Translating Among Processable Multi-media Document Formats Using ODA

Jonathan Rosenberg
Mark S. Sherman
Ann Marks
Information Technology Center
Carnegie Mellon University
Pittsburgh, PA 15213

Frank Giuffrida
Center for Information Technology Integration
University of Michigan
Ann Arbor, MI 48105

Introduction

This paper describes our experiences in translating processable multi-media documents among diverse systems. We performed these translations using the international standard Office Document Architecture (ODA) as an intermediate representation. The next two sections provide brief descriptions of the NSF-funded EXPRES project (under which this work was performed) and our goals for this phase of the project.

Following that is a brief description of the multimedia document systems at Carnegie Mellon University (CMU) and the University of Michigan (UM), the two institutions at which this work was performed. This is followed by a brief introduction to ODA and the ODA Tool Kit, a subroutine library that was used in the construction of the translators.

These sections are followed by a discussion of general issues relating to the fidelity of document translations. We then describe our approach to document format translation using ODA, the problems we encountered and our solutions. We provide some details about the implementation of the translators, our current status and our conclusions on the effort. The last section discusses some directions for further work on document interchange and an appendix contains a listing of the document format features that we supported in our translations.

The EXPRES Project

In June 1986, the National Science Foundation (NSF) solicited proposals for the Experimental Research in Electronic Submission (EXPRES) project. EXPRES was designed to promote the electronic interchange of multi-media documents among the scientific research community. The NSF proposal submittal process was chosen as a controlled sub-environment in which to pursue the goals of the project, because this process embodies many of the issues that must be addressed by EXPRES. In particular, proposals are documents that usually contain information expressed in several media, such as text, line drawings and raster images.

In September of 1986, the NSF made equal three-year awards to the Information Technology Center at CMU, and to the Center for Information Technology Integration at UM. In addition, several vendors have donated equipment and provided technical support to the EXPRES project: IBM, DEC, Apollo Computer, Sun Microsystems and Apple.

There are many issues that must be addressed to improve the ability of researchers to interchange multimedia electronic documents. To make the task manageable, we are concentrating on the problem of effective interchange of processable multi-media documents among diverse systems. In particular, we are ignoring the method by which a document is transferred from one system to another. Instead, we are concerning ourselves with the question of how a multi-media document created on one system can be viewed and edited on another system.

The proposal submission process, like the scientific collaboration process, involves many different hardware and software tools. Therefore, any attempt to automate these processes must account for broad heterogeneity. Devising an interchange scheme that accounts only for a fixed set of multi-media document formats (for example, the formats in use at CMU and UM) fails to address a critical requirement: the ability of a new document system to engage effectively in free interchange among existing systems.

The obvious technique of performing translations between each pair of systems is impractical, because the addition of a new system would require the construction of new translators for, at least, some of the existing systems. In order to attack this problem efficiently, the EXPRES project uses a standard representation that is translated to and from for each system. This technique requires no modifications to the existing set of translators for the entry of a new system. We have settled on the international standard Office Document Architecture (ODA) as the intermediate format [4].

To demonstrate the effectiveness of this translation strategy, we are building applications to translate between two multi-media document systems and ODA. The CMU project is building translators for the Andrew Toolkit (ATK) document system [8]. UM EXPRES is writing translators for translating between the Diamond multi-media system [11] and ODA. ATK and Diamond were chosen because they are powerful multi-media systems that are in active use within the computing environments at CMU and UM.

For the first phase of the interchange experiment we have restricted our multi-media types to multi-font, structured text and raster images (a listing of the kinds of text structuring we have chosen appears in an appendix). We chose text and rasters because they are, in our opinion, the two most useful multi-media types and are well supported by both the CMU and UM systems, as well as by many other document processing systems.

If we are able to demonstrate that interchange between dissimilar systems using ODA is effective, we would like to ease the task of others that wish to pursue this strategy. We believe that one of the most practical means for doing this is to produce code for applications that manipulate ODA documents. The ODA Tool Kit described in this article is such a tool: it is a highly-portable C subroutine library that provides a useful set of facilities for applications that process ODA documents.

Andrew and UM EXPRES

The Andrew project [6] at CMU is a natural environment in which to further the goals of EXPRES. Andrew is a joint project between IBM and CMU and is designed to support multi-media document preparation, electronic messaging and educational software construction on the CMU campus.

The primary application software provided by Andrew is the Andrew Toolkit and an associated set of applications [3]. The ATK is a subroutine library for high-function workstations (such as IBM PC-RT's) that can be used by application programs to manipulate multi-media documents. The ATK contains support for creating and modifying media objects, for displaying these objects on a screen and for producing hardcopy. Currently, the ATK supports multi-font text, hierarchical line drawings, equations, spread sheets, raster images and simple animated line drawings.

The EXPRES project at UM is based on the Diamond multi-media system, which was developed under a DARPA contract by Bolt, Beranek and Newman, Incorporated. Diamond is a system that allows the creation and modification of multi-media documents. Like the Andrew Toolkit, the Diamond system runs on high-function workstations and provides the capability for dealing with a number of multi-media objects: multi-font text, raster images, hierarchical line drawings, spread sheets, equations and voice.

While the Diamond and Andrew multi-media systems are quite similar in their capabilities, they were developed independently and are strikingly different in their underlying implementations. This provides the EXPRES project an ideal environment for the investigation of document interchange between dissimilar systems.

The Office Document Architecture

The Office Document Architecture (ODA) is an international standard designed to facilitate the interchange of multi-media documents. One of the important factors in our choice of ODA was that it includes a complete semantics for specifying the layout of a document. We felt that interchanging only the logical structure of a document would not be sufficient for effective interchange, but that users would insist on the ability to specify the appearance of the document.

ODA defines a document architecture, several content architectures and two datastream formats. The *document architecture* is the means by which the structure of a document, irrespective of its content, is represented. In general, an ODA document is represented using two sets of structures. The *logical structure* is based on the meaning of various divisions of the document. For example, the logical structure of a document might consist of chapters, sections and paragraphs. In the *layout structure*, the document is structured on the basis of presentation. For example, the layout structure of a document might consist of pages and, within the pages, frames and blocks that define headers, footers and paragraphs.

In addition, each structure may exist in two forms: *generic* and *specific*. A generic structure may be thought of as a template or macro that allows structure information to be collected and referenced. For example, the *generic logical structure* of a document might indicate that the document consists of a title, followed by one or more sections, followed by a set of references. Correspondingly, a *generic layout structure* for the same document might indicate that the title is a block that appears two inches from the top of the first page and is centered, each section is preceded by a section number and the first line of each paragraph is indented 0.5 centimeters.

If the generic structures of a document can be thought of as macros, then the specific structures represent invocations of those macros. The *specific logical structure* is, thus, the actual structure of a document. For example, the specific logical structure might show that a particular document consists of a title, five sections and a set of seven references. There is a *specific layout structure*, corresponding to the generic layout structure, but it is used only for the representation of a final form document (one that may be imaged). Since we are concerned only with editable documents, our translation schemes do not use any specific layout structures.

The actual content of an ODA document consists of instances of *content architectures*. Each content architecture defines its own internal structure, which may consist of logical and layout structures. There are currently three content architectures defined within ODA. *Character content architecture* defines the presentation and processing of characters and allows the specification of graphic character sets, multiple fonts, ligatures and formatting directives such as indentation and justification. *Raster graphics content architecture* defines pictorial information represented by an array of picture elements. *Geometric graphics content architecture* defines representations of picture description information such as arcs and lines.

A *datastream* is an out-of-memory representation for a document that is suitable for storage in a file or transmission over a network. The ODA standard defines a binary datastream format known as the Office Document Interchange Format (ODIF).[1]

Documents represented in ODA are graphs, the nodes of which are known as *constituents*. Each constituent has a set of *attribute-value* pairs. Attributes have values that control the presentation and layout of the document. For example, the value of the attribute "Separation" at a constituent will control the distance between blocks of text when the document is displayed or imaged.

Figures 1 and 2 provide an example of a two page document and a graphic representation of a corresponding ODA document. In figure 2 only the specific logical and specific layout structures are used.

The ODA Tool Kit

To ease the task of working with ODA, the ODA Tool Kit [9] provides a set of facilities that are common to all ODA applications: data type definitions, reading and writing the ODIF datastream and managing an in-memory representation of an ODA document. The ODA Tool Kit is a C subroutine library that has been designed to be extremely portable. In particular, we expect that the Tool Kit can be installed with a small amount of effort on virtually any machine-operating system combination provided that a viable C compiler is available.

Besides saving the time required for each group to write code to support ODA, use of the Tool Kit had some additional benefits. Using common code minimized the opportunities for each group to assign different interpretations to parts of ODA. While using different implementations may be a good way to detect incorrect interpretations, the time frame in which we

[1]ODA defines another datastream representation, the Office Document Language (ODL), which is a clear text representation that conforms to the Standard Generalized Markup Language (SGML) standard [5]. Note that this does not imply that there is a direct relationship between an ODA document and the equivalent document marked up using SGML.

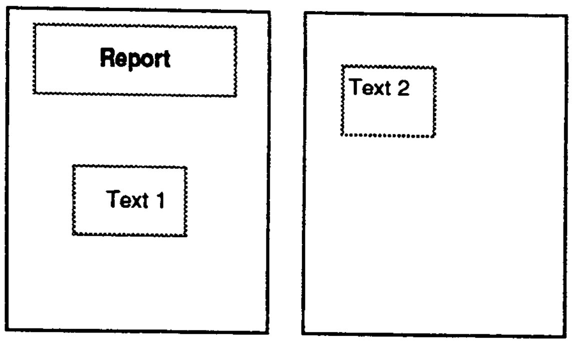

Figure 1: The sample document consists of two pages. The first page contains a title ("Report"), that is centered and bold-faced and a centered paragraph of text ("text1"). The second page contains a left-justified paragraph of text ("text2").

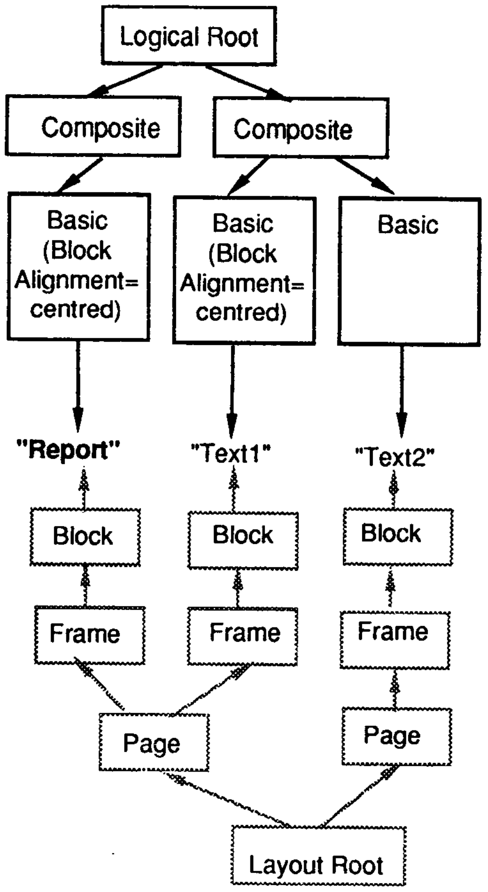

Figure 2: The ODA specific logical structure consists of a *composite* object for each page and a *basic* object for each paragraph. The centering of the text is accomplished by attaching the attribute "Block Alignment" with the value 'centred' to two of the basic objects. The specific layout structure consists of two *page* objects each of which contains a *frame* and *block* object to position the paragraphs.

had to work and the complexity of ODA made the use of common code imperative. In addition, using the Tool Kit in the early stages of its development ensured that the same functionality was available to all translators. This allowed us to begin interchanging documents at an early stage without concern about a mismatch in the degree of implementation of each project.

Translation Fidelity

When translating a document created on one system into a document to be viewed and edited on another system, it is necessary to decide the kind of fidelity is desired. The simplest kind of fidelity is *hardcopy fidelity*: the document should look the same when printed on both systems. This is rather easy to accomplish by interchanging a *page description language* representation of the document, such as PostScript [1, 2] or Interpress [12].

The next level of fidelity is *display fidelity*, in which we are interested in retaining the general appearance of the document when it is viewed on the screen. For example, if the size of the screen region is made narrower, then the editor must be able to shrink the line length and rewrap lines, as necessary. Display fidelity is trickier to obtain than hardcopy fidelity, because it requires that some of the logical structure of the document be interchanged. The receiving system, for example, must be able to distinguish hard line breaks (which must be preserved) and soft line breaks (which may be readjusted for viewing).

The hardest kind of fidelity to obtain is *editing fidelity*. Editing fidelity requires display fidelity but, in addition, the document must be editable in a manner similar to that on the originating system. This is particularly important for EXPRES because we are concerned with allowing collaboration on the construction of a multi-media document among people using dissimilar systems.

Our most compelling example of an editing feature to be retained across translations is *style sheet* information. It is common among document systems to provide a mechanism by which the user can package formatting information into a template, frequently known as a *style sheet*. For example, the user may define a style known as "quotation", in which the right and left margins are indented and which uses an italic font face. This style may then be applied to various regions of text. The crucial point about this style is not just that the designated regions appear indented and in italics, but that if a modification is made to the definition of "quotation", then the change will be reflected in each appropriate region.

The preservation of editing information requires that much of the logical structure of the document as perceived by a document system be preserved across translations. The next section discusses the preservation of information using ODA as the intermediate translation format.

Translation Using ODA

The problems that can occur when translating from one document format to another using an intermediate representation can be classified into four categories. In addition, each problem has two opportunities to occur: during the translation to ODA and during the translation from ODA. The four categories are:

1. *Missing functionality*: the intermediate format does not support some feature. For example, ODA does not support mathematical equations, which are supported by ATK and Diamond.

2. *Detail mismatch*: there is a minor difference in semantics among the formats for some feature. For example, each format may impose a slightly different semantics on underlining (such as how far under the baseline the line appears or whether the space character is underlined).

3. *Semantic redundancy*: a document format provides multiple methods for accomplishing a single task. In ODA, for example, there are at least eight distinct ways to represent a change to the left margin of a paragraph.

4. *Editing information loss*: editing information is lost during one of the translations. For example, some document systems allow a region of text to indicate that it is to be indented one inch further that the surrounding text. If this region of text is moved within the document, its indentation will change to be one inch inset from its new surrounding text region. ODA does not allow a relative margin change, so the region of text will be translated with an absolute positional specification. The relative information has been lost.

In the short term, there is not much to do about the first two problems: missing functionality and detail mismatch. Fortunately, the detail mismatch problem promises to be only a minor nuisance; they are, after all, only details. Missing functionality is more problematical as it means that whole pieces of functionality are unavailable through translation. There are various schemes that could be imagined to circumvent some of the loss. For example, equations could be transmitted between Diamond and ATK by rendering them as raster images, which ODA supports. Of course, they would no longer be equations and we consider this an unacceptable long term solution (although it might be of practical use in the short term). We are investigating a number of solutions to this situation as possible future EXPRES work.

Unlike the first two problems, semantic redundancy is manageable: the translator just needs to contain enough code to recognize all possible combinations of interacting features. Unfortunately, this solution may be impractical for many translators due to time constraints or the number of combinations that must be considered.

The creators of ODA recognized that the size of ODA would be a problem and so provided for *Document Application Profiles* (DAP's). A DAP is a specification of a subset of ODA and may, therefore, restrict the features of ODA that an implementation must support and the manner in which these features may be combined. A DAP is a recognized ODA concept that may be registered with a central authority and assigned a unique identifier that is used within an ODA document to indicate adherence to the DAP.

A DAP is some help in solving the semantic redundancy problem simply by its restriction of the amount of ODA that must be considered. For EXPRES, we have used the the National Bureau of Standards (NBS) DAP [7] as a starting point for our efforts. The NBS DAP is being specified by one of the OSI Implementor's Groups, which meet regularly at the National Bureau of Standards to define agreements among implementors of OSI standards. The NBS DAP is concerned largely with the high-level structure of a document and provides for restricted forms of numbered chapters and sections, footnotes and other architectural features. The NBS group has wide international representation and we expect the NBS DAP to have widespread acceptance. We have, however, found it necessary to deviate from the NBS DAP in several instances and we are lobbying the NBS group to modify the DAP to accommodate our needs.

To make the translation problem manageable within the time frame of EXPRES, it was necessary for us to make additional agreements beyond those in this DAP. These agreements specify a single ODA representation for a set of document features, such as "left margin" and "superscript".[2] For example, we have an agreement that states, in effect, that, "the left margin of a paragraph will be represented in ODA by the use of the 'Offset' attribute". This set of agreements greatly simplifies the translation task. A strict requirement for us was that any document represented according to our conventions must be processed correctly by a translator (or native ODA system) that does not understand our conventions. This means that our agreements must not change any of the semantics of ODA.

Note that agreements such as this are strictly outside of ODA and it is, in fact, not possible to specify them in a DAP. This is because a DAP may only specify a subset of ODA and a subset is, in general, not a precise enough mechanism for ensuring a single representation for a document feature.

Unfortunately, the *editing information loss* problem is inherent and no amount of programming will overcome it. There are many kinds of editing information that might be lost during translation to ODA. Just as importantly, the style sheet systems of ATK and Diamond are more flexible than the mechanisms provided in ODA for grouping sets of attributes. The next two sections discuss our approaches to these problems.

The Andrew Toolkit Translators

There are several ways that the ATK translators lose editing information. Ultimately, these losses are caused by differences in the document models of ATK and ODA. In this section, we briefly discuss our general translation strategy and our attempts to deal with various kinds of editing information loss in the ATK translators.

The formatting of an ATK document is controlled by the *formatting state*. The formatting state is a collection of values for document features such as the font to use and margin sizes. A *style sheet* is a collection of changes to the values of specified features. A style sheet is applied by surrounding a contiguous portion of the document with an instance of the style sheet. For translation purposes, the important aspects of the ATK style sheet mechanism are that it provides:

- *Relative changes*: changes to the value of a feature can be a function of the current formatting state. For example, a style sheet could specify that the left margin is to be increased by one inch from its value in the formatting state. Notice that the value of the left margin for this style sheet is determined at application time, not definition time.

- *Fine grain control*: the value of any feature can be changed without affecting any other feature's value. For example, each part of a font's description (family, size, face design) can be specified independently.

- *Hierarchical application*: applications of style sheets are strictly nested but otherwise unrelated to document structure. The changes specified by the style are in effect throughout the enclosed portion.

The ODA document model also supports the factoring and abstraction of changes to the formatting state through the use of styles. Certain groups of ODA attributes can be attached to a *presentation style* or a *layout style* and this style can be referenced from the document structures. The styles provided by ODA differ from the ATK style mechanism in several important ways:

- ODA provides no relative changes; all attribute values are absolute.

- Many ODA attributes do not allow fine grain control over the values of document features. For example, when specifying a font, it is necessary to specify all parts of the font's description.

- An ODA style affects a subtree of the document architecture, not an arbitrary contiguous portion of the document.

ODA's non-uniform treatment of styles makes it difficult to find a mapping for ATK style sheets and their applications into ODA. The basic architecture of our solution is to translate an ATK style sheet into parts of the generic layout structure: a combination of a layout style, a presentation style and a generic page. (A *generic page* is a constituent of the generic layout structure used to provide a template for the layout of a page.) The two kinds of styles are needed because ODA requires the separation of the attributes that affect document content (attached to a presentation style) and those attributes that affect the document layout architecture (attached to a layout style). The generic page is required because there is no way to associate header and footer information with an ODA style.

Because ODA insists on non-relative values for attributes, there is no unambiguous way to represent an ATK style sheet that contains a relative change. Our answer is to represent a relative change by an ODA attribute whose value is the result of applying the rela-

[2] A full listing of these document features may be found at the end of this paper.

tive change to the default value in ATK. For example, assume that the default left margin in ATK is 1 inch and that a style sheet changes the left margin to be 2 inches greater than its value at application time. This relative change appears in ODA as an attribute with the value 3. Thus, the relative change can be inferred by computing the difference between a particular style's value and the default style's corresponding value.

Unfortunately, this is an ambiguous mapping: if the style sheet in the example above had specified an absolute left margin of 3 inches, the representation in ODA would be identical. After many years of creating documents with ATK, it has been our experience that the vast majority of style sheet changes are relative.[3] The ODA to ATK translator, therefore, creates relative changes in style sheets whenever possible.

Translating a style sheet using this technique leads to cases where the style information in the generic structure is not appropriate for a specific style sheet application. Continuing the previous example, if the style sheet were applied to a portion of the document where the current left margin were 4 inches, then the appropriate changed left margin would be 6 inches. But, our scheme has the value 3 in the translated style sheet. The solution is to attach an attribute with the appropriate value, 6 in this example, directly to the affected portion of the document in the specific logical structure. By ODA's inheritance rules, the locally specified value will override the value specified in the generic structure.

This technique (specifying the actual attribute value locally) also allows us to simulate the fine grain control available in ATK. The trick is to provide ODA values different from the default values only for those features that were changed in the ATK style sheet. By comparing the values, the ODA to ATK translator can infer the fine grain changes for ATK. Notice that, in general, it will be necessary to specify the correct values locally in the specific logical structure in order to override the style's value (which has been encoded to represent ATK features).

The difference in style application between ATK (over a contiguous portion of the document) and ODA (over a subtree of the document architecture) is problematical when translating from ATK to ODA. This is because an ATK style sheet application may cross arbitrary boundaries in the corresponding ODA structure. In such cases, the translator must be careful to shuffle attribute values around so that unnecessary side effects are not obtained.

The Diamond Translators

Like ATK and ODA, the Diamond system also allows changes to document features to be collected into a style sheet, which can then be used to affect the presentation of portions of the document. The general translation scheme described in the previous section, representing a style sheet in ODA as a generic page and associated layout and presentation styles, is part of our translation conventions and, therefore, also used by the Diamond translators. Because Diamond's style sheet mechanism is different than ATK's, additional problems arise. For this discussion, the salient aspects of the Diamond style mechanism are:

- *Global features*: Diamond has a set of document features whose values apply implicitly to the entire document: page margins and header and footer text, for example. Once set, these values may not be changed.

- *Limited relative changes*: some features in Diamond require relative changes. For these features it is not possible to specify an absolute value; for all other features an absolute value is required.

- *Hierarchical definition*: style sheet definitions form a hierarchy. A style sheet definition, thus, inherits values from its ancestors' definitions. (This is unlike ATK, in which style sheet definitions form a flat structure.)

- *No nested applications*: style sheet applications may not be nested: any portion of the document is surrounded by at most one style sheet.[4]

The unmodifiability of Diamond's global features may cause information loss during translation from ODA to Diamond and back to ODA. Consider an ODA document that includes a style that changes the top margin in the middle of the document. The Diamond translator represents this as a style sheet that changes the space above paragraphs, and then applies this style to the first paragraph on each page. This is necessary because the top margin in Diamond is a global feature and cannot be modified within the document. Although the document will be rendered correctly in Diamond,

[3]This is likely true because the ATK editor is not a true WYSIWYG editor and is not really appropriate for producing camera-ready copy. This makes the use of absolute specifications less desirable.

[4]There is a restricted kind of style sheet that can be nested within other style sheets. These are known as *font delta's* and may only specify changes to the current font.

the translation back to ODA will contain a style for changing space above paragraphs. The change to the document top margin has been lost.

Unlike ATK, there is no loss of relative change information in the Diamond translators. This is because each document feature is predetermined as requiring either relative or absolute specification, there is no choice.

Unlike ATK and ODA, Diamond does not allow applications of style sheets to be nested. Diamond does, however, allow the definitions of style sheets to be nested. The general translation strategy for Diamond, therefore, is to define style sheets whose definitions represent the nested applications of the styles found in the ODA document. These style sheets may then be applied in a non-nested fashion corresponding to their original nested applications.

Unfortunately, this works only if the relationships among style applications in the ODA document are strictly hierarchical. This will not be true if two style sheets, say A and B, are nested within each other in the document. In this case the naive approach would have A's definition nested in B and B's definition nested in A. The Diamond translator solves this dilemma by creating two new style sheets: one for representing the style sheet that results from applying A to B and one for the result of applying B to A. The price of this solution is that there is no way for the Diamond to ODA translator to reconstruct the original style application structure.

Status

During July 1988 we completed our conventions for the current phase of the interchange experiment [10]. Implementation of the ODA Tool Kit has been underway since February and the Tool Kit currently contains support for BSD Unix, System V Unix and MS-DOS. We expect to provide support for MPW on the Macintosh and VAX/VMS in the near future.

So far we have built only skeleton translators for producing ODA from ATK and Diamond. These translators generate the architecture and content for an input document; they do not yet produce all of the necessary attributes. The translation from a multi-media format to ODA is much more difficult than the translation from ODA to a particular format. For this reason, we have been concentrating on the translators from ATK to ODA and Diamond to ODA. As of September 1, 1988, we have been able to translate simple documents represented in ODA into ATK or Diamond. These documents consist of paragraphs of text, left-flushed, right-flushed, justified or centered.

Although the documents we have translated to this date have been simple looking, their representation in ODA is structurally rich as they obey the complete set of EXPRES interchange conventions. Because of this, we can state some useful conclusions about translation among processable multi-media document formats using ODA.

Conclusions

We can state unequivocally that building the ODA Tool Kit was a good idea. The Tool Kit has proven to be extremely portable and its use has, in fact, allowed us to begin interchanging documents early.

Our experience in collaboration on papers has convinced us that users will not be satisfied with exchanging only the logical structure of a document; they will insist on being able to convey particular presentation syles, such as indentation and inter-paragraph spacing. Style sheets are a powerful means for abstracting author intentions about document layout. Our extensive use of advanced multi-media editors has also shown us the importance of style sheet information when modifying a document.

We believe, therefore, that the key to effective interchange of processable multi-media documents is the ability to translate both presentation information and style sheet information as intended by users. All of our experience to date leads us to believe that translating to and from an intermediate format is a viable technique for the interchange of documents with presentation and style sheet information.

ODA is particularly appropriate for the interchange of layout information because of its rich layout architecture. The power of ODA's layout structures is due to two features: a well thought out semantics for layout attributes and the fact that the layout structure of an ODA document is disjoint from the logical structure. This means that the layout specification of a document need not be tied in a direct way to the logical structure, but is free to span parts of the document.

Unfortunately, using ODA to transmit style sheet information is problematical. Although ODA does provide style abstraction mechanisms, they are not powerful or flexible enough to represent the kinds of style sheets that ATK or Diamond have. In particular, ODA's style mechanisms suffer from the following problems:

• The designation in ODA of an attribute as either a *layout* or *presentation* attribute, which controls the use of the attribute, seems arbitrary to us. This makes it awkward to make consistent decisions about the translation of document format features.

• Similarly, the distinction between logical and layout attributes appears to be inconsistent. This necessitates the awkward separation of some information between the logical and layout structures and makes translation from ODA, in particular, more difficult.

• The values of the attributes that may be grouped in a style are all absolute. This means that there is no unambiguous way to translate relative document presentation changes.

• Inheritance of style information in ODA is tied directly to the logical and layout structures of a document. It is not possible, therefore, to represent a style sheet hierarchy such as that allowed by Diamond.

Despite the shortcomings of the ODA style mechanism, we still feel that our choice of ODA as the intermediate representation was correct. The presence of a powerful, defined layout semantics for ODA continues to be a strong factor in our recommendation of ODA as an interchange format. The committee responsible for the definition of ODA is aware of many of the deficiencies in the style mechanism and we hope to influence future changes to the standard as a result of our work.

There remain many unanswered questions about our approach to document interchange among diverse systems. The overriding issue is whether it is possible to retain enough editing and presentation information for a particular set of document systems to make it feasible to collaborate on multi-media documents. This question can only be answered empirically by people using the translators.

A question that has plagued us for some time is that of the status of a document as it is repeatedly translated among document systems with differing capabilities. One scenario is that the document quickly reaches a fixed point or lowest common denominator. In this case, the document has been reduced to one with features with which all participating systems can cope. If this set of features is rich enough, the document has reached a happy medium. Of course, it is also possible that the document will be reduced to nothing as each translator throws away information that can not be used in its system. A somewhat more ominous scenario is that each translator would continue to attach redundant information to the document and the representation will become enormous or over-constrained. Once again, these questions can be answered only with use.

Further Work

Our work until the end of January 1989 is clear: finish the translators for the selected document features. At that point, there are a number of directions in which our work could proceed. We could continue to refine the translators so that they handle more document features or so that the subset of ODA accepted is enlarged. Another possibility is the translation of geometric graphics, which is supported in ATK, Diamond and ODA.

It is likely that we will make use of our experience with ODA to attempt to influence the future direction of ODA and the NBS DAP. We have learned a great deal about the use of these standards and we believe that we could contribute to their continued development and practical use.

Lastly, we are investigating the integration of our translation techniques with other systems. The multi-media mail and bulletin board systems of Andrew and Diamond are good candidates for these translators. The incorporation of automatic translation to and from ODA would allow the interchange of multi-media messages between diverse mail systems.

Acknowledgements

The EXPRES project is funded by the National Science Foundation under grant number ASC-8617695, and we would like to thank Al Thaler, the program manager, for his help in administering and directing this work.

In addition, several people at CMU have been instrumental in this project. Barry Silverman has provided administrative and public relations work, as well as general advice and counsel. Michael Zheng, a graduate student at Marquette University who has worked with us at CMU since July 1988, has written all of the ODIF support in the ODA Tool Kit. Lastly, Paul Crumley and Maria Wadlow aided us greatly during the initial design phase of the ODA Tool Kit.

References

[1] Adobe Systems Incorporated. *PostScript Language: Tutorial and Cookbook.* Addison-Wesley, 1985.

[2] Adobe Systems Incorporated. *PostScript Language: Reference Manual.* Addison-Wesley, 1985.

[3] Nathaniel Borenstein, Craig Everhart, Jonathan Rosenberg and Adam Stoller. A Multi-media Message System for Andrew. In *Proceedings of the USENIX Winter Conference*, pages 37-42. February, 1988.

[4] International Organization for Standardization. *(ISO 8613) Office Document Architecture (ODA) and Interchange Format* International Organization for Standardization (ISO), 1988.

[5] International Organization for Standardization. *(ISO 8879) Information processing -- Text and office systems -- Standard Generalized Markup Language (SGML)* International Organization for Standardization (ISO), 1986.

[6] James H. Morris, Mahadev Satyanarayanan, Michael H. Conner, John H. Howard, David S. H. Rosenthal and F. Donelson Smith. Andrew: A Distributed Personal Computing Environment. *Communications of the ACM* 29(3):184-201, March, 1986.

[7] *NBS Implementor's Agreements: ODA Document Application Profile* National Bureau of Standards, 1988.

[8] Andrew J. Palay, Wilfred J. Hansen, Mark Sherman, Maria G. Wadlow, Thomas P. Neuendorffer, Zalman Stern, Miles Bader and Thom Peters. The Andrew Toolkit - An Overview. In *Proceedings of the USENIX Winter Conference*, pages 9-21. February, 1988.

[9] Jonathan Rosenberg, Ann Marks, Mark Sherman, Paul Crumley and Maria Wadlow. *The CMU ODA Tool Kit: Site Installation Guide & Application Programmer's Interface.* Technical Report CMU-ITC-071, Information Technology Center, Carnegie Mellon, March, 1988.

[10] Mark S. Sherman. EXPRES Interchange Conventions for ODA. Carnegie Mellon University, Unpublished report.

[11] Robert H. Thomas, Harry C. Forsdick, Terrence R. Crowley, Richard W. Schaaf, Raymond S. Tomlinson, Virginia M. Travers. Diamond: A Multimedia Message System Built on a Distributed Architecture. *IEEE Computer* 18(12):65-78, December, 1985.

[12] Xerox Corporation. Interpress Electronic Printing Standard. In *Xerox System Integration Guide.* Xerox Corporation, Stamford, Connecticut, 1983.

Document Format Features

This appendix contains a list of document format features that we have included in this phase of the EXPRES project. This is not meant to provide a precise description of the features, but the names should be descriptive enough to indicate their intent.

The comment *(conventions only)* indicates that at the time we built the translators neither Diamond nor ATK could handle the associated features, but that our agreements cover their representation.

- Text margins: left, right, top, bottom
- Font changes: size, family, italics, bold, underlined, superscript, subscript
- Paragraph indentation, including outdenting
- Alignment: left, right, centered, justified/filled
- Tabs: left-aligned
- Inter-line spacing
- Inter-paragraph spacing
- Explicit page breaks
- Style sheets
- Page headers and footers *(conventions only)*
- Footnotes *(conventions only)*