An Algebra for Multidimensional Documents
as Abstraction Mechanism for Cross Media Publishing

Ulrike Lucke
University of Rostock, Institute of Computer Science, Chair for Computer Architecture
Albert-Einstein-Str. 21, 18059 Rostock, Germany
ul@informatik.uni-rostock.de

Abstract. The concept of multidimensionality allows to specify contrary variants of a content object on a common, abstract level. This prevents redundancy, simplifies the maintenance and increases the adaptability of content. This is crucial for automated cross media delivery. Though some implementations of multidimensional document systems are available, existing formalisms do not yet cover this concept. This article presents the multidimensional extension of a document algebra, as well as a reference implementation of this algebra in an authoring tool. The algebraic approach is a valuable basis for interoperability of formats and tools in order to realize the whole content workflow in cross media publishing.

1 Introduction

1.1 Concept and Benefits of Multidimensionality

The distribution of content to different channels requires powerful and reliable mechanisms to automatically adapt the content’s appearance as well as internal structure to the specific requirements of this channel. This is more than a simple layout problem, which can be solved sufficiently by using the XML and stylesheets technology. Cross media publishing of given content requires to select appropriate components, to arrange them in the expected structure, and to cover them with a useful layout. Nevertheless, there will be some similarities between different published variants of the same content.

The concept of multidimensionality takes advantage of these similarities. Here, every characteristic of content regarding its appearance in a specific scenario of use is interpreted as dimension. To give some examples, one may think of the amount of text to be presented, the ability to display dynamic media, or interactivity patterns to be applied – all these make up the context that a given content can be automatically adopted to. From this background, the term multidimensionality is used as a more precise understanding of context, here. Thus, multidimensional content is the abstract encapsulation of its inherent variants in order to provide a powerful common basis for automated extraction of the required subset. This prevents redundant description of repeating elements, and thus decreases the costs of content maintenance, and increases flexibility as well as re-usability of content, significantly.

To give an example, a very profitable scenario of use for this concept is eLearning. Multidimensional learning objects can be easily published as web based course, as printable manuscript or on mobile phones. Every distribution channel is associated with specific devices, each with varying capabilities. These can be automatically addressed, if the content is enriched with sufficient contextualization information. In the course of commercialization of the educational market the value added by multidimensionality (in terms of adaptivity or personalization [5]) can justify prices resulting from expensive multimedia production.

Since most multimedia formats are monolithic, the main anchor point for multidimensionality or contextualization, respectively, are objects with a more textual emphasis like traditional documents [19]. Nevertheless, the definition of varying behaviour in different dimensions does not stop at multimedia objects. As an example, the call of a Java applet can be parameterized in order to adjust to a specific level of detail, or interface type.

The process of production, management, transformation, and distribution of multidimensional content is rather complex. An abstract common layer can provide a valuable basis for interoperability of several formats and tools in this workflow. This article describes an algebraic formalism for multidimensional content in chapter 2. For implementation of this concept existing systems should be re-used as far as possible in order to keep down development costs. Still, the crucial point will be the author’s awareness of possible future application scenarios in order to model his content in a broader context. That’s why the focus of our work on multidimensionality was not only on publication mechanisms, but also on transparent authoring systems as described in chapter 3.
1.2 Related Work

There is a variety of content models available. Models for document structures mainly originate from the area of XML [17]. Formal approaches [3][4][6][14] represent the internal structure of a document as nested sequences of tuples (NST). They sufficiently model the capabilities of XML, but do not cover specification of multidimensional documents nor their processing.

Powerful formalisms for processing of data structures can be borrowed from data base theory. Several NST algebras provide an extensive set of operations on large and uniform XML-like structures [2][7]. But, they do not deal with single documents in potentially high degrees of freedom. Again, there is no support for multidimensionality.

Finally, there are some XML-based implementations of multidimensionality [12][15][18]. Unfortunately, they do not consider a formal basis and thus are not interoperable with other mechanisms and tools.

Putting it all together, the aim of our work was to combine formal document models and data base algebras to an algebra for structured documents [9]. The algebra was equipped with the concept of multidimensionality [10]. This article introduces the formalism as well as a reference implementation.

2 A Formal Model for Processing of Multidimensional Documents

2.1 Modelling of Document Structures

First of all, a short definition of data structures is needed here in order to specify contextualization of objects, and operations on objects. The following definitions are to a great extent familiar from existing HTML or XML models, especially from algebraic approaches [6].

A variety of media objects are atomic components of a document, primarily visible to the user. Examples are text, image, animation, video, or sound – depending on the document type and application scenario. All these types shall be summarized to the data type ATOMIC.

Definition 1: Atomic or Media Object
An object \( x = \text{name} : \text{value} \) of the data type ATOMIC is called media object. \( \text{name} \), followed by a colon, is an optional identifier of the object. \( \text{value} \) refers to or is the content of the object.

Media objects can be internally built from individual media, like a text consisting of characters, a pixel-based drawing, or a video with images and sound. Nevertheless, media objects shall be considered as atomic if they appear as inseparable to the user and are recognizable without knowledge on their internal structure and/or physical representation.

Fine-grained objects in a document are step by step composed to larger structures, building so called composite objects. This data type COMP is defined as follows:

Definition 2: Composite Object
A tuple \( x = \text{name} : (y_1, \ldots, y_n) \) of objects with \( y_1, \ldots, y_n \in (\text{ATOMIC} \cup \text{COMP}) \) \( (n \in \mathbb{N}^+) \) is called composite object of data type COMP. \( \text{name} \), followed by a colon, is an optional identifier of the object. The set \( \{y_1, \ldots, y_n\} \) of components of an object \( x \) is defined as \( \text{comp}(x) \).

Composite objects are nested tuples and can be called structure objects. They define the internal structure of a document, but are perceived by the user just in passing. As an example, they might have an effect on the order and appearance of their components. In the remainder, the entirety of media and composite objects shall be called just objects. The symbol for an empty object is \( \epsilon \), where \( (a, \epsilon, b) = (a, b) \) and \( (\epsilon) = \epsilon \).

The definition of identifiers for objects allows to clearly address and associate parts of documents without knowing their internal structure. Alternatively, position numbers can be used. In this way, the definition of paths and links leads to an overlay structure of documents in addition to the fundamental hierarchy of their components. Nevertheless, links as well as attributes of objects shall be omitted here, since they do not make a special contribution to the concept of multidimensionality.

Type, order, and incidence of all elements are defined by the chosen description language. The used grammar can be formalized and mapped to instances of documents, too. But, this is outside the focus of this article.

2.2 Formalization of Context

The document structure is now enriched with dimensional or context information, which is a novel feature of the algebra introduced here. So far, no formal descriptions for multidimensional documents are available. Thus, the concepts of existing implementations [12][18] have been formally modelled here. First, this implies to specify constraints for the validity of an object regarding certain aspects (or dimensions) of an environment.

Definition 3: Dimension Specifier
A dimension specifier is expressed as \( DS = D \circ W \). The dimension \( D \) identifies one aspect of the system to be described, and \( \circ \) is an operator from the set \( \{\pm, \times, \epsilon, \varepsilon\} \). \( W \) is an atomic value from the domain of \( D \), if \( \circ \in \{\pm, \times\} \), or a non-empty set of atomic values from the domain of \( D \), otherwise.
Now, these constraints can be grouped and assigned to an object and/or its components.

**Definition 4: Context Specifier**
A context specifier \( c \) is expressed as a set \( c = \{ DS_1, ..., DS_m \} \) of dimension specifiers \( DS_i = W_i, W_i^c, W_i^ε \), where all dimensions \( D_i \) are different.

**Definition 5: Contextualized Object**
The contextualization of an object \( x \) with the context specifier \( c \) is expressed as \( x[c] \).

The case \( n = 0 \) in Definition 4 is marked by \( ε \) and named empty context. The declaration \( [ε] \) in Definition 5 can be omitted if \( c = ε \). Missings dimensions in a context specifier express no restrictions in validity of the object regarding to this dimension.

The definitions given up to here shall be illustrated using a short example. Consider a tourist information system that provides some historic and cultural knowledge. Depending on the current access type, information will be presented in full length or shortened, and with only static or also with dynamic media elements. In the content, this is modelled as:

```plaintext
touristInfo : {  
    title : "Rome",  
    text : "The city of Rome is the capital of Italy and with more than 2.8 million inhabitants also its spiritual and cultural centre. Antique legends describe the foundation of Rome ... " [length=full],  
    list : {  
        text : "capital of Italy",  
        text : "2.8 mio inhabitants",  
        text : "aged more than 2500 years",  
        text : "residence of pope"  
    } [length = short],  
    image : rome.jpg [media=static],  
    video : rome.avi [media=dynamic]  
}
```

If the user request a printable guide, full texts and static images are published. Smart phones will receive short texts in a list accompanied by a video, elder mobile phones only the text, and so on. Further dimensions might consider the user’s current geographical position in order to provide navigation information, tips for restaurant and hotel, etc. In this way, just three dimensions with each a domain of three values lead to a total of \( 3^3 = 27 \) potential variants of a content object.

Mapping between a contextualized object and a given scenario of use is realized by specifying distinct values for all dimensions appearing in the context of this object. Alternatively, dimensions can be treated separately for generation of temporary internal formats.

**Definition 6: Environment**
An environment \( E = \{ D_1 = w_1, ..., D_n = w_n \} \) consists of a number of distinct dimensions \( D_i \) \((1 \leq i \leq n)\), to which atomic values \( w_i \) from the domain of \( D_i \) are assigned.

**Definition 7: Validity of an Object in an Environment**
The object \( x[c] \) with the context \( c = \{ DS_1, ..., DS_m \} \) is valid in the environment \( E = \{ D_1 = w_1, ..., D_n = w_n \} \) if \( ∀ DS_i \in c \) with \( DS_i = W_i, W_i^c, W_i^ε \) \((1 \leq i \leq m)\) and \( ∀ (D_j = w_j) \in E \ (1 \leq j \leq n) \) with \( D_j = D_i \) the expression \( (w_j, W_i^c) \) is true.

Applying this to the example given above, the environment \( E_{guide} = \{ length=full, media=static \} \) brings up the printable tourist guide, consisting of title, full text and image. The whole list and the video are skipped, as they are invalid in this environment. The title has no context; it will thus be valid and appear in all possible environments.

The context of an object affects all its (sub-)components, too, as demonstrated above with the list object. In order to prevent the user from description of contradictory contexts, which would cause non-deterministic algorithms for the manipulation of contextualized objects, this has to be verified by appropriate tools before further processing – in analogy to syntax checks. A formal definition of context validity shall be omitted here due to lack of space.

### 2.3 Operations on Multidimensional Documents

Besides contextualization, the second novelty of the algebra is the transfer of processing mechanisms from database algebras to document structures. The 14 operations of the document algebra are partly familiar from existing NST algebras [7]:

- 8 dyadic operations for document composition
  - (union, intersection, difference, cartesian product, join, concat, pair, and insert)
- 6 monadic operations for document transformation
  - (selection, rename, delete, remove duplicates, unnest, and nest)

A significant difference to database theory is that here, operations deal on single varied documents, instead of on sets of fine-grained uniform data. All operations were extended for manipulation not only of nested objects, but also of their attributes, links, and contexts. Obviously, not all 14 operations in full extent can be described here, but two representative ones are explained in the following:

**Select.** The selection is a very powerful operation, but comparatively easy to define and to implement. It returns all components of an object, for that a given predicate is true. In contrast to relational algebra, where different
operators for selection $\sigma$ and projection $\pi$ are defined, the predicate allows to select regarding both, names as well as content of objects, here. The selection is defined as follows:

**Definition 8: Selection from an Object**

The Selection $\sigma$ returns all components of an object $x = (x_1, \ldots, x_n)$ for that a given predicate $p$: $(\text{COMP} \cup \text{ATOMIC}) \rightarrow \text{BOOL}$ is true.

\[ \sigma x = (x'_1, \ldots, x'_n) \]

where \[ x'_i = \begin{cases} x_i & \text{if } p(x_i) = \text{true} \\ \epsilon & \text{otherwise} \end{cases} \]

The predicate may consider not only the name or content of objects, but also attributes, links or contexts. Especially for operations on context, a recursive definition of the select operator is desirable due to inheritance of context.

**Definition 9: Recursive Selection from an Object**

The recursive selection $\rho$ returns all components, sub-components and so on of an object $x = (x_1, \ldots, x_n)$ for that the predicate $p$: $(\text{COMP} \cup \text{ATOMIC}) \rightarrow \text{BOOL}$ is true.

\[ \rho x = (x'_1, \ldots, x'_n) \]

where \[ x'_i = \begin{cases} x_i \rho & \text{if } p(x_i) = \text{true} \\ \epsilon & \text{otherwise} \end{cases} \]

Considering again the example given in section 2.2, a recursive selection on the touristInfo object regarding the predicate $p(x) = \text{true} \Leftrightarrow x$ is valid in $E_{\text{guide}}$ will create the printable tourist guide.

Analogously, a literal search on a content object can be realized as a recursive selection. The predicate has to go down in the hierarchy and to return only composite objects (with a recursive call) and text elements. Neglecting markup and object borders, a full-text search can be described in this way.

In contrast to other approaches [2], the select operator is the only recursive one in the document algebra. The reason is that it is the only non-manipulating operator. Since the algebra aims at describing which tasks an author can perform, too much functionality in a single operation (with a lot of possible side effects in a complex structure) could tend to confuse the author rather than supporting him. Nevertheless, recursion is a proper instrument to efficiently manipulate large sets of uniform data (like in database algebras), or for powerful search mechanisms as described above.

**Union.** In contrast to simple selection, the manipulation of nested, ordered, and contextualized objects is rather complex. Imagine the tourist information system to provide content from different sources, that has to be united before distribution. There are two information sets available with different text length but same title and image for a given point of interest:

**touristInfoX:**

- title: "Colosseum",
- list: 
  - text: "opened 80 b.C.",
  - text: "capacity 73.000",
- image: colosseum.gif [media=static]

**touristInfoY:**

- title: "Colosseum",
- text: "This is the largest amphitheatre ever built.",
- image: colosseum.gif [media=static]

Combining these two objects requires not only to check the equivalence of their components, but also to process their contexts. Due to its inheritance, the original contexts of both father objects have to be merged with those of their components in order to keep all information. In addition, the contexts of equivalent components in both objects have to be united.

**Definition 10: Union of Objects**

The union $\cup$ of the two objects $x = (x_1, \ldots, x_n)(c_x)$ and $y = (y_1, \ldots, y_m)(c_y)$ returns all components of both objects except those components of $y$ that exist in $x$.

\[ x \cup y = (x'_1, \ldots, x'_n, y'_1, \ldots, y'_m)(c_x \cup c_y) \]

where \[ x'_i = \begin{cases} x_i \cup (c_x \cap c_y) & \text{if } i \in \text{comp}(y) \\ x_i \cap c_x & \text{otherwise} \end{cases} \]

and \[ y'_j = \begin{cases} y_j \cap c_y & \text{if } j \in \text{comp}(x) \\ \epsilon & \text{otherwise} \end{cases} \]

Likewise, the union of contexts is defined. Here, one has to consider that a missing dimension specifier expresses the validity of that object for all possible values in the domain of this dimension ($D \in \text{dom}(D)$). In this case, a constraint for this dimension in the other context specifier can be ignored.

**Definition 11: Union of Contexts**

The union $\cup$ of the two contexts $c_x = \{D_{x_1}^{\xi_{x_1}}W_{x_1}', \ldots, D_{x_n}^{\xi_{x_n}}W_{x_n}'\}$ and $c_y = \{D_{y_1}^{\xi_{y_1}}W_{y_1}', \ldots, D_{y_m}^{\xi_{y_m}}W_{y_m}'\}$ returns the
combined dimension specifiers for the dimensions appearing in both contexts.

\[ c_1 \cup c_2 = \{ (D_{x1} \circ x_1 W_{x1})', \ldots, (D_{xn} \circ x_n W_{xn})' \} \]

where \((D_{xi} \circ x_i W_{xi})' = \begin{cases} (D_{xi} \circ x_i W_{xi}) \lor (D_{yj} \circ y_j W_{yj}) & \text{if } \exists (D_{yj} \circ y_j W_{yj}) \in c_y \\ \epsilon & \text{with } D_{yj} = D_{xi} \\ \epsilon & \text{otherwise} \end{cases} \]

The interpretation of the Boolean expressions \((D_{xi} \circ x_i W_{xi}) \land (D_{yj} \circ y_j W_{yj})\) depends on the type of the operators \(\circ x_i\) and \(\circ y_j\), as well as the type and value of \(W_{xi}\) and \(W_{yj}\). Calculation is as usual and thus not stated here.

For the given example objects, the application of the union operation as defined above leads to the following result:

```plaintext
touristInfo: {
    title: "Colosseum",
    list: {
        text: "opened 80 b.C.",
        text: "capacity 73.000"
    }

    image: colosseum.gif [media=static]
    text: "This is the largest amphitheatre ever built." [length = full]

    [length = short, full]
}
```

Since the domain of the dimension length only consists of the values short and full, the dimension specifier length \(\epsilon\) short, full has been ignored in all sub-objects.

### 3 Reference Implementation

Today, native XML editors offer extensive possibilities for handling of semi-structured documents, schemas, and stylesheets. But they are only suitable for users from a technical background. Fortunately, the XML capabilities of standard word processors increased meanwhile [8]. As an example, Adobe FrameMaker [1] sufficiently supports the creation of structured documents. The software is powerful and robust. Especially, the cross-referencing mechanism is known as efficient and reliable. But, FrameMaker is also expensive and only extendable using a simple C-API. Anyway, because of the comprehensive functionality we have implemented a plug-in for full support of the data structures, multidimensionality, and operations of the document algebra presented in chapter 2.

Figure 1 depicts the extended FrameMaker (Version 7.0) with the sample multidimensional content object from section 2.2. The document view (1) shows all media objects in appropriate formatting, i.e. the primary content visible to the user. The structure view (2) adds information on the composite objects and thus the structure of the document. Both windows act synchronously, that means selections and modifications always affect both sides. The definition of document dimensions, contextualization of objects, check for context-validity, as well as generation of zero-dimensional variants are not possible in the standard version of FrameMaker. After our extension, context is visualized in the document and the structure view, and modifications on context are realized in a dialogue (3). Additional features of the algebra are grouped in a new menu entry. This includes, for instance, to define or import a document’s dimensions and their domains, or to show or hide the context visualizations.

The operations on documents offered by FrameMaker are comparatively powerful, e.g. nesting and unnesting of objects as fundamental operations in structured documents are available. But, several operations (like union, intersection, and difference) had to be added by our plug-in. They are executed using a context menu in the structure view. Figure 2 demonstrates this using the union example introduced in section 2.3.

Finally, context-based selection shall be demonstrated. The content object as depicted in Figure 1 includes several components for full and short output, as well as for static and dynamic variants. According to the examples in section 2.2 a printable version of the tourist guide shall be extracted. This requires to define an environment for the document. This is done in a dialogue window opening from the document algebra menu, which allows to select a certain value for each dimension from its defined domain. As a result all components that are not valid in the new environment are hidden in document and structure view,
Figure 1 A multidimensional document in the extended FrameMaker (German interface): 1 – media objects, 2 – structure objects, 3 – dimensions

Figure 2 Union of two content objects in the extended FrameMaker (German interface)
which is depicted in Figure 3. Hidden components are still part of the document and appear to the author if the environment is deactivated. Obviously, formatting of the resulting „tourist guide” does not yet meet professional design requirements. This is just a simplified preview during the authoring process. Additional layout is added by appropriate stylesheets during the publication process, afterwards.

The extended Adobe FrameMaker is a suitable reference implementation for the multidimensional document algebra introduced in chapter 2. Large-scale practical experiences in the field of eLearning [20] with multidimensional content for a complete study on computer engineering (produced by users coming from technical as well as non-technical background) have proven the validity of both, the multidimensional approach as well as the developed tool. Nevertheless, a second extension of an authoring tool based on the algebra (targeting the open source sector) is intended.

4 Conclusion

The article introduced a formal model for contextualization of content or multidimensionality, respectively, which is based on a recently developed document algebra. The algebra was designed in order to meet the requirements of cross media publishing as well as pre-requisitions given by existing algebraic approaches from data base theory.

The data model of the algebra consists of media and composite objects in a hierarchical structure. Objects can be enriched with manifold validity constraints, that make up their context in an multidimensional environment. 14 operations for manipulation of objects have been stated. Selection and recursive selection from a document, as well as union of two document parts have been explained in more detail, including helper operations for processing of contexts. The developed algebra does not depend on specific description languages or standards, nor does it compete with them. Rather, the algebra is intended to provide a general reference model for multidimensional content on a higher level of abstraction in order to ensure interoperability of different formats and/or tools.

Formal specification of data structures and algorithms can be directly implemented in tools for production, management, transformation and use of multidimensional content. Especially, high-quality authoring is a crucial point in the content life-cycle. Tools have to be powerful, but also transparent to the user, and should integrate into the common working environment, seamlessly. That’s why we selected the desktop publishing program FrameMaker to be extended by the features of the multidimensional document algebra. Hiding the inherent complexity of multidimensional documents reduces the effort involved in building up such content and helps to not outweigh the general usefulness of the concept. Thus, a consistent approach for the development of multidimensional content from theory to practice has been presented in this article.

Additional effort was made in order to develop a suitable infrastructures for delivery of multidimensional content. In the WWR project [20], a web service based approach for publication of multidimensional learning objects has been followed, successfully [11][13][16].

Current work is extending the document algebra to cover not only the structure and processing of multidimensional content, but also its management and deployment. Thus, the full life-cycle of digital content is addressed. The formal basis of the algebra allows to implement all tools in a consistent workflow, and also to model their integration into a modular, distributed architecture. Here, the formal model helps to determine, which components of a content engineering system are rather closely related and therefore should be arranged close together, e. g. in a single service. Thus, a straighter design is expected, compared to a more intuitive implementation. An increased performance and inter-operability could be the benefit.

The overall goal of multidimensionality is to provide a basis for automated adaption of content for distribution on different channels. Here, the flexibility of content provided by its full coverage of a multidimensional application space is of higher value compared to conventional, even selective classification of content with metadata. Thus, the concept of multidimensional content can be seen as an evolution of traditional component-based approaches, leading to easier maintenance of data due to less redundancy. This will further increase the flexibility and efficiency of cross media publishing.
References


