Multimedia Authoring Systems

Ross Cutler* Kasım S. Candan†

1 Abstract

In this paper we have surveyed three multimedia authoring systems (Multimedia Toolbook 3.0, Director 4.0, and Authorware 3.0). Each system uses a different metaphor (book, movie, and icon-based flowchart) for creating multimedia applications. A sample application is developed in each MAS and the effects of the corresponding metaphors are compared. Technologies like *ODBC*, *OLE*, *DDE*, *DLL*, and *MCI* are discussed and we have shown how MAS's can benefit from them. We have also discussed the limitations of the current systems, and where the research in this area is headed.

2 Introduction

Multimedia software applications have become a multi-billion dollar business. Typical multimedia applications include:

- Product demos (e.g. Multimedia Toolkit demo)
- Kiosk applications (e.g. concert ticket ordering)
- Computer Based Training (CBT) (e.g. Microsoft Excel tutorial)
- Games (e.g. Iron Helix is written in Director)
- Multimedia References (e.g. MS Encarta, 747 repair manual)
- Education (Multimedia textbooks and courseware)

However, the growth of the multimedia software market has not kept up with the hype that multimedia could get in the press. This slowness is in large part due to the inadequacies of the tools supplied for the developers. For example, it is only with in the last two years that cross platform tools have become available, and it is even more recent that these tools have started to provide interprocess communication and database connectivity.

Multimedia applications can be developed using traditional tools, such as C++ (if provided with graphics, sound, hypertext, animation, video, and database libraries). However, the number of tools that a developer would have to learn in order to write even a simple multimedia application is quite daunting. A better alternative is to use a multimedia authoring system (MAS).

*Email: rgc@cs.umd.edu
†Email: candan@cs.umd.edu

MAS's are high-level systems providing a unified environment for creating multimedia applications. A major advantage of MAS's is that they can be used by non-programmers (e.g. domain experts) too. Nevertheless most systems come with programming languages for advanced applications. A typical MAS includes tools for hypertext, pictures, animation, video, sound, database connectivity, and interprocess communication.

In this paper we will concentrate on three MAS's:

- Asymetrix's Multimedia Toolbook 3.0
- AimTech's IconAuthor 6.0
- Macromedia's Director 4.0

These systems are chosen because (1) each have proved to be very successful in the market, and (2) each uses a different metaphor to create multimedia applications. More specifically, Toolbook uses a *book* interface, Director uses a *movie* metaphor, and IconAuthor uses an icon-based *flowchart*. Director and IconAuthor are cross-platform systems, and run on Microsoft Windows and MacOS systems (Director applications also run on the 3DO which is an inexpensive multimedia game system). Toolbook only runs on Windows platforms, but because of its popularity and the predominance of Windows in the marketplace, we have chosen to include it in our paper as the representative of the *book* metaphor.

In order to evaluate and compare each MAS, we have built a simple application in each system. This application is one that you might find in a video tape rental store (or in a video-on-demand system). The application allows users to query a database of available movies. For example, the user could search for all in-stock movies that are dramas, casts Gary Cooper, and won an Oscar. For each of such movies, the user would be able to recieve a short review and a small video clip. The user can also check out the movie using the application, which would notify a clerk to get the movie from storage and hold it at the checkout desk.

Before getting to the details of each MAS, we first discuss some of the common technologies: $ODBC$, OLE, DDE, DLL, and MCI(note that Director does not support OLE, $DDE$and $ODBC$).

3 Underlying Technology

3.1 ODBC

The Open Database Connectivity ($ODBC$) interface allows applications to access data in database management systems (DBMS) using the Structured Query Language (SQL). The interface permits maximum interoperability – a single application can access different database management systems.

This technology allows an application developer to prepare, compile, and ship an application without targeting a specific DBMS. Users, then, can use some modules (database drivers) to link their choice of database management systems to their application.

As a result, developers do not have to write database specific code for connections or communications of queries and results.

3.2 OLE

Object Linking and Embedding (OLE) is a technique that is extensively used in Windows and MacOS that enables users to create an object in one application (the server application), and then incorporate it into a file from another application (the client application).

There are three types of OLE objects:

- **Linked Object**: An object that is stored in a separate file called the source file.
- **Embedded object**: An object that is a copy of the original object but stored within your document.
- **Static object**: An object that is originally imported into a document as an OLE object, but whose connection to the server application has been severed to ensure that it cannot be edited.

Using OLE, it is possible to create applications that integrate the capabilities of many different OLE compliant programs. For instance, Toolbook does not include an equation editor, so to include equations in an application one needs to pick an equation editor (e.g. EquationEdit) and write/edit the equations through OLE. If one wishes to prevent the readers from modifying the resulting equations, then s/he can covert them into static objects.

A future version of OLE is expected to let applications share objects across a network.

3.3 DDE

Dynamic Data Exchange (DDE) is a Windows and MacOS communication protocol used for exchanging data between applications or for executing commands in other applications.

A DDE conversation consists of two or more applications. The client application initiates the conversation, while the server application responds to the client's requests.

DDE has also been extended to $NetDDE$ allowing conversations across networks.

3.4 DLL

Dynamically Linked Libraries (DLL's) is a technology that allows libraries to be dynamically loaded at runtime. DLL's can be used to add libraries of routines to MAS's. For example, if one wants to add a MPEG-2 video player to a MAS that does not come with one, s/he can do so via a DLL. The MPEG video player can then be called in a similar fashion as other (built-in) library calls.

3.5 MCI

The Multimedia Control Interface (MCI) is a communications standard that is used to control multimedia devices like video disks, CD players, camcorders, etc. For example, a multimedia application can use a video disk player to provide video clips, or a CD jukebox player to provide sample sound bites.

4 Sample Application - "Find-Movie"

We now discuss the sample application that we have implemented in all three systems. The application mainly consists of three screens: *title*, *query*, and *result* screens.

The *title* screen is what is displayed when the application is idle. It contains a moving text string ("Press any key") and random movie stills (which fade in and out in random locations on the screen). When a key is pressed, the query screen is activated.

The *query* screen allows a user to construct a query using the following criteria:

Attribute	Type
title	char[]
director	char[]
release_date	date
movie_id	int
genres	char[]
mpaa_rating	char[]
num_in_stock	char[]
star_rating	int
clip_filename	char[]

Table 1: `Movies` Table

Attribute	Type
movie_id	int
actor	char[]

Table 2: `Actors` Table

- Movie Title
- Director
- Genres
- MPAA Rating
- Actor/actress
- Star Rating (1-4)
- Release Date (range)
- In Stock (Y/N)
- Awards

Once a query has been constructed, the user can press the "Search" button to send a SQL query to a movies database system through *ODBC*. In all likelihood, there would be one database server per store, which would be accessible via a network from each kiosk. The sample database consists of the following three tables: `Movies` (table 1), `Actors` (table 2 and `Awards` (table 3).

The *result* screen displays a list of movies satisfying the given query. When the user selects a movie from this list, a short review and video clip of the movie is displayed. Video clips and reviews are stored on a central file server. The format for the video is MPEG, and the format for the reviews is Rich Text Format (RTF). If a user decides to check the movie out, the application sends a *NetDDE* message to the clerk's computer (which runs a *DDE* server) to inform the clerk.

Attribute	Type
movie_id	int
award	char[]

Table 3: `Awards` Table

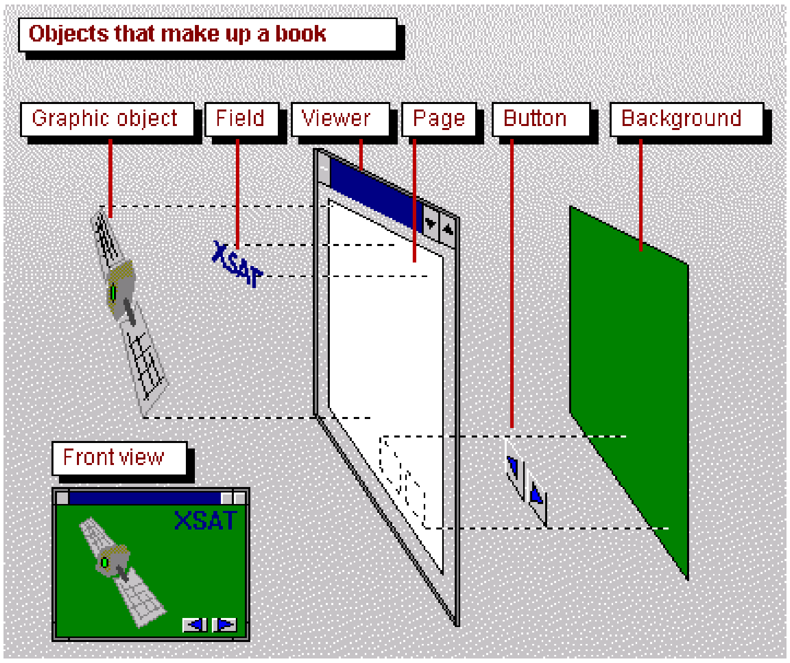

Figure 1: Toolbook book components

5 Multimedia Toolbook 3.0

Multimedia Toolbook uses a book metaphor to build multimedia applications. The components of a book include

- graphic objects,
- fields,
- viewers,
- pages,
- buttons, and
- backgrounds.

A book is built by creating pages, placing objects on the pages, and writing scripts in Toolbook's OpenScript language to perform actions (see Figure 1).

Toolbook provides an object-oriented event driven programming environment; however the scripting language is not object oriented (i.e., it doesn't provide classes, inheritance, or encapsulation).

In our sample application, the *title* page is expected to display random movie stills and a scrolling the text ("Press any key to continue") across the bottom of the screen. Figure 2 shows a snapshot of this screen. The text label that scrolls in this page is a *Field* object with the following attached script:

```
-- on idle, makes text scroll horizontally
notifyBefore idle
        x = item 1 of my position
        y = item 2 of my position
        dx = item 1 of my size
        width = item 1 of size of this book

-- wrap the text to the left hand side of the screen
        if x > width
             x = -dx
        end

        set my position to x + 30, y
end
```

The text label is animated simply by changing its position by 30 units to the right each time the idle handle is called. Note that the idle handle is activated whenever no other events are being processed.

A *Paint* object is used to display a series of movie stills. Unfortunately, it is very difficult (but still possible) in Toolbook to dissolve individual items, so in our sample application no special effects are performed when the stills are changed.

Finally, to handle the key press, the following script is attached to the *Page* object (which is called "titlePage"):

```
to handle keyDown
        fxDissolve fast to page ''queryPage''
end
```

When a key is pressed, the *title* page dissolves and the *query* page appears instead (page transition special effects are easy to handle in Toolbook).

The *query* page (Figure 3) uses a series of *ComboBox* and *Button* objects to build a query. Once the user constructs the query and presses the "Search" button, a SQL statement is generated (using the *ComboBox* and *Button* objects). This SQL statement is then sent to the store's database server via *ODBC*. The results are read into a global array[1] with elements of the form {movie title, review file, MPEG filename}. These results are then displayed on the *Results* page (Figure 4).

If the user clicks on a particular movie in the *movies* field, an *OLE* connection is made with the application "QuickTime Player" to play the associated MPEG file. The *play*, *stop*, and *pause* buttons can be used to communicate with the "QuickTime Player" to control the movie.

If the user wants to check out a movie, s/he can do so by pressing the *Check-out* button, which has the following script:

[1]The array is created global because it is the most efficient way to transfer data between event handlers.

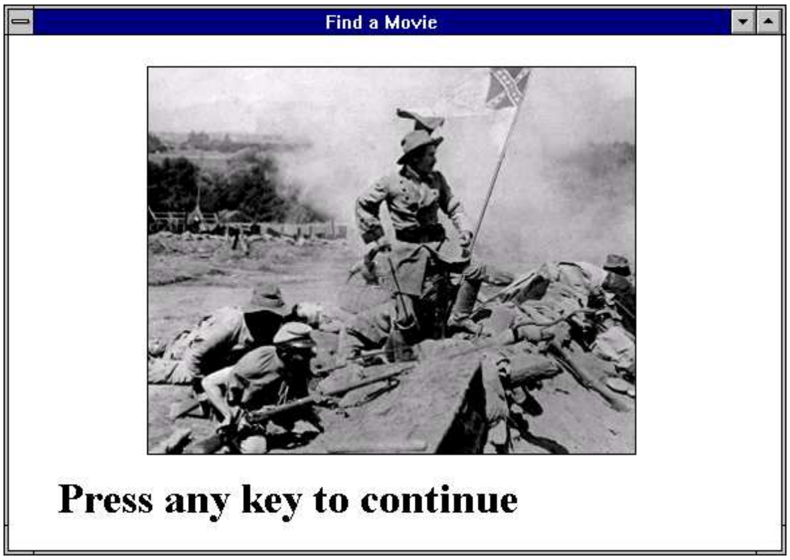

Figure 2: Title page

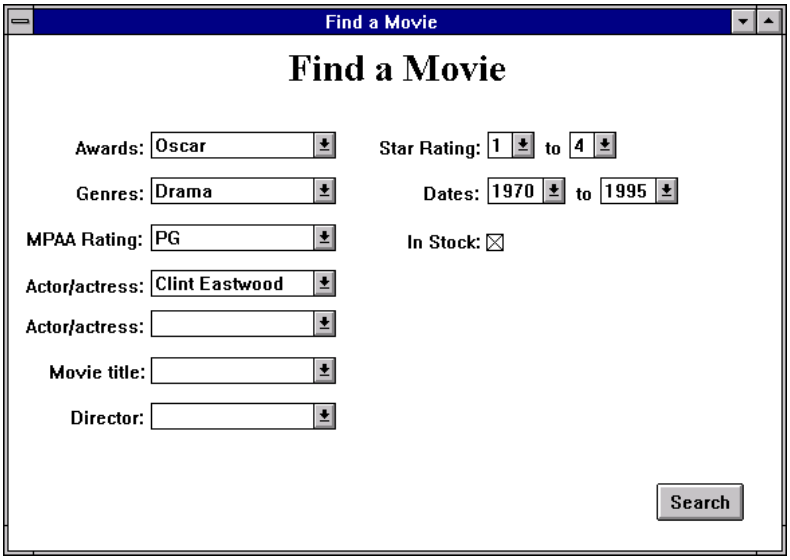

Figure 3: Query page

Figure 4: Results page

```
to handle buttonClick
        -- check out the selected tape
        tape = selectedText of field ''moviesField''
        executeRemote ''send checkout'' & tape application ''clerk''
end
```

This script uses *NetDDE*to communicate with the clerk's Toolbook program which acts as a *DDE* server. When this server recieves a "checkout tape" message, it adds the name of the tape to a scrolling field object on the clerk's screen and beeps to notify the clerk. Note that there is no need to specify what computer the clerk is at on the network, this flexibility is achieved by using a network administrative program to setup a *DDE* "*share*" (which is similar to a disk or printer share used in Microsoft networks).

As one can see, Toolbook is quite a powerful tool for authoring multimedia applications. Unfortunately, even our simple application required significant programming.

6 IconAuthor 6.0

Unlike Toolbook, IconAuthor does not include a programming language. Instead, applications are described using an icon-based flowchart called a structure (see Figure 5). An icon is a small picture that represents a function that can be performed. An application starts off at the *Start* icon and executes the icon immediately below it; each subsequent icon executes the icon immediately below or the the right of itself (depending on the type of the icon and the state of the program).

It is important to note that the flow of control is procedural and not event driven. IconAuthor is geared for non-programmers, and procedural control flow is probably easier for non-programmers to conceptualize.

Figure 5 shows the implemention of the *title* page for our application. The "Scrolling text" icon is a *Display* icon used for displaying bitmap graphics, animation scripts, or SmartObject files (a collection

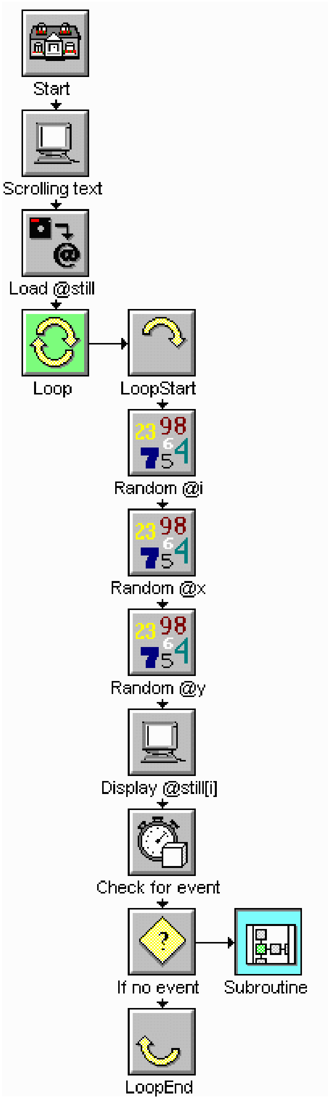

Figure 5: Title page structure

of user interface objects). In our case, we display a simple animation script which brings the scrolling text "Press any key to continue" to the screen. This script is very similar in concept to a Toolbook script, but it described using icons instead of words.

IconAuthor supports both scalar and array variables. In the icon "Load @still", we load an array named "@still" and fill it with values from a text file. In particular, @still contains a list of path names to bitmap files containing the stills that we want to display on the *title* page.

The next icon starts a loop, which can be terminated by an *Exit* icon. The "LoopStart" and "LoopEnd" icons are nothing but two dummy icons, automatically included to delimit the loop.

The next three icons in the figure assign random numbers to the variables @i, @x, and @y. The range of @i is $[1,N]$, where N is the number of stills. The range of @x and @y are $[1,dx-px]$ and $[1,dy-py]$, respectively. Here, dx, dy are the dimensions of the screen and px, py are the dimensions of the stills.

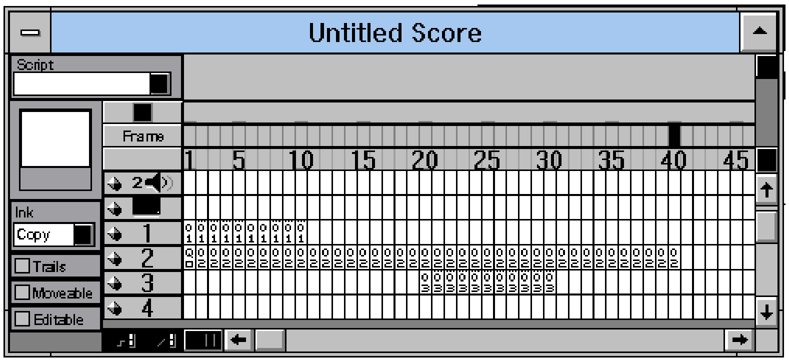

Figure 6: Director score window

The "Display @still[i]" icon displays the still @still[@i], with the dissolving effect. This is followed by the "Check for event" icon waiting for an event (e.g. key press or mouse click). If no events occur in five seconds, the icon times out and sets the appropriate system variable. If one wishes to have a reverse dissolve effect on the still, then s/he can do so by adding "Display @still[i]" icon below the "Check for event" icon.

The "If no event" icon is an *If* icon which checks whether an event is received (via the mentioned system variable) or not. If a time-out occured then then the icon below the *If* icon is executed; otherwise, the other icon (which is located to the right) is executed. In the latter case, the "Subroutine" icon is executed, and as a result the *query* page is displayed and the execution is continued from a new event loop.

IconAuthor uses icons to describe the flow of the program; objects are used for creating the user interface and providing connectivity (via *ODBC*, *OLE*, *DDE*, *DLL*, and *MCI*). For the query page, *Combo* and *CheckBox* objects are used (as in Toolbook) to generate a page very similar to Figure 3. These objects are used to build an SQL query string, and a *Database* object uses this SQL string to query the database via *ODBC*. The results are stored in a *List Box* object in a page very similar to Figure 4. IconAuthor was the only MAS to directly support MPEG among the ones that we have studied, hence an *OLE* was not necessary for implementation of the *result* page.

7 Director 4.0

Director uses a *movie* metaphor to create applications. This metaphor consist of

- a stage,
- cast members (e.g. graphics, animation, video, text, and sound), and
- a score.

A score can be thought of as a virtual piece of film. It is described by a score window (see Figure 6), which contains a matrix of cells; the matrix columns represent individual frames of the movie, and rows represent layers on the stage where cast members can appear. A cell can contain scripts, special effects, timing instructions, color palettes, and sound control. The score window allows up to 48 interactive media elements or 32,000 static objects to be onstage simultaneously.

Director uses an object oriented scripting language (Lingo) to enhance the power of the movie metaphor. Each cast member can have a script associated with it. Objects can catch events and modify the control flow of the application by jumping to a particular frame.

In the remaining part of this section, we describe how to implement our sample application in Director.

For the scrolling text, Director provides a Banner animation which does just what we want (with no programming). A Banner animation is a cast member, so we include that member in the score whenever the *title* page is visible. For example, Figure 6 shows cast member 2 (which is the Banner animation) in frames 1-40. We also wish to display a series of movie stills (in this case the movie stills are not random). Each movie still is included in the score as a cast member. For example, in Figure 6, two stills (cast members 1,3) are displayed in frames 1-10 and 20-30, respectively. By modifying the blend attribute for these members, we can make them fade in and out (other special effects are also possible). The next thing to do is to include more movie stills in the score, put a loop at the end of the score so that it repeats, and add an event handler to the stage which will cause a jump to the *query* page when a key is pressed.

During the implemention of our sample application, we ran into some major deficiencies of Director: it does not support $ODBC$, DDE, or OLE; and it cannot import RTF (or any formated text). In fact, it does not have any built in database support. It does, however, support DLL's, and an extension mechanism to Lingo, XObject, which allows us to use a third party solution for database connectivity (e.g. $ODBC$). In theory, DDEcould also be implemented, since it is provided as DLL's (at least on the Windows platform). But the fact that these essential items are not standard, causes a detraction from one of the main goals of MAS's: "minimizing the number of tools to be learned".

8 MAS's and Current Technology

As discussed in the previous sections, multimedia authoring tools are getting more sophisticated everyday. They are becoming more attractive to both naive users and experts as they are enhanced with:

- the portability of the applications to multiple hardware/OS platforms,
- the development of GUI's which make them very easy to learn and use,
- the introduction of application independent structures allowing a large range of users to benefit from them,
- the use of technologies like $ODBC$, OLE, DDE and DLL which let the users fit their MAS's to their own needs.

In this section we study how the current research in multimedia technologies can be helpful in increasing the strength of the multimedia authoring tools, and how MAS's can be used in increasing the efficiency and productivity of the multimedia research.

8.1 How to improve MAS's?

Current research in multimedia technologies can roughly be classified as follows:

1. Research in multimedia hardware:

- Fast and efficient storage devices for multimedia objects (CD-ROM's etc.),
- Fast and high quality displays,
- High bandwidth, low delay communication medium.

2. Research in multimedia software:

   - Efficient data structures for representing multimedia objects,
   - Methods for merging/synchronizing homogeneous/heterogeneous media objects,
   - Indexing/searching of non-traditional data (video etc.)
   - Automatic feature extraction and semantic indexing,
   - Computer vision and image processing,
   - High speed operating systems (especially for mutlimedia),
   - Network protocols which satisfy a given set of quality of service (QoS) requirements such as:
     - low delay,
     - low jitter,
     - low loss rate.
   - Fast compression algorithms.

Although the above list is far from being complete, it is large enough to show that there is a huge amount of open problems waiting to be solved in this area. Despite of these problems, the demand for sophisticated applications (like *video-on-demand*) increases and this gives the researchers a certain incentive to concentrate their efforts on multimedia research. Although the current technology does not allow very sophisticated applications to be widely used yet, it is not too optimistic to expect them to be up and running in the near future. Various companies already have built prototype *video-on-demand* servers, and they are testing their systems in pilot areas.

The demand for highly sophisticated multimedia softwares also affects the designers of *Multimedia Authoring Systems*. They also suffer from similar problems multimedia researchers are currently facing, and it is clear that any advance in multimedia area will also help the designers of MAS's in building better tools. For instance, the lack of a well-studied and well-understood knowledge representation for multimedia objects prevents the MAS designers from building a system which can bring various type of media together and which can be easily extended with the addition of different media types. Most of the tools available today deal with a fixed set of media types, and they do not allow the users to add new media types to these systems as they need. For such an extensible system to be running, it should be possible to "introduce" a new *media type* to the MAS by presenting information related to the characteristics of the new media, and the corresponding objects:

- the description of the information that may be attached to the objects (i.e. metadata [**?**, **?**]),
- the constraints that should be satisfied when such an object is retrieved and/or displayed (e.g. time constraints),
- the interactions of these objects with objects of different types (e.g. How an audio object can be linked to a video object? What are the constraints that needs to be satisfied when an object of *audio* type is displayed along with the an object of the *video* type?),

- the interactions of the objects of the same type (e.g. How two play two objects of the *audio* type together? Or is it possible to do so?),

- the preferred indexing and/or storage strategies that should be used with the objects of the new type,

- etc.

Similarly, users of MAS's may also want to be able to describe how different multimedia objects can be composed together to create new types of multimedia objects. Some of the questions that may arise in such compositions are:

- Which characteristics of the parent object types are inherited?

- What are the additional characteristics that needs to be added to the new object?

- How to perform the dynamic merging and decomposition of the objects?

All of the above questions point to the the necessity of a uniform and comprehensive knowledge representation for multimedia data. Unfortunately, although MAS users desire such an ease of use and such expressive powers from the authoring systems, they and the MAS designers are limited with the status of the current technology.

Most of the existing MAS's still rely on databases which are well suited for traditional data such as *text* or *numbers*, for both their internal data represantation/storage and for the database capabilities they provide to the applications. This basically makes it difficult for MAS's to use the properties of multimedia items for increasing the storage and retrieval efficiency. However, the characteristics of such media (like *video* or *hand writings*) are inherently different from those of the more traditional data, and they require a totally different approach in most cases. For instance, video data

- has a temporal extent which can not be efficiently represented in the relational databases,

- contains objects, roles and events which need to be extracted and indexed for content-based retrieval,

- has associated QoS constraints that need to be satisfied for network transmission,

- usually has some metadata associated with it.

These properties of the video data makes it hard to implement a sophisticated multimedia application with video capabilities using relational databases. Today, some of the multimedia systems use 'Object Oriented' approach, which overcomes many of the deficiencies of the relational databases. Unfortunately, even the *OO* approach is not completely suited for all multimedia purposes.

Some of the current MAS's provide *ODBC* which enables their users to attach their favorite DBMS's to the MAS's. In theory, such systems could be integrated with more advanced multimedia database systems using database specific drivers, solving the problem of storage for multimedia items.

However, since the standard *SQL* does not have an expressive power suitable for most multimedia applications either *SQL* needs to be extended which address different or a totally new and powerful query language needs to be build in *ODBC* to use this paradigm for multimedia databases. In fact, there are various extensions to SQL such as "Spatial SQL"[**?**] which handle different types of data, and a specially designed *ODBC* driver could be used to link a spatial database to a multimedia application using "Spatial SQL".

Many researchers are working on building query languages which will address the requirements of the multimedia databases. However, the lack of a solid knowledge represantation in which all the properties of the multimedia items, their interactions and their requirements can be fully encapsulated, makes it almost impossible to create a comprehensive multimedia query language.

OLE is a technique which enables the creation-exportation-reuse of objects in multimedia applications. The reuse of OLE objects necessitates a search mechanism that will provide access to the existing multimedia objects. For instance, such a mechanism would be very useful in applications like "video editing" in which parts of the existing video segments are brought together to create new videos. Such an application, requires the understanding of temporal aspects of video segments as well as the annotation methods. They also require content based or annotation based indexing which will let the users search for the video segments (i.e. video objects) in a video database.

Note that in such applications the objects in question are not explicitly stored, but they are extracted dynamically upon need. Unfortunately, it is not clear how to perform such a dynamic extraction. Furthermore these extracted objects needs to be put together to create new video segments. If the interactions between multimedia objects could be modeled properly, the merging/synchronizing of them would be much easier to perform.

Properties like Dynamic Data Exchange (DDE), on the other hand, provide a platform in which communications can be established between external softwares and the MAS's. Data can flow back and forth between the MAS and the external software, and MAS's can benefit from the capabilities of other softwares. Note again that, to design a communication protocol through which MAS's and external softwares may exchange information, it is necessary to understand the characteristics of the media objects.

Similarly, if it was clear how to satisfy QoS requirements in a distributed multimedia environment, then it could be possible to design a MAS which enables their users build an efficient distributed multimedia environment. The current advances in the network area attracts people to use distributed information instead of storing it at their own site. Hence, the need for MAS's which are capable of building distributed multimedia systems is increasing. However, the communication characteristics that are applicable for the traditional networks is much different than the characteristics of those that will carry multimedia information. These characteristics needs to be fully explored, and efficient protocols need to be developed for providing fast and high quality multimedia communication.

One of the uses of DDE is to exchange data among softwares that are geographically distributed. As mentioned above it is still an open question how to guarantee QoS requirements in multimedia environments, and DDE's are usually slow for this purpose. Today MAS's provide their users the power of building distributed systems mostly using network file systems. Note however that there are lots of networking problems to be solved to be able to use the networks in the most efficient way for multimedia communication.

As you can see, although the current technologies like $ODBC$, OLE, DDE and DLL are capable of providing many nice properties to MAS's, in practice they still lack most of the crucial aspects. There is a large amount of research going on in multimedia technologies, and it is clear that MAS's will highly benefit from the results of these researches.

8.2 How to Benefit from MAS's in Multimedia Research

One of the biggest problems that multimedia researchers face today is the lack of time and man power. In order to design a working multimedia system, many researchers need to devote a large portion of their time to the unrelated components of their systems.

Preparing GUI's and creating databases in which multimedia objects will be stored are usually among the necessary steps for building a multimedia system. However, designing such a multimedia environment from scratch is usually to cumbersome and most of the time impossible.

Multimedia authoring tools, on the other hand, provide a framework in which many desired properties of such multimedia systems can be built very easily. For instance, a researcher who uses a MAS does not need to design a whole database; instead s/he can integrate an already existing database to the system and use it along with the tool s/he has designed. Similarly MAS's let researchers build GUI's for their multimedia systems with great ease and they increase the portability of the resulting systems.

These nice properties of MAS's help minimizing the time researchers spend for research-unrelated issues, and let them concentrate on more important topics.

9 Conclusion

In this work, we have studied three popular MAS's and we have implemented a sample multimedia application in each of these systems.

Among the three systems we have studied we have found that, Multimedia Toolbook is the best for developers with programming experience. Toolbook also turned out to be the most powerful and elegant MAS we have studied. The only real deficiency of this tool is its lack of portability (it is available for Windows only).

Although IconAuthor's *icon-flowchart* metaphor is easy to learn, and the procedural control flow is suitable for non-programmers, it can be quite awkward for experienced users. Furthermore, procedural control flow is not nearly as concise as event programming in many cases. On the plus side, it has excellent OS connectivity, and superb special effects and media support (much better than Toolbook). For non-programmers, this would probably be the tool of choice.

Finally, although Director was not suited for our sample application, it is very well suited for many others (it is the single popular MAS in use). For example, Director excels in animation and video control.

We have also looked at the deficiencies of the current MAS's and we have discussed how the reserach in the multimedia area will effect the design of future MAS's. It appears that there are various properties that would be nice to have in MAS's, but that are not currently provided.

Lastly, we have seen that multimedia researchers can benefit a lot from the MAS in their work, because it significantly reduces the time they would spend on the overheads of system design.

References

[1] mw[1] *Microsoft Windows for Workgroups Resource Kit.* Microsoft Press, Chapter 11.

[2] mt[2] *Multimedia Toolbook 3.0 Users Guide.* Asymetrix

[3] ia[3] *IconAuthor 6.0 User's Guide.* AimTech

[4] dir[4] *Director 4.0 User's Guide.* Macromedia

[5] ss[5] M.J. Egenhofer *Spatial SQL: A Query and Presentation Language.* In *IEEE Transactions on Knowledge Engineering*, Vol.6, No. 1, pages 86–94, February 1994.

[6] er[6] L.E. McKenzie and R. Snodgrass *Evaluation of Relational Algebras Incorporating the Time Dimension in Databases.* In *ACM Computing Suirveys*, Vol.23, No. 4, pages 501–543, December 1991.

[7] cr[7] T.-S. Chua, S.-K. Lim and H.-K. Pung *Content-based Retrieval of Segmented Images.* In *ACM Multimedia Conference*, October 1994.

[8] ps[8] N. Dimitrova and F. Golshani *Px for Semantic Video Database Retrieval.* In *ACM Multimedia Conference*, October 1994.

[9] sv[9] S. Ghandeharizadeh *Stream-based vs. Structured Video Objects: Issues, Solutions, and Challenges.*

[10] mmd[10] K. Bohm and T.C. Rakow *Metadata for Multimedia Documents.* In *SIGMOD Record*, Vol. 23, No. 4, pages 21–26, December 1994

[11] mvd[11] R. Jain and A. Hampapur *Metadata in Video Databases.* In *SIGMOD Record*, Vol. 23, No. 4, pages 27–33, December 1994

[12] fmd[12] S. Marcus and V.S. Subrahmanian *Foundations of Multimedia Database Systems.* submitted for publication (1994).

[13] mqvd[13] R. Hjelsvold and R. Midstraum *Modelling and Querying Video Data.* in Proc. of the 20th VLDB Conference 1994

[14] avis[14] S. Adali, K.S. Candan, S.-S. Chen, K. Erol and V.S. Subrahmanian *Advanced Video Information System: Data structures and Query Processing.* submitted for publication (1995).

[15] osr[15] B. Ozden, R. Rastogi and A. Silberschatz *On the Storage and Retrieval of Continuous Media Data.*

[16] mim[16] D. Woelk and W. Kim *Multimedia Information Management in an Object-Oriented Database System.* in Proc. of the 13th VLDB Conference 1987