Title of the Submission: The MATILDA Data Model and its Implications

Primary Contact Person: Dr David Lowe

Affiliation: Computer Systems Engineering
University of Technology, Sydney

Street Address: No. 1 Broadway, Ultimo
City: Sydney
State/Province: NSW
Postal/Zip Code: 2007
Country: Australia

Telephone: +61 2 330 2526 (before July 1, 1996)
+61 2 9514 2526 (after July 1, 1996)
Fax: +61 2 330 2435 (before July 1, 1996)
+61 2 9514 2435 (after July 1, 1996)
Email: dbl@ee.uts.edu.au

Names and affiliation(s) of additional author(s):

A/Prof Athula Ginige
Computer Systems Engineering
University of Technology, Sydney

Mr. Mark Sifer
Computer Systems Engineering
University of Technology, Sydney

Dr. John Potter
Microsoft Research Institute
School of MPCE, Macquarie University
Sydney, Australia

Topic Keywords:

Note: The presentation of this paper will be accompanied by a demonstration of the MIST information structuring application which has been built based on many of the principles outlined in this paper.
MATILDA Data Model and its Implications

David Lowe, Athula Ginige, Mark Sifer
School of Electrical Engineering,
University of Technology, Sydney
E-mail: {dbl, athula, marks}@ee.uts.edu.au

John Potter
Microsoft Research Institute,
Macquarie University
E-mail: potter@mpce.mq.edu.au

Abstract

Present authoring issues, such as managing large-scale complexity, information reuse and application maintainability have yet to be effectively addressed by current authoring approaches. This problem has motivated the MATILDA hypermedia project, which has three primary development areas: frameworks, information models, and process models. This paper presents the MATILDA information model. This has been developed within the context of an information management paradigm defined by the MATILDA framework. We discuss the problems and issues associated with hypermedia authoring of large scale applications. We then present our information model and the visual formalism (based on structured graphs) which we have adapted as a foundation for the model. We also show how this information model can assist us in addressing many of the issues raised.

Introduction

Still most authoring systems require authors to handcraft individual screens and manually link related information. As a result the cost of producing large hypermedia information systems is very high and the process is time consuming. Also the manual linking of related information is prone to errors resulting in lower quality applications (Buford, 1994).

Very little attention has been paid to maintainability of systems such as legal information systems that has information which change over time.

For our work we consider information as data provided within a context. This context consist of a structure, interpretation of the contents where applicable and a presentation format. For example if we take a research paper it has a structure typically consisting of an abstract, introduction, methodology, results and a conclusion. Where applicable it will have pointers to footnotes and references which will help to interpret and find out more about the contents. Also the paper will have a particular layout, ie a specific font type and size for headings and another for text in the paragraph etc. In hypermedia the interpretation of information is provided by creating navigational links to related information.
Most authoring approaches do not separate the structure, navigation and presentation of information. For example in HTML the structure of information and the links are all coded into the document. This make it very difficult to use a document in a different context as this will require change to the structure and the navigational links. Thus if we want to re-use the information we have to separate the structure, navigational information and the presentation information.

We developed the MATILDA (Multimedia Authoring Through Intelligent Linking and Directed Assistance) framework to identify the issues that give rise to above problems and to investigate possible approaches to solve these problems. We see the purpose of the MATILDA framework as providing a structure or framework within which we can refine our ideas on information handling and hypermedia authoring. Typically, the framework will be used to guide us in investigating information representations, various aspects of the development process, and broader issues such as information re-use and application maintainability. This will be achieved through the development of an appropriate architecture, which can then incorporate suitable information, data models, tools, etc.

The MATILDA authoring framework has evolved over a period of time. It was first proposed in January 1995 (Ginige 1995a). The structure of the MATILDA framework was refined during 1995 and was published in January 1996 (Lowe 1996a). After realising the complexities involved in managing and visualising the syntactic and semantic relationships we explored the use of structured graphs. The results of these investigations were published in June 1996 (Lowe 1996b).

This paper describes the current status of the MATILDA Information Model and the benefits we obtained by adopting the structured graphs visual formalism to represent the syntactic and semantic relationships in the Information Model.

**Authoring Issues**

There are authoring tools that have implemented elegant solutions to authoring issues such as combining and synchronising media, easy design of screen layouts, basic structuring and linking. Still there are some unresolved issues in relation to development of large hypermedia information systems. The resolution of these issues within an authoring paradigm is critical if we are to significantly improve the effectiveness of the hypermedia authoring process, and the quality of the resultant applications.

Important issues are:

- **Productivity issues**: Various authoring applications can be classified into one of the following approaches; program language based, screen based and information centred (Ginige 1995a). The first approach is suitable for very specialised applications such as some games that need vary fast response time. The second type of approach is very good for quickly developing a small application; but this approach will not scale up for large systems. The information centred approach can be automated and used for developing large systems. A main feature of information centred approach is it separates information structuring, linking and presentation so that we can individually optimise individual processors and also develop schemes to automate them.
• **Cognitive issues** major advantage of hypermedia information systems is the ability to use non linear information structures and to cross link related information. While authoring for small systems the author can draw the structure on paper and prepare a list of nodes that needs to be crossed link. These methods does not scale up for large systems.

If the authoring paradigm supports an information representation which allows reasoning about the information, then the authoring process can utilise this reasoning to provide more intelligent support to the author. For example, if the model provides support for suggesting semantic relationships based on alternative semantic relationships and information structure, or for suggesting decompositions of information into a lower level structure, then these can be used to assist the author in generating the information structures to be used.

• **Maintainability**: 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.

• **Information re-use**: As applications become progressively 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 independent, 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.

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.

**The MATILDA Project**

The HyVIS (Hypermedia and Visual Information Systems) group at the University of Technology, Sydney has 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 should be addressed when developing information systems: a suitable framework (which provides an environment within which the issues can be addressed); an information model (for representing the structure and manipulation of the information); and a set of processes, algorithms and tools for manipulating the information.

**MATILDA Framework**

The MATILDA framework aims to provide a context for developing information models and processes which facilitates addressing the issues outlined above. The basic premise behind the development of the MATILDA framework is that it should make explicit various forms of information (and *meta*-information). In particular, the application-independent information (such as the underlying syntax) is separated from the application-dependant information (such as presentation information). At present our research has been predominantly focussing on the information domain. We wish to make explicit the meta-information which is either embedded in the media, or in the way in which we interpret the media. Much of our thinking in this process was influenced by natural languages. A more detailed outline of the MATILDA framework is given in (Lowe, 1996a).

**MATILDA Information Model**

The MATILDA Information Model defines an information structure and the way in which this structure can be manipulated. The basic form of the structure, as given by the MATILDA framework, is a series of seven layers of seven layers of information - split into an information domain and an application domain. We have decomposed the information domain into four
primary layers (shown in figure 1). The bottom layer is the raw data layer, representing the constituent media files (such as a raw text file, a MPEG video file, or an Excel spreadsheet). The next level up makes explicit the lexical structure of these media files, identifying the components within the media at an arbitrary level of granularity (such as a word or chapter in the text, or a scene in a video). The third level identifies the syntactic structure between the various lexical components (such as a chapter containing various sections), and the fourth level makes explicit the semantic relationships within the information (an image and some text relate to the same object). Refer to the example given later in the paper for a more concrete illustration of these information layers.

The application domain codifies the information which will be application specific. In particular, a specific set of viewpoints on the information are identified (such a given application specific structure, and mapping of information content into this structure) followed by the interlinking of the information which the application is to use. Finally, presentation issues can be identified as the final layer of information to be included in the application.

Beyond this basic informal description of the layers, the MATILDA information model has incorporated (and to some degree, had its evolution driven by) an underlying visual formalism. This formalism is an adaptation of the structured graph formalism. This provides both a formal description of the information components and their relationships, and a sets of structure preserving operations. The structured graph formalism was explicitly developed to address various issues (including scalability of browsing and editing of large graph-based structures) which are directly relevant to large information systems.

The final aspect of the information model is the development of specific data models and database schema to use in the implementation of these concepts.

**MATILDA Process Model**

It makes sense, given that the information structure has been broken into application-independent and application-specific, to also break the processing of the information down. The MATILDA Process Model separates the authoring process into information structuring and application structuring. The information structuring processes (which are the basis of the MATILDA Information Structuring Toolset - MIST) are responsible for identifying and formalising the structure of the information. This is described in detail in (Lowe, 1996b). The application structuring processes (which are the basis of the MATILDA Application Structuring Toolset - MAST) utilise this information structure to generate specific applications. It is important to note that this does not imply a specific process model - though we do envisage that these two toolsets will be used in conjunction with each other in an iterative fashion.

**Structured Graph Formalism**

Hypermedia information models typically yield complex graphs. Interacting with such graphs on a computer is difficult, because they are typically large and cannot be displayed in full detail on one screen. Even, if one had a large enough screen this is unlikely to solve the problem of understanding the graph. One needs to be able to select an arbitrary portion of the
graph, then show it at an arbitrary level of summarisation, and then to edit these summary graphs directly, all without loosing context. Structured graphs were designed to address this need.

Structured graphs is a visual formalism. It consists of:
- a data structure
- structure preserving operations on that data structure (eg. editing and browsing)
- associated visual notation that enables one to see the data structure or a portion of that data structure.

Another way to look at structured graphs is as a network of nodes and links, a node hierarchy and a link hierarchy. Each node may have several links as inputs and as outputs. Each link may have several nodes as producers and as consumers. Appendix A gives a simplified mathematical definition of structured graphs.

Browsing operations on structured graphs allow arbitrary cross-sections of nodes to be selected, and summary graphs to be produced for those nodes. Editing operations allow direct editing in essentially a single step on any summary graph (such as changing a link or moving a subgraph).

The major point of the structured graph model is that arbitrary summary graphs can be viewed and edited. The views are not restricted to single nodes or to fixed levels.

The use of the structured graph visual formalism within the MATILDA information model is best illustrated with an example, as shown in figure 2 (based around an information representation of Shakespeare's Macbeth).

*Figure 2: Structured graph for a very simple text/image example*
In this example, the circles represent lexical elements, or information components, at various levels of granularity (an entire image file or an object within the image, the entire text of a play, or single statements spoken by one of the characters). The squares represent syntactic elements, or the structural relationships between lexical elements (which can be either containment relationships, or contributing relationships - these relationships are described below). Notice that this allows us to specify different partitionings of a parent lexical elements (such as the temporal partitioning into acts, or the instructional partitioning into various characters and commentaries). The graph also includes semantic relationships between the various lexical and syntactic elements.

**Browsing a Structured Graph**

Browsing a large graph on a computer screen is unwieldy, so the hypertext approach has been to just show one node and its immediate links (anchor points). The structured graph provides more flexible browsing options: being able to see implied semantic links between nodes; being able to select several rather than one arbitrary node and see the semantic links (including derived links) between them.

For example when we suppress information below nodes P and Q an implied semantic link appears between nodes, P and Q if there is a semantic link between a node contained in P, say X, and a node contained in Q, say Y. Figure 3 shows this. Another way to think about this is: the higher nodes P and Q partition the graph of links below them, and those links which pass between these partitions are the implied links between P and Q. Now if a user is browsing from node P the user will have an anchor point leading to Q automatically provided (without the author having to explicitly add this).

![Diagram](image.png)

*Figure 3: An implied semantic link*

An example of the use of this concept is shown in figure 4. As shown in figure 4(a), we can initially display both the *PlayImage* and *PlayText* nodes, with a derived link displayed showing that there is a semantic relationship between these nodes. We can then expand each of these nodes (or possibly just one) as shown in figure 4(b) with the link being progressively refined until we reach the ultimate link between leaf nodes as shown in figure 4(c). At no point have we lost context, or shown more information than was required. This process can work in reverse as well, allowing us to hide information whilst retaining the fact that relationship exists (at some deeper level in the node hierarchy). In practice, link summarisation would also be used in conjunction with the use of derived links, so that we did not end up with unmanageably large numbers of derived links at high levels in the hierarchy.
Editing a Structured Graph

The structured graph formalism also supports the editing of large models. For example, we can initially add unresolved semantic links between parent nodes of an incomplete graph during the initial creation (such as between Image II-ii and Act II, Scene ii) which will later (when required) be resolved to a link between the relevant children nodes (such as the image and the specific passage of speech by Macbeth). Similarly the reverse can apply - given an existing semantic link at a low level, we can generate implied semantic links between parent nodes when suppressing low-level structural elements.

Containment versus Contribution

In establishing the mapping from information models to a structured graph, the most difficult part was the interpretation of syntactic elements. A syntactic element may be contained within another syntactic element. For example, a paragraph being fully contained with a chapter. Also there can be syntactic elements that only contribute to another syntactic element. For example a particular characters dialog will contribute to many scenes, but not be contained entirely by any one scene.

In this section we show contribution can be thought of as the partial specification of a containment relationship. This is important as it makes the distinction between these two interpretations clear. When we first started applying the formalism we thought containment could simply be treated as a special case of contribution, however this proved to be inappropriate. The reason for this will now be presented.
Figure 5: Containment and Contribution links

Figure 5(i) shown the nodes A to G, with a line from D down to G indicating that node D contains all the information in G (and more), while the dotted line from B down to D indicates that B contains only some of the information in D (but also contains other information). Containment is a transitive relationship, that is, B contains C and C contains E means that B contains E. Looking at figure 5(i) we have D contributes to B and B contributes to A, so perhaps D contributes to A. But this may not be true.

Let the part of node B which contributes to A be node C. Let the part of node D which contributes to B be node F. After resolving the contribution relationship into containment, nodes A and D are left unordered, that is, A does not contain D and there is no outstanding contribution relationship between A and D. This is shown in figure 5(ii). In summary, the contribution relationship is not transitive. This realisation was critical in applying structured graphs to MATILDA, as a structured graph node hierarchy is formally an order relation, and hence must be transitive, otherwise the resultant implied links would be meaningless. So only containment relationships between elements were mapped to the node order.

Conclusions

We have developed an information model within the context of the MATILDA framework that will make the structure of the information contained within raw media explicit. This structure consists of lexical elements and syntactic and semantic relationships among these elements.

When this information structure becomes large, as will be the case with any substantial hypermedia application, visualising and manipulating the information and its structure becomes difficult. We have begun the process of solving this problem by adopting a visual formalism based on structured graphs. Structured graphs provide a mechanism for editing and browsing large graphs in a scalable fashion. There is a relatively straightforward mapping from the structured graph components (nodes and links) to the information elements we are utilising in the MATILDA information model. This mapping has provided us with an enhanced ability to manipulate the information structures in a flexible, scalable, and consistent manner.
The MATILDA information model will subsequently allow us to provide substantial benefits in the authoring process, especially with respect to the cognitive load placed on the author (through improved visualisation of the information structures), information reuse (through making the structure explicit) and application maintainability (through separation of application structure from information structure).

Acknowledgements

The authors would like to thank the Australian Research Council (small grants scheme) and the University of Technology, Sydney (internal research grants scheme) for funding aspects of the work associated with this project.

References


Appendix 1: Definition of Structured Graphs

This appendix provides the formal definition of structured graphs\(^\text{1}\) with its browsing operation. We have used [Davis, 1990] as our reference for order theory.

Definition A.1 An ordered set (also referred to as partially ordered set) is a set \(P\) equipped with a binary relation \(\leq\) on \(P\) such that, for all \(x,y,z \in P\):

- \(x \leq x\) (reflexive)
- \(x \leq y\) and \(y \leq x\) imply \(x = y\) (anti-symmetric)
- \(x \leq y\) and \(y \leq z\) imply \(x \leq z\) (transitive)

Definition A.2 A order relation \(\leq\) on \(P\) gives rise to the relation, \(\parallel\) of non-comparability such that, for all \(x,y \in P\):

\(x \parallel y\) if and only if not \((x \leq y\) or \(y \leq x)\)

Definition A.3 Let \(\langle P,\leq\rangle\) be an ordered set and let \(x,y,z \in P\). Then \(x \prec y\), which is read as \(x\) is covered by \(y\), is given by:

\(x \prec y\) if \(x < y\) and \(z < y\) implies \(z = x\)

For visual presentation of ordered sets, the graph of the covering relation (Hasse diagram) is typically used. To draw an order, if \(x \prec y\) then \(x\) is placed below \(y\) in the diagram, and \(x\) and \(y\) are joined by a line. Once we are working with the covering relation we can use the terms parent and child so that, \(x\) is covered by \(y\) can be restated as \(x\) is a child of \(y\). For finite ordered sets it is the reflexive transitive closure of the covering relation which gives back the original ordering relation.

Minimals are those elements at the bottom of an order. Though minimals are usually defined for a whole ordered set, we are often interested in the local minimals of a subset of an ordered set.

Definition A.4 Let \(P\) be an ordered set and let \(S \subseteq P\). Then minimals \(S\) shown as \(\mathcal{S}\) is a subset of \(S\) which satisfy:

\[\mathcal{S} = \{ m : S \mid \forall a : S \cdot a \leq m \Rightarrow a = m \}\]

A structured graph is a bi-partite graph whose vertices are nodes and links where edges are characterised by prods and cons relations, equipped with ordering relations on nodes.

Definition A.5 Given nodes: \(\mathbb{P}\) Node, and links: \(\mathbb{P}\) Link, and an ordering on nodes \(\leq_n\), the tuple \((\text{prods}, \text{cons})\) where:

---

\(^{1}\) A more complete treatment of structure graphs, which includes a link order and better support for partial graphs is given in [Sifer, 1996]. For example, this includes links which may have only producers or only consumers.
**prods, cons :** Link $\rightarrow$ P Node

where:

\[
\forall l : \text{links} \bullet \\
\text{prods } l \neq \emptyset \quad \text{and} \quad \text{cons } l \neq \emptyset
\]

This definition requires every link to have producer nodes and consumer nodes. Derived links are added to a structured graph by applying an abstraction operation.

**Definition A.6** Let $G$ be the structured graph (prods, cons). Let the abstraction operator $A$ map from structured graphs to structured graphs. Then $A (G) = (\text{prods}', \text{cons}')$ where:

\[
\forall l : \text{links}, n : \text{nodes} \bullet \\
\text{n } \in \text{prods'} l \iff \exists p \in \text{prods } l, c \in \text{cons } l \bullet p \leq n \quad \text{and} \quad n \parallel c \quad \text{and} \quad p \parallel c
\]

\[
\text{n } \in \text{cons'} l \iff \exists p \in \text{prods } l, c \in \text{cons } l \bullet c \leq n \quad \text{and} \quad n \parallel p \quad \text{and} \quad p \parallel c
\]

A node produces a link when it is above a node which produces the link, and there is an outside node which consumes the link.

A structured graph is browsed by selecting the nodes and links of interest. By default this may be all links for example. This then determines a portion of the original structured graph to be shown with its derived links.

**Definition A.7** Let $G$ be the structured graph (prods, cons). Let $N$ and $L$ be subsets of nodes and links respectively. The *view* of $G$, selecting on $N$ and $L$ is given by:

\[
\text{view} \left( G, N, L \right) = L \triangleleft A \left( \text{prods}, \text{cons} \right) \rhd N
\]

where,

\[
L \triangleleft G \rhd N = \left( L \triangleleft \text{prods} \rhd N, L \triangleleft \text{cons} \rhd N \right)
\]

symbols $\triangleleft$ and $\rhd$ denote respectively, relational domain and range restriction

Page 13