EMMA – Towards a Query Algebra for Enhanced Multimedia Meta Objects

Sonja Zillner, Utz Westermann, Werner Winiwarter
Department of Computer Science and Business Informatics
University of Vienna, Austria
{sonja.zillner,gerd-utz.westermann,werner.winiwarter}@univie.ac.at

Abstract

For efficient access to multimedia content the media data has to be enriched with additional information about the content's semantic description and functionality, e.g. style sheets for rendering. Current approaches for semantic modeling of multimedia content store the information about the content's semantics and functionality in different files and at different locations, which makes the retrieval and reuse of multimedia content very difficult. We have proposed Enhanced Multimedia Meta Objects (EMMOs) as a new means for semantic multimedia modeling that indivisibly combines multimedia content with its description and functionality. In this paper, we give an overview of the structure of EMMA, a query algebra, which is adequate and complete with regard to the EMMO model, enables query optimization, allows the integration of ontology knowledge, and supports distributive collaborative authoring.

1. Introduction

The way how multimedia content can be searched and accessed depends on the way how this content is stored. Up to now, there are several *multimedia documents models* for storing multimedia content, such as SMIL [1] or SVG [5]. But by primarily encoding the presentation of multimedia content and not the information the content conveys, those approaches only permit the hard-wired presentation of content in the specified way. Valuable information about the semantics of content is lacking for advanced operations like effective retrieval, reuse of content, or adaptation of content to a user's interest. In parallel with research concerning the Semantic Web, a variety of *semantic standards* have emerged, e.g. RDF [10, 4] or Topic Maps [8], which focus on the semantic modeling of multimedia content; i.e. not the presentation of media but their semantic interrelationships are described.

As a novel approach to semantic multimedia content modeling, we have developed *Enhanced Multimedia Meta Objects (EMMOs)* [13] in the context of the EU-project CULTOS[1]. An EMMO constitutes a self-contained piece of multimedia content that indivisibly unites three of the content's aspects: the *media aspect*, i.e. the media which make up the multimedia content, the *semantic aspect*, which describes the content, and the *functional aspect* by which an EMMO can offer meaningful operations on the content and its description that can be invoked and shared by applications. EMMOs are *tradeable* – they can be bundled and exchanged in their entirety including media, content description, and functionality – and are *versionable* – they can be modified concurrently in a distributed collaborative scenario.

As part of the CULTOS project, a distributed infrastructure of *EMMO containers* and an *authoring tool* for the creation of EMMOs were developed. The remaining missing link in this infrastructure has been an adequate query mechanism allowing for the fast access to the knowledge captured by EMMOs, i.e. the three aspects and the versioning lattice.

Our approach to close this gap is to provide a query algebra for EMMOs named *EMMA* as a formally sound foundation for the realization of EMMO query services. This paper gives a high-level overview of the overall structure of EMMA. EMMA is adequate and complete with regard to the EMMO model, enables query optimization, and allows the integration of ontology knowledge into queries. We have defined five classes of modular orthogonal operators – extraction operators, navigational operators, selection predicates, constructors, and a join operator – that can be combined to build complex queries.

The remainder of the paper is organized as follows. Section 2 explains the basic aspects of EMMOs. In Sect. 3 we analyze the requirements of a query algebra for EMMOs. Section 4 takes a look at related query approaches. In Sect. 5

1 CULTOS was carried out from 2001 to 2003 by partners from 11 EU countries and Israel and aimed at providing a collaborative multimedia platform for researchers in intertextual studies enabling them to share and communicate their knowledge about the relationships between cultural artefacts. See http://www.cultos.org for more information.

we present EMMA's design principles and briefly sketch its five classes of operators. Section 6 concludes this paper.

2. The EMMO Model

An EMMO is a self-contained unit of multimedia content that encompasses three aspects, which we would like to illustrate using Fig. 1 depicting an example knowledge structure as used in the domain of intertextual studies.

The *media aspect* describes that an EMMO aggregates the media objects of which the multimedia content consists. In Fig. 1, the EMMO "Jesus Christ" contains the MPEG video "JesusSuperstar.mpeg", the text document "KingJews.doc", and the JPEG image "YellowChrist.jpg". Containment of media objects can be realized either by *inclusion*, i.e. the raw media data is embedded within an EMMO, or by *reference* via a URI, in cases where embedding media data is not feasible.

The *semantic aspect* reflects that an EMMO further encapsulates semantic associations between its contained media objects by means of a graph-based model similar to conceptual graphs. Hence, an EMMO constitutes a unit of expert knowledge about multimedia content. In Fig. 1 the media objects contained within the EMMO are digital manifestations of Jewison's movie "Jesus Christ Superstar", the text "The King of the Jews", and Gauguin's painting "The Yellow Christ". By labeling the associations with the corresponding concepts of an ontology, we can express that the movie "Jesus Christ Superstar" retells the text "The King of the Jews" and resembles the painting "The Yellow Christ". By modelling the semantic associations and EMMOs as first-class objects, the EMMO model becomes very expressive in a way that it is possible to establish references to other EMMOs and to reify associations.

The *functional aspect* expresses that an EMMO offers operations for dealing with its content, which can be invoked by applications. In Fig. 1, the EMMO is associated with a rendering operation, which might return a presentation of the EMMO content in different formats, such as SMIL or SVG.

An EMMO can be *serialized* into a bundle that completely encompasses all three aspects, and is thus *transferable* in its entirety between different EMMO providers, including its contained media objects, semantic associations between these objects, and functionality. Moreover, *versioning support* has been a central design objective: all the constituents of an EMMO can be versioned, thereby paving the way for the distributed, *collaborative construction* of EMMOs.

The formal basis of the EMMO model are *entities* which are defined as 13-*tuples* [13] with the 13 values covering the following information:

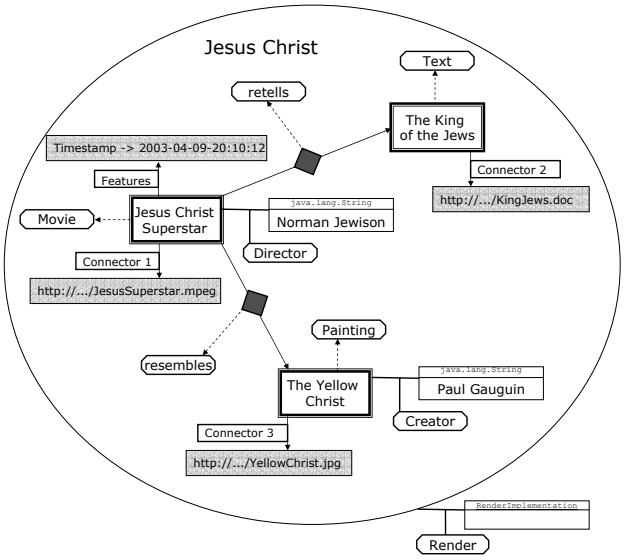

Figure 1. EMMO "Jesus Christ" (e_{jesus})

- Each entity is globally and uniquely identified by its *OID* and carries a human readable *name*.
- The entity's *kind* attribute determines which of the four concrete specializations of entities the entity represents:
  - *ontology objects* represent concepts of an ontology,
  - *logical media parts* represent media objects or parts of media objects, e.g. book chapters or video scenes,
  - *associations* model binary relationships,
  - *EMMOs* aggregate semantically related entities.
- Each entity is labeled by an arbitrary number of concepts of the ontology subsumed in the *types* set.
- The *attributes* of an entity describe an arbitrary number of attribute-value pairs with the attribute being a concept of the ontology, whereas the *features* represent a fixed set of primitive attribute-value pairs allowing to augment an entity with domain independent attributes, e.g. the timestamps information.
- For enabling versioning support, each entity specifies an arbitrary number of preceding and succeeding versions in the *predecessors* and *successors* sets.

The remaining elements and sets specified within the 13-tuple are only relevant for certain kinds of entities, they are left empty for all other kinds of entities:

- The set of *connectors* is only specified for logical media parts and points to the physical representations.

- The *source* and *target* entities, establishing a directed binary relationship between those two entities, are only appropriate for associations.
- Only EMMOs can be further described by the set of *nodes* specifying all the entities contained within an EMMO and by the set of *operations* including an arbitrary number of functions that can process the EMMO's content.

3. Why Do We Need an Algebra for EMMOs?

As the knowledge structures described within the EMMO model are quite complex and, in addition, include multimedia data, the EMMO query language has to enable query optimization for fast query access. Therefore, the querying of EMMOs has to be based on a formal algebra, i.e. a set of *operators* with precise semantics suitable for the EMMO data model, which are defined in a declarative way only describing what needs to be accessed without specifying the underlying implementation. Those operators further need to be *orthogonal*, which means that complex queries can be constructed by sequencing simple queries. Hence, it is necessary that the result of a query can again be used as input for another operator. In addition, the algebra needs an operator for nesting a specified set of entities into an EMMO knowledge structure. On the basis of a formal orthogonal algebra, the equivalence of queries can be described and proved so that in situations where queries can be formulated in different ways and these variants show different performance characteristics, the fastest query can be identified.

Furthermore, the querying of EMMOs should be *adequate* and *complete* with regard to the EMMO model, i.e. all data stored within EMMOs and entities must be accessible. Thus, the algebra should provide operators for the access of all three aspects of multimedia content that are covered by an EMMO:

- *Media aspect:* Operators should be available to extract all data concerning the logical media parts and their connector information.
- *Semantic aspect:* The algebra should offer operators reflecting the graph structure established by the nodes of an EMMO by providing means to navigate this graphical structure, and operators for traversing the established recursive containment hierarchy of EMMOs.
- *Functional aspect:* The algebra should give access to and permit the execution of operators of an EMMO.

In order to fully suit the EMMO model, the algebra should be able to deal with versioning by providing operators to traverse the *successor* and *predecessor* lattice specified within an EMMO.

Moreover, the algebra should provide operators for the *modification* and *construction* of EMMOs. As the database is already supplied with an authoring tool for creating, updating, and deleting EMMOs, the algebra only needs to provide basic modifications of EMMOs, e.g. the union, intersection, difference, or flattening of EMMOs. Finally, as EMMOs are stored in distributed containers, the algebra has to be extended to queries across multiple EMMOs and entities by providing a join operator.

Although the EMMO model describes data on the instance and not the schema level, ontologies can be very useful by holding additional information about the particular application domain. For example, by knowing that an entity or an association is an instance of a specific concept of the ontology, information about superclass relationships between concepts, or about properties of concepts, e.g. the transitivity of an relation, can be incorporated into the query. Thus, in order to make full use of the characteristics of the application domain, the query algebra should enable the integration of *ontology knowledge*.

Similar to the SQL approach, Emma's queries will be written by programs, and not by end users. Therefore, it was not our concern to create a user-friendly language, or to use a human-readable and elegant query syntax.

4. Related Approaches

In this section, we will discuss related approaches to the EMMO query algebra EMMA. Compared to other standards for multimedia content representation, the EMMO approach is unique in such a way, that none of the query languages for those standards can fulfill all the requirements regarding the expressiveness of a language for querying EMMOs. However, valuable aspects of their design have been integrated into the design of the algebra EMMA.

As the EMMO model describes a graph-like knowledge structure, we focus on query algebras based on graph data models. We analyzed query languages for RDF and Topic Maps, regarding whether they allow for query optimization, for navigating the graph structure, and for integrating ontology knowledge.

Although there exist several query languages for *RDF*, there is no official standard yet. Furthermore, as far as we know, there is only one approach, RAL [6], which provides an algebra for querying RDF and thus a basis for query optimization. RAL provides operators for graph navigation and allows to integrate basic ontological knowledge, i.e. the modeling construct within RDF Schema, within the queries.

All the other RDF query languages are not founded on a formal basis; query optimization remains an open issue. RQL [9] provides operators for the navigation along the edges of an RDF graph, and is suitable for querying both RDF description and schema, thus allowing for the integra-

tion of ontology knowledge. There exist graph-navigation languages, such as SquishQL [11] or Versa [12], which specify the syntax of operators for simple graph-navigation, but provide no support for schema integration.

The situation for *Topic Maps* is quite similar to RDF. All approaches for querying Topic Maps introduced so far, such as Tolog [7], TMPath [3], orXTMPath [2], only provide a query syntax and do not address the aspects of query optimization. All those approaches allow the navigation within the graph structure, but only Tolog enables the representation of basic ontological knowledge, i.e. the class-instance relationship, within its queries.

To summarize, there are several approaches, like RQL or Tolog, that allow for graph navigation and for integration of ontology knowledge, but only one approach, RAL, which defines a query algebra suitable for query optimization. However, just like all the other querying languages, RAL does not provide access to the functional aspect and the versioning information. As the EMMO model establishes a unique way of representing multimedia content, the existing approaches are neither adequate nor complete with regard to the expressiveness of EMMOs.

5. The EMMO Algebra – EMMA

The design of the EMMO algebra EMMA was in the first place driven by the requirement of accessing the complete information stored within the EMMO knowledge structures. As the EMMO model constitutes a unique approach of modeling multimedia content, only an algebra which provides fast access to all its characteristic information is adequate for the EMMO model. To enable query optimization, the algebra's operators are simple and modular, and through the combination of modular operators, complex queries can be formulated. The combination of operators can be either realized by sequentially applying the operators or by combining the operators' return values via basic set operators. In this way, the orthogonality of the algebra operators can be realized.

This section gives a brief overview of EMMA's different kinds of query operators. They can be divided into five general classes. The *extraction operators* provide means to query all the attributes of the entities of the EMMO model, the *navigational operators* enable the navigation along an EMMO's semantic graph structure, the *selection predicates* render it possible to select only those entities fulfilling a specific characteristic, the *constructors* allow to modify, combine, and create new EMMOs, and finally, the *join operator* relates several entities or EMMOs with a join condition. In the following subsections, we introduce all five classes of EMMA operators along with illustrative examples, and explain how the operators contribute to fulfill the requirements for an EMMO algebra.

5.1. Extraction Operators

The extraction operators allow to access all the attributes within a 13-tuple representing an entity, and thus are similar to the SQL projection operator. Most attributes of the 13-tuple are simple elements or sets, therefore their access operators are simply designed as projection. Only some of the set attributes describe quite complex structures, e.g. the set of nodes which allows nesting to arbitrary depth, so that their access operators, reflecting this complexity, can become quite elaborate. For example, applying the operator *AllEncEnt*, which allows to access *All Enc*apsulated *Ent*ities of an EMMO, to EMMO "Jesus Christ" (e_{jesus}) in Fig. 1, returns a set consisting of three logical media parts representing the movie "Jesus Christ Superstar" ($l_{superstar}$), the text document "The King of the Jews" (l_{jews}), and the painting "The Yellow Christ" (l_{yellow}), as well as the two associations "Retells" ($a_{retells}$) and "Resembles" ($a_{resembles}$):

$$\textit{AllEncEnt}(e_{jesus}) = \{l_{superstar}, l_{jews}, l_{yellow}, a_{retells}, a_{resembles}\}.$$

By providing extraction operators to access all information stored within the set of *connectors*, we can access the media aspect, and by providing extraction operators for accessing EMMO's *operations*, we can cope with the functional aspect. Moreover, extraction operators for traversing an entity's predecessor and successor versions allow to access EMMO's versioning tree, and extraction operators to access an association's source entity and target entity (describing the semantic relationship between those two entities) provide means for accessing parts of an EMMO's semantic aspect. As a consequence of the EMMO model's complexity, the list of extraction operators is quite long.

5.2. Navigational Operators

The navigational operators allow to traverse the semantic aspect of an EMMO and provide support for logical inference. An EMMO describes a graph-like knowledge structure of entities with associations being labeled by ontology objects (representing concepts of the domain ontology) describing the edges of the graph structure. The navigation is determined by a navigation path, which is defined as a set of sequences of ontology objects. For each ontology object in a sequence, a mapping to the corresponding association within the EMMO is established to navigate through the graph. We have defined *regular path expressions* over ontology objects for describing the syntax of a navigation path; and the navigational operators determine the semantics of those syntactic expressions. For example, the navigational operator *JumpRight* returns for a given EMMO, start entity, and path expression, the set of all entities that can be

reached by traversing the navigation path in the right direction, e.g. applying the operator *JumpRight* to EMMO "Jesus Christ", the starting entity "Jesus Christ Superstar", and the very simple regular path expression consisting of only one ontology object "Retells" ($o_{retells}$) yields the logical media part representing the text document "The King of the Jews":

$$\textit{JumpRight}(e_{jesus}, l_{superstar}, o_{retells}) = \{l_{jews}\}.$$

The regular path expressions specify the wildcard symbol "_" referring to any arbitrary ontology object, the unary operator "$*$" indicating an iteration of path expressions, and the union operator "|". The latter allows to combine two regular path expressions, such that the return value of the operation

$$\textit{JumpRight}(e_{jesus}, l_{superstar}, o_{retells} \,|\, o_{resembles})$$

is interpreted as the union of the two return values of the operator *JumpRight* applied to the two regular path expressions $o_{retells}$ and $o_{resembles}$, which yields a set consisting of the two logical media parts "The King of the Jews" and "The Yellow Christ":

$$\begin{aligned} & \textit{JumpRight}(e_{jesus}, l_{superstar}, o_{retells}) \cup \\ & \textit{JumpRight}(e_{jesus}, l_{superstar}, o_{resembles}) \\ = \; & \{l_{jews}, l_{yellow}\}. \end{aligned}$$

EMMA also provides a *JumpLeft* operator to navigate along associations in the opposite direction, i.e. for a given target entity it retrieves all start entities from which the target entity can be reached via a regular path expression.

By providing means to navigate along the graph structure of an EMMO, its semantic aspect is addressed. Furthermore, the syntax of the regular path expressions provides means to reflect the expressive power of ontology languages, e.g. the inclusion of subclasses of properties can be expressed by the union operator "|", and the transitivity of properties by the iteration operator "$*$".

5.3. Selection Predicates

The selection predicates allow to select only those entities fulfilling a specific characteristic. They basically use the return values of extraction operators to create Boolean operators. The selection predicates can be used with the generic *Select* operator, which takes a predicate and an arbitrary set as input value, and returns all elements of the set that satisfy the condition of the specified predicate. For instance, the operator *IsType* applied to the logical media part "The Yellow Christ" and the ontology object "Painting" in Fig. 1 returns true:

$$\textit{IsType}(l_{yellow}, o_{painting}) = true.$$

If we apply the *Select* operator to the selection predicate *IsType* with the ontology object "Painting" as fixed parameter value and to the nodes of EMMO "Jesus Christ", the result set consists of only one entity, i.e. the logical media part representing the painting "The Yellow Christ". It is the only entity within the nodes of EMMO "Jesus Christ" that contains the ontology object "Painting" in its *types* set:

$$\textit{Select}(\textit{IsType}_{[\$, o_{painting}]}, \textit{nodes}(e_{jesus})) = \{l_{yellow}\}.$$

The selection predicates enable the selection of relevant entities, and contribute to the orthogonality of EMMA's operators.

5.4. Constructors

EMMA specifies five constructors for EMMOs, i.e. the operators *Union*, *Nest*, *Flatten*, *Difference*, and *Intersection*. These operators have all in common that they take at least one EMMO and possibly other parameters as input value, and return exactly one EMMO as output value. For example, the *Nest* operator takes an EMMO and a set of associations and creates a subgraph consisting of only those associations together with their source and target entities. Applying the *Nest* operator to EMMO "Jesus Christ" and the association labeled "Retells",

$$\textit{Nest}(e_{jesus}, \{a_{retells}\}),$$

constructs the new EMMO shown in Fig. 2 consisting of three entities describing the movie "Jesus Christ Superstar", the text "The King of the Jews", and their connecting retelling association.

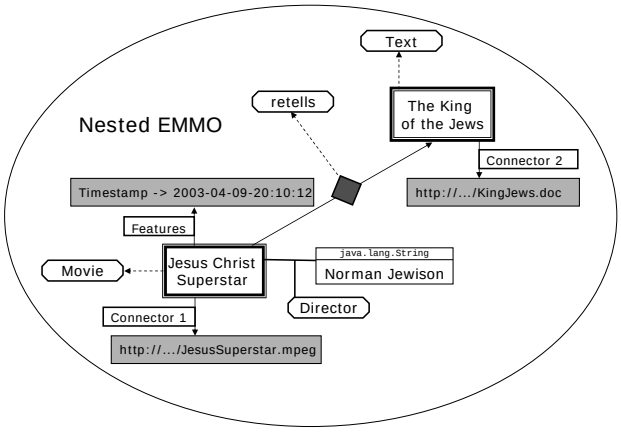

Figure 2. Nested EMMO

By allowing to nest an arbitrary set of associations into an EMMO knowledge container, the construction operators are essential for the orthogonality of EMMA's operators.

5.5. Join Operator

The *Join* operator makes it possible to extend queries across multiple EMMOs. It specifies how to relate n sets of entities, possibly originating from different EMMOs, within a query. The *Join* operator takes n entity sets, n operators, and a predicate as input value. We compute the Cartesian product of the n entity sets and select only those tuples that satisfy the predicate after applying the n operators to the n entities. The result set of tuples is projected onto the first entry. In this way, the *Join* operator is a generalization of the *Select* operator accounting for constraints defined on not only one but several entity sets.

For example, asking for all encapsulated entities within EMMO "Jesus Christ" which have at least one concept (accessed by the extraction operator *types*) in common with the logical media part representing the movie "Life of Brian" in Fig. 3 corresponds to

$$\begin{aligned} &\mathit{Join}(\mathit{AllEncEnt}(e_{jesus}), \{l_{brian}\}, \\ &\mathit{types}, \mathit{types}, \mathit{Not} \circ \mathit{Empty} \circ \cap) \\ =&\{l_{superstar}\} \end{aligned}$$

and yields the logical media part representing the movie "Jesus Christ Superstar", because this is the only entity, which is also labeled as movie.

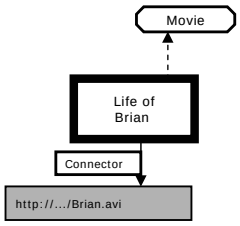

Figure 3. Logical media part "Life of Brian"

6. Conclusion

In this paper, we have given an overview of the EMMA query algebra for EMMOs. EMMA allows access to the media, semantic, and functional aspect, as well as the versioning tree of EMMOs. Moreover, EMMA provides a formal basis for query optimization and the integration of ontology knowledge.

Currently, we are in the process of implementing the query algebra and compiling a comprehensive set of use cases for query evaluation. Future work will focus on the definition and formal proof of the equivalence of operator sequences for variants of queries as an important prerequisite for query optimization. Furthermore, we will develop a language for the definition of ontologies that is compatible with EMMOs, and therefore can be integrated into the query processing. Finally, we plan to carry out a case study in the domain of eLearning to evaluate the feasibility of our approach in a real-world setting.

References

[1] J. Ayars et al. Synchronized Multimedia Integration Language (SMIL 2.0). W3C Recommendation, World Wide Web Consortium (W3C), Aug. 2001.

[2] R. Barta and J. Gylta. XTM::Path – Topic Map management, XPath like retrieval and construction facility. Online Article, available under http://cpan.uwinnipeg.ca/htdocs/XTM/XTM/Path.html, 2002.

[3] D. Bogachev. TMPath – Revisited. Online Article, available under http://homepage.mac.com/dmitryv/TopicMaps/TMPath/TMPathRevisited.html, 2004.

[4] D. Brickely and R. Guha. Resource Description Framework (RDF) Vocabulary Description Language 1.0: RDF Schema. W3C Working Draft, World Wide Web Consortium (W3C), Apr. 2002.

[5] J. Ferraiolo, F. Jun, and D. Jackson. Scalable Vector Graphics (SVG) 1.1. W3C Recommendation, World Wide Web Consortium (W3C), Jan. 2003.

[6] F. Frasincar et al. RAL: An Algebra for Querying RDF. In *Proc. of the Third International Conference on Web Information Systems Engineering (WISE 2000)*, Singapore, 2002.

[7] L. Garshol. tolog 0.1. Ontopia Technical Report, Ontopia, 2003.

[8] ISO/IEC JTC 1/SC 34/WG 3. Information Technology – SGML Applications – Topic Maps. ISO/IEC International Standard 13250:2000, International Organization for Standardization/International Electrotechnical Commission (ISO/IEC), Feb. 2000.

[9] G. Karvounarakis et al. RQL: A Declarative Query Language for RDF. In *Proc. of the 11th Intl. World Wide Web Conference (WWW 2002)*, Honolulu, Hawaii, 2002.

[10] O. Lassila and R. Swick. Resource Description Framework (RDF) Model and Syntax Specification. W3C Recommendation, World Wide Web Consortium (W3C), Feb. 1999.

[11] L. Miller, A. Seaborn, and A. Reggiori. Three Implementations of SqishQl, a Simple RDF Query Language. In *Proc. of the First International Semantic Web Conference (ISWC2002)*, Sardinia, Italy, 2002.

[12] U. Ogbuji and M. Olson. Versa. Online Article, available under http://uche.ogbuji.net/uche.ogbuji.net/tech/rdf/versa/versa.doc?xslt=/ftss/data/docbook_html1.xslt, 2004.

[13] K. Schellner, U. Westermann, S. Zillner, and W. Klas. CULTOS: Towards a World-Wide Digital Collection of Exchangeable Units of Multimedia Content for Intertextual Studies. In *Proc. of the Conference on Distributed Multimedia Systems (DMS 2003)*, Miami, Florida, 2003.