An Object-oriented Multimedia Distributed Meta-database

Sha Xin Wei*

March 25, 1994

Abstract

We describe a multimedia distributed databases system (mmdd) which manages associations of arbitrary renderable objects (such as text, 3D-graphics, sound, video, certain executables) using a hybrid object-oriented, relational database abstraction. Strengths include author-modifiable schema, author-reconfigurable front ends, and relatively straightforward extensibility to arbitrary media types and search methods.

The mmdd enables authoring and navigation in rich user interface models. It does not address low-level issues such as synchronization of time-based media, or algebras of virtual media devices.

Key features of the architecture include (1) atoms that are not traditional wordprocessor-like documents but *abstract entities*, which endow the mmdd with unusual flexibility, and (2) object-oriented core mediating classes which provide coherent but extensible meta-data- and content-based services. The mmdd was designed around the projected needs of faculty and their assistant authors, based on extensive experience with authoring multimedia simulations. The mmdd supports Stanford University's Multimedia Distributed Library experiment. [1]

*ASD, Stanford University, *xinwei@jessica.stanford.edu*

[1] *draft 0.26; submitted to ACM Multimedia 94*

Language is a labyrinth of paths. You approach from one side and you know your way about; you approach the same place from another side and no longer know your way about.
– Wittgenstein, Philosophical Investigations

1 Introduction

This paper discusses the *mmdd*: a set of meta-database, content-based searcher, hyperlinker, media abstractor/filter, and user interface classes which assemble into widely varying network-based multimedia applications. The mmdd mediates services such as relational database engines, grammar-driven indexers, format conversion utilities, and natural language filters. It makes such services available via several distributed mechanisms to author-configurable interfaces. The user interface classes have been constructed as Macintosh XCMD's, Hypercard objects, NeXTSTEP Interface Builder and Objective-C classes.

In section 2, we discuss the users and tasks we wish to support with our multimedia management and authoring system. In section 3, we list a few design desiderata, followed by a discussion of the currently implemented architecture in section 4, which also contains some example interfaces. In section 5 we briefly indicate related work, and in section 6 we discuss present limitations of the mmdd and directions for future work.

2 Audience and tasks

We are placing the mmdd in the hands of faculty and student authors who are integrating multimedia technology into their work. These authors typically have very rich media organized in associative structures which do not map well onto static, linear forms like film. We have chosen to particularize the mmdd in several pilot projects: a History of Silicon Valley which is a growing web of research to be used and extended in a class this winter quarter, a History of Renaissance Theater used by members of the Drama department, and a multimedia messaging board supporting collaborative learning. A fourth application is a research and teaching archive of electroacoustic music at Stanford's Center for Research in Music and Acoustics. These are

described in the Stanford Multimedia Distributed Library proposal. [21]

A typical use is the composition, close analysis and annotation of a distributed compound document by several people. Another use is the application of content-based analysis and database operators to any personal or commonly-held media in the distributed filesystem. Some related tasks include entering digitized data, text composition, annotating slides or video with text, or conversely, annotating text with sound, slides or video. In general we want to make it easy to associate any piece or stream of renderable media – a *blob*[2] – to another, to view blobs from multiple phenomenological or epistemic perspectives, and to repurpose media across projects.

We identify several classes of users of the mmdd: (1) the browser who makes few permanent changes in the corpora's blob content or topology, (2) the author/designer who may edit corpora or re-arrange the interface in simple ways, (3) a designer/programmer who creates new interfaces and scripts behavior using graphical or textual programming environments. The mmdd provides environments which support all these activities. Although users should be able to pass freely from one sort of interaction to another, we believe that this model clarifies the tasks that confront a prospective participant.

3 Some desiderata for media authoring systems

Faced with the need to manage and depict compound media in very fluid simulation projects based on relational and rule-driven models, we sought a fairly flexible media management system which could be adjoined to our existing authoring practices. We also needed a system which would let us migrate our work to other systems as the state of the art evolved. Unfortunately, this ruled out some very interesting experiments (described in section 5). We discuss here a few of the design desiderata which guided the mmdd.

[2]John Cage rather puckishly defined music as any sound to which you pay attention. "Multimedia" – loosely defined as any juxtaposition of different types of *renderable media* – also has such an epistemologically subjective sense. The mmdd includes at least certain types of interpretable or executable media such as Hypercard stacks or Mathematica scripts; see Table in the Appendix.

- distinguish between conceptual *model*, data *format*, and data *depiction*, and support maps between each of these ontologies. To illustrate, word-processors typically have a fixed depiction – e.g. running text with embedded graphics, allow no control over internal data format, and do not encode models. (Of course this simplicity is also their strength.) Hypercard gives a great deal of control over the depiction, but limited control over internal data format and model. On the other hand, traditional database programs which are designed to handle quite complex relational models typically had fixed notions of data format, and fairly inflexible user interfaces. Most importantly, we needed to let authors dynamically reshape their database schema.

- anticipate a moving target in multimedia formats. This has at least four corollaries: (A) adopt no "holy format." (B) make as few assumptions as possible about the internal structure of a blob; (C) retain the ability to translate authors' work to future object-oriented file-systems and other databases; (D) make it possible to upgrade blobs transparently.

  As better digitization technology or better sources are acquired, we should be able to feed new data into the mmdd without troubling the authors unless they request notification.[3]

- support multiple, variable depictions and associations of media.

  Rather than spend much effort casting multimedia structures into some universal document formats like Adobe Acrobat or HTML, the mmdd sidesteps altogether the document metaphor used by word- processors, at least its static conception, in favor of dynamically composing elements, and even depiction formats depending on the user's state.

- scale: decay gracefully on low-end personal computers, but take advantage of more powerful services from the net if available. The mmdd works on heterogeneous network of computers, including Macintosh and NeXTSTEP (NS) systems, backed by plain unix servers.[4]

[3] A radical approach to this versioning problem would be to treat every single blob as a media stream rather than a static datum. Thus the mmdd treats time-based media as well as atemporal media on a uniform footing.

[4] We wish to balance functional power of the infrastructure with flexibility and adaptibility of the human-computer interface. This strongly motivated our decision to marry envi-

- accommodate a constellation of cooperating applications, some of which may be supplied by authors, rather than rely on a monolithic library. We encourage our authors to use their own favorite commercial applications, though for the base media formats (e.g. RTF or TIFF) we supply some lightweight viewers

- be useful to non-technical authors in non-technical endeavors by winter 1993, yet remain extensible to arbitrary media formats, annotation/indexing schemes and search metaphors.

We investigated a variety of commercial multimedia databases, but found none which satisfied all our design requirements at that time.[5]

---

ronments such as unix which can perform low-level and network functions quite well, with NS and Macintosh which carry the most carefully conceptualized HCI design environments available across the campus.

[5] We evaluated FileMaker, Atlas Pro– a geographic information system, Art Access – a flat database aimed at curators of art museums and at wirephoto services, Fetch – a flat multimedia database, Oracle, Sybase, and Versant – a robust object-oriented database aimed at large medical, geographic and engineering systems. We also examined some more experimental systems, including Parabase, Gain and Atasca.

4 Architecture

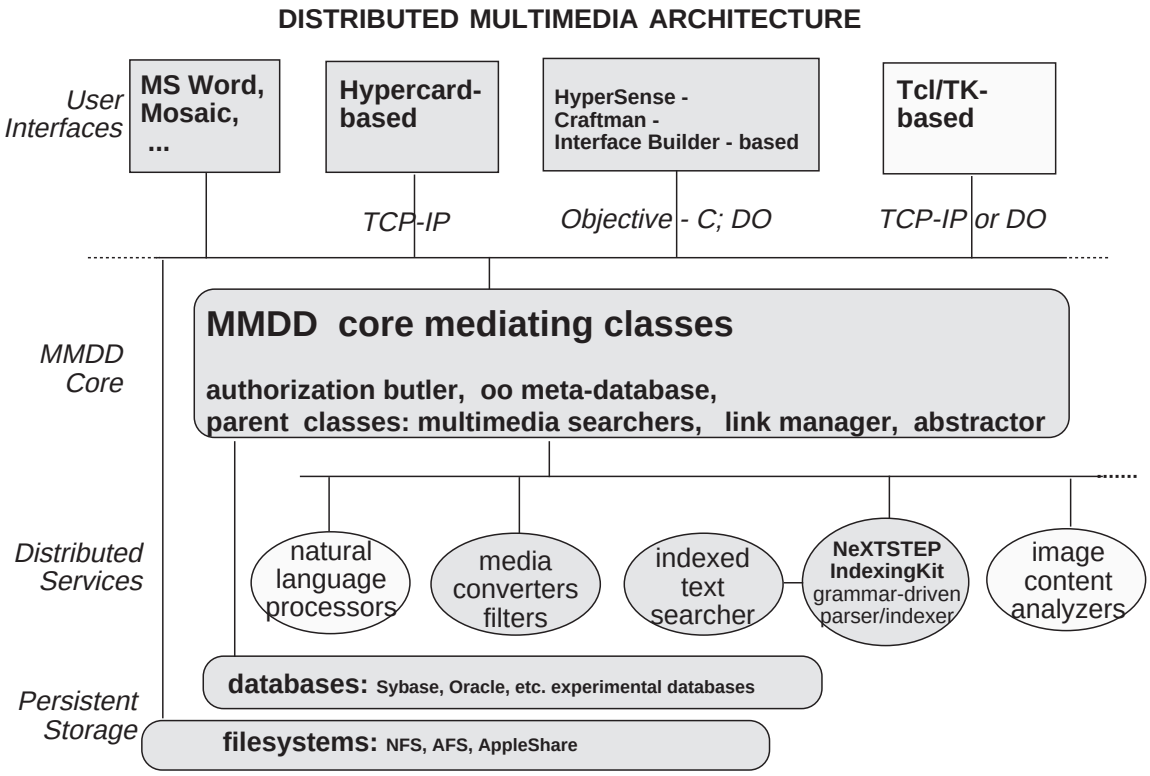

Figure 1. Schematic of the mmdd architecture (light gray indicates where future components will fit). DO = Distributed Objects/CORBA, a cross-platform object-level inter-operation protocol being developed by NeXT, HP and Sun.

Our prototype system uses NeXTSTEP (NS) and Macintosh computers, supported by Sun file-servers and an IBM database server, with inter-object communication by Objective-C, a TCP-IP based API, and Distributed Objects.

4.1 Front ends – *optics*

Figure 2. A NeXTSTEP(NS) virtual light table showing databased abstracts of a scorefile, a Mathematica program, a WriteNow document, a sound sample, a live video channel, a several TIFF images, an MPEG digital video (also in foreground), and a 3D RIB file, viewed by *geomview* (in background). Abstracts may be of any type – text, voice, etc.

A key aspect of the mmdd is that there are multiple "optics" through which one can view into and manipulate the multimedia web structures. For each environment which we support, the mmdd provides a set of reconfigurable interface objects out of which author-designers can assemble front ends. Front ends talk to the mmdd either by invoking the public methods of the mmdd classes (section 4.3), or via TCP-IP using a simple application programming interface. (see 4.5) The user interface objects include a polymorphic MediaView which can display multiple types of media, a CompositeView which performs some layout and tiling of daughter MediaViews, TableViews optimized for matrices of scalar types, as well as a FetchGroup which buffers queries to the Mom and caches returned data.

For Macintosh clients, we have built a set of such local data-caching classes and interface classes in HyperTalk, and a set of XCMD's which use the mmdd's API, all packaged into template Hypercard stacks. Obviously, a more robust application developer workbench would be desirable, but that will depend on the evolution of environments from innovative industrial partners.

Under NS we have extended the dbkit's user interface classes in the Interface Builder.[14] From these components it is quite easy to (re)fashion viewer/composer applications customized to the needs of our faculty and student authors. We have produced half a dozen viewer applications tailored to various tasks, such as cataloguing, navigating and weaving compound media, allowing various intensities of reading or authoring. (see videotape)

For example, Figure 3 shows one early Macintosh workspace for the the Elizabethan Renaissance Theater project. The item of interest occupies the large focus field, with linked media around the border. This particular deliberately simple interface emphasizes direct manipulation. A browser can pull a media object into the center of attention, make links, search, and add voice or text annotations. Since the user interfaces are written in Hypercard and Interface Builder, we can quickly re-arrange the interfaces without breaking database service.

Figure 3. A Macintosh workspace, showing drag-drop of an item from the Elizabethan Renaissance Theater corpus into the focus field, with linked media in border.

We emphasize that the mmdd supports the dynamic composition of views in content and even form, depending on the state of the user. Figure 4 shows a NS "detail-view," the main image comes from a fileserver, annotations and

links' abstracts are culled from the mmdd databases. Some reformatting can also be performed dynamically, for example, to present musical media instead of visual media.

Figure 4. A dynamically constructed view of one media entity – Pantaloon – from the Renaissance Project. Linked "daughter" images appear as sound and image abstracts on the left, and research assistant's comments appear below.

This particular interface shown in Figure 4 shows several navigational methods with which we are experimenting: tracing links represented by non-text media objects, profile search (query by example), term search in selected meta-data fields, and full content text search.

To complement standard boolean queries on meta-data, which we provide to expert users who are familar with their application's schema, we also experimented with alternatives such as multiple pools for cascaded search and more free-form gathering (Figure 5). Authors can assemble personal sets of media by combining boolean searches with more intuitive gathering and winnowing gestures. Authors may catalog new material by dragging items from other pools or directly from the filesystem into personal pools. Authors can also make links by a similar gesture. This takes advantage of the NS appkit's universal *drag-drop* service.

Figure 5. Multiple pools for cascaded search and browsing of personal "light tables."

The mmdd interfaces are integrated with sister applications via NS Services: selections can be sent to applications such as Mathematica with graphical or formatted text results returned to originating views. Figure 6 shows NS Digital Librarian, which provides simple full text indexed search on words or regular expressions.

Figure 6. Digital Librarian, a NeXTSTEP text-search application which we have used as a prototypical indexed content-search service.

4.2 Core multimedia database classes

The mmdd's heart is written as a set of Objective-C classes, epitomized by Butler – authenticator/accessor, Mom – multimedia object manager which is the Objective-C representation of multimedia database entities, Searcher – object-oriented query, and Linker – generalized hyperlink manager. We describe only these principal parent objects.

4.2.1 Blob structure

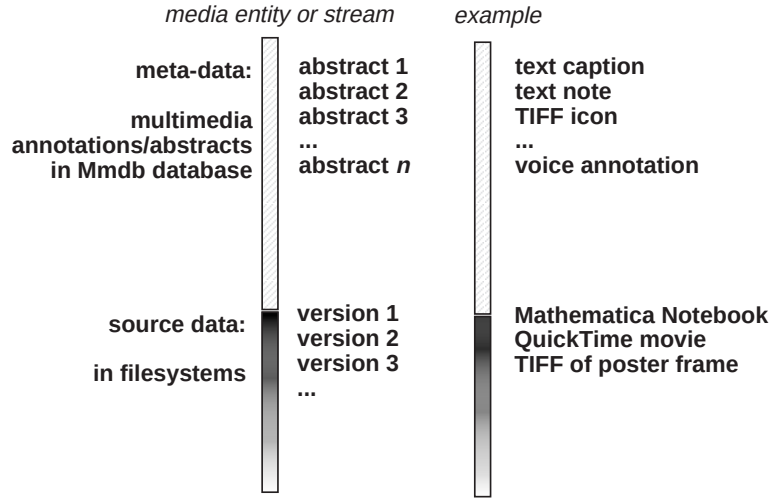

Figure 7. The media entity is a logical construct with a meta-data vector of abstracts in multiple media, and an equivalence class of representations stored in the filesystems.

Blobs are described by meta-data descriptors such as *id, name, location...* in an extensible structure where only the location is expected but not required of the client – the Mom generates an id and fills in a timestamp upon creation of a new meta-data record.[6]

The heart of the mmdd's flexible handling of media lies in the fact that its descriptor fields may be arbitrary (Objective-C) objects, or even raw data wrapped in Objective-C Data objects. A further, practical flexibility comes from allowing multiple versions of a logical media entity, where the equivalence is defined by the application, not the database. Virtual blobs may be created and linked to support arbitrary logical hierarchies. Note that our atomic blobs are abstract media objects, not physical files or logical documents in the traditional sense of word-processor programs. Thus, the Mom can manage composite blobs as well as *selections* within blobs, an important uniformization of the hypermedia model.

[6]For some (most?) purposes, network time-service could provide a "univeral" timestamp, but some clients may wish to define their own time-stamp service.

4.2.2 Butler

A Butler maps users to equivalence classes of authorship ranging from full creation/destruction of database schema to passive browsing, connects to requested databases (see discussion of SQL servers below) and instantiates Mom objects which represent the Objective-C incarnations of database entities.

4.2.3 Multimedia object manager

Blobs are always stored by reference in a Mom, along with multimedia indices and other descriptors. The heuristic is that indices, of whatever type, be it text, graphic, or sound, must be small enough to be delivered *en masse* across the network in a few beats. Some indices may be Objective-C objects of arbitrary type, including any serialized object from any client which adheres to the API; thus, in a descriptor field of object type a Mom can archive entirely different data such as a Macintosh PICT or an NS Sound, though in practice this is not done, to simplify life for the client applications.

The Mom makes essentially no assumption about the internal structure of a blob (see discussion of blob type below). It stores abstracts, generalized thumbnails, of a blob, and can serve up references to one of several versions of a given blob in support of the graceful decay design principle. For example, we can take Kodak Photo-CD images and keep versions in different sizes and bit-depths to speed transport to machines with limited graphics rendering. A more interesting example would be to keep layered wavelet-compressions of a digital video.

4.2.4 Search class

The Searcher class abstractly supports search on indices against a pattern, plus methods to generate an arbitrary boolean query against object-patterns where the comparison operators may be defined either by subclasses of the Searcher or by the index and pattern objects themselves. Content-based searches, being slower and format-dependent, are relegated to subclasses which may call upon parallelized methods. (see section 5 below) Some blob descriptors are defined with an eye toward storing metrics for proximity, fuzzy, and more general searches. While we expect specialized search classes

to maintain their own dynamically computed metrics, the mmdd can store precomputed norms. (See our discussion

4.2.5 Links and Linker class

Links now refer to whole blobs as atoms, so that the granularity is fixed at the moment of scanning/digitization or fairly soon thereafter. (see section 5 on limitations and future work.) Links are databased separately from the media, which avoids marking up data files. For portability, we decided against linking mechanisms specific to an operating system or application.[7]

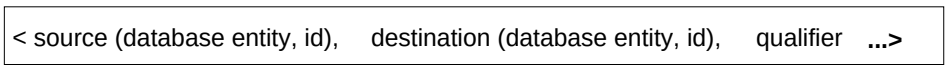

Figure 8. Extensible link structure: source and destination blobs may be virtual, or *selections* within physical media.

The link structure is quite simple, but extensible. We attempt to give universally unambiguous blob ids by joining database entity identification with an identifier unique within that database. Thus we can annotate one blob even by entities drawn from foreign databases, such as a bibliography database extracted from the university library system and from our faculty authors' private databases.

With such a general link structure, we can link any two blobs in the mmdd, which may be files, Objective-C objects, or even virtual objects, logical entities like "The Commedia Harlequin" to which the author can link other entities. We have traded efficiency for generality, but have the option of writing link maps into the SQL engine's tables to take advantage of its native relational operators.

The parent Linker class provides a few convenience functions to trace links based on source, destination, or a string qualifier. This provides the basis for handling link maps which may be owned by a particular person or class of people. Since links are first-class entities in the mmdd, applications may apply Mom and Searcher objects directly to the set of links, or even apply their own functions. This provides great flexibility and extensibility to alternative link management strategies.

[7] However we have encouraged students and faculty to compose links using native viewer applications such as Microsoft Word or NS Edit, to gain some experience with hypermedia forms.

4.3 Distributed services

4.3.1 Relational database services

We use commercial entity-relationship engines to provide ordinary search and sort on atomic types, relational database functions, and meta-data storage. We take advantage of the NS dbkit[14] to insulate our mmdd from the peculiarities of particular SQL engines, so we can easily connect to any number of popular commercial relational database engines on the net. For example, we are currently experimenting with Sybase servers on IBM RS6000 as well as NeXT computers.[8]

4.3.2 Filtering and abstraction

The mmdd currently creates abstracts of images automatically, but is able to store abstracts of arbitrary type. Although the mmdd allows clients to store arbitrary, even OS-specific blob types (e.g. voice annotations), it guarantees access across unix, Macintosh, and DOS/Windows only to the equivalence classes of TIFF and RTF. By an equivalence, we mean any approximate information-preserving transformation of data-format. For example, public-domain pbmplus and its cousins provides inter-conversion among approximately 40 graphics types, including TIFF, EPS, Mac PICT, and GIF. Although only isomorphic filters will be allowed for some critical blobs, in practice, somewhat lossy transformations are acceptable because our authoring model follows a policy of flowing media from archival locations to the clients.

4.4 Communication

4.4.1 Inter-object communication

The mmdd's core objects communicate via three alternative protocols: (1) direct Objective-C method invocation, (2) TCP-IP based application programming interface modelled after SQL but extended to arbitrary non-text patterns and attributes, and (3) NS Distributed Objects (DO, PDO). The mmdd's NS viewer applications simply invoke the methods, though of course

[8] Any database service for which there is a NS dbkit adaptor may be used. These include Oracle, dbase, INGRES, and NS IndexingKit.

they could use distributed objects, while the Macintosh viewers use the API. The distributed objects protocol provides an elegant way to migrate computation across different hosts and base operating systems.

4.4.2 Storage, transport, conversion

Here we lie at the mercy of our networks, but the mmdd makes no assumptions about the media formats or network protocols, and so can inherit whatever solutions are implemented. Again, the mmdd should be viewed as a distributed media manager which is not directly concerned with network issues such as synchronization and latency.

Blobs are archived as ordinary Macintosh and unix typed files. We anticipate that databased file-systems will eliminate many of the circumlocutions currently required by this file-based object store.

To import foreign text-based databases, we have writtten Mathematica functions which map string tensor spaces of any dimensions. We sacrifice speed for extensibility and generality across operating systems and table formats. Although we recognize that this is unacceptable substitute for dynamically adjoining foreign database servers, we see no alternative until certain protocols like Z39.50 or distributed objects are standardized and widely supported.

As noted, we also can import PICTs or RTF structured text from the Macintosh to our unix servers. Conversion and filtering services have been designed as distributed services in a parallel project, a modest version of the general media transcoding initiative.

5 Related work

5.1 Information depiction, user interfaces

A prototype LISP-based, object-oriented information system was written by Cutting et al. at Xerox PARC [6], but it was specialized for text data. Some pioneering work has been done by Rao and colleagues [19] in Xerox PARC's Information Grid project in the domain of object-oriented viewer application kits. Xerox PARC's Information Grid shows promise as an embedding architecture in the similar spirit as the mmdd, though as described, it focuses on retrieval of application objects as its top level mechanism for accessing

user information, documents, or services, an approach echoed by some commercial firms such as Oracle. Such application-centric systems may be too coarse-grained in the paradigm of lightfooted search agents in databases of information.

In the domain of user interface kits, one system which has garnered some attention is Ousterhout's X-based user interface specification language Tcl/TK [16]. However, we need user interface systems which can be used by fairly non-technical designers, yet achieve the broad design space needed to support present-day multimedia simulations like the Paris Theater (Regnaut [22]), not to speak of even richer interface gestural models supported by systems such as J. Lanier's VPL [10] which provided a language by which applications could do unto each other as humans would do unto them.

5.2 Model – format – depiction trichotomy

Klas et al. discussed similar fundamental principles, such as distinction between "the modelling and the presentation of objects" [9]. They spent a large amount of effort on versioning, and claimed that nonlinear document structure allowed "easier" navigation than linear structure. Many of the "subtle" issues concerned with connecting the presentation layer to the conceptual layer have been solved at a fair degree of generality in the mmdd, though much more work remains, of course.

Trehan and colleagues [20] devised an object-oriented hypermedia system with features similar to some aspects of the mmdd. However, they re-derive the software superstructure from a custom programming language (concurrent OO C), and their system is intimately bound to a set of X-Windows editor tools. This results in a fairly porous set of classes ill-matched to the rich application universe found on Macs and PCs. Nonetheless, Trehan et al.'s abstraction of database collections, media classes, and tools agrees in spirit with corresponding aspects of the mmdd. The mmdd enjoys certain advantages by relegating constructs such as stream objects and inter-object communication via distributed objects to a layer of which can be provided by the ambient development environment (NS). Thus the mmdd builds upon a much richer stratum and can directly support the experiments at hand.

Andreas Paepcke and colleagues [17] designed an object-oriented cover for a database to support more flexible forms of text-searching mediated by some query and presentation classes. They have also recognized the need for a

deeper semantic connection between user interface objects and computational services.

5.3 Services

While Gaines and Shaw [7] also aim to provide services such as semantic nets, (numerical) simulation engines in addition to hypermedia, unfortunately their entire layout is rigidly fixed on the metaphor of the paper document. This excludes a huge design space which could integrate such facilities in radically different ways, as some of the mmdd applications require. Moreover, they locked their extensions underneath a monolithic Macintosh application, rather than cooperating with relatively lighter weight modules.

5.4 Case study, comparison with World Wide Web and NCSA Mosaic

. While we have exported some of our corpora also via World Wide Web [3], [2], WWW and NCSA's design significantly differs from the mmdd's (see section 4), and do not serve the mmdd's spectrum of media and interactivity. We briefly contrast the two designs to more clearly situate the mmdd:[9]

- NCSA Mosaic, and HTML, are based on a word-processor document model: text with inlined media "raisins," which is too rigid for the sort of interactions the mmdd is designed to support, such as staged video sprites and dynamically defined compound objects like "classroom activity" or "historical actor." The mmdd supports a world of co-equal media entities, which may be anything ranging from QuickTime movies, Hypercard stacks or Mathematica programs to streams of music or video.

- Mosaic and WWW have no native design for mediating database services. CGI's represent a relatively limited stopgap measure via static data forms which do not streamline the real work of cross-database communication, query-generation, model management, all of which are

[9] While we focus most of our comparison at the level of the user interfaces, it is important of course, to distinguish the underlying World Wide Web architecture from the interfaces that happen to be written for it.

uniformly and dynamically handled by the NS dbkit's object-oriented methods.

- Mosaic requires that its documents be marked up in a "holy format" derived from SGML. The mmdd, via its object-oriented database structures, requires essentially no assumption about the internal structure of a blob, yet maintains meta-information and supports database functions on the blobs. Specifically, whereas links must be explicitly written into HTML documents, a laborious and error-prone operation, mmdd's link-maps are transparently written into independent databases by simple user gestures, such as *drag-drop* under the NS Workspace and Mac front ends.

- Mosaic front ends have frozen layout and functionality and, most importantly no schema editing at all. By contrast, the mmdd is designed to let authors revise their own schema. Mmdd interfaces can be reconfigured in minutes. Whereas XMosaic and Windows require installation of viewers which must conform to some Mosaic-specific protocols, mmdd front ends exchange structured data with ordinary commercial applications familiar to users.

- In contrast to WWW's strategy of providing libraries including special-case media viewers on an *ad hoc* basis, the mmdd's object architecture provides a rational and flexible framework (described below) for integrating any service.

6 present limitations and directions for future work

We intend to migrate client corpora from the mmdd to true databased, object-oriented persistent storage systems which should replace current filesystems in the forseeable future.

Currently though it is quite easy for a nonexpert to edit the base database schema using existing schema editing tools, the end-author may not dynamically change the schema. We intend to allow the end-author to define new typed attributes dynamically. Experience with other multimedia simulation

projects suggests that dynamic schema editing is vital during the design phase of a mmdd project, but not used by browsers. (also see Batra[1].)

We wish to test the extensibility of our system in at least two significant dimensions, perhaps by implementing reasonably refined query by image content methods (IBM, [15]), natural language content analysis (NASA [11]; Apple [18]), or video content analysis and captioning (Media Lab [13], SRI).

6.1 Content-based filter services

A fundamental problem with any media service is the *conversion* of file formats. While the mmdd itself needs minimal information about internal blob structure, its clients must be able to communicate media in their preferred formats. For example, a Macintosh drawing program could profitably be used by an author to edit a non-Macintosh document on the network. Although some mature applications attempt to import and export multiple file formats, the efficient solution would be to have the *ambient operating system* provide such services to all applications. This elegantly provides extensibility and factors out demanding computational tasks from the mmdd.

A related problem is that, sophisticated searches usually benefit from some content-based pre-processing of the documents. This is especially critical with large text corpora or nontext media. Such pre-processing can be regarded as an *abstraction* service, which again, should be provided as a general service.

Formally, abstraction and conversion services are independent of the core mmdd classes, but we expect to tightly integrate them into our general scheme.

Modelled loosely after the elegant NS Services which automatically interposes filters between vendor and consumer applications, such abstraction and conversion services would be dynamically extensible. Two key differences are that non-NS clients of the mmdd should be able to request filtering via the mmdd, and that the services may distribute to whatever compute servers can best perform them. Of course, this places the burden on siting compute servers close to either the fileservers or the client applications.

- **Abstraction service**

To search a database rapidly, one usually needs a set of indices. Content-based searches require richer indices which may be generally viewed as "abstracts," like the abstract of a traditional research article, or the poster frame of a QuickTime movie.

The polymorphic abstractor automatically summarizes files in a way which is sensitive to type, and perhaps content. The abstractor is easily modulated, e.g. by scripts or subclassing. For example, upon encountering a directory which is actually bundled as a single hyper-linked document (as is common with unix word processors or mm mail) the abstractor might create a single RTF file summarizing the contents. Or, encountering a Mac PICT file, it may create Mac icon, as well as TIFFs in several sizes and bit-depths. Another example would be to extract the "author," "title" and "abstract" structures from TeX documents. A Searcher object can invoke abstractors and request special work like pre-processing graphics to prepare for queries by image content, or running FFT on a soundfile.

There is a large body of research, partitioned by media type, on automated interpretation of documents ranging from natural language text interpretation [4], to image interpretation (Niblack et al. [15], Hirata et al. [8]) and video interpretation (Matthews [13]). Note that we do not require semantic analysis, merely "dumb" heuristics to make thin versions of documents for transport and for search or classification. Examples are the scatter-gather text methods by Cutting et al. [5] and music recognition (e.g. Bruce Penny-Cooke at MacGill University) and music pattern recognition (e.g. David Huron at the University of Waterloo).

- **Conversion service**

  On a more basic level, it would be quite useful to extend the model provided by the NS Services, where producer applications post announcements of the formats of a selection which they are prepared to vend and consumer applications request a selection in a preferred format. The dynamically extensible set of system services should interpose filters between media streams and convert data in transit. It would be quite useful have such services mediating Macintosh-unix connections. Such an intermediary radically simplifies the development of applica-

tions which must work with heterogeneous media on different operating systems.

We are using combinations of netatalk, AFS and NFS filesystems, which provide limited facilities for dynamic intervention by custom filters. Experimental systems in the genre of "open documents" will be quite useful, but serve a different metaphor, that of the paper document (qv. Xerox PARC's System 33), whereas we need to support more supple transport models.[10]

6.2 Other services

We will investigate how best to provide connections to other network services. Two currently interesting enterprises are World Wide Web (WWW) and Z39.50. WWW provides a literally global but boundlessly complicated topology of interlinked documents. Its principal constraint from our perspective is that it was designed as a document delivery system, rather than as a multimedia authoring system. Nonetheless, by delivering the mmdd via a WWW server, we gain a few simple but widely distributed viewers at little cost. One may view the mmdd in this context as simply another WWW server, which could potentially become a WWW service. However, a WWW server gains no uniform deep access to sibling network services a la Z39.50. For the latter, we will depend on commercial solutions, for example in the form of NS dbkit adaptors or some other equally powerful API. Such compliance will allow mmdd clients to connect to sibling information sources.

6.3 Detached media

One of the most serious limitations of the mmdd is its dependence on network services. We would like to have authors "check out" subsets of associated media from the mmdd, annotate them perhaps at home or in the field, and write the updates back to the mmdd. This requires local applications which contain embedded instances of the Mom and its sister classes.

[10] One example comes from scientific computation: we derive a model within Mathematica on a Macintosh, but have it numerically computed on a remote compute server, and have the results converted for review on a Silicon Graphics machine. Alternatively, we could send the resulting animation to videotape, using for example, a QuickTime converter.

6.4 Links *vs.* queries

We intend to take advantage of the database's object fields to attach links to *selections* within blobs.

We are interested in exploring how far we can replace the notion of hypertext links with a more fluid mechanisms of query, and query-generation, for which we have prepared the architecture. For example, through an adaptor to the underlying grammar-driven Indexing Kit, a general grammar-driven database service which can parse and index any structure for which a grammar can be specified, the mmdd can provide such grammar-driven content-based indexed searches to any of its clients. This is potentially a a very powerful and general indexed search mechanism for the class of formally parsable media structures.

We also are planning to refine or replace our link-maps by more sophisticated methods of personalizing or contextualizing the views into the media web. We are exploring the possibility of interposing an adaptive network which would build user-class profiles from users' browsing and linking behavior or from users' more formal queries on meta-data attributes. Such work has been done, for example, by Mathé & Chen in the context of adaptive hypertext [12].

6.5 Other platforms

Finally, we have not supported X or Windows clients, except by providing a API and a TCP-IP service in lieu of a common distributed objects protocol. We regard our current distribution of media via Mosaic to such environments as a stop-gap measure to be replaced by reconfigurable UI environments.

7 Conclusion

We have described a hybrid object-oriented relational database system which manages and presents graphs of compound media on Macintosh, NeXTSTEP environments, supported by unix and AppleShare file-systems. We have concentrated our efforts on what seem to be some of the lacunae in approaches toward a casually usable multimedia authoring system spanning unix and Macintosh operating systems which may be extended to arbitrary media

formats, and arbitrary search methods in a relatively elegant and efficient fashion.

8 Acknowledgments

Deborah Zimmerman in the Academic Development Group was responsible for the Macintosh HyperCard front end, and Rick Wong implemented the tcp-ip API supporting blob-transport. We thank Tim Lenoir, Henry Lowood, Judy Dolan, Bill Eddelman, and the student assistants for participating in the design process, and we are indebted to Charles Kerns for his constant interest. We thank also Bill Verplank and Terry Winograd for their criticism.

9 Appendix

9.1 Concise representation of the mmdd

The mmdd model has a simple structure which can be more clearly expressed in more precise terms.

The mmdd structure can be viewed as a directed graph $\{, , \Sigma\}$where the nodes $\gamma \in ,$ are logical entities representing generalized *selections* in multimedia objects, and arcs $\sigma \in \Sigma$ are associations of ordered pairs $(\gamma_i, \gamma_j), \gamma_i \in , ,$ carrying extra structure. Both , and Σ carry simple but extensible structure. Each γ is represented by $\alpha \oplus [r]$, where α is a a vector of annotations and abstracts in various media types, and $[r]$ is an equivalence class of persistent data – for example a file or, more generally, a selection in a blob – modulo an equivalence defined by the application. In practice the α are stored in a relational database while the r are stored in the filesystem. For efficiency and to keep the database relatively compact with an eye toward detachable sub- databases, our policy dictates that the sizes $|\alpha_i| << |r|$, and that most queries will preferentially compare patterns against the α_i rather than the full content r. The associations may be qualified to allow individuals to create their own link maps.

In this setting, abstraction and conversion services are contraction maps $\phi : r \to \alpha$ and (pseudo-)isomorphisms $\psi : r \to r'$. For precise estimates of the complexity of the functions required, we would need to define information-theoretic topologies on , and its function spaces.

9.2 Pictures

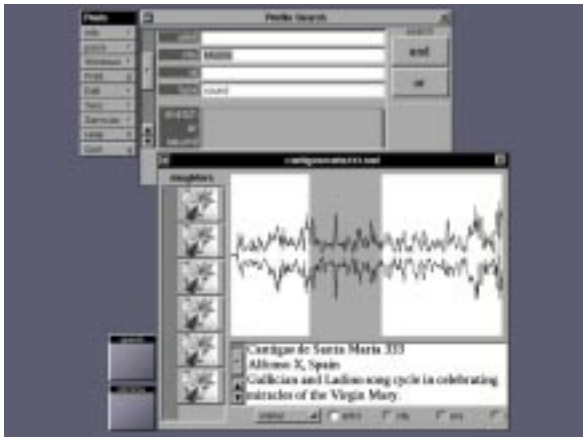

Figure 9. The Search classes support comparisons against arbitrary objects; we will accommodate several non-text content-based search methods.

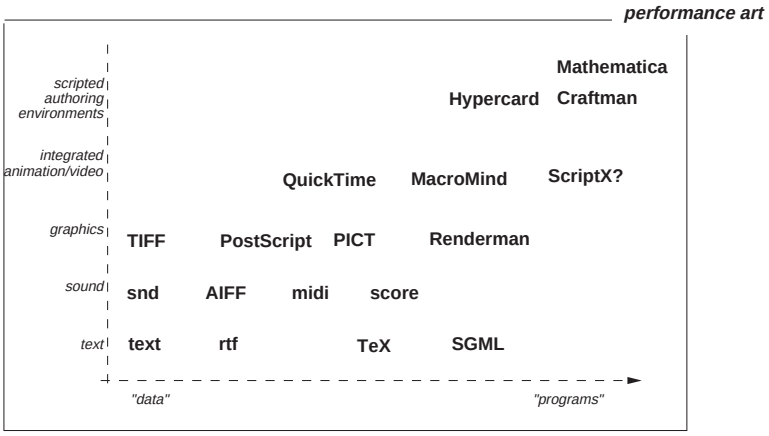

Figure 10. What is multimedia? This table is intended only to give a sense of the dimensions along which we can array some renderable media, and to indicate the scope of the need for a well-integrated, databased filesystem, perhaps based on an object-oriented model.

References

[1] D. Batra and J. G. Davis. Conceptual data modelling in database design: similarities and differences between expert and novice designers. *Intl. J. Man-Machine Studies*, 37:83–101, 1992.

[2] T. J. Berners-Lee. The world wide web, talk at Stanford PCD seminar CS 547, 1993.

[3] T. J. Berners-Lee, R. Cailliau, J-F Groff, and B. Pollermann. World-wide web: The information universe. In *Electronic Networking: Research, Applications and Policy*, volume 2, pages 52–58, 1992.

[4] Bran Boguraev and J. Pustejovsky. *Lexical Knowledge: Acquisition and Representation*. MIT Press, in preparation.

[5] D. Cutting, D. Karger, and Jan Pedersen. Constant interaction-time scatter/gather browsing of large document collections. In *Proceedings of SIGIR'93*, 1993.

[6] D. Cutting, Jan Pedersen, and P.-K. Halvorsen. An object-oriented architecture for text retrieval. In *Proceedings of RIAO'91*, 1991.

[7] Gaines and Shaw. ... In *Proceedings ACM Multimedia*, 1993.

[8] Kyoji Hirata, Yoshinori Hara, Naoki Shibata, and Fusako Hirabayashi. Media-based navigation for hypermedia systems. In *Proceedings of ACM Hypertext*, pages 159–173, 1993.

[9] W. Klas, E. Neuhold, and M. Schrefl. Visual databases need data models for multimedia data. *Visual Database Systems*, pages 433–462, 1993.

[10] Jaron Lanier. personal communication. (see reference in Sci. Am), 1988.

[11] Nathalie Mathe'. Facilitating access to information in large documents with an intelligent hypertext system. In *Proceedings of the 9th AIAA Conf. on Computing in Aerospace*, 1993.

[12] Nathalie Mathe' and James Chen. A user-centered approach to adaptive hypertext based on an information relevance model. In *Fourth Int'l Conf. on User Modeling*, Aug 1994.

[13] J. Matthews, P. Gloor, and F. Makedon. Videoscheme: A programmable video editing system for automation and media recognition. In *Proceedings ACM Multimedia*, pages 419–426, 1993.

[14] Inc. NeXT. *NeXTSTEP*. Addison Wesley, 1992.

[15] W. Niblack, R. Barber, W. Equitz, M. Flickner, E. Glasman, D. Petkovic, P. Yanker, and C. Falouttsos. The QBIC project: querying images by content using color, texture, and shape. Technical report, IBM Almaden, 1993. Research Report RJ 9203 (81511).

[16] J. K. Ousterhout. An X11 toolkit based on the TCL language. In *Proceedings of the Winter 1991 USENIX Conference*, pages 105–115, 1991.

[17] Andreas Paepcke. Viewing heterogeneous text databases: object technology works for the view and its implementation. Technical report, Hewlett-Packard Lab, 1992.

[18] J. Pustejovsky and Bran Boguraev. Lexical knowledge representation and natural language processing. *Artificial Intelligence*, 1993.

[19] R. Rao, S. K. Card, H. D. Jellinek, J. D. Mackinlay, and J. D. Robertson. The information grid: A framework for building information retrieval and retrieval-centered applications. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, 1992.

[20] Trehan, Sawashima, Yamaguchi, and Hasebe. Toolkit for shared hypermedia on a distributed object oriented architecture. In *Proceedings of ACM Multimedia*, pages 175–182, 1993.

[21] Sha Xin Wei, Robert Street, and Michael Keller. NSF proposal: A multimedia distributed library. Technical report, Stanford University, 1994.

[22] Fabienne Zimmermann-Regnaut. Interactive memory of theater: Paris Theater. In *Conference at Univ. Paris VIII*, Dec 1993.