MATILDA: A Framework for the Representation and Processing of Information in Multimedia Systems

Dr. D. B. Lowe, A/Prof A Ginige
School of Electrical Engineering,
University of Technology, Sydney

Abstract:
Current multimedia systems are clouded in their distinction between raw data, the information represented by that data, the structure of the data, and the inter-relationships between the data. Similarly, development paradigms for information systems do not typically take into account issues beyond the immediate functionality of the given system. This results in information systems which are difficult to maintain, difficult to develop, difficult to reuse, and which do not reach their potential in terms of functionality and useability. This work attempts to begin to redress this situation by considering both appropriate frameworks and data models which encourage suitable treatment of information from a broader perspective. These ideas are encapsulated in the MATILDA framework, discussed in this paper.

Introduction
Multimedia systems are growing at a rapid pace, both in terms of their use and acceptance and also in terms of the size and complexity of applications. Multimedia applications gain their strength from integrating a number of different media into an interactive whole for the purpose of facilitating access to information. The process of developing multimedia applications - multimedia authoring - is essentially about handling information. Like the development process for any complex system, multimedia authoring should incorporate an appropriate process. This will be increasingly true as these applications become more complex, and the information contained grows (both in quantity and diversity).

These development processes should take into account any factors which are likely to improve the development productivity or application quality. Apart from process management issues, these factors tend to focus on information. This would, for example, include data models and processes aimed at improving application maintenance, and information reuse. Current multimedia authoring approaches typically do not address these issues particularly well. In particular, the development process is confused by the tendency to cloud the distinction between raw data and the information represented by that data, between the data content, the structure of the data, and the inter-relationships between the data. Similarly, current development paradigms for information systems do not typically take into account issues beyond the immediate functionality of the given system. This results in information systems which are difficult to maintain, difficult to develop, difficult to reuse, and which have poor functionality and useability. A common example of these problems is the use of markup languages such as HTML, where the information content, syntactic structure and semantic relationships are embedded into a single document. The criticism is not with the use of markup languages for storing multimedia information, but with their use during the authoring process.

In the work described in this paper, we are attempting to begin to address many of the issues raised above. In particular, we wish to consider both appropriate frameworks and data models which encourage suitable treatment of information from a broader perspective. This includes such issues as clear distinctions between information, structure, and application, and facilitating reuse and maintainability. In order to improve our ability to assimilate and/or manipulate information, to improve the maintainability of information systems, and to improve information reuse between applications, we need to improve our techniques for representing and utilising these information sources. The first step in this process is to begin to improve the organisation of the
information - to develop a paradigm and framework which supports suitable models, and the development of models within the context of this framework.

**Issues in Developing an Information Framework**

**Information Structure**

If we are to develop a framework for improving the handling of information in the multimedia authoring process, then we need to initially consider the requirements for such a framework. The logical place to start this process is to begin to understand the structure of information and ways in which we use information. Specifically, multimedia information systems essentially operate at two levels with respect to the information being used: the raw information, and meta-information about the structure and meaning of the information. For example, in multimedia anchor text is typically underlined or highlighted in some way. Additionally, the information has been broken into nodes or pages. In both of these cases the user is provided with meta-information regarding units of information which can be used in some way. Similarly the anchor text is related to other information (made explicit in the linking) and the use can make use of this meta-information.

Philology can provide us with many worthwhile insights. A lexicon is the stock of words within a language. Syntax describes the rules governing how the lexical elements can be combined. Semantics describes the meaning of the resultant constructions. Pragmatics considers the context and implications of this. These concepts can be extended to other. Indeed, we believe that this will allow us to solve several of the criticisms given above. We mentioned that current multimedia systems typically do not differentiate between the data content and the way in which we decompose the information into meaningful units (i.e. lexicon), between the data and the structure of the data (i.e. syntax), between the data and the inter-relationships within the data (semantics), and between the data and what we can learn from the data (pragmatics).

It is worthwhile considering existing data models in this context. Although many multimedia information data models have been developed (Halasz, 1994, Grosky, 1994, Niblett 1989) most of these have not been developed as part of an overall information framework. This means that much of the strength of these models is lost due to a lack of context for the development process with. Although the models (in many cases) have contributed significantly to improving the ability to utilise specific forms of information, these data models have typically not been developed in the context of an overall framework of information handling. Rather, they have been very focussed on supporting the specific functionality required by the system under development.

**Information Reuse and Application Maintainability**

As applications become progressivley more complex, the ability to reuse information from other applications will become increasingly more beneficial. A major component of the authoring process is the structuring of the information. If this part of the process can be abstracted out and made application independant, then the information can be readily reused across many applications. For example, a specified video sequence can be decomposed into scenes, shots and objects, and the relationships between these components identified as part of the information structuring process. This information can then be used by multiple applications as needed.

Another issue is the maintainability of applications. As the applications grow in scope and size, it will become more important that they are readily maintainable. This is not the case with most existing systems. For example, with HTML documents, both information structure and presentation are coded using the same language, leading to difficulties in making large scale modifications.

It is worthwhile pointing out that many of the ideas included here owe their origin to analogies with the field of software engineering. Traditionally software was predominantly handcrafted (much as multimedia applications are currently handcrafted). As software systems grew in scope and complexity this approach broke down, with many systems failing to deliver the required performance or being completely unmaintainable. This was (and still is) addressed through the development of appropriate sophisticated software engineering paradigms, process models, methodologies, techniques and tools (Gibbs, 1994). The focus within software development has shifted away from technical constraints and issues (such as software coding) towards broader issues (such
as appropriate paradigms and process models) which resolve problems such as software maintainability and reuse. A fundamental premise in most of our current work is that a similar shift needs to occur within information systems - away from specific technical considerations towards the broader issues of information engineering - such as suitable paradigms and frameworks.

Separation of Information and Application Domains

In order to facilitate appropriate handling of information, we consider it important that the information and application be separated (at least logically, during the development process, if not in the final application). This approach improves information maintainability and reusability. Maintainability since the underlying information representation is not embedded into structuring or presentation formats (taken to the extreme by models such as HTML). Reusability since the information exists in isolation from the application (indeed has been represented in an application-independent way) and can therefore be utilised by multiple applications.

The MATILDA Framework

We have been investigating the development of large scale hypermedia systems for some years (Ginige, 1995). We have identified an approach to efficiently convert text to hypertext (Robertson, 1994). Also we have been investigating issues specific to non textual information; especially visual information. These have been integrated into an experimental HyperImage system (Lowe, 1995). Based on these experiences we have identified three core elements which need to be addressed: suitable frameworks for representing information utilisation, appropriate representations and data models of the information, and tools, processes and algorithms for manipulating the information.

To address many of the issues raised above, we have developed an initial framework - the MATILDA framework (Multimedia Authoring Through Intelligent Linking and Directed Assistance - shown in Figure 1). In this framework, layers represent specific representations of the information. This framework can be used to identify both the information representations and the various aspects of the development process. The initial stages of the MATILDA project aim to develop the MATILDA framework, and the underlying technologies (data models, tools, etc.) to the point where they can be used to investigate novel approaches to Multimedia authoring, including but not limited to design issues, information representations, pedagogical issues, and authoring processes.

We will now consider a detailed description of the layers which make up the framework.
Data Layer

The original raw information will need to be represented digitally in order for it to be utilised in the information systems we are interested in developing. This digital information will be obtained either by appropriate data capture techniques (e.g. OCR, image scanners, audio digitisers, video frame grabbers, etc.) or it may be generated directly in this form (e.g. text entered using a text editor, graphics created using CAD programs). This information will, in some cases, be further processed to convert it into a form more appropriate than the original digital canonical format. This processing may be a result of requirements such as the need for compression (resulting in, for example, JPEG images), the need to be able to more readily perform analysis on the information, or in some cases a combination of these. The information that results - the data layer information - is the lowest level of information which is represented digitally and is accessible to applications. The ability to access the data layer information by client programs should be retained. This therefore means that the use of information in the data layer should be non-intrusive - i.e. it is not acceptable to modify, say, a LaTeX document with the intent of allowing anchor points to be inserted, unless this is transparent to any possible client application.

Lexical Layer

Information typically has a certain inherent structure - the structure implicit in the original information. In order to facilitate access to this information we wish to make the structure explicit without enforcing a specific (and hence limiting) viewpoint of the information. This information incorporates both lexical information and syntactic information. Within the basic media we can identify components which represent a “unit” of information at some arbitrary level of granularity. For example, a single media file - say a text document - might contain chapters, sections, paragraphs, sentences, phrases, words, etc. - all which can be treated as a unit of information. The lexical layer is used to represent these components. In developing the data models for this layer we need to keep in mind that we wish to support an arbitrary level of granularity. This layer of the model partially maps to the “information chunking” process of many multimedia authoring tools.

It is worthwhile noting that this layer acts as an interface between the “media dependant” Data Layer (i.e. we need to know the media type and format to access and utilise the information) and the “media independent” higher layers - since the higher layers can simply refer to lexical components independently of the media type which underlies the lexical component. Finally, we should note that since different lexical components will be defining regions of different media, the data models used in this layer will need to incorporate a mechanism (probably both media and format dependant) for defining these regions.

Syntactic Layer

Within the original information a physical or logical structure can be explicitly identified. Although this structure will be implicit within the data, we wish to identify this structure and make it explicit. For example, video information contains scenes, shots, frames and objects within those frames. The lexical layer makes these components explicit but does not define the relationships between them. In the syntactic layer we identify the syntactic interrelationships between these components (such as a specific sentence is part of a specific paragraph, or one scene in a video temporally follows another scene).

In developing data models for this layer of the framework we need to consider what forms of structure we wish to support (especially with respect to the forms of relationships between components). Typical examples might include relations such as consists-of, is-part-of, contains, after, before, next-to, above, below. The list is not exhaustive but gives an idea of the types of relationships expected. Note that interconnections can be redundant, such as after implies a before. If we consider a typical example - say a text file, it will be constructed from various sections, which will in turn have pages, which might have paragraphs, etc. - i.e. a hierarchical structure.

The relationships between the structural elements in the syntactic layer often indicate a hierarchical tree, the root being the entire document, the branches (relationships) leading to sub-structural components and again branching further to an arbitrary level of granularity. We can however find examples which are not purely tree-structured. For example, a document may be broken down into either pages, or chapters, each of which
can contain (possibly through several layers) the same paragraphs (see figure 2). Another example might be a musical composition which can be broken into either specific instruments, or movements, which can intersect. The result is that the syntactic structure forms a directed graph. Each graph node represents a lexical component, and each graph link represents a relationship between a parent component and a number of children components. The result is therefore that the Syntactic Layer includes a single relationship - a parent/children relationship - which can be used to represent the entire gamut of syntactic relationships. The specific structure of this relationship and how it supports the desired relationships will be considered in the section on data models.

**Semantic Layer**

The lexical and syntactic layers only identify the structure of information components, and does not attempt to associate meaning to this structure. The semantic layer adds appropriate meaning to the identified components. This will include tagging the components with appropriate attributes. Note that this implies that interpretation is occurring as well as the addition of information which is not explicitly contained within the original information. Grosky addresses this issue quite well, in proposing a taxonomy of information types which can be applied to both component attributes and relationships (information bearing content based, non-information bearing content based, content independent) (Grosky, 1994).

In developing data models for this layer of the framework, we need to identify the forms of information which we wish to include as well as their behaviours when considering the relationships represented in the syntactic information layer. For example, some attributes of component X may be inherited by the components which are part-of component X (such as the components author) whereas others may not (such as the components length). The definition of this layer (including the data models) is still ongoing.

**The MATILDA Application Layers**

The information layers within the application domain will vary from application to application. A video archive application will be different from a multimedia authoring system. To illustrate a typical set of layers, those which we are investigating for use in Multimedia authoring are described.

**Viewpoint Layer:** A viewpoint is essentially a point of view or perspective. This can be used to limit the direction of movement through information, steering users along a defined path, or providing a particular
perspective on certain information by restricting access to some information, whilst emphasising other information. An example of this is utilising the same set of information for both teaching and reference material. For teaching, basic concepts need to be explored initially, and a staging process, whereby progressive and directional development occurs, ensuring that a person develops an adequate grasp of the material. The same information will be viewed from a totally different perspective in situations where it is being used as a reference. Although it is easy to see what the viewpoint layer means with the previous example, it is not as clear for other applications. This is an area that requires additional work to enhance its definition so as to justify its existence within the framework.

**Linking Layer:** One can deem the viewpoint as a linking process itself, however the distinction between the two layers is that the viewpoint layer identifies what information will be ‘seen’, and how it is to be accessed, whereas the linking layer formalises the interrelationships between information which we wish to support.

**Presentation Layer:** The final layer is mechanism and tools used to display or publish the linked information.

### An Authoring Architecture Based on the MATILDA framework

#### Architecture

The Matilda framework maps directly to the development process, and hence it provides a structured approach to hypermedia authoring. A typical system architecture based on the MATILDA framework described above is shown in figure 3. In this architecture, structured information is extracted or generated based on the raw media files, and stored into a suitable database repository. A series of generic information structuring tools are used to populate the repository with the lexical, syntactic and semantic information. This information is then used by a series of application-specific multimedia development tools to develop the required multimedia systems.

#### Data Models

The first step in developing the architecture is to design data models for the repository information. At present we have simple prototype data models for the lexical and syntactic levels of the repository.
**Lexical Data Model**

We will begin by introducing the *Lexical Data Model* and then explain the justification for the various aspects of the model. As was discussed previously the lexical sub-layer identifies the chunks of a document stored in the *data layer*. The key components of the *Lexical Data Model* are: A *reference component* which specifies the source file containing the media document, and is declared in the format of a URL (allowing the specification of remotely located files). A *content definition* which is used to define a region of the media document being referenced. This specific format of the content definition will be media and format dependant - i.e. it will need to be explicitly defined for each new media format. A set of *attributes* which specify an extension to the regional information such as spatial, temporal, logical positioning. The attributes are free form, allowing the specification of any desired attributes, though a set of ‘standard’ attributes are listed and can be specified where appropriate.

These can be formalised as:

- **lexical-component** = document-spec + content-definition + 0{lexical-attribute}n
- **document-spec** = *UFL* (Uniform File Location)
- **content-definition** = *media and format dependant region abstraction*
- **lexical-attribute** = name + value + units
  - **name** = standard_name|...
  - **value** = string
  - **units** = string
- **standard_name** = NAME | UID | POSx | TIMx | LOGx

**Syntactic Data Model**

As with natural language theory, the syntactic information relates to the manner in which the building blocks (‘chunks’) are combined. The data model specifies both the parent and children lexical blocks, attributes of the syntactic block, and a series of flags. These include an ordering flag which specifies the relative sequencing of the children, a contiguous flag to specify whether the children are contiguous and a containment flag which indicates whether the children are entirely within the parent (such as chapters in a book), or simply contribute to the parent (such as musical instruments in a symphony movement). The resultant model is:

- **syntactic-block** = ordering-flag + contiguous-flag + containment-flag + parent + 0{child}n + 0{syntactic-attribute}n
  - **ordering-flag** = irrelevant | unordered | spatial | temporal | logical
  - **contiguous-flag** = contiguous | non-contiguous
  - **containment-flag** = containing | contributing
  - **parent** = lexical component
  - **child** = lexical component
  - **syntactic attribute** = name + value + units
    - **name** = standard_name|...
    - **value** = string
    - **units** = string
  - **standard_name** = *still to be defined*

**Information Structuring and Multimedia Development Tools**

Once we have data models, we need to develop appropriate tools to use in extracting the desired information to populate the information repository. The following tools are being developed:

**Information Capture and Conversion:** The first step of the process in developing multimedia application is to capture the information from its original form (*raw information*: paper-based, raw video stream, etc.) into a digital form which can be stored, utilised and manipulated on computers (*data layer* of the framework). This base information may be either in a raw canonical form or as coded information (based on requirements such as data compression, or analysability). In some situations the information may be generated directly in its base form (e.g. computer graphics and text). Techniques for performing this process are well established are not the focus of this work.
**Lexical Structuring:** As discussed above, *lexical information* is the information (which is often implicit) defining the media subcomponents. We wish to facilitate the process of making the *lexical information* explicit without enforcing a specific (and hence limiting) viewpoint of the information (equivalent to the chunking process in many other authoring approaches). Lexical structuring will typically incorporate identifying the components within the specific media data. For example, this might include identifying and recording the scenes and shots within a video file, and then the objects within these scenes. In addressing the development of this process step we need to consider techniques for segmenting information (in various media) into the constituent components. A sophisticated system would, for example, include intelligent tools to separate keywords out of text, objects out of images, or shots of video sequences, and suggest them as possible lexical elements.

**Syntactic Structuring:** Once we have performed the lexical structuring we need to identify the syntactic relationships between the various lexical components. As with lexical structuring the tools used perform the syntactic structuring can incorporate progressively more sophisticated assistance. A typical example would be tools that suggest syntactic relationships between chapters of a book.

**Semantic Structuring:** The *lexical* and *syntactic structuring* were both focussed on the information structure, rather than the information meaning. The *semantic layer* takes the next step and begins to define relationships between information components which have no connection indicated by the structure of the information. Tools to support the identification of *semantic* structures will, as with *lexical* and *syntactic structuring* incorporate intelligent assistance. At its simplest level this might take the form of identifying a high level of correlation between certain words in textual documents. As media analysis tools become progressively more sophisticated they can be incorporated into this process stage.

**Access Structuring:** The next stage is to utilise the information in the development of specific applications. Multimedia systems achieve this through layering additional structure on the information. This structure provides a specific application focus to the information, and supports appropriate interactivity. This process involves two phases. The first is identifying the viewpoint from which we wish to utilise the information - the *information viewpoint layer* (which will in turn influence decisions on which information is relevant and how it is to be used). The second phase is to identify the inter-relationships within the information which we wish to make explicit - the *information linking layer* - in order to support user navigation through the information. Often these two phases are combined (i.e. *access structuring*). We need to develop appropriate tools which support this process. This has not yet been addressed.

**Presentation Formatting:** The presentation layer takes specific information, structured for a specific application, and presents in such a way as to maximise the useability of the application. Issues such as screen layout, general navigation facilities, etc. are addressed at this stage. The information is typically presented in a way which minimises the user disorientation, and improves the clarity of the information and its inter-relationships. Typically a given set of linked information from a given viewpoint can be presented in numerous fashions according to personal taste and available technologies. These issues are beyond the scope of this project as there are many third party presentation systems which can be readily used.

**Implementation**

In order to begin investigating many of the ideas outlined above we have developed an initial prototype system based on the MATILDA framework. This prototype implements the MATILDA Information Structuring tool (shown in figure 4). The tool incorporates a series of subtools (corresponding to the various processes described above). The first tool - the *lexical structurer* - (this is actually a series of tools, depending upon the media and media format) is used to present a raw media file (at present only raw text and windows bitmap image data are supported). The user can then select various regions which are committed to a database as lexical components. These lexical components are then used as the input to the second tool - the *syntactic structurer* - which is used to add syntactic relationships to the database. The third tool - the *semantic structurer* - takes the output from the previous two tools and generates semantic relationships. All information generated is stored in a database and can be accessed by other systems during the publishing process. At present the tool is still evolving and is predominantly being used to investigate many of the issues described above (such as information reuse, data models and authoring processes).
Conclusions

This paper has described the initial work on the MATILDA framework. This has included a description of the framework structure, data models for the initial layers of the framework, a multimedia authoring architecture based on the framework, and a brief description of the initial work on developing a system based on the MATILDA framework.

The key precept underlying much of this work is that information management systems in general, and multimedia authoring systems in particular, need to consider more actively issues related to information reuse, process management, and application maintainability. This includes such issues as clear distinctions between information content, information structure, and information usage within applications. We have attempted to begin considering these issues in the contact of an information framework - MATILDA. This framework begins by separating the information domain from the application domain (thereby promoting information reuse and a cleaner identification of information structure).

Future work will focus in two areas: refining the data and process models, and investigating the effect of these models on some of the issues described above. The immediate project revolves around the continuing development of the authoring tools described in the paper, and their use to obtain feedback to be used in the next generation of development.

Acknowledgments

The authors would like to thank both the Australian Research Council, and the University of Technology, Sydney for supporting research which has contributed to many of the ideas outlined in this paper. We would also like to thank John Robertson, John Leaney, Philip Martyn, Zhenya Yourlo, Andrew Dorrell and Peter Knight for the comments, discussions and development work which has assisted with this work.
References


Contact Details

Dr. David Lowe
Senior Lecturer, Computer Systems Engineering
University of Technology, Sydney
P.O. Box 123, Broadway, 2007, NSW, Australia
Tel: +61 2 330 2526
Fax: +61 2 330 2435
Email: dbl@ee.uts.edu.au

Associate Prof. Athula Ginige
Associate Professor, Computer Systems Engineering
University of Technology, Sydney
P.O. Box 123, Broadway, 2007, NSW, Australia
Tel: +61 2 330 2393
Fax: +61 2 330 2435
Email: athula@ee.uts.edu.au