MIST: THE MATILDA INFORMATION STRUCTURING TOOLSET

D.B. LOWE, P. MARTYN
Faculty of Engineering, University of Technology, Sydney
P.O. Box 123 Broadway 2007 Sydney NSW Australia
Email: db@eng.uts.edu.au

The MATILDA hypermedia authoring framework has been developed in an attempt to address a number of issues related to authoring process, information reuse and application maintainability. In particular, the framework addresses information representation and usage issues, and has been used in the development of a series of data models. The architecture of MATILDA separates the information domain from the application domain and is underpinned by a graph-based information representation. This paper presents details of MIST - the MATILDA Information Structuring Toolset used within the information domain to identify and represent the structure inherent within hypermedia information. MIST incorporates a graph-based information model, information space structure browsers, and media browsers, which when combined provide a powerful yet flexible hypermedia authoring tool. In particular the scalable nature of the information representation and tools make it well suited to the development of very large and complex information spaces. The design and implementation of MIST is presented, along with results illustrating its application within the hypermedia authoring process.

1 Introduction

Hypermedia 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. Hypermedia 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 hypermedia applications is essentially about handling information. Like the development process for any complex system, hypermedia development should incorporate an appropriate process for the handling of this information. This will be increasingly true as these applications become more complex, and the information content of applications 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 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 usability. 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, making reuse of the information and maintenance of the application difficult.

Given these observations, it becomes apparent that we need to improve current approaches to handling information. A major facet of this is the development of architectures or frameworks within which we can develop improved information handling systems. If we are to develop a framework for improving the handling of information in the hypermedia 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 itself and the ways in which we use this structure. Specifically, hypermedia 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 hypermedia, 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 user can make use of this meta-information.

An important issue affecting how we make use of this meta-information is information reuse. 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. At present, this information structuring is typically embedded into the authoring process, and so much of the knowledge about the information structure which is utilised during the authoring process is lost once the application has been constructed.
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 content and information structure (especially interlinking) are coded using the same language - the meta-information is made explicit, but left embedded in the content. This leads to difficulties in making large scale modifications, or reusing the meta-information. This will be increasingly important as evolutionary systems become more common. As with the issue of reuse, separating the information structuring process from the application development process will enable applications to be maintained more effectively. This will be a direct consequence of an improved degree of visibility of the structure of the information being manipulated and maintained.

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 hypermedia 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 above observations have been the basis of an ongoing development project called MATILDA (Multimedia Authoring Through Information Linking and Directed Assistance). This project is based around an information handling and multimedia authoring architecture which separates the information domain from the application domain and makes explicit the structure of the information being handled. MATILDA has been used as the basis for the development of an authoring platform which utilises two toolkits - the MATILDA Information Structuring Toolset (MIST) and the MATILDA Application Structuring Toolset (MAST). In this paper we provide a brief overview of MATILDA and then focus on the role, philosophy, development and results of MIST. We discuss the development and implementation of MIST (including the underlying visual formalism used to define the information structures used.
2 Background

2.1 Authoring Issues

Prior to discussing the specific approach which we have adopted, it is worthwhile raising a number of issues. 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 (in terms of usability, interactivity, and especially maintainability):

Handling of large systems: The size and scope of application is growing considerably, any attempt to develop an authoring process or system needs to consider how to cope with this. In particular, any information representation or authoring process needs to be easily scalable. This especially applies to scalability in terms of cognitive issues. A major constraint of current systems is the lack of support for assist the author in effectively handling the large information space. This issue is also reflected in a number of the following points.

Accessibility: The authoring paradigm adopted must explicitly address problems associated with the access to (both by the author and the user) the information space. For example, during the authoring process, the author is initially working with an unstructured or poorly structured information space and needs explicit support in manipulating this space whilst ensuring access to the information contained within.

Partially complete representations: Related to the above point is that during the authoring process, the information space (especially within the under-development application) will be constantly under manipulation. The authoring paradigm should ensure that even partially completed representations of the applications information space can be utilised, access, evaluated etc. This is typically not the case with most current system - where under-development systems will typically contain dangling links, incorrect structuring etc.

Reasoning about the information: 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.

2.2 The MATILDA Framework

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). This has resulted in the development of the MATILDA framework (Multimedia Authoring Through Information Linking and Directed Assistance) - shown in Figure 1. This framework is aimed at supporting research into many of the issues identified above, especially those related to information reuse, application maintainability, and authoring process.

The MATILDA framework has been developed (and is continuing to evolve) in an attempt to address many of the higher level issues in hypermedia authoring which we have mentioned above. A much fuller treatment of the MATILDA framework and its application is given in (Lowe, 1996a). We will give a brief introduction here in order to place the remainder of the paper into context.

The MATILDA framework aims to 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.

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.

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 framework 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. 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. Further details of the MATILDA framework are given in (Lowe, 1996a).

Before continuing, it is worthwhile making explicit the meaning of an "information framework". 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 reuse and application maintainability. This will be achieved through the development of an appropriate architecture, which can then incorporate suitable data models, tools, etc.

2.3 Structured Graph Formalism

An important issue which needed to be addressed as part of the development of the MATILDA framework in general, and MIST in particular, was the structure of the information to be represented. This needs to take account of both information theoretic issues, as well as the authoring process issues identified above. Although the MATILDA architecture described above provides an overall information structure within which these issues can be addressed, it has not provided the specific information models required to develop a working system. This has been achieved through the development of a set of information models based on an adaptation of the structured graph visual formalism (Sifer 1996, Lowe 1996b).

Structure graphs were originally designed to address the need for handling
graph based information at an arbitrary level of granularity and summarisation, without losing context. A structured graph is a network of nodes and links, a node hierarchy and a link hierarchy. Each node may have several link inputs and several link outputs. Each link may have several node producers and several node consumers. Formally the node and link hierarchies are ordered sets. The structured graph formalism also defines browsing and editing operations. The browsing operation allows an arbitrary cross-section of nodes to be selected, with a summary graph being automatically produced for these nodes. The editing operations also allow direct editing on any summary graph (such as changing a link or moving a subgraph).

The structured graph formalism has been adapted to the information structures used within MATILDA. This is best illustrated with an example, as shown in figure 2 (based around an information representation of Shakespeare's Macbeth. 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, which the formalism tells us are simply unresolved containment relationships). 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. The formalism and its mapping to
MATILDA is described in detail in (Lowe, 1996a)

The structured graph formalism can be used to assist in addressing many of the issues raised above. For example, we can simplify the browsing of a network by suppressing detail below a certain level in the graph, or zooming in to specific sections of a graph. Similarly, the structured graph formalism actively supports editing of the large information structures. 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. Essentially, the formalism, and its implementation as a set of data models, allows us to handle large complex information structures in a consistent, effective, and scalable fashion.

3 MIST

The MATILDA Information Structuring Toolset (MIST) is the implementation of the information management concepts outlined above. Essentially MIST is a toolset which allows us to identify and record the meta-information contained within the information used in hypermedia applications. This includes identification of lexical components, syntactic structure and semantic relationships.

MIST utilises many of the concepts outlined above with respect to handling large information systems. This especially relates to browsing the information and its structure at varying levels of detail through suppression of detail, and
supporting the generation of incomplete or partial structures. It is also worthwhile keeping in mind that MIST (indeed all of MATILDA) is a research tool. MIST has therefore been developed in such a way as to promote flexibility and extensibility. This will be illustrated through an initial discussion of the MIST architecture.

3.1 **MIST Architecture**

The core philosophy behind the development of MIST is that it should provide a scalable, extensible mechanism for identifying and recording the structure of information in an application independent fashion. It is important to keep in mind that MIST is not intended as a commercial system. Rather, it is aimed at providing a research infrastructure which can be used to investigate a broad range of information structuring issues. The development of MIST has gone through a number of iterations, during which time the architecture, design, and implementation has been progressively refined.

Prior to discussing the MIST architecture itself, it is worthwhile considering a number of issues which need to be taken into account in developing an appropriate architecture.

Support for structured graph formalism: The architecture adopted for MIST must be able to provide active support for the underlying information representation and the associated formalism. In particular, the architecture needs to consider the mechanism which will be used to provide an interface between the information structures and the browsing and editing tools. A good example of the types of interface paradigms (such as graphs, trees, sets) which can be used is given in (Feiner, 1988).

Open and Extensible: MIST is being used as a research tool and it is therefore critically important that the structure be relatively open and extensible. By this, we mean that the architecture should support the gradual evolution of existing components and addition of new components into the overall framework. The need for openness in the architecture goes beyond research systems however. For commercial systems, the level of assistance provided to the author in the structuring process will evolve over time, as more intelligent support is provided. This concept, which will be discussed in more detail later in this paper, is well recognised in a number of existing systems, such as Microcosm (Hall, 1996).

Distributed functionality: Related to the above concept of extensibility is the need to distribute the functionality. The development of MIST encompasses a large number of smaller subdevelopments (such as the inclusion of video segmentation tools, text analysis tools, etc.). Such developments can
The MATILDA Framework

Figure 3: The MIST Architecture

best be accommodated if the functionality of MIST is distributed, facilitating the separate development of a number of separate subprojects within the MIST umbrella.

Well defined interface to the information repository: The core of MIST is the information structure and the scalable interface to the information repositories. The interface to this repository needs to be consistent yet flexible.

Based on the above principles, we developed an architecture for MIST. The basic structure of this architecture is shown in Figure 3. MIST consists of a core information repository (actually, the repository is not considered part of MIST, but rather the central element of MATILDA which incorporates both MIST and MAST) an MGraph module (which provides the graph-based interface to the repository), the main MIST interface (which is used to manage databases, such as requesting the MGraph module to load a new database), and a series of extensible structure and media browsers. It is these browsers which provide the core functionality in terms of browsing and editing the information structure.
The two elements which require some explanation are the structure browsers and the media browsers. The structure browsers are used to provide views onto the information space. A typical view may be a section of the graph, a display of the node or link ordering, or some other arbitrary collection of nodes satisfying a given query. We can open multiple browsers to provide different views or perspectives of the same information space. The MGraph module actively supports different types of views onto the information representation. The modular nature of the architecture means that as different types of structure browsers are developed they can be readily integrated into the overall system.

While the structure browsers manage the overall information space, the media browsers are used to manage individual nodes of information within this space. For each type of media (text data, images of varying formats, video components, audio elements, etc) a separate media browser can be created. These browsers can then make use of information analysis tools to provide sophisticated intelligent assistance to the author and handling the information components. A structure browser can interface to these elements and launch the appropriate media browser when required. Each of the interfaces between components has been specifically designed for modularity within the system. This allows additional components (especially browsers) to be readily added to the system as needed, resulting in a flexible open architecture.

3.2 MIST Implementation

The development of MIST is very evolutionary. A progressively more sophisticated set of prototypes has been developed. This development has been performed on a PC environment under Windows 3.1. Due to the rapid prototyping nature of the development, the system had been developed in a combination of Microsoft Visual Basic (for the main interface elements), and C++ for the MGraph module and underlying media analysis functionality. This development environment provided us with a high level of flexibility as the project evolved.

An example of a MIST structure browser (for the node ordering) and a media browser (for a text element) are shown in Figure 4. The structure browser interface uses a tree metaphor for displaying the node ordering (even though the node structure is actually a directed acyclic graph - this is mapped to a tree representation). The main interface for the system allows multiple databases to be opened and information copied between them. In the example shown (which is based on a partial information representation of Shakespeare's "Macbeth") we can see that the main database contains both images (stage
Figure 4: Screendump of the Matilda Prototype
layouts, actors costumes, shots from productions of the play etc.) and the text of the play. The text of the play has been partitioned in two primary ways - either temporally into acts and scenes, or logically into different components of information - such as dialog and commentary (this structure was shown in figure 2). The main interface also allows elements to be dragged and dropped to modify or edit the structure. Lexical elements can be created at any level and then copied or moved to exist as part of an existing lexical element (with the automatic creation of a syntactic relationship).

Once a lexical element has been selected in the structure browser, the toolbar allows various media browsers and analysis tools to be launched. The three most significant controls are:

1. Attribute control: The attributes of any element can be viewed (as shown in figure 4b). These attributes allow the information elements to be tagged with attributes. The specifics of the attributes are dependant upon the type of element being addressed. 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).

2. Semantic relationship control: The semantic relationships in which the media element participates can be viewed. The semantic relationships may either be primitive relationships, or derived from relationships with information which is currently suppressed.

3. Lexical element control: This control is actually an extensible set of arbitrary controls designed to allow manipulation of the lexical elements. For example, figure 4a shows an early lexical text control (which allows the specification of the content of a lexical text element) and figure 4b shows a lexical image control (which can be used to define image regions). Lexical elements are automatically attached to the relevant lexical element control. These controls can, in turn, provide progressively more sophisticated support for analysis of the information and its structure. For example, the image control can automatically, when requested, segment an existing image lexical element (such as Image II-ii in the above example) and add to the database a set of lexical elements corresponding to the segmented components (such as Macbeth and Lady Macbeth) as well as the syntactic element linking these components together. The flexibility of the architecture which has been adopted is such that these tools can be progressively built up and linked into the system as they are developed.
4 Discussion and Further Work

4.1 Discussion

Having demonstrated the architecture and design of MIST, we will consider how effective MIST is in addressing the various authoring issues which were raised at the beginning of this paper (section 2.1):

Handling of large systems and accessibility: MIST provides a natural mechanism for accessing and manipulating large information systems. Lower level detail can be suppressed, whilst maintaining the information context. As an example, consider the semantic relationship between Image II-ii and a specific passage of speech by Macbeth from the play (shown as the link Macbeth Talking in figure 2). If we suppress the level of detail, so that the specific passages of speech by Macbeth are not shown then we can still show a "derived" semantic link between Image II-ii and Act II-Scene ii (this derived link is shown as Scene II-ii in figure 2). This automatically derived link lets us know that Image II-ii has a semantic relationship to some part of Act II-Scene ii, even if this specific link is not being shown. This mechanism can continue as we suppress more and more detail. We can also use a second mechanism - link summaries as given by the link hierarchy - to summarise semantic relationships between two lexical or syntactic elements, providing another mechanism for handling the cognitive burden. For example, if we suppressed ALL detail in the Macbeth example, so that only the two top-level elements (Play Images and Play Text) were being shown, then we would have a large number of derived relationships between these two elements (corresponding to each of the individual relationships between lower level elements). These could all be combined or summarised into a single semantic relationship showing that Play Images and Play Text are describing the same play!

Although at present MIST has only been applied to a relatively small number of applications, and no extremely large systems (this work is ongoing) early indications are that MIST provides a very flexible mechanism for accessing large information spaces in a consistent scalable fashion. The interface (after a number of developmental iterations) is intuitive and straightforward. The user can view subregions of the information space without losing context, and the interface provides a natural mechanism for providing summary information (through the collapsing/removal of detail, and the use of derived and summarised semantic relationships).

Partially complete representations: During the creation of the information space (or alternatively, population of the information repository) the overall information structure will be incomplete. The formalism underlying MIST ensures that a structured graph representation is consistent, and that all al-
lowable operations are structure-preserving. This implies that MIST ensures that even partial representations of the information space maintain internal self-consistency and can therefore be easily browsed. MIST however goes further than this, in that it allows incomplete representations to be built as intermediate steps. For example, MIST has the ability to add semantic links at a high level (before the detail has been added) or to initially create contributory relationships (before they are resolved into the lower level containment relationships) before the low-level information structure has been created. Continuing with the example used in the previous point, we could create a structure which includes the Acts and Scenes, then add the the semantic relationship Scene II-ii. Later, when we add the specific passages of speech, we can resolve the Scene II-ii relationship into the Macbeth Talking relationship. Support for these forms of manipulations has resulted in a tool which is particularly effective in the construction of very large information spaces.

Information reuse: The expected scenario for use of MATILDA is that MIST and MAST (the MATILDA Application Structuring Toolset) be used together iteratively. MAST is used to determine the application structure, interlinking and presentation, and MIST is used to create the information space (i.e. both information content and information structure - or meta-information) which is used to populate the application. For the initial application, only the required information space would be created (as an integral part of the application development, but as a logically separate task). For a second application, components of this existing information space could be reused, and the space extended where required. As more applications were developed, the space would grow, and the degree of information reuse would expand. For example, parts of the structure developed for our Macbeth example could later be reused in an application talking about Shakespeare's writing style.

Essentially, we are removing the need to repeat the information structuring process (which is often only performed implicitly) with each new development. At present we have been focusing on developing the information structuring toolsets, and do as of yet have significant data on the development of a number of hypermedia applications from similar information bases. Several small scale experiments have however indicated that MIST does indeed assist in addressing the problem of information reuse. The degree to which this will occur is yet to be adequately quantified.

Information maintainability: MIST should support hypermedia application maintainability in much the same way that it supports information reuse. In MIST the information structure is drawn out and made explicit as part of the information space. In MAST, this information (and meta-information) is then linked into a specific application structure. We can modify the structure
of an application without needing to restructure the information space. Similarly, we can change or update the underlying information content without needing to modify the application structure (provided that the linkages between the application domain and the information domain remain unchanged - something which the MATILDA information representation makes simple to evaluate). As with the issue of information reuse, we do not yet have sufficient experience with the use of MATILDA/MIST to evaluate the degree to which the above will hold true.

4.2 Future Work and Development

Although the MIST prototype provides a fully functioning system, there are a number of areas where we are continuing development. The MIST architecture was developed with extensibility in mind. It is intended to use MIST as the basis of many small development projects, to investigate a large number of issues. The immediate plans for short-term development encompass:

Improvement of the MIST structure browsers: A number of minor inconsistencies still exist with the main MIST structure browsers which we have implemented. For example, the graphical tree display of the node ordering can be significantly enhanced to incorporate more accurately the information on the different forms of syntactic relationships and the different media types for lexical elements. We also wish to integrate the semantic relationship control into the node viewer.

Addition of further structure browsers: We are currently developing a wider range of structure browsers so that we can investigate the effects of browsing flexibility on the ease of access to information. These will include complex graph browsers which incorporate sophisticated graph layout algorithms.

Addition of further media browsers: At present MIST includes media browsers for raw text, BMP images, and (under development) AVI video and WAV audio files. We will expand this set to encompass additional media (animations, music, etc.) and formats (such as various formatted text, image and video formats).

Development of Information Analysis Tools: We will continue to improve the level of support for the information structuring which is provided by the media browsers. For example, we are developing sophisticated text analysis tools aimed at progressively decomposing text into its components and video analysis tools for segmenting scenes in videos and tracking selected objects through a scene.

Apart from the specific ongoing development work associated with the
evolution of MIST there are a number of areas which will be the focus of ongoing research. We will be performing useability analyses to investigate the effectiveness of the techniques which are used to interface with the information space. This will be used in longer term research on techniques for lightening the cognitive burden placed on authors during the authoring process. We will also be investigating the effectiveness of MIST in particular, and MATILDA in general, in addressing the issues of reusability and maintainability. This will require the application of MIST to very large information systems.

Finally, it is worth noting that the structured graph formalism contains some features which have only been touched upon in this paper. For example, the inclusion of a link order allows improved browsing selection as a layer of links can be selected, improved browsing by the automatic display of summary links in summary graphs, and top-down editing which permits the partial definition of links, which are completed later. In future work, we hope to explore the application of these structured graph features to MIST. We also wish to continue the formalisation process. For example, making more explicit the difference between containment and contribution in syntactic elements.

5 Conclusions

In this paper we have briefly introduced the MATILDA framework and discussed the separation of the information and application domains during the hypermedia authoring process. Within the information domain, we have process the MATILDA Information Structuring Toolset (MIST). MIST is a system which facilitates, in a scalable and extensible fashion, the structuring of large information sets.

Underlying MIST is a visual formalism based on structured graphs. This formalism served several critical purposes. Firstly, it assisted in the resolution of a number of issues relating to the specific structure of the information (such as the relationship between contribution and containment syntactic elements). More importantly for this discussion however is the guide that it provided in terms of providing a scalable mechanism for consistently editing and browsing information structures. MIST provides an elegant solution to problems related to the difficulty with browsing of large information structures, maintaining consistency during the creation of these structures, and how these structures can be used to understand the information space. In particular MIST allows the information space to be browsed at arbitrary levels of summarisation, whilst maintaining the integrity of the relationships and structure which are being represented.

Although the development of MIST is ongoing (especially the gradual ex-
pansion of the toolsets which provide much of the sophisticated functionality within MIST), early results have already indicated the significant benefits which it can provide. In conjunction with MAST (the MATILDA Application Structuring Toolset) it provides a powerful tool for performing hypermedia authoring in a way which addresses a number of fundamental issues. Specifically, MATILDA facilitates both information reuse and application maintainability through the explicit separation of information from application. These two issues which will become progressively more important as hypermedia applications grow in scope and complexity. This will be especially true for those applications which evolve over time, requiring ongoing maintenance and development.

In summary, we have found that the MIST implementation of the ideas which evolved out of the MATILDA framework and its underlying formalism has allowed us to extend the capability of the browsing and editing operations for these models. In particular, this will improve the ability to model hypermedia information in a way which will facilitate the development of authoring tools which support information reuse and application maintainability for large complex systems.

Prototype Availability

The current version of the MATILDA Information Structuring Tool (MIST) along with documentation and several example information databases is available from the authors website at http://www.ee.uts.edu.au/eeo/hyvis/. The documentation includes a brief tour of the capabilities of MIST. Comments and feedback are most welcome and should be directed to dbl@ee.uts.edu.au. The authors would also welcome suggestions for collaboration or extensions to MATILDA.

Acknowledgments

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. The authors also wish to thank Mr. Mark Sifer, one of the originators of the work on structured graphs, whose work and fruitful discussions provided many of the ideas which are at the core of the formalism underlying MIST.
References