www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 2 Issue 3 March 2013 Page No. 806-810

ODA : PROCESSING MODEL DESIGN FOR LINKING DOCUMENT

R. N. Jugele*[*]*, Dr. V. N. Chavan

[*]Department of Computer Science, Science College, Congress Nagar, Nagpur. Maharashtra, **rn_jugele@yahoo.com**
Head, Department of Computer Science, S. K. Porwal College, Kamptee, Dist : Nagpur. Maharashtra.

Abstract-*Designers of systems have always been deeply concerned about their systems demise. The idea address some thoughts on the implementation of ODA Link systems, particularly with respect to processing design platform that remain malleable and stable through periods of experimentation and radical change.*
When documents are to be accessed the link structure of the documents should be employed. The structures also form the basis for the presentation of the content of the document to the user. The content of document is not just text but a highly structured collection of text fragments and figures.
The model defines link to capture relations and consists of specifications of its starting point and its ending point. The link context structure defines that the displayed component should only be replaced when it is not the component the activated link is pointing to.

Keywords :

Architecture, Link, ODA, Object, Style, logical, Layout, Generic, Content.

1. INTRODUCTION

Traditional document architecture has been thoroughly studied in the area of open information systems. The ISO's Open Document Architecture (ODA) standard covers almost all the concepts developed for document structuring[3]. The ODA standard defines a document as composed of logical and layout architecture and content portions. The logical architecture partitions the content of a document by using semantic rules defined by the author. The layout architecture defines the rules that govern the presentation of the document. The content represents the information that normally associate with documents, for example, the text and the diagrams of a report. ODA-like structures have been also proposed in several multimedia information systems [2][5].
A document architecture is defined to represent the document view. This document architecture is composed of a logical architecture and a layout architecture. The logical architecture partitions the content of the document according to the semantic rules defined[7] i.e. content can be divided into scenes, subscenes, sections, subsections, etc. The layout architecture partitions the content of the document by defining the rules that govern the presentation of the content [6].
Consider a simple model whose structural part is composed of nodes and links. Each node has a unique identity and a content. The content of a node is a sequence of elements which may be character strings and images. A sequence of elements within the content of a node serves as a starting or ending point of a link. A link is defined by its starting and ending point and by its category it is either ‘reference’ or ‘inclusion’. Reference links are intended to create a navigation structure within the nodes. Inclusion links are intended to create nested structures that represent complex contents[4].
A reference link creates an active element whose action consists in jumping to the referred link. A link specification refers to a node through its identity, which is composed of its schema name together with actual parameter values. The source anchor of a link can be any element or list of elements[1]. A reference link is specified with the following syntax :

```
href node_name [ parameter_list ] ( element,...)
```

2. MODEL DESIGN

To provide a user with current information in a timely manner the linking process should be automated but the size of the document, manually updating the links between it, its dynamic nature and the graphical guidance front-end application proved that it is inefficient process. The process has to account for deletions and changes to regulations that are linked to from the graphical guidance front-end and additions of information that might be pertinent to the waste characterization process.
With the original application design it is observed that the major problem is how to establish the links to the referenced document after it underwent changes. For such situation following two steps can be consider :

i. Costs can be reduced while developing similar applications
ii. Improve responsive changes to dynamic reference documents

The approach recommended is to design links into the application so that the linktext references can be maintained as data, which will minimize recoding and redesign when the reference documents change.

The links to the document can be stored in any responsive database system. For each link built a record that included the search criteria and the information required to instruct the engine to directly go to that reference. In this case it needed only the document name and the line number of the referenced paragraph to display the desired information for the application. When the application needed to access the reference engine, it first retrieved the information needed, then passed control with the appropriate parameters to the module via its interface. This concept also made it simple to change engines with minimal or no changes to the application.

Three events will occur for new reference document, a powerful engine can support all three tasks using techniques of searching and knowledge discovery.

i. **The links will have moved:** It is the most common and simplest. It simply involves locating the search field from the database and its new line number in the document. The line number will be updated in the database. Generally this process can be automated with no changes required in the application.

ii. **There is new material in the document:** Preferably it seems that there will be no references to by the application and no action is required. But the database can be used to scan the new document to check for any new information related to existing references. This new knowledge can be reviewed and added to existing links where appropriate otherwise new concepts are added, no changes to the application.

iii. **The reference was deleted:** It require recoding any application dependent upon this reference, for this approach there are following considerations:
   a) If the engines discovery capabilities reveal that a new, related reference has been created the new reference can be substituted again, no change to the application is required.
   b) If the application uses the reference as supplemental information then the database record can be changed to indicate that the supplemental information is no longer available. The application will be designed to interpret this possibility and not offer the information to the user. However, some deletions may require changes to the application when they are essential to overall context.

The ODA architectures, services and applications is developed to support the creation, interchange and processing of documents. It emerged as the result of a fundamental goal to develop a coherent set of standards and capabilities for data interchange. The main focus is the document, its logical structures and data linkages. ODA components key design decisions are made with reference to international standards. Multiple workstations, central processing graphics and data processing applications need to interchange information in a revisable format to produce documents.

Main aims of ODA architecture are :

- Support easy handling of text, graphics and image document content
- Support extensible to new media/data types
- Support for linked applications incorporation
- Deal with data and documents at the end-user level
- Support layered services for document handling
- Support incorporation of industry and international standards
- Support heterogeneous systems

3. THE ODA ARCHITECTURE MODEL

There are four levels as shown in Figure 1.

- The first level based on applications that are the source of information for inclusion in documents.
- The second level is a revisable document where content is added and manipulated. This level is important for logical structure and data linkages.
- The third level is document results which applying formatting rules and layout characteristics to document.
- The fourth level is the transformation of document into specific protocol for display or printing.

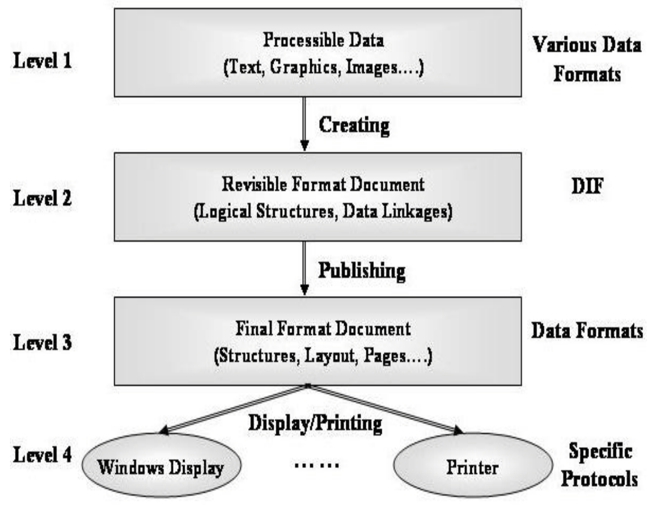

Fig. 1: Open Document Processing Model

The processing steps and data formats of the first level is domain specific because it determined by the needs of many different processing environments. The first level is important because it is the foundation for the access to other applications and data. Application specific data viewing modules provide the link between the level one and level two functions. Interchange of data take place in all levels but the architecture focuses on the second phase documents and their processing.

Document contains abstract relationships between components of the document because these relationships are logical, any aspect of the document that has stated to the creating application can be changed, updated or recalculated more easily than in final form documents. These are more easily developed and operate more efficiently. The revisable form for structured text, graphics and image are specified by the document interchange format which provide inclusion of related data in other formats.

The final form represents the logical document component relationships and attributes. These attributes include text fonts, character positions, positioned, sized graphics frames and final page layout. The final form is produced from revisable form and specifically formatted for a particular class of display.

ODA/ODIF defines the DDIF data format as its extension or specialization. The advantages of its are:

- ODA is not compatible, therefore one makes compatibility with the other some what problematic
- The DDIF format required semantics
- ODA is incomplete in several areas, so extensions could make the result nonconforming and handle linkages between multiple documents and applications.
- ODA does not integrate text and non-text data, it simply combines the existing and nonintegrated standards by layering structure primitives on top.

4. ARCHITECTURE OF DOCUMENT CONVERSION

The ODA converter architecture comprises a layer that supports document formats. Following are the goals :

- Support procedural interface for data format conversion
- Support many document formats
- Incorporate non-Digital conversion
- Dynamically conversion modules
- It build on existing ODA services

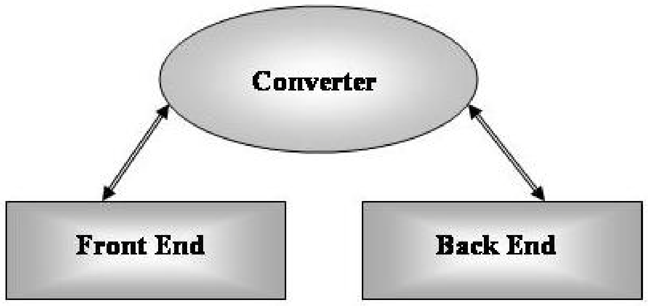

Fig. 2: Document Conversion

The converter architecture is based on a conversion model. A front-end converts an input document to a format. A back-end module then converts the format to an output format. The format are in DIF format.

The converter control procedure assembles the complete conversion program at execution time based on the conversion modules installed and available at the time they are referenced shown in figure 2. Document data flow through the conversion. Each operating system includes a small number of essential conversion modules as standard equipment shown in figure 3.

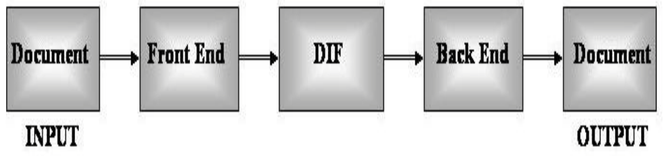

Fig. 3: Flow of Data in Conversion

It works on DIF conversion formats. A front-end or back-end module operates in DIF domain or in many cases the front end and back end operate in the same domain. For example word processing document conversion. Module added additional logic to the converter to permit such domain-crossing conversions which receives aggregates from one domain and translates them to aggregates of another domain.

5. LINK CLASS

The link class fulfill two tasks :

i. It specifies which actions are sent to which objects.
ii. The conditions are specified under which this process occurs.

From these tasks, it can be derived that the processing is based on an event-driven processing model. This model is suitable for the mapping of parallel running, synchronized processes which exist often in interactive multimedia presentations. Their sequentialisation depends on the performing system.

A link object consists of a set of links. The semantics of a link is shown in Figure 4. A link connects a source object with one or several destination objects. The source can be a virtual views. The performance of a link is always dependent on a condition, which can be expressed through the possible state transition in the source object. Only if this condition is satisfied, extra conditions are checked. If all conditions are satisfied, the link is active. In this case, the action objects, specified in the action link are sent to all destination objects.

Although the standard does not specify the implementation, it may be appropriate to mention that the link is checked only if a state transition of the particular source objects occurred. The attachment point is used to position destination objects relative to the source object. It means that coordinates in an action object express a relative position of the destination with respect to the source.

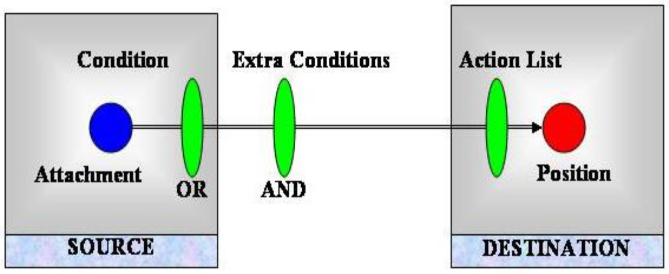

Fig. 4: Link Construction

5.1. Processing Link Connections

Link connections are of three types :

- Link to picture : It references a data file that is imaged DIF file on the page.
- Link to application : It references a data file and application names . The application can be invoked to process the data and deliver DIF content for presentation on the page.
- Link to document : It references a document whose pages are merged with referencing document from a number of smaller documents.

A link is a reference link (href) that triggers action when traversed. In addition to usual elements the source of an active link can have one or more input elements and must have one action element. The general syntax of link is :

href node_name[parameters] (standard or input or action elements)

6. INDEX IMPLEMENTATION

The ODA model track ODA evolution compatibility even though philosophy maintains that the relationship between structure and content should be more highly integrated. Document model including the treatment of generic and specific structure but need to adopt a different treatment for combining structure with content.

6.1. Generic structure extension

The generic structures are made extensible in such a way that they can accept the pieces of generic structure needed by an application. The elements and the attributes required to handle the indexes are defined as a generic structure extension which occur dynamically called the Index. Thus document have index tables which defines number of elements and attributes.

6.2. Passage delimiters

A paired component is called passage delimiter and is defined in the Index. Several logical elements are concerned by an index entry. The delimiters comply with the logical structure of the document. The pairs are siblings but several passages may overlap in a sequence of three paragraphs, the first and the second may constitute a passage which is referred by the index and the second and the third may constitute another passage which is also referred by another index entry as shown in figure 5.

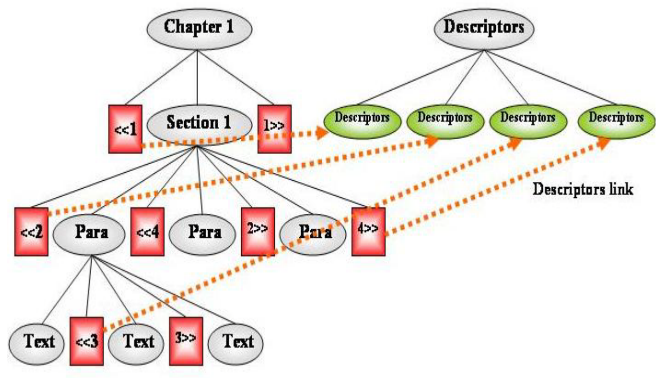

Fig. 5: Delimiters in the document structure

Indexes delimiters and descriptors of generic structure is :

STRUCTURE EXTENSION Index
EXTENS
ROOT + (passage delimiter);
STRUCT
passage delimiter (ATTR !descriptors Link = REFERENCE (passage descriptor)) = PAIR;
ASSOC
passage descriptor (ATTR !delimiters Link = REFERENCE (passage delimiter)) =
BEGIN
Keys (ATTR Level = low, normal, high) = LIST [1..3] OF (Key = Text);
? Subject= Text;
? Semantics = Text;
? Gloss = Text;
END

<<1>> Delimits Section 1 of Chapter 1
<<2>> Delimits First and Second Para in section 1
<<3>> Delimits few words in the First Para of section 1
<<4>> Delimits Second and Third Para of section 1

It is inserted around a previously selected passage within the structured document. When an opening or closing paired component is selected the editor automatically selects the paired component. When the delimited passage runs over a series of consecutive pages, the indexing system will display a range of pages in the index table. The passages delimited by passage delimiters are linked with passage descriptors. This is achieved with reference attributes, called descriptors Link, associated with each opening paired component. This specification extends the root element (ROOT) of any document. It defines the passage delimiter as a paired component (PAIR). A mandatory (!) attribute, called descriptors link is a reference to an element of type passage descriptor.

6.3. Descriptors

Passage descriptor describes an index entry. These descriptors are associated elements (ASSOC). Each descriptor contains a term (KEY) and sometimes a subject. All the descriptors are displayed in a separate view of the document. By double-clicking on passage delimiter the descriptor view is automatically scrolled to display the linked passage descriptor.

7. CONCLUSION

ODA family of architectures, services and applications form support environment for documents. This support is designed and implemented to operate consistently on multiple hardware and software platforms. Significant efforts have been undertaken to ensure that the ODA benefits are not restricted to Digital-produced products. There is still much work to do, however, as the applications and requirements for documents

continue to expand into new media types and into more dynamic relationships. These areas are now active areas of ODA architecture research and development.
The proposed methods have been used in a case study. The performance is sufficient to identify most of the structures automatically. The large number of links found between text and figures. However, they relieve the document creator from the tedious task of identifying all links which are difficult to identify automatically.

8. FUTURE STUDY

ODA offers the possibility to implement an open standard with integration of continuous media. This would allow the exchange of multimedia documents in the same way as we exchange text documents through the mailing systems today. But, there are still many missing aspects.

- ODA will have to consider security and color aspects.
- Backwards compatibility should be preserved besides text and graphic.
- Tables and data will be supported in documents. This will require a data exchange between a document and spreadsheet as well as a transformation from data to text.
- The notion of partial documents should be introduced. Partial documents are incomplete documents which include external pointers.
- Formulas should be included as part of ODA.
- A version management should be introduced.
- Content architectures for others should be defined.

REFERENCES

[1] B.C. Watson and R.J. Davis. ODA and SGML: an assessment of co-existence possibilities. Computer Standards & Interfaces, 11:169{176, 1991.

[2] Christodoulalds S, Theodoridou M, Ho F, Papa M, Pathria A (1986) Multimedia document presentation, information extraction, and document formation in MINOS: a model and a system. ACM Trans Office Information Syst 4:345-383.

[3] ISO (1988) Information processing: text and office systems; office document (ODA) and interchange format. ISO 8613, parts 1-8

[4] J. Nanard, M. Nanard. (1993)."Shoud Anchors Be Typed Too? An Experiment with MacWeb". In Proc. of the Hypertext'93 Conf., Seattle, 51-62, 1993.

[5] Megheni C, Rabetti F, Thanos C (1991) Conceptual modeling of multimedia documents. IEEE Comput 24:23-30.

[6] R. N. Jugele and V. N. Chavan, Structure of Multimedia Documents : Theoretical Approach, International Journal of Computer Science and Telecommunications, ISSN 2047-3338,Vol. 3, issue 9, Sep. 2012, pp. 26-31.

[7] R.N. Jugele and V.N. Chavan, ODA: A Study of Document Design, International Journal of Emerging Trends & Technology in Computer Science**,** ISSN:2278-6856,Vol. 2, issue 1, Jan-Feb - 2013.

Books :

01. Principles of Multimedia
By. Ranjan Parekh
Tata McGraw Hill Companies.

02. Hypertext and Hypermedia.
By. J. Nielsen
Academic Press.

BIOGRAPHIES :

R. N. Jugele received his B.Sc. degree from Nagpur University and M.Sc. degrees in Computer Science from Marathwada University, Aurangabad, Maharashtra, India in 1991 and 1993 respectively. Currently, he is working as a Associate Professor in Department of Computer Science, Science College, Congress Nagar, Nagpur. Maharashtra.

Dr. V. N. Chavan Head, Department of Computer Science, S. K. Porwal College, Kamptee, Dist : Nagpur. Maharashtra. He is research guide in Computer Science subject in various universities and have wide knowledge in research field since last 22 years. He is a member of various bodies in Computer Science subject.