedic subjects within a knowledge lattice*, so that information items can be not just retrieved, but also reasoned about through the use of *pattern-directed inferencing mechanisms*. Thus in contrast to the "conventional" electronic encyclopediae previously noted, this long-term effort is directed at developing and providing a substantial *dictionary of objects with characteristic attributes, values, relations, constraints, and processing hueristics* to convey the meaning of

[6]This collection is available on CD-ROM occupies 20 volumes in printed form, while the subsequent hypertext version consists of 60M bytes of text and 50Mbytes of indices (including pointers to each occurence of every word).

informational concepts appearing in the articles.

Hypertext also holds promise for expanded studies of the languages, literature, and semiotics through the written words of ancient and modern cultures. The Thesaurus Linguae Graecae Project at UC Irvine and the Perseus Project at Harvard are entering the entire opus of available ancient Greek literature and associated images into an online accesible form, a text base of size 100M bytes. Such a text base is intended to support both traditional and radically new kinds of scholarly studies of interest to historians, philosophers, and literary critics. Such studies of the History of Greek Religion, Aristotle, the Trojan Women and the like seek *to identify and interrelate (link) key literary passages, rhetorical structures, and themata evolution* found in the literature [8, 21]. Some of these studies further employ automated processing mechanisms such as *concordance preparation, cross-reference distributions, and co-word/co-term frequency analysis* for quantitative support [33]. Hypertext systems in use at Brown University have been demonstrated to provide highly interactive navigation, viewing, and linking mechanisms for classical studies applications. However, such systems provide a non-traditional view of classic literature which makes citation to particular textual passages on designated pages in a reference edition of the classic text unconventional. Thus, Crane argues that hypertext systems for scholarly studies in the humanties require conventions for linking and classifying text passages within standardized text base formats before widespread adoption and routine use occur [8].

3.2. Creative Writing and Interactive Fiction

The development of computer-supported "writers' workbenches" seeks to employ a variety of automated mechanisms to support the production of literary works. The visionary potential of this emerging literary medium has been foretold by Ted Nelson [29] among others. The basic idea is to combine conventional text processing systems--*text editors, outline processors, online dictionaries and thesauri, spelling checkers, document formatters, page layout utilities*, and the like--with hypertext mechanisms similar to those used for literary studies to create a new electronic literature. This literature would in turn be distributed using new media such as computer networks, electronic bulletin boards, CD-ROMs, etc. An automated writing environment under development at University of North Carolina to support the technical writing needs of professionals who work within a network of computer workstations is an example effort along these lines [39].

In addition to (re)structuring and streamlining the writing and publication process, writers also seek to *access online reference services and catalog/archive holdings* maintained by libraries and similar institutions.[7] The Jefferson Project at USC seeks to develop a writing environment for use by individual students in introductory humanties and social science courses that incorporate access to these library services [5]. Their purpose is to improve basic "freshman writing" skills of these students as well as increasing the students awareness and routine use of online library services for research and writing purposes.

Finally, also along the lines envisioned by Ted Nelson, the emergence of interactive writing environments coincides with the appearance of interactive fiction and story presentation. Although first appearing within computerized adventure games--of which many are also hypertext applications--interactive fiction is now appearing as *a writing and reading medium for serious novels and other*

[7]These latter capabilities are increasingly an essential part of the workplace infrastructure of librarians as well.

experimental literature. The migration to the electronic medium removes the restrictions imposed by the linear printed medium and therefore allows new experiments in literary structure [4]. Such experiments might eventually include reflexive narrative forms such as "war stories" that tell of hypertext system malfunctions, their cooperative diagnosis, repair or modification (cf. [30]). This reflexive literature in turn might aid in constructing deeper understandings of how hypertexts are created, and how the supporting hypertext environment mitigates their evolution.

3.3. Journal and Book Publication

Professional journals are often produced and disseminated at regular intervals. Mass audience periodicals may be specialized into *published versions or editions* for distinct geographic regions, target audiences, etc., possibly even for parallel mass-production of printed copies. In either case, editorial entities within a journal--articles, editorial masthead, editorial page, table of contents, special event and new product announcements, photographic images and graphic displays, classified advertisements, information for authors, etc.--all have different *space, typeface, and layout requirements within budgeted allotments*. Further, each of these entities is subject to editorial content policies (for what is acceptable or isn't), standardized formats, style guidelines for presenting ordered information content. However, one benefit that such an ordered information content accomodates is the *parallel authorship and review of multiple articles* and the like. This of course aids in the timely production of sufficient editorial content to meet budgeted allotments and distribution schedules.

Selection of entities to include for publication involves the review and judgements of an editorial panel, as will decisions for revisions requested of authors, including length or space restrictions. The degree and extent of editorial staff participation will vary depending on the type of periodical--archival journal, magazine, newsletter, newspaper, or electronic data base.

The primary functions of journal or book publication activity include (a) manuscript storage and retrieval, (b) document binding, and (c) job management [23]. The document storage and retrieval system must manage all information necessary to produce the document. This includes publication style rules (standard document format markup), spelling dictionaries, contracts, specifications, production schedules, and accounting data. Further, documents will be revised, and so must be maintained as a sequence of multiple parallel versions. When several people work on a publications in parallel, there is a need to check that all the parts are appropriately numbered, indexed, cross referenced in a consistent manner. *Document binding*--the composition of document parts into a complete publishable assembly--provides automatic numbering and cross referencing across all parts in a consistent manner. Finally, job management supports the organizing of document production. This includes support for publication cost estimation, tracking of work in progress, production schedules, and maintenance of contracts and accounts.

Last, the emergence of automated document production on a wide-scale through the advent of personal computers and desktop publishing packages, together with the growing organizational interconnectivity of computer networks, is giving rise to *electronic journals*.[8] In electronic journals, documents for publication and dissemination are composed, formatted, submitted, revised, distributed,

[8]The July 1988 issue of the COMMUNICATIONS OF THE ACM is one of the first professional journals serving the Computer Science community to be widely distributed as an electronic hypertext edition on diskettes.

and criticized online [34]. *Electronic bulletin boards* are the most widespread form of this embryonic literary medium. In addition, a small but growing number of book publication companies are beginning to establish electronic communication channels such as those for handling book manuscripts destined for printed production. Similarly, *electronic data bases* of bibliographic citations, reference/service manuals, journal abstracts, images, empirical data sets and the like are also beginning to appear. Finally, we should soon expect to see *electronic knowledge bases* of attributed and interrelated object lattices and rule bases for different domains being published, extended, and recirculated.

3.4. Insurance Policy Management

It seems that the use of information systems in insurance organizations is one of the more widely studied forms of information work [20, 17, 10, 6]. Perhaps, this is because the insurance industry is one of the most computerized of all. This may be due to the fact that most insurance office work entails processing and verification of a high-volume of *form-based transactions*, such insurance policy premium payins and claim payouts. In turn, these transactions are entered into *data base management systems*, that can be accessed through *fourth generation languages, report generators, spreadsheet and statistical packages* in support of acturial and portfolio investment analysis. Much of this information work is often considered routine, so that *automated procedural scripts for information and document flow through the organization* can be developed and operationalized [37].

However, insurance offices, like other information processing offices, are open systems [18, 17]. As an open system, various kinds of information forms are processed by different actors including clerks, actuaries, data base administrators, document controllers, portfolio investors, work supervisors, clients, insurance agents, etc. Accordingly, the textual information or documents embed multiple simultaneous viewpoints for coding, review, and interpretation. This means that no piece of insurance form information is simple, nor will the *organizational interfaces* to the insurance information system be singular or simple [6, 24]. Similarly, this means that not all viewpoints will be represented completely in any one representation. Subsequently, insurance information systems must deal with multiple competing, possibly irreconcilable codings and data processing requirements across different actors [17]. Resolving the conflicts and anomalies that subsequently emerge hinges on developing local closures to these information coordination problems. Elsewhere, Dunham and associates describe their experience in evaluating the experimental use of a *coordination system*[9] to discuss and seek (or force) resolution to similar kinds of problems [10].

3.5. Computer-Aided Software Engineering

Software engineering is concerned with the development, use, and evolution of large software system applications and environments. "Large" refers to the fact that such software systems are often developed by teams of 10-100 engineers who work on development projects that take months to years to complete, and whose resultant programs are usually in the range of 50,000 to 500,000 source code statements (between 1000 to 10,000 single-spaced pages). Further, due to the complexity of these programs, it is widespread practice to also develop a series of *interrelated multi-version documents* that describe the life

[9]The coordination system they used is intended to structure electronic communications of conversations for action (what to do next) and conversations for possibilities (brainstorming for deciding what to do next). See [41] for additional details on coordination systems.

cycle--the requirements, functional specifications, designs, test plans, implementations, user manuals, and maintenance guides--for the system being developed. These documents often represent 5-10 times the volume of the source code. As such, it is more efficient for the structure and to some extent the content of these *documents are standardized* in order to facilitate *parallel/multi-person development, review and quality assurance*.

Software systems are documented using fully structured descriptions (functional specifications, designs, and source code) *whose syntax and semantics can be formally defined and automatically analyzed*, and weakly structured descriptions (narrative requirements, user manuals, maintenance guides) whose content can text-processed and understood by people, but may be ambiguous and incomplete [13]. As such, automated mechanisms must be provided to *identify and trace relationships across multiple semi-structured descriptions* of the same system in order to *configure, validate, and maintain the consistency of interrelated software descriptions* as they evolve [27, 28].

Large software engineering projects produce encyclopedic volumes of semi-structured and interrelated descriptions. As such, the production of system life cycle documentation can represent a substantial fraction of the system's development cost. However, such cost is often necessary since the large scale of the system development effort often implies that the system's documentation will be the focal *medium for coordinating the engineering tasks and information flows among developers working at different times and places*. Software development requires the substantial coordination of diverse specialists, automated tools, multiple system descriptions, and other organizational resources. *Electronic mail and bulletin boards* are increasingly essential, but usually lack *knowledge of the software development products, processes, workplaces, and their interrelationships*. However, a new generation of *knowledge based software engineering technologies* are beginning to appear that represent and manipulate these kinds of knowledge [2, 14]. Similarly, to increase the rate and quality of software system production, other technologies including *intelligent message management systems* [19, 24] and *online catalogs of reusable software components* [11] are being investigated.

Finally, it is interesting to observe that the kinds of information structures and processing mechanisms used for a CASE hypertext are generally a superset of those appearing in the other domains described above. This of course does not imply such a covering with respect to the document content or professional expertise of the other domains. Nonetheless, we can use this domain to examine how the structure of information in a CASE hypertext is organized, and how the attendant processing mechanisms are organized into as comprehensive DSHE in the sections that follow.

4. The Structure of Information in CASE

Within the domain of CASE, documentation is a primary record of engineering activities. The documents that are produced are interrelated through various kinds of *links*. Documents are structured into development folders or *partitions* associated with particular *system/subsystem projects and associated staff*. Each partition may contain a standard set of chapters, sections, and subsections which store and organize software product descriptions. These can be represented as *forms* which organize collections of related descriptions as filled-in *templates*. These templates are the nodes of the CASE hypertext. Further, as the volume of documents produced is often quite large, then additional document structuring mechanisms are included to provide views that cut across standard document structures. Such views constitute interlinked document *compositions*. These compositions are used for selecting

partitions, forms, or templates which can be structured and related following standard linking schemes or ad hoc, user-defined schemes. As such, compositions can provide cross-cutting views of software documents that are useful for distribution, quality assurance review, browsing, or simply printing an individual's project document contributions. Next, the development, use, and evolution of large software system *descriptions can be explicitly organized and represented in terms of the processes and settings* that engineer them. Last, the structure of information within the domain of CASE defined in these terms thus accomodates a rigorous formalization of the underlying abstract relationships that are embedded in a CASE hypertext [15].

In the remainder of this section, I elaborate the information structures highlighted in the preceding paragraph based upon our experiences with a CASE hypertext environment in the System Factory Project at USC [36, 13, 14].

4.1. CASE Hypertext Information Structures

Links: Links represent typed relations that identify specific associations within or between software descriptions. Keywords and annotations are special kinds of links that are supported with predefined processing mechanisms. Keywords and user-defined links may serve as simple pointers for tracing the occurence of designated terms across and between CASE documents. Annotations represent hierarchically linked narratives that further describe the designated term or object to indicate its meaning, explanation of its use, or decisions pertaining to its definition and purpose. Users can also define links as semantic relations which in turn can be static or operational. Static links denote a simple logical relation between linked software descriptions, whereas operational links imply some standard or user-defined computational processing when linked items are visited or modified. Overall, as all links indicate some kind of relation, they can be stored and managed by a relational data base management system (rdbms) [13, 28]. In this way, linked software object descriptions can be indexed, browsed, and relationally queried through the mechanisms of the rdbms including query processors, report generators, pattern matchers, and fourth generation languages.

Basic Templates: BTs denote semi-structured object descriptions of various length and substructure [24, 13]. Semi-structured indicates that object descriptions are structured to some degree--that is, all objects possess descriptive attributes which characterize the structure, content, and purpose of the object. The range of possible values these attributes take on may be either completely defined (e.g., by formal language specification or enumeration) or weakly defined. However, the presence and formalization of these attributes determines the degree to which the descriptions can be parsed, analyzed, and interpreted by processing mechanisms such as language-directed editors, application generators, specification and design analyzers, and compilers to determine their consistency and completeness [13, 36]. Subsequently, BTs and forms provide a common, persistently typed substrate for representing and managing software descriptions in ways that facilitate engineering tool integration.

Forms: In simple terms, forms represent aggregations of BTs. For example, a BT might be used to denote the sections and subsections within the form of the Requirements Analysis document for a project. More generally, forms can be used to define the structure and type of software system information organized as a collection of life cycle document structures that can be standardized and reused across many project teams of software engineers. This standardization can therefore support parallel authorship of multiple software documents when the interrelationships between documents are made explicit *a priori*.

This arrangement thus creates opportunity for improving the coordination and productivity of the system development effort.

Compositions: These are threaded networks of forms and/or BTs that denote some user-defined association. For example, configurations can be composed for printing only the sections or subsections of an article modified by an author since the last revision. In multi-authored documents such as large software systems, this is a particularly useful feature that facilitates coordination and project status monitoring. Similarly, compositions allow linked software descriptions to be managed, viewed, or evolved in ways that approach the rigors of software configuration management [27, 28].

Partitions for Projects and Teams: Partitions provide a structuring mechanism whereby collection of hierarchical forms of BTs can be composed, standardized, and shared by classes of users. For example, in the System Factory Project, partitions can organized by work group, project, or project site [36]. Partitions represent contexts [9] for structuring access to a hypertext of software object descriptions. Thus individual form or BT instances cannot appear in more than one partition class. However, people working within a partition may browse, link, or compose across multiple partitions. This supports the assignment of standard BT/form processing mechanisms to designated types of software descriptions [13, 14]. Similarly, it helps to minimize getting lost in a software information hyperspace since a user always works within a known context with defined forms and BTs.

CASE Processes: The preceding information structures are used to describe the organization of software object descriptions as product documents that cover the software life cycle. However, it is oftentimes the case that the software life cycle *process*--the sequence of tasks and associated processing mechanisms that create and manipulate software descriptions--is itself subject to alternative definitions and task compositions. Thus, the tasks which software engineers perform should also be developed and organized in ways analogous to other software object descriptions.

The information structures for specifying CASE tasks need to describe the sequences of processing mechanism invocations that simplify the routine manipulation and interchange of software documents. Consider source code program documents for example: Here the processing mechanisms for manipulating source code descriptions include editors, compilers, debuggers, program linkers and loaders, as well as formatters for displaying highlighted ("pretty-printed") program listings. The structure of the task of developing a working program usually requires the use of each of these mechanisms. The sequence of their invocation is mostly non-deterministic and non-procedural, but easily tracked by individual software engineers for small programs. However, if the programs being developed are large, built by teams developing multi-version program components according to an elaborate life cycle engineering methodology [36, 13], then the program writing task becomes complex and costly if not well-coordinated.

The description of software object/document processing tasks can be decomposed at many levels of detail and interrelationship. First, there must be support for specifying the task of task specification. This *meta-task* description is needed for organizing tasks, specifying their interrelationships, and assigning them to appropriate partitions. Meta-task descriptions must be maintained since the content and structure of task descriptions may evolve in open system workplaces in unexpected ways [18, 16, 17, 3]. Second, there are two classes of document tasks for which many subtypes can be identified. These are *management tasks* of project administrators, and *engineering tasks* of technical project staff.

Management tasks focus on activities such as breaking down system development projects into subsystems, assigning staff and processing mechanisms, scheduling and budgeting subsystem description development, monitoring project progress and productivity, acquiring and maintaining an adequate supply of staff and computing resources, assuring the quality of the integrated and validated final product document assembly, and redoing any of these when things breakdown, foul-up, or when external conditions dictate [35]. Engineering tasks are performed by individual or small groups of engineers who create, manipulate, and interchange component documents assigned to them among other things [35, 3]. These tasks include analyzing system requirements, developing system functional specifications and designs, program development and test, maintaing existing systems, and so forth.

We should recognize that the management and engineering tasks are interrelated in many ways. For example, in order to validate that a system is fully operational and ready for delivery, the tasks of assuring that (a) all the right versions of (b) all of the right system components were selected and (c) composed in the right order and (d) appropriately tested. Each of these subtasks in turn requires management subtasking. For example, each of these subtasks assumes that the required source code programs were developed and run through the assigned set of source code processing mechanisms. However, they also assume the existence of either an automated processing mechanism or (manual) administrative mechanism for checking to see that the components fit together in a consistent and complete manner. When the number of components is in the hundreds or thousands, each existing in one or more versions, each potentially fitting into many alternative configurations, and each having one or more sets of data for testing, then a combinatorial explosion of alternatives must be both engineered, configured and managed. The complexity of the emerging system begs for coordination and automation rules that simplify and maintain a closed system description whose consistency and completeness can be directly assessed. But the complexity of system artifacts emerging from the development effort within an embedding environment requires that the successful performance of the management and engineering tasks will be interdependent [20, 35, 16, 17, 3].

All tasks, regardless of level of description, describe a potentially *non-linear sequence of actions*[10] These actions affect some concrete or abstract transformation of a software BT, form, composition or partition. For example, in the task of implementing a software system design as a program, the creation of a successfully compilable program component is a necessary action. Other actions in the sequence include (a) understanding the software design, (b) developing a program implementation strategy, (c) establishing which processing mechanisms are available, (d) debugging an anomalous program behavior, etc.

In turn, all actions can be further decomposed into *primitive actions*. These represent commands issued in dialogues between an individual and the current processing mechanism in use. For example, understanding the software design can entail (a) browsing a design document, (b) following embedded cross-reference links, (c) querying the design dictionary for information about a particular software object, (d) tracing design details back to the originating system requirements, (e) searching for an existing program component which performs a similarly designed computation, etc. As before, the sequences of primitive actions are also non-linear [14].

Overall, primitive actions, actions, tasks, and meta-tasks must be articulated, aligned, and coalesced

[10]In other places, these are called *task chains* [20, 16, 3].

vis-a-vis one another when a system of software life cycle documents are produced. This can become an emerging, open-ended activity that we seek to structure, control, and close so that we can assure its consistency and completeness. As such, these software process descriptions at varying levels of detail can not be guaranteed *a priori* to be consistent, complete, or correct under all possible performance circumstances. Hence the need arises for viewing the creation and manipulation of CASE process tasks as structured descriptions that should be managed as domain-specific hypertext [14].

Organizational Settings for CASE: As previously noted, the workplaces for using domain-specific hypertext include people with different skills, processing mechanisms, and various shares of organizational resources. Thus, this information should also be described and linked into the CASE hypertext work environment. I will limit our focus here to the project participants.

People in CASE projects are typically divided in terms of management, engineering, customer, and maintainer skills. Further decompositions (or sub-types) of each can be identified.

For management skills, we sometimes see specialists for process management, quality assurance, scheduling and budgeting, and configuration management. On small projects, these skills might be possessed and put into practice by a single person. However, on large projects divided into many subsystems and small group teams, then these management skills will often be distributed across a few key people. Similarly, on the engineering side, there may be specialists for each software life cycle activity, or some subset of life cycle activities. For example, the principal software engineer and a small group of trusted senior engineers may be primarily responsible for defining the overall software system architecture, major subsystems, as well as specifying their operational requirements. These specialists however will usually not be found with the additional responsibilities of performing the detail design and implementation of source code modules.[11] Similarly, the majority of system programmers will not have the skills for producing a viable set of system requirements or overall design.

However, there are also other people whose participation and skills can affect a CASE project. These would be the end-users and clients who typically will define the systems' general requirements, and the system's maintainers. If the customers have a history of experience in specifying or using diverse software applications, their skills in specifying system requirements will be different from someone acquiring or using an unfamiliar application technology. These latter type of customers may as a result of their inexperience or uncertain knowledge of the exigencies of the new technology might then end up frequently changing the specification of their requirements. However, it is widely recognized that changing system requirements is one of the most frequent causes of projects being late, over budget, or otherwise a technical failure or maintenance nightmare. This brings us to another class of project participants, software system maintainers.

For a large system, maintenance activities go on for years. Maintenace tasks--adding functional enhancements, resolving anomalies, tuning system performance, migrating the system to ohter environments--are divided among staff according to subsystem responsibilities. Software maintainers for large systems are generally not the same people who originally developed the system. As such, their knowledge of the system's operational behavior, function, and structure must be derived from either the

[11]One possible exception to this might arise for modules designated as critical to overall system performance or integration. But on most projects this will be uncommon.

existing software object descriptions, informal conversations with others, or direct experience. One frequent problem here is that the existing descriptions (source code versus system design) are typically inconsistent, incomplete, or otherwise inadequate [3]. Thus, software maintainers are often at quite a disadvanage in keeping operational systems viable, unless there are automated maintenance support systems to assist them [27, 28], or else unless the available software object descriptions were engineered and maintained throughout the project up to this point. As a result, the success of the maintainers tasks depends on their ability to accomodate or negotiate alternative definitions of their tasks or work arrangements [3].

As such, I can identify four classes of participants for CASE projects--managers, engineers, customers, and maintainers--which can be further decomposed into task specialtists that are interrelated and interdependent. However, the task-skill combinations are constrained by the limits of the organizational resources and automated processing mechanisms allocated to their interlinked project partitions. Thus, the structure, content, and flow of project participants' task-skill organization should be described and managed as evolving software object descriptions [14].

Last, we should also delineate the structure of processing mechanisms and organizational resources in terms of their compositions or class-subclass hierarchies as well as their links to other setting, process, and software object description structures. These are discussed elsewhere [36, 14].

5. The Power of Domain-Specific Hypertext Technology

Given this examination of different DSHE, I can identify seven dimensions along which the potential power of DSHE can be realized. In addition, we should recognize the the acquisition, exercise, and accumlation of such power comes with varying costs. Similarly, we recognize that it is often the case for information systems that their benefits are overestimated, while their costs are underestimated [20]. Thus, I turn to examine these benefits and costs.

Productivity enhancement: DSHE pose many opportunities for enhancing the productivity of their users in different domains. For example, DSHE support explicit articulation of an application domain in terms of hypertext content structures. Further, DSHE can also organize processing mechanisms, processing tasks and settings. These provide a framework for organizing information processing work in a systematic rather than ad hoc manner. Standardized document contents and structures support the cataloging and re-use of information across different application instances. Formalization of the hypertext documents, processes and settings provides a basis for use of knowledge base processing mechanisms. Hypertext can serve as a integrating medium for coordinating the interconnections and interfaces among multi-author documents. Similarly, DHSE can utilize cooperation and composition mechanisms that allow mutiple authors to work in parallel, as well as across different document versions.

Improved quality of document production and consumption: The "look and feel" qualities of documents increase, especially as both the writers and readers acquire skills and perform tasks that increasingly necessitate the use of hypertext environments in their work. This may be especially true in domains where hypertext technology serves as a groupwork support system where such systems have been found to increase the participation of "writers" and increase the quality of decisions for action by "readers" [1]. Similarly, standardized content structures streamline the composition and binding of multi-contribution documents. In turn, interactive readers of such documents can acquire skills or processing mechanisms that exploit available structured (or restructurable) contents. Finally, when document

contents and structures can be formalized, then automated mechanisms can be employed to check on the consistency and completeness of domain documents as they are created or modified, and then prevent or report errors when they are detected.

Hypertext as a medium for coordinating work: When DSHE allow explicit representation and modelling of the relationships between documents, processes, and workplace, then automated mechanisms that integrate, manage, communicate, transform, or rapidly recompose product-task-setting information can be introduced. Similarly, when combined with an intelligent message/mail management system, then a DSHE can support mechanisms for allocating, collecting, and integrating documents that are interchanged among collaborating workers, as well as tracking, triggering, and filtering documents whose production was scheduled and budgetted.

Active system participation: As hypertext systems become increasingly intelligent and knowledge based, such systems can be empowered to act not merely as passive information repositories, but also active information gatherers. When knowledge pertaining to the information structures in some document can be developed with automated mechanisms, a DSHE begins to move beyond passive storage and user-directed retrieval capabilities. When information pertaining to knowledge of the domain-specific document structures and contents, the processes of their production and use, and the workplaces where such documents and processes are articulated can be acquired through interaction with system users--through system-directed queries, inferential analysis of gathered information, and reuse of analogous knowledge bases--then DSHE will move toward becoming active agents capable of producing and using hypertextual information content and structure [14].

Reinforcing biases and structured perspectives: Structured views of large bodies of textual information enable the the view developer to try to focus the attention of the reader to particular "points of interest" or connections within the text. These view structures may be intended to direct or persuade the reader of the appropriateness of the viewmaker's interpretation of the importance or meaning of the underlying text.[12] Therefore, the viewmaker can impose a set of standard ways of producing and consuming information structures explicitly represented in a application hypertext, and thus encourage "shared" interpretations of its meaning.

Enrichment of information processing skill and work: For some people, the use of DSHE for text searching, browsing, query processing and other actions intrinsically motivating or stimulating [1, 24]. When groups of these people use DSHE in their collective work, they may then learn how to more effectively coordinate the production of multi-author documents. Further, the aquisition of hypertext creation and manipulation skills will help create new career and professional job opportunities for those people who first seek to master such systems within their domain of expertise.

Competitive advantage through technological innovation: Employing hypertext systems in particular application domains may realize a competitive advantage to its organizational users [31, 42]. For example, the use of new technologies in academic disciplines to study either established research problems in alternative ways or to open up new problem domains is a frequent subject of scholarly research publication. Researchers who have access to the new technologies can realize a new

[12]This is familiar to readers of used textbooks that are marked with highlighter pens. It is often easy to have one's attention drawn to the marked passages, and accordingly difficult to ignore or overlook these same markings.

opportunity to extend their line of research work by publishing articles in professional conferences or archival journals that describe their experience in developing, using, evolving, or evaluating hypertext mechanisms. Most of the scholarly publication outlets are highly competitive regarding articles accepted for publication. The mere use of new technology does not guarantee acceptance of subsequent research publication. But the opportunity to innovate with hypertext environments appropriate for academic or commercial domains exists for those capable of exploiting its technical and professional potential.

5.1. The Cost of Power: The Hypertext Package

DSHE represent more than merely a set of programs. Instead, DSHE represent a set of requirements for combinations of computing hardware, software, communications interfaces, mass storage systems, as well as people with skills to develop, use, and evolve hypertext applications, and policies, procedures, budgets and schedules for efficient utilization of DSHE in regular information work activities. Unless these requirements are adequately satisfied and sustained, then the power of DSHE wiil not be realized to its full potential. Thus, I refer to these organizational requirements as *the hypertext package* (cf. [20]).

The potential power of DSHE comes at a price. The cost can be expressed or evaluated in terms of the technological and social resources that get consumed, allocated, or displaced to accomodate the use of DSHE. However, a favorable power-cost ratio can more likely be realized when a DSHE is managed as a package with implicit trade-offs, rather than as application programs with instruction manuals.

Consider the following trade-offs: First, the potential for productivity enhancement can emerge from articulate information work structures and integration mechanisms that enable higher level of parallel activities. However, this could decrease "hidden" pockets of discretionary (slack) staff resources which are considered inefficient but are potentially very effective in mitigating workflow failures or breakdowns [3]. Second, the potential of hypertext systems as a coordination mechanism depends on the continuity of staff participation in its use. What happens to people not online, or who drop offline for unpredictable periods? Coordination of work in these situations must occur within the workplace in an ad hoc manner to be realized. Third, skill, work, and professional enrichment may emerge from mastery of hypertext production and manipulation. However, as is the situation with text processing systems, many technical professionals have broadened their skill base downward through the assimilation of more clerical responsibilities. Similarly, many clerks and typists have broaden their skills upward by acquiring computing specialist skills needed for the correct and sustained operation of office information systems. For clerical staff, their upward skill mobility can translate into promotion, salary increase, new career opportunities, or increased workload. On the other hand, for technical professionals, their expanded skills enable increased workloads and more control over document production, but little or no new upward mobility. Will hypertext follow text processing in this regard?

Each of these example trade-offs again underscore that DSHE should be recognized and managed as a web of information work products, processes, and settings, rather than as simply a program that comes in a box ready for use on a personal workstation. The potential power as well as the costs of DSHE lies within the hypertext package.

6. Conclusions

Five conclusions follow from the preceding analysis and discussion. First, DSHE represent a new generation of organizational information infrastructure. As such, investment into their development and evolution should be long-term. This will lead to both a realization of their maximum potential power, but at the same time making clear the costs that will be borne to make this infrastructure effective and efficient.

Second, the power of DSHE is derived through structuring hypertext applications to fit with the documents, processing mechanisms, tasks, settings, and their interrelationships specific within a work domain. The power is not inherent in the hypertext system technology. Thus, it might be reasonable to expect that those people who choose to treat hypertext more as a tool than a package, might have more difficulty in understanding or affecting how the DSHE works.

Third, DSHE must support the information work for many kinds of information producers and consumers. This becomes clear from examining DSHE in classical studies, creative writing, journal and book publication, insurance policy management, and CASE domains. Each kind of user may require a distinct viewpoint of the hypertext information, as well as process information with different mechanisms according to different information task sequences. In addition, application domains differ in how hypertext environments and processing mechanisms are structured and standardized.

Fourth, I sought to draw increased attention to viewing DSHE as operational information structures that focus on producing interlinked document compositions by reproducing the structure, content, and flow of information work in an application workplace. As a result, I did not focus attention the technical details underlying user interface display, authoring issues, alternative retrieval algorithms, hypertext system implementation issues, and the like. These are important contributions being investigated by others. Instead, I chose to focus attention to specific application domains and task environments which have mostly been ignored up to this time.

Last, I identified topics for further investigation. These include (a) identifying alternative techniques for integrating message management systems and knowledge based processing mechanisms into hypertext systems, (b) scaling-up hypertext environments to support projects or organizations with hundreds to thousands of users, (c) how to increase the strategic value and competitive advantage attributal to DSHE, (d) how to determine the requirements for other DSHE,[13] and (e) identifying strategies that maximize the power of DSHE that minimize adverse costs.

7. References

1. L. Applegate, B. Konsynski, and J. Nunamaker. A Group Decision Support System for Idea Generation and Issue Analysis in Organizational Planning. Proc. Computer-Supported Cooperative Work, MCC, Austin, TX, 1986, pp. 16-34.

2. R. Balzer, T. Cheatham, and C. Green. "Software Technology in the 1990's: Using a New Paradigm". *Computer 16*, 11 (1983), 39-46.

[13]For example, is a DSHE for computer-aided design of VLSI devices similar to that for software engineering? Would a DSHE for Banking be similar to one for Insurance?

3. S. Bendifallah and W. Scacchi. "Understanding Software Maintenance Work". *IEEE Trans. Software Engineering 13*, 3 (1987), 311-323.

4. J. Bolter and Michael Joyce. Hypertext and Creative Writing. Proc. Hypertext '87, ACM, Univ. North Carolina, 1987, pp. 41-50.

5. M. Chignell and R. Lacy. "Integrating Research and Instruction: Project Jefferson". *Academic Computing 3*, (September 1988), (to appear).

6. A. Clement and C.C. Gottleib. "Evolution of an Organizational Interface: The New Business Department at a Large Insurance Firm". *ACM Trans. Office Info. Systems 5*, 4 (1987), 328-339.

7. J. Conklin. "Hypertext: An Introduction and Survey". *Computer 20*, 9 (1987), 17-41.

8. G. Crane. From the Old to the New: Integrating Hypertext into Traditional Scholarship. Proc. Hypertext '87, ACM, Univ. North Carolina, 1987, pp. 51-56.

9. N. Delisle and M. Schwartz. "Contexts--A Partitioning Concept for Hypertext". *ACM Trans. Office Info. Systems 5*, 2 (1987), 168-186.

10. R. Dunham, B. Johnson, G. McGonagill, M. Olsen, G. Weaver. Using a Computer Based Tool to Support Collaboration: A Field Experiment. Proc. Conf. Computer-Supported Cooperative Work, ACM, Austin, TX, 1986, pp. 343-352.

11. W. Frakes and B. Nejmeh. Software Reuse through Information Systems. Proc. COMPCON '87, IEEE Computer Society, San Francisco, CA, 1987, pp. 380-384.

12. M. Frisse. "Searching for Information in a Hypertext Medical Handbook". *Communications ACM 31*, 7 (1988), 880-886.

13. P.K. Garg and W. Scacchi. A Hypertext Environment for Managing Software Life Cycle Documents. Proc. 21st. Hawaii Intern. Conf. Systems Sciences, Vol. 2, IEEE Computer Society, Kona, Hawaii, 1988, pp. 337-346.

14. P.K. Garg and W. Scacchi. "The Design of an Intelligent Software Hypertext System". *IEEE Software 5*, (1988), to appear.

15. P.K. Garg. "Abstraction Mechanisms for Hypertext". *Communications ACM 31*, 7 (1988), 862-870.

16. L. Gasser. "The Integration of Computing and Routine Work". *ACM Trans. Office Info. Sys. 4*, 3 (1986), 205-225.

17. E.M. Gerson and S.L. Star. "Analyzing Due Process in the Workplace". *ACM Trans. Office Info. Systems 4*, 3 (1986), 257-270.

18. C. Hewitt. "Offices are Open Systems". *ACM Trans. Office Info. Sys. 4*, 3 (1986), 271-285.

19. B.I. Kedzierski. Knowledge-Based Project Management and Communication Support in a System Development Environment. Proc. 4th. Jerusalem Conf. Info. Techology, ACM, 1984, pp. 444-451.

20. R. Kling and W. Scacchi. "The Web of Computing: Computer Technology as Social Organization". *Advances in Computers 21* (1982), 1-90. Academic Press, New York.

21. G. Landow. Relationally Encoded Links and the Rhetoric of Hypertext. Proc. Hypertext '87, ACM, Univ. North Carolina, 1987, pp. 331-344.

22. D. Lenat, M. Prakash, and M. Shepard. "CYC: Using Common Sense Knowledge to Oversome Brittleness and Knowledge Acquisition Bottlenecks". *AI Magazine VI*, 4 (1986), 65-85.

23. B. Lewis and J. Hodges. Shared Books: Collaborative Publication for an Office Information System. Proc. Conf. Office Info. Systems, ACM, Palo Alto, CA, 1988, pp. 197-204.

24. T. Malone, K. Grant, K. Lai, R. Rao, and D. Rosenblitt. "Semi-structured Messages are Suprisingly Useful for Computer-Supported Coordination". *ACM Trans. Office Info. Systems 5*, 2 (1987), 115-131.

25. G. Marchionini and B. Schneiderman. "Finding Facts vs. Browsing Knowledge in Hypertext Systems". *Computer 21*, 1 (1988), 70-80.

26. G. Marchionini. "Information-Seeking Strategies of Novices Using a Full-Text Electronic Encyclopedia". *J. Am. Society Info. Sciences* , (1988), .

27. K. Narayanaswamy and W. Scacchi. "Maintaining Configurations of Evolving Software Systems". *IEEE Trans. Soft. Engr. 13*, 3 (1987), 324-334.

28. K. Narayanaswamy and W. Scacchi. "A Database Foundation to Support Software System Evolution". *J. Systems and Software 7* (1987), 37-49.

29. T. Nelson. *Literary Machines.* (available from the author), 1981.

30. J. Orr. Narratives at Work: Story Telling as Cooperative Diagnostic Activity. Proc. Computer-Supported Cooperative Work, MCC, Austin, TX, 1986, pp. 62-72.

31. M. Porter. *Competitive Advantage.* Free Press, 1985.

32. R. Rada and B. Martin. "Augmenting Thesauri for Information Systems". *ACM Trans. Office Info. Systems 5*, 4 (1987), 378-392.

33. D. Raymond and F. Tompa. "Hypertext and the New Oxford English Dictionary". *Communications ACM 31*, 7 (1988), 871-879.

34. B.K. Reid. "USENET Cookbook: An Experiment in Electronic Publication". *Electronic Publishing 1*, 1 (May 1988), .

35. W. Scacchi. "Managing Software Engineering Projects: A Social Analysis". *IEEE Trans. Soft. Engr. SE-10*, 1 (January 1984), 49-59.

36. W. Scacchi. The USC System Factory Project. Proc. Software Symposium ’88, Japan Software Engineers Association, Tokyo, Japan, 1988, pp. 11-37.

37. A. Sen and L. Kerschberg. "Enterprise Modeling for Database Specification and Design". *Data and Knowledge Engineering 2*, 1 (1987), 31-58.

38. R.F. Simmons. "A Text Knowledge Base from the AI Handbook". *Info. Processing and Management 23*, (1987), 321-339.

39. J. Smith, S. Weiss and G. Ferguson. A Hypertext Writing Environment and its Cognitive Basis. Proc. Hypertext ’87, ACM, Univ. North Carolina, 1987, pp. 195-214.

40. S. Weyer and A. Borning. "A Prototype Electronic Encyclopedia". *ACM Trans. Office Info. Sys. 3*, 1 (1985), 63-88.

41. T, Winograd and F. Flores. *Understanding Computers and Cognition: A New Foundation for Design.* Ablex, 1986.

42. C. Wiseman. *Strategy and Computers: Information Systems as Competitive Weapons.* Dow Jones-Irwin, 1986.

Table of Contents