The eText Project: Publishing Hypermedia Textbooks

Rohit Khare and the eText Group
The eText Project at Caltech
MSC 256-80, Computer Science
Pasadena, CA, 91125
fax : (818) 792-4257
e-mail: khare@cs.caltech.edu

April 11, 1994

ABSTRACT

The eText Project at Caltech seeks to construct a hypermedia textbook for teaching sequential and parallel programming. We present a *document-centric* hypermedia model that supports 1) portable, atomic compound-documents, 2) customizable hypermedia annotation objects, and 3) navigation and personalization. The model is used to characterize the development of the Project's prototype implementations; facilities for teaching and reference; and hypermedia publishing efforts. The Project is also directly implementing its model in a new hypermedia environment, the eText Engine.

Keywords: Hypermedia, Education, Publishing, Object-Oriented Design, User Interface Design.

1 INTRODUCTION

The eText Project at Caltech was initiated to help teach parallel and distributed processing techniques. This research encompasses many areas, including the development of the Archetypes eBook, an interactive hypermedia text- and reference-book. This paper communicates some of the lessons learned from our experience building our "electronic book."

1.1 The eText Project

Today, most software designers and scientists write sequential code, in applications ranging from database queries to physical simulations such as crash testing. Moore's Law dictates that hardware speeds double every eighteen months,

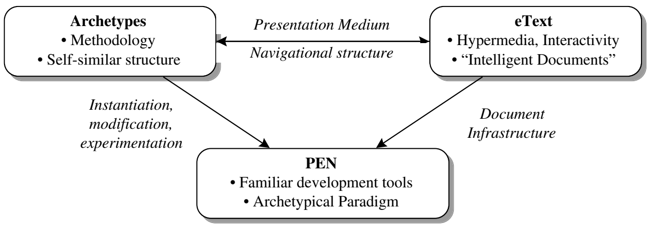

Figure 1: *The eText Project Triad.*

but soon sequential technology will have realized its peak, and to garner more peformance these developers will need to embrace parallel computing. They will want extend their knowledge of sequential software design to accommodate new theories, tools, and technologies.

The eText Project is developing a three-fold approach to enable such learning, as depicted in Figure 1. The teaching methods are based on archetypes [19, 20], which systematize patterns of computation in both the sequential and parallel domains. Archetypes naturally lend themselves to a specialized Programming ENvironment that enables students use and extend Parallel Archetype Libraries (PEN PAL). The theory of Archetypes and the PEN tool are enabled by the eBook, which in turn is built atop the eText Engine. The eBook incorporates three key design lessons:

- *Document-Centricity.* A large-scale, authoritative teaching and reference tool is not compatible the fine-grained focus (e.g., cards, screens, and frames) of most current hypertext and hypermedia systems.
- *Genuine Interactivity.* The eBook requires an environment where "live", interactive, custom-coded simulations reside within the hypertext. Such "true" interactivity is as important for eBook as standard hypermedia features like navigational links and multimedia annotations.

[0]Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copyright is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.
©1994 ACM 0-89791-xxx-x/xx/xxxx...

- *Data Portability.* Our compound-document architecture, built with open, portable component data formats, is explicitly designed to migrate to future standards and support "distillation" of native documents into other formats, all maintained from a single source tree.

The remainder of this paper presents the eText hypermedia model and uses it to discuss the evolution of our prototypes and the eText Engine, the requirements analysis of several educational scenarios, and implications for publishing and compatibility with other hypermedia systems.

2 THE eText HYPERMEDIA MODEL

While developing our textbook, we tried to map our visions onto several different hypermedia models (e.g., databases, timelines, or cards), which ultimately proved inadequate. Synthesizing the features we needed, we designed a hypermedia model for our applications, illustrated in Figure 2. The eText Engine [11] implements this model as an actual system.

2.1 Related Work

In our initial survey of commercial and academic hypermedia systems [15, 7, 18], we discovered that although each has its strengths, none is appropriate for building a complete programmers' learning and reference enviroment. Instead, we encountered prescriptions for:

- *Aircraft-Repair Manuals,* exemplifying large-scale, rigidly structured text applications. Usually positioned as SGML-based enterprise information repositories, they support some combination of document-to-document links, indexing, querying, and simple, static graphics. The most advanced of this genre, derived from high-end publishing systems, add multimedia capabilities to render audio and video.

- *Walking, Talking, Sales Pitches In A Box.* Several overgrown PC and Macintosh presentation software packages, such as Macromind *Director* and Asymmetrix *Astound*, as well as workstation-based Computer-Based Training (CBT) setups such as Imagine *Callisto* are built around a metaphor of a timeline and a screen. These systems cannot be adapted to present large volumes of information, nor to allow real freedom in navigation. While they can competently integrate multimedia, and some even had hooks to call external function libraries, these systems have minimal support to incorporate interactive simulations or navigation/linking. Hence, the timeline metaphor and screen-size scaling of these systems sets them at odds with the true goal of *hyper*media.

- *MacFrameHyperWebCard,* encompassing the tools that have succeeded in the hyperliterature community. While they excel at producing scholarly webs, commentaries, and occasional multimedia presentations, they are inappropriate for handling large-scale and potentially distributed projects. None appeared to be extensible, and fewer were compatible with other systems (except the subset of Claris *HyperCard* clones). One system of this kind, Brown's *Intermedia*, did seem suited for our use, but is no longer available [13].

- *Do-It-Yourself,* a category of systems that are not really hypermedia systems *per se*. Generally touted as prototyping tools, such as GAIN *Momentum* or Xanthus *CraftMan*, or as multimedia extensions to databases (ORACLE *Cooperative Development Environment*), they allow developers to build systems from scratch that can call system libraries to handle hypermedia functions, and are inherently extensible by compiling in more code. These tools did not offer any foundation for a hypermedia project, such as a data model, navigation tools, or the ability to publish the product widely.

- *The Ultimate Word Processor.* Recently we have seen the rise of multimedia word processors that can handle audio and video; some, notably Microsoft *Word 6*, even incorporate full-strength programming languages. University of Southampton's *Microcosm* [10] is an example of leveraging these commercial applications in concert to provide a hypermedia system. An alternate formulation, and the one that proved closest to our own model, is that of the compound document, in which the processor is no longer of words alone, but of arbitrary "objects" as well.

During this process, we experimented with several prototypes, which will be discussed in §3.1. From that experience and our informal survey, we adopted the compound-document model and used it as an integrating platform to bring together elements from all of these systems. The balance of this section will present the resulting architecture in greater detail, referring to Figure 2.

2.2 Compound Documents

At the core of our model, both graphically and conceptually, is the notion that a document is a quantum of one to ten pages. The document is a formatted text stream with any combination of executing objects encoded within. We assert that such a document is at least as powerful as any other hypermedia model that does not incorporate temporal synchronization; this is further discussed in §5.1.

In the circle labeled "eText Kernel", we have a concrete realization of that idea. This system currently encodes the formatted text stream as an extension of Microsoft Rich Text Format (RTF), and offers an atomic document-storage mechanism that binds that text store together with component storage for any external data used by objects in the document (e.g.

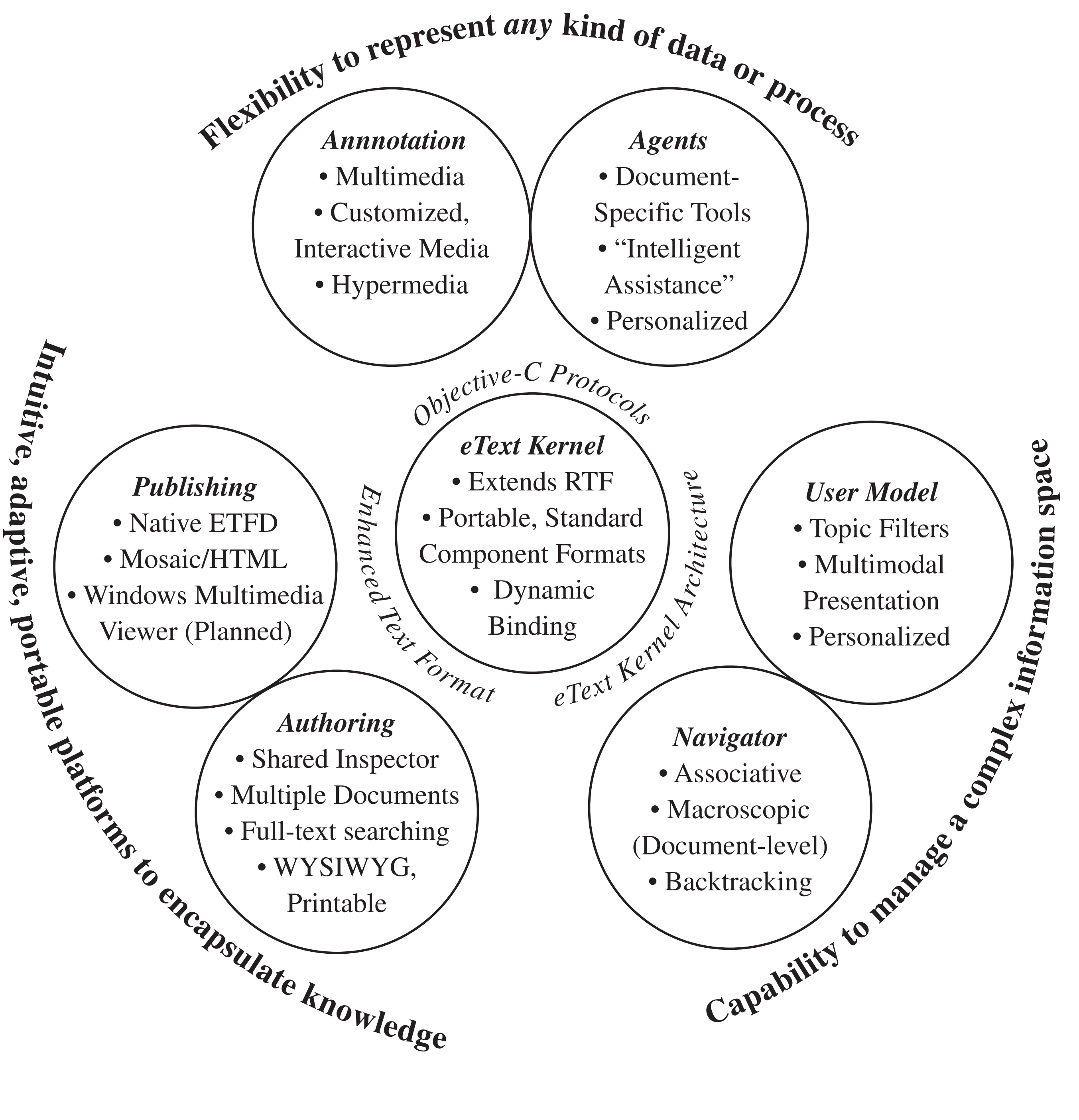

Figure 2: *The eText Publishing System Architectural Overview. Clusters denote related subsystems for Interactivity, Navigation, and Publishing support.; within each circle is a set of relevant features from the eText Engine. Light curved type denotes system affordances. Bold curved type is the "mission statement" for each subsystem.*

images, audio, simulation state. The Kernel also leverages an object-oriented environment by publishing an Application Programming Interface (API) for the object to be loaded into the system at runtime. As a result, the Kernel guarantees the provision of a standard document format for Publishing purposes, a standard architecture for plugging in user services for Navigation, and an API for loading in new kinds of objects to be instantiated within documents.

Alternative formulations might replace the text stream with a frame-based layout metaphor, or a structured text encoding. Document storage may be in a shared OO database rather than as atomic directories in a filesystem. [11] discusses these and other implementation details in the context of the eText Engine.

2.3 Interaction

The key to making this model a hypermedia model rather than just a text editor, is the Interaction support, represented by the cluster labeled "Annotations" and "Agents". By defining standard protocols, the document can load in objects that give it the expressive power of other hypermedia systems.

An annotation is defined as an object that the user can instantiate within the text stream. The protocol we use contains only a bare handful of methods for allocating a region of the document for the annotation, drawing it, encoding and decoding alternate representations, and for publishing a user interface (e.g., menu commands, inspectors, and toolbars). Multimedia annotations can present time-based media; hypermedia annotations provide link buttons, anchors, and margin notes; and interactive media annotations can exercise the full power of the host system (See Figure 10).

This application technology has been embraced by the commercial market as well. Microsoft has *Object Linking and Embedding 2*; Component Integration Laboratories (backed by IBM, Apple, WordPerfect, Novell, and others) offers *System Object Model* and *OpenDoc*; and Go's *PenPoint* pen-based operating system predicated its entire user interface and API on it. Although all of these efforts have APIs that are conceptually identical to the one described here, they are more complex to program, by an order of magnitude; we maintain simplicity and ease of development as project goals. On the academic side, systems such as Andrew [4] paved the way for current commercial interests. We would have used one of these systems, if they actually existed or were developer-friendly enough to build a hypermedia system.

An agent is a variant that defines an object that binds to the entire document, not just a location within the text stream. The agent helps model processes rather than data, since it has the authority to help the user edit the entire document, and even manage it in relation to other documents. The PEN mentioned in Figure 1 is one such application, guiding the user through the creation of a hypermedia source file, in conjunction with other files in the project. The use of agents is a rapidly-moving field, and it is unclear what other features this facility might enable. Nevertheless, while few systems have any similar hooks today, we feel this is an important part of a generic hypermedia authoring system. We are currently investigating the potentials of this within the eText Engine development project.

2.4 Navigation

The most slippery part of any information system to characterize is its user interface (i.e., its look-and-feel). Nevertheless, we endeavor to specify the functional power that any such interface must provide.

First, any such system should track user preferences and help personalize the information space. In a training system, this may be as sophisticated as cognitive modeling and performance tracking; for a news feed, it may simply be denoting which sections most interest the reader. Another purpose is multimodal presentation, to represent an concept formally, or with graphics, or with a voice-over, or tailoring the content to different languages and physical abilities.

Second, the system must include tools for dealing with the "Navigation Problem" [3]. This involves: a history mechanism, so the Kernel should notify the navigation subsystems when documents are opened and closed; associative retrieval, so the navigation subsystem maintains its own databases and indices of documents; and some macroscopic, document-to-document links, which cannot be handled by annotations alone. Beyond this simple model, one can envision systems that combine user-tracking and associativity to provide the user with guided tours of the information space, such as "suggested reading" buttons, but these systems can still be classified within this model.

2.5 Publishing

Finally, the most contentious part of embarking upon a project such as eText is the chicken-and-egg problem of creating a hypermedia infrastructure. During the past year, the explosion of growth on the World-Wide-Web, fueled by *Mosaic*, has made HyperText Markup Language (HTML) [1] a *de facto* standard for presenting hypermedia documents. Nevertheless, it is neither a panacea nor a permanent solution; interactivity, in particular, goes unsupported in this medium.

The publishing model of a hypermedia system should be able to cope with this uncertainty, and it is our conviction that open, portable publishing will remain the most crucial

[2] The National Center for Supercomputing Applications at Urbana-Champaign estimated that Mosaic use was growing at a weekly rate of 11% at the end of 1993. (from Dr. Bob Lucky, Bellcore, lecture at UCLA 4/5/94.)

part of the hypermedia deployment puzzle for years to come. Our solution has relied on the inherent power of compound-documents. In this model, the Kernel creates a corpus of all the data in the document. Individual component data is the responsibility of the annotations, and thus the issue of portability is moved out of the Kernel and into the mutable, user-level layers. To write out an HTML file rather than our Enhanced Text Format (ETF), the Kernel reencodes the text stream in HTML, and the Image annotations simply respond to `writeHTML:` instead of `writeRTF:`. The key is that some subset of this fully-interactive native ETF document can be automatically reparsed into another document-oriented system. A more dramatic example has come from our investigation into planning Microsoft *Multimedia Viewer* document conversion, which also has an ETF-like model.

Publishing hypermedia documents is not just about publishing data, though. Publishing should also address custom, interactive annotations. The "Authoring" cluster in Figure 2 mentions those aspects involved in supporting a flexible, extensible user interface; for further details, see [11].

3 SYSTEMS DEVELOPMENT

The evolution of our model can be traced through our implementation efforts. In each phase we learned about the proper emphasis on each of the three subsystems of our model.

3.1 Prototypes

For the first several months, we experimented with different platforms and approaches to hypermedia. Our concrete short-term goal was to produce a system for teaching an undergraduate course. The three systems below taught us about interactivity and extensibility, interface and navigation, and publishing, respectively.

3.1.1 Inspiration: *MediaView*

In the beginning, we were inspired to investigate the interactive textbook example set by *MediaView* [17]. It had been used to produce an interactive computer graphics paper that featured a running simulation of the reflectance model being discussed, right inside the paper. The model, shown in Figure 3, featured several such embedded *interactive figures*. In addition to direct manipulation of the parameters through controls on the Inspector panel, the paper also had "journals" which could replay the mousing, typing, and voiceover of the author as he guided readers through the figure.

MediaView has a compound-document architecture and interactivity support, but no real navigation or publication facilities. The extensibility, though, was quite powerful. Objects included with the system supported audio, video, drawings, and formatted notes. Extensions included 3D wireframe viewers, multithreaded animations, and algorithm visualizations. *MediaView* allowed each such extension to create a control panel for itself, leading to a haphazard look-and-feel; our later efforts would standardize this into an Inspector paradigm, which we incorporated directly into our implementation.

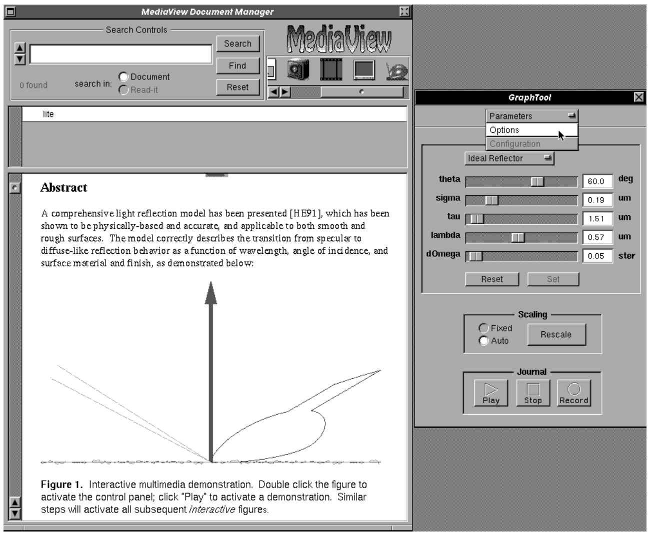

Figure 3: *The* MediaView *interactive paper,* A Fast and Accurate Light Reflection Model. MediaView *features extensibility through custom objects, as with the reflection simulation above, but did not support hyperlinks.*

3.1.2 Development: *CraftMan*

Since *MediaView* is no longer supported, we decided to prototype a *NeXTSTEP*-based system of similar functionality to build the first few chapters of the eBook. With Xanthus *CraftMan*, we adopted an interpreted prototyping environment somewhat similar to a SmallTalk environment. Here our group developed some of its most advanced concepts on interactive media, and added a navigational layer.

Figure 4 shows the main chapter interface; there are explicit link-buttons to other chapters and links within the chapter are all hard-coded. In line with the criticisms leveled earlier against prototyping enviroments, it has no hypermedia model foundation, no storage system, no history mechanism, and every chapter ends up as a separately compiled application. Figure 5 shows one of the system's successes: the ease of constructing interactive figures, in this case a skyline simulation, which also supported a journaled walk-through. Other successful aspects include narrated animations, interactive sorting exercises, user-tracked quizzes, source code browsers, and multimodal presentations (Basic and Expert modes).

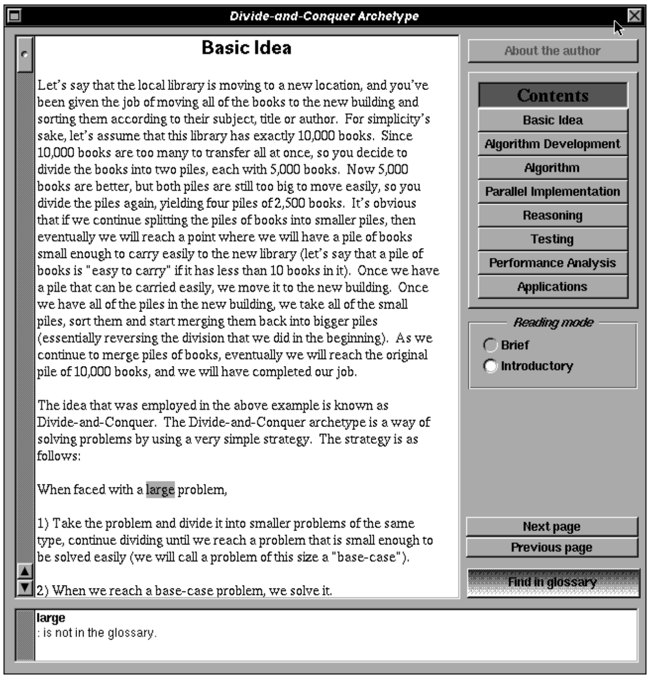

Figure 4: *The* CraftMan *version of the Divide-and-Conquer chapter. The chapter interface has quick reference-buttons, computed glossary links, and links to other chapters.*

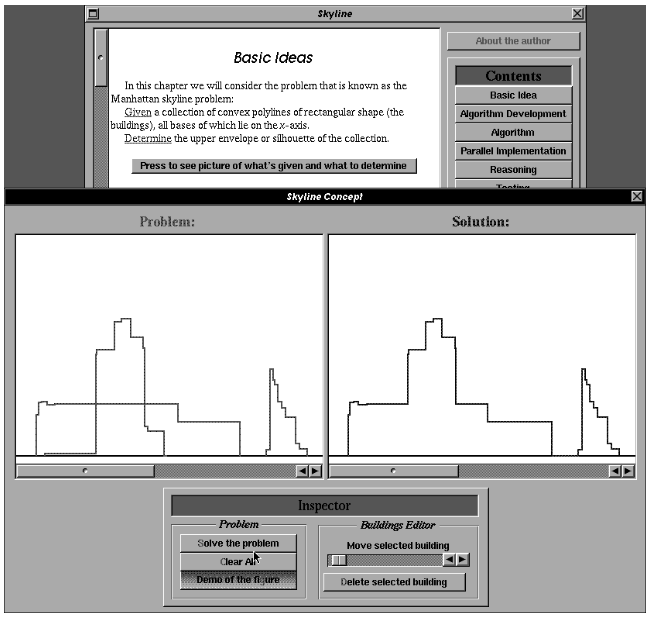

Figure 5: *The Skyline interactive figure. Students can investigate the behavior of the algorithm by drawing a set of buildings on the left side and watching the eBook construct the upper envelope on the right. The figure includes a narrated walkthrough.*

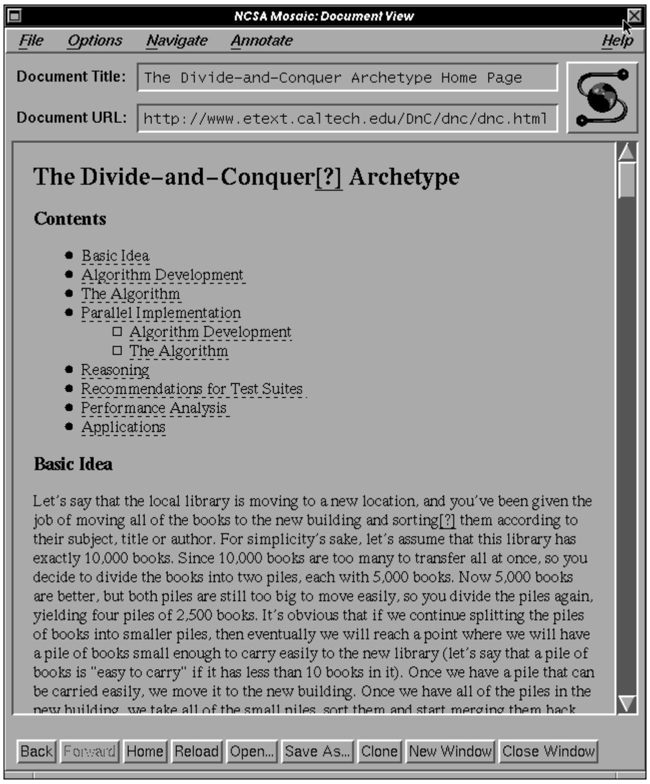

Figure 6: *The* Mosaic *version of the Divide-and-Conquer chapter seen in Figure 4. Note that the glossary links are explicit question marks.*

3.1.3 Publishing: *World-Wide-Web*

As our user community heard about our work with eText, the call for a World-Wide-Web version of our textbook grew. This was our third lesson: the importance of open publishing standards cannot be underemphasized.

While most of the component data ports to HTML/Mosaic formats easily, some formatted text (notably, equations) caused problems. Our solution was to provide bitmap images of the equations and tables. We were also able to transform the slide shows by creating a separate HTML file for each frame and adding "forward" and "back" links. Unfortunately, other interactive media experiments could not be replicated for a distributed audience, although our group has had recent success with TCL/Tk [16] in porting the skyline figure to X-Windows. We concluded from this experience that we could automate the conversion of a subset of our compound documents to HTML – and possibly other formats.

3.2 The eText Engine

As the eText hypermedia model took shape, we found that no one system would be able to satisfy all of our different goals.

The eText Engine is an implementation designed specifically to meet the architectural specifications of our model.

3.2.1 Justification

In particular, eText fills the need for a document-centric hypermedia system. Previous work about compound documents uses an “object” as a user interface abstraction rather than as a context for software engineering. Hypermedia systems are too often bound to a time-centric “mulitmedia” past; hypertext systems are mature enough to handle textbooks with ease, but aren’t extensible. The Dexter model of hypermedia systems [9] describes a system with sufficient expressivity, but no successful Dexter-inspired system has been developed yet. With the exception of the document-atomicity assumption within eText, the publication support of our model is reminiscent of Dexter, including identification and linking schemes.

eText also reflects commercial research and development trends. The eText architecture model is appropriate for expressing many different kinds of applications, once the object infrastructure is in place. Soon, the same models of navigation the hypermedia community creates for textbooks will resurface in financial analyses and interactive shopping.

3.2.2 Implementation

The eText Engine is being developed under *NeXTSTEP* [8], an advanced object-oriented development enviroment, and is shown in Figure 7. For creating, viewing, and converting documents, it is comparable to a word processor, with multiple documents, multiple undo/redo, rulers, WYSIWYG, printing, faxing, spell-checking, drag-and-drop, and so forth. New annotations can be created in a variety of ways. For example, an audio annotation can be created by importing an audio file, pasting audio data from another application, dragging in a sound icon, or choosing a menu command. Click on the sound-icon, and the Inspector displays the waveform, editing, and playback tools. Similar ease-of-use applies for other media types, even running custom simulations. Linking is achieved through a drag-and-drop operation; simply create a bookmarked region, and drag it out to another document.

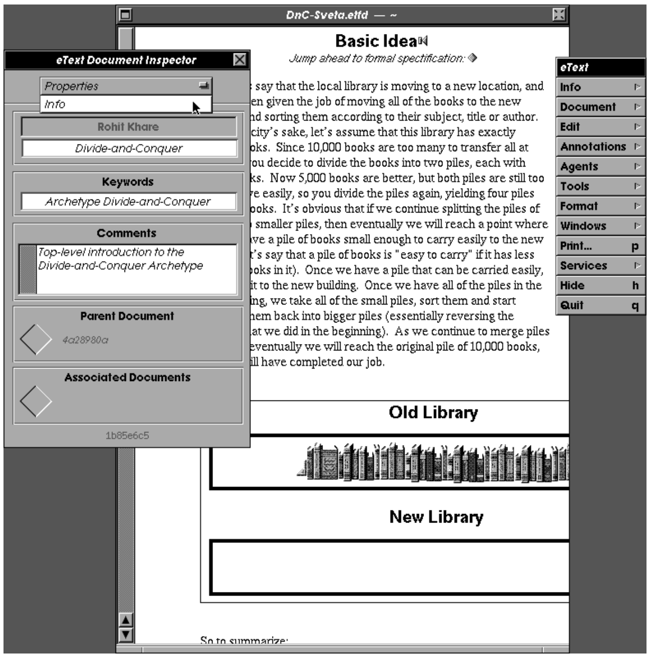

Figure 7: *An early* eText Engine *version of the Divide-and-Conquer chapter seen in Figure 4. The first button is a miniature speaker icon; the diamond is a link. The graphic is an inline slide show. The inspector is currently displaying the basic navigational information about this document: author, date, etc, and has diamond-shaped link-wells for dropping links to associated documents.*

4 EDUCATION & REFERENCE

The nature and purposes of a textbook define many aspects of our model. A reference book might only exercise the publishing aspects, a picture book just the interaction, and a literary criticism only the navigation, but a full textbook will employ all three. In this section, we present the lessons learned from analyzing potential uses of the book.

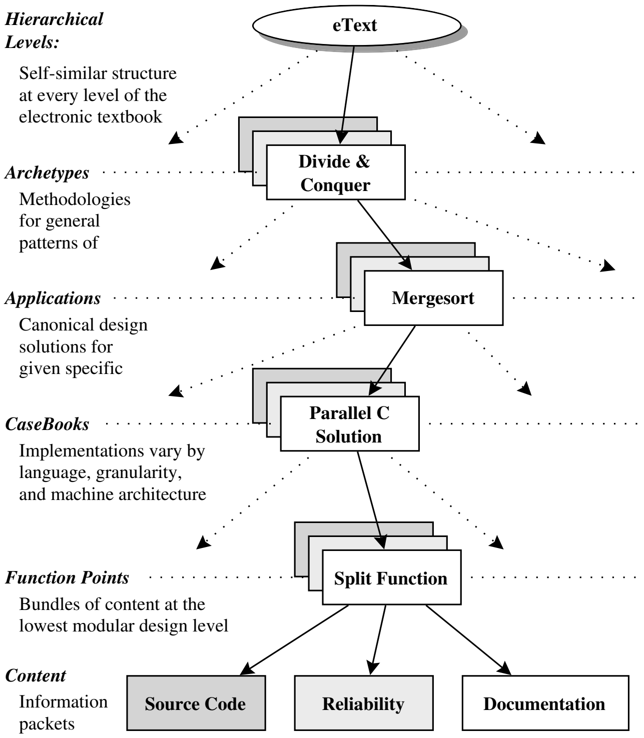

Figure 8: *The eText Archetype textbook structure.*

4.1 Archetypes

The most fundamental quality of our entire programming-education effort is the the recursive self-similarity accompanying the archetype model of programming. At every level, from archetype to application to casebook, the symmetry dictates the existence and structure of the algorithm, reliability argument, perfomance analysis, testing, and design documentation. This structure is shown, somewhat simplified, in figure 8.

Consider how the Divide-and-Conquer archetype is presented in figure 9. At the highest level, we present the metalgorithm, proof outlines, and design documentation for *split*, *merge*, *isBaseCase*, etc. At the Application level, we present pseudocode, generic proofs, and documentation of creative steps necessary to flesh out the archetype to yield algorithms to solve specific problems (e.g., Mergesort, Skyline, and Fast Fourier Transform). Finally, each Application is implemented within several CaseBooks, with concrete proofs, project documentation, and actual code tailored for a particular programming language, data and process granularity, and machine architecture. Note that the Archetype approach unifies sequential, parallel, and distributed versions of the solutions; it only diverges at the CaseBook level.

The lesson of this analysis is that the nature of the information space can help greatly reduce the navigational load. A well-

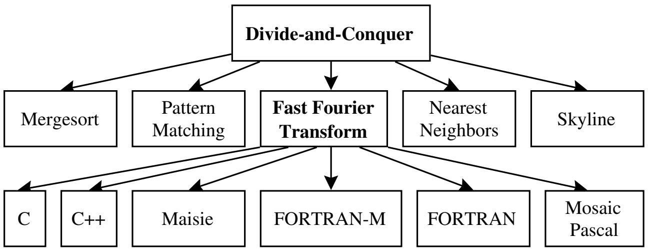

Figure 9: *A sample decomposition of the Divide-and-Conquer archetype.*

designed editorial architecture can keep readers more focused than any permutation of graphical navigation widgets.

4.2 Teaching

Teaching undergraduates and motivated users learning independently, we want to present the material as effectively as possible, since the eBook is intended for self-directed use. Not only does this mean that the material should be written and edited for that purpose, but that the eBook should offer multiple paths of entry. The multimedia and interactive media portions must be able to present different versions based on the user's proficiency and interest. Students should be able to take tours through the book (e.g., looking just at sorting algorithms), rather than repeatedly drilling the hierarchical structure. All of this indicates the need for intelligent curriculum design, but more subtly highlights the need for strong system support.

In this scenario, the system needs a strong user model. For example, the bookmark annotation delimits a section of a document to which other documents can link. A bookmark can also be collapsed behind an icon, and conditioned to display only depending on user preferences; that behavior should be tied to a persistent user model. One cannot have a lab full of students fill out a profile questionnaire starting class each day, and yet that is precisely how mulitmodal presentation support is achieved on current hypermedia authoring systems.

This scenario also exercises the need for a flexible annotation development system. Considering all of the effort that goes into designing a custom-coded simulation, the hypermedia model iself should not impose any further overhead. Also, to reach a wider base of students, all of whom have access to personal computers, we need to publish compatible subsets of our information to other, more accessible systems.

4.3 Reference

We envision the eBook as a cooperative repository for computational scientists. They would like help to skim the space to choose an appropriate archetype and refine it further. Consider a chemical engineer performing a smog modeling of the Los Angeles Basin [6]. She inherits thousands of lines of legacy Fortran code. Her goal is to perform regressions on several axes, and this will require lots of program runs. Fortunately, her management has just purchased a parallel machine. Although scientists are aware that they can solve bigger problems, faster, using parallel and distributed machines, we must overcome their prejudices of how difficult they perceive parallel programming to be.

Presently she has sequential code, and she understands her application domain, so a few index searches or browsing a list of applications in the eBook should guide her to the proper archetype. Suppose that a low-level climate model is already in the system, as an Application of the Mesh Computation archetype. The navigational process is simplified by the hierarchical organization of the space. When she reads the actual documents though, she doesn't need to see the slideshows or hear the narration. In fact, since she has told the user model that she has a Fortran code, she is not distracted by any information specific to C++.

This scenario highlights the importance of navigation. In the next phase of her development of code, when using PEN to develop a prototype instantiation of the her archetype, she can use agents to manage her project.

4.4 Knowledge Capture

In fact, if we extend these scenarios just a little further, the eText Engine enables an engineering design knowledge capture system. Most engineering design processes, like archetypes, have a recursive structure and problems that can be solved through successive refinement. One can just as easily posit archetypes for electrical circuit design or for satellite design. In each of these cases, a hypermedia engineer's notebook for capturing the design process and reusing it can yield a significant productivity boost, as discussed by the CSCW community [12].

5 HYPERMEDIA PUBLISHING

As mentioned earlier, the challenge of actually getting hypermedia materials out to our user community is a bottleneck for increased application of hypermedia techniques from the classroom to the boardroom. Developing content for the eText model is a risky proposition unless it is possible to migrate existing information from other systems.

5.1 Expressive Power

Developers can grow within the eText model. By opening the architecture up for loadable Annotations and Agents, we can

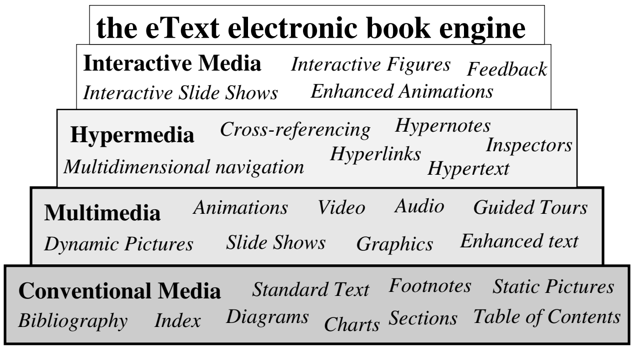

Figure 10: *The eText Media Pyramid. We assert that all of these media types can be enabled by Annotations in the eText Model.*

present almost any kind of hypermedia data or process. Annotations can model any of the media types shown in Figure 10, which covers the known spectrum of hypermedia applications. Agents, in the model's definition if not in the current implementation of eText Engine, can model any process mediating between users and document creation. Between these two and the prospect of replacable, upgradeable navigation support, the Engine plays the role of a working testbed for trying out new hypermedia techniques. The only "limitation" is that the document-centric metaphor, and the eText model in particular, is orthogonal to timeline-based and real-time schemes.

5.2 Standard Platforms

Currently, the most popular hypermedia platform in the world is the World-Wide-Web [2]. Accesible to a wide variety of clients, it is bound by a lowest-common-denominator phenomenon. With the addition of forms, Web servers can get feedback from users, but the heterogenous, distributed nature of the web still impedes progress towards publishing interactive media on it.

Also a volume leader, Microsoft *Multimedia Viewer* [14] has been used to develop a wide variety of hypermedia titles. While its files can be automatically resued to a degree on Windows, Macintosh, and X/Unix setups, its expressive power is limited by its modest capacity, limitations of DOS, and arcane development methods. Adding interactivity in particular seems to be a problem area.

The eText Engine, currently, is tied to NeXT's *NeXTSTEP* and forthcoming SUN *Solaris*-based versions of *OpenStep*, which is a narrow slice of our audience; we believe in producing courseware versions with the native engine and a distributable subset over the Web.

5.3 Standard Representations

There are several successful efforts underway in the community to standardize the representation and interchange of hypermedia data. We certainly do not purport to be one of them. We also don't know which standard to bet on, either, so we are watching with the assurance that we can bridge between our Enhanced Text Format (ETF) compound-stream representation and forthcoming standards.

Currently, we have a crop of standards based on Standard Generalized Markup Language (SGML), which work by adding structural encoding to documents. Direct descendants include HTML and the Dexter interchange format. A related type is MIME, the Multipurpose Internet Mail Extensions [5]. These systems all share the concept of a document unit and of objects embedded in text. The main difficulty in mapping ETF to this class is the implementation of the eText Engine using RTF streams, which store formatting information rather than structural encoding. The mapping of the text is automatic, but fragile; the component media types that are in common will probably be rosbust, since there is growing consensus on multimedia data formats.

The "other" category includes proprietary markup schemes, as used in *Multimedia Viewer*, and future standards associated with commercial development, tossed under the rubric of "object-oriented file systems." Interoperability with these schemes will be possible, but the efficiency of conversion is unclear as yet. Once these systems arrive, though, we will have a platform for publishing not merely hypermedia data, but through distributed object technology, simulations and processes.

6 CONCLUSIONS

The lessons learned from the eText Project to date fall into three categories. In the search for a system appropriate for producing interactive textbooks, we designed the eText hypermedia model and built the eText Engine. In designing the educational content of a hypermedia textbook, we see how all the parts of the model must be present to enable learning and reference. Finally, the model must be robust enough to produce documents that will survive the model's inevitable passing. Building new hypermedia systems is rewarding, but rebuilidng content is a mistake.

- *Hypermedia Systems.* We believe that a document-centric model for hypermedia is a compelling solution to a wide variety of applications. Within the eText hypermedia model it has the potential to match the expressive power of any other model that does not intorduce temporal contraints. It provides a granularity appropriate for many different kinds of applications, especially in engineering. Finally, the document/publishing model it is based on is being validated by the success of systems such as WWW and the commercial push for compound documents.
- *Hypermedia Education.* A sucessful electonic textbook should be able to reach as many students as possible. The material should not be broadened to the point of meaninglessness (as is the fate of some paper texts), but should be able to address each student at their own levels. It means only simple things like selectively presenting material according to a user model or having alternate versions of a demonstration, but it is useful. Navigation, even with an overall structure for the document-space, remains as important as ever. To get the information out to as many students as possible, we also need tools that can re publish data into other formats.
- *Hypermedia Publishing.* To close, we reiterate our belief that compound document architectures – leveraging real objects, not "objects" – is the future of application development in many fields, and constitutes an integrating paradigm for publishing documents of any format. While such structures may be inefficient and wasteful of resources for display, indexing, and storage, they conserve the most precious resource – the time and talent of those who create the knowledge.

7 ACKNOWLEDGEMENTS

Special thanks go to Adam Rifkin, for many productive discussions, and to other members of the eText project group: Paul Ainsworth, Svetlana Kryukova, and Rajit Manohar. Last, but not least, thanks to K. Mani Chandy for establishing, leading, and supporting the eText Project.

This research is sponsored in part by the Air Force Office of Scientific Research grant AFOSR-91-0070 and CRPC support for education and parallel scientific applications under cooperative agreement CCR-9120008.

References

[1] T-J. Berners-Lee, R. Cailliau, and J.-F. Groff. The world-wide web, computer networks and ISDN systems. In *Proceedings of the 1992 Joint European Networking Conference*, number 25, pages 454–459. North-Holland, 1992.

[2] T.J. et.al Berners-Lee. The world wide web initiative. In *Proceedings of INET'93*. CERN, 1993.

[3] Mark Bernstein. The navigation problem reconsidered. In Emily Berk and Joseph Devlin, editors, *Hypertext/Hypermedia Handbook*, pages 285–298. Mc-Graw Hill, 1991.

[4] Nathaniel S. Borenstein. *Multimedia Applications Development with the Andrew Toolkit*. Prentice-Hall, 1990.

[5] Nathaniel S. Borenstein. Mime: A portable and robust multimedia format for internet mail. *Multimedia Systems*, 1(1):29–36, 1993.

[6] K. Mani Chandy, Rajit Manohar, Berna Massingill, and Dan Meiron. Integrating task and data parallelism with the collective communication archetype. *submitted to Supercomputing 94*, 1994.

[7] Bob Cotton and Richard Oliver. *Understanding Hypermedia*. Phaidon Press, 1993.

[8] Simson Garfinkel and Mike Mahoney. *NeXTSTEP Programmin: Step One*. TELOS, a division of Springer-Verlag, 1993.

[9] Frank Halasz and Mayer Schwartz. The Dexter hypertext reference model. *Communications of the ACM*, February 1994.

[10] Wendy Hall, Gary Hill, and Hugh Davis. The Microcosm link service. In *Hypertext '93 Proceedings*, November 1993.

[11] Rohit Khare and the eText Group. The eText engine: An extensible object-oriented hypermedia publishing system. In *Submitted to Proceedings of the 1994 European Conference on Hypermedia Technology*, 1994.

[12] Vinay Kumar, Jay Glicksman, and Glenn A. Kramer. A shared web to support design teams. In *IEEE Third Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*. Enterprise Integration Technologies Corp, April 1994.

[13] Norman K. Meyrowitz. Intermedia: The architecture and construction of an object-oriented hypermedia system and applications framework. In *OOPSLA '86 Proceedings*, 1986.

[14] Microsoft, One Microsoft Way, Redmond, WA. *Multimedia Viewer Technical Reference*, 2.0 edition, 1993.

[15] Jakob Nielsen. *Hypertext and Hypermedia*. Academic Press, 1990.

[16] John Ousterhout. *TCL and the Tk Toolkit*. Addison-Wesley, 1994.

[17] Richard L. Phillips. MediaView: a general multimedia digital publication system. *Communications of the ACM*, July 1991.

[18] Roy Rada. *From Text to Expertext*. McGraw-Hill, 1991.

[19] Adam Rifkin. Teaching archetypical design with an electronic textbook. *Proceedings of the 25th ACM SIG CSE Conference*, March 1994.

[20] Adam Rifkin. Teaching archetypical design with an electronic textbook. Technical Report Caltech-CS-TR-94-02, Center for Research on Parallel Computing, California Institut e of Technology, 1994.