Concealed structures in data

Revealing patterns in digital documents

Jakob Voß

Verbundzentrale des GBV (VZG), Göttingen, Germany,
`jakob.voss@gbv.de`

Abstract. This paper introduces to a research project to analyze how digital documents are structured and described in general. Using a phenomenological approach, it reveals common paradigms and patterns that are used in data, independent from particular technologies in which the data is available. A better understanding of data patterns will not only help to better capture singular characteristics of data by metadata, but it will also recover intended structures of digital objects.[1]

We do not, it seems, have a very clear and
commonly agreed upon set of notions about data.
— George Mealy (1967): Another look at data [18]

1 Introduction

Both documents and descriptions of documents increasingly exist as digital objects, basically as streams of bits, abstracted from any storage medium and location. While traditional concepts such as 'document', 'page', 'edition', and 'copy' blur, forms such as 'files', 'records', and 'objects', are rather different views on the same thing, than inherent properties of a digital document. These inherent properties depend on context both for documents [4] and metadata [7], but you still need to actually look at data at some level of description, especially if context and formats an not fully known. A deeper look at data is required, to reveal how digital documents are actually structured and described. The question should not be answered by simply pointing to concrete technologies and formats, which are subject to rapid change and obsolescence, but at a more fundamental level. The main hypothesis is that all methods to structure and describe data share common patterns, independent from technology and level of description.

[1] A first version of this paper is made available at arXiv (`http://arxiv.org/abs/1105.5832v1`) and another version appeared in the *Proceedings of the International Conference on Theory and Practice of Digital Libraries 2011*, p. 527-530. See `http://aboutdata.org` for more material.

2 Background and related works

The concept of data is used in many disciplines with various meanings. It can range from the product of objective, reproducible measurements ('data as hard numbers') to the product of any recorded observations ('data as observations'), and processable encodings of information and knowledge ('data as bits') [2]. Current 'data science' mainly uses the first two connotations and deals with aggregating, filtering, and visualizing data based on statistical methods. The main concern of data science evangelists seems to be "big data", that is "when the size of the data itself becomes part of the problem" [17].

The problem, that I want to tackle, does not depend on the size of the data or on basic problems of preservation [20], but on the inherent complexity of data, independent from its specific form. I commit to the third concept of 'data as bits' which subsumes any kinds of digital documents. This concept is also found in computer science and in library and information science, but both disciplines tend to use information as core concept instead of data.

The best examination of data by libraries so far is found first in long-term preservation of digital material, and second in metadata research and practice. In long-term preservation you must either emulate the environment of digital objects or you must regularly migrate them to new formats. Both strategies require good metadata, which themselves become subject of preservation, so documents may get buried in nested layers of metadata. Metadata research provides at least some guidelines for interoperability by metadata registries, application profiles, and crosswalks. However, there are numerous ways to describe the same object by data, and the same data can describe different things. In practice manual work is needed because context and function are not fully known or creators of data just do not comply to assumed standards – there is no silver bullet neither in data description. Just pointing at new standards such as RDF does not help, first because libraries must deal with digital documents as they exist 'in the wild' and second because all meaningful data is a simplified, context-dependent image of reality. Good criticism of particular data encoding languages has been given by Kent [13,14,15] and several failures of attempts to create a 'perfect language' are illustrated by Eco [10].

The novel approach of this research is to use *patterns* to show how data is structured. Patterns as structured methods of describing good design practice, were first introduced by Alexander et al. in the field of architecture [1]. In their words "each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem." Patterns were later adopted in other fields of engineering, especially in (object-oriented) software design [3,11]. There are some works that describe pattern for specific data structuring or data modeling languages, among them Linked Data in RDF [9], Markup languages [8], data models in enterprises [12,21]. A general limitation of existing approaches is the focus to one specific formalization method. This practical limitation blocks the view to more general data patterns, independent from a particular encoding, and it conceals blind spots and weaknesses of a chosen formalism.

3 Method

A preliminary analysis of different structuring methods shows that each data language highlights some structuring features. These features are then overused and even conceal intended structures of data. For instance, the nesting and order of elements in an XML document can be chosen with intent, but they can also be chosen in an arbitrary manner because an XML document must always be an ordered tree. For this reason we cannot rely only on official descriptions and specifications to reveal patterns in data. Most existing approaches to analyze data structuring are either normative (theoretical descriptions how data should be), or empirical but limited. Existing empirical approaches only view data at one level of description, in order to have a base for statistical methods (data mining) and other automatic methods (machine learning). In contrast, I use a *phenomenological* research method that includes all aspects of data structuring and description. Beside technical standards that specify data, you must consider software that shapes data, and typical examples how data is actually used by people. The phenomenological method views data as social artifacts, that cannot be described from an absolute, objective point of view. Instead data are studied as "'phenomena': appearances of things, or things as they appear in our experience" [22]. The analysis begins with a detailed review of methods and systems for structuring and describing data, from simple character encodings to data languages and even graphical notations. The focus is on conceptual properties. Details of implementation, such as performance and security, are only mentioned where they show how and why specific techniques have evolved.

4 Findings

The first outcome of this work is a broad *typology* of existing methods to structure and describe data. These methods are normally described as data codes, systems, languages, or models without consistent terminology jointly among technologies. Table 1 lists six general prototypes of methods together with their primary but not exclusive purpose and some typical examples. It should be noted that actual methods rarely fit exactely to one category but prototypes should better be viewed as dimensions or cognitive reference points as described by Rosch [19].

method prototype	**main purpose**	**examples**
encodings	express data	Unicode, Base64
storage systems	store data	NTFS, RDBMS
identifier and query languages	refer to data	URI, XPath
structuring and markup languages	structure data	XML, CSV, RDF
schema languages	constrain data	BNF, XSD
conceptual models	describe data	Mind Maps, ERM

Table 1. Prototype categorization of data structuring methods

The main empirical part of the analysis consists of a detailed description and placement of the most relevant instances and subgroups from the typology above. Each structuring method has its strengths and limitations, and each method shapes digital objects independent from the object's characteristic properties. It is shown how popular instances from the typology above each highlight a specific set of patterns and make other patterns less visible or more difficult to apply. Instances of data further show that in practice, patterns and levels of abstraction often overlap and that methods of structuring are often used against their original purpose. Typical examples include; the creation of dummy values for non-existing mandatory elements and the use of separators to add lists to non-repeatable fields. It appears that in practice it is often difficult to judge which properties of data are intended and which arise as artifacts from the constraints of a given modeling language. The example of XML was already mentioned above: XML structures data in form of an ordered tree, but many instances of XML documents use this feature only to apply other patterns but hierarchy and strict ordering. Figure 1 shows an example from a *catalog of data patterns* as result of this work.

A third result, more general than the catalog of patterns, is a list of five *paradigms* that describe kinds of viewing at and dealing with data in general and digital documents in particular. Unlike scientific paradigms [16], the paradigms identified in this work are less complete and disruptive. You can identify some paradigm shifts in the history of data structuring, [6,5,11, to give some examples], but old methods are rarely fully replaced by new ones. The paradigms are:

documents and objects are two rivaling approaches to realize digital documents either as given or as created. Both provide useful methods to describe digital artifacts as a whole, but it is often not clear whether a particular datum is actually used as value or as object. Once the distinction is fixed in a data description language, it is difficult two switch the point of view.

standards and rules specify how document should be structured and which parts are relevant or redundant. However, standards can only be as good as their conformance can be tested.

collections and types both group parts of digital documents. By this they are inevitable to reduce the number of objects to handle and they allow for identification of objects. On a closer look all grouping is artificial as there are only individual, distinct data objects.

entities and connections seem to be basic building blocks of all data. They help to separate independent, primary elements and dependent, secondary elements. But this separation often conceals that there is no final separation between entities and connections, as both can be transformed into the other.

levels of abstraction provide descriptions of the same document with different granularity and help to separate relevant and irrelevant parts. Too many levels, however, may more confuse then they simplify and levels often cannot clearly be separated.

name

sequence pattern

idea

strictly order multiple objects, one after another

context

a collection of multiple objects

motivation

sequences are a natural method to model one-dimensional phenomena, for instance sequences of events in time. As digital storage is structured as sequence of bits, sequences seem to be the natural form of data and counterexamples, such as formal diagrams and visual programming languages, are often not considered as data.

implementations

- If objects have a known size, they can be directly concatenated. If objects have same size, this results in the array pattern.
- The separator pattern can be used to separate each object from its successor object. To distinguish objects and separators, this implies the forbidden objects pattern. If separators may occur directly after each other, this may also imply the empty object pattern.
- You can link one object to its successor with an identifier. To avoid link structures that result in other patterns (tree, graph, . . .) additional constraints must apply.
- If objects have consecutive positions, a sequence is implied by their order.

examples

- string of ASCII characters (array)
- string of Unicode characters in UTF-8 (each character has known size)
- '`Kernighan and Ritchie`' (sequence with '` and `' as separator)
- $extract \to transform$, $transform \to load$, (sequence of linked steps)

counter examples

files in a file system, records in a database table, any unordered collection

problems

empty sequences and sequences of only one element are difficult to spot, like in other collection patterns.

similar patterns

without context, sequences are difficult to distinguish from other collection patterns. Many implementations of other patterns use sequences on a lower level.

implied patterns

position pattern

specialized patterns

array, ordered set, ring

Fig. 1. Example of a pattern description. Pattern names are underlined.

5 Evaluation and application

First results confirm that common paradigms and patterns (for instance identifiers, repeatability, grouping, sequences and ordering) are used on all levels of data description in different variants, implicitely and explicitly. Some patterns are already recognized, but it lacks a more systematic view, independent from the constraints of particular technologies. An examination of pattern that are actually applied to data shows that many description methods result in other structures than originally intended. It is unlikely that one single technology like XML or RDF will provide the final metadata tool. For this reason it is important to identify and apply patterns for both the creation of data and its consumption, especially in digital libraries.

As the selection of patterns is a creative act of design, their recognition will to some degree free data designers from apologies and unquestioned habits that are justified as enforced by natural needs or technical requirements. Knowledge of general paradigms and data patterns can help to reveal concealed structures in digital documents. Such retrospective analysis of incompletely defined or unknown data could be named "data archeology" and be located in the humanities, as it involves study of the cultural context of data creation and usage. Data patterns provide a contribution to intellectual data analysis. This analysis is needed to underpin and interpret algorithmic data analysis, which cannot reveal meaning of data as part of social practice. A general understanding of data and data patterns, as provided with this research, can therefore help libraries at least as much as the current understanding of physical publication types and materials.

References

1. Alexander, C., Ishikawa, S., Silverstein, M.: A Pattern Language: Towns, Buildings, Construction. Oxford University Press (1977)
2. Ballsun-Stanton, B.: Asking about data: Experimental philosophy of information technology. In: 5th International Conference on Computer Sciences and Convergence Information Technology. pp. 119–124 (2010)
3. Beck, K., Cunningham, W.: Using pattern languages for object oriented programs. In: Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA) (1987)
4. Buckland, M.: What is a "digital document"? Document Numérique 2(2), 221–230 (1998)
5. Chen, P.P.: The Entity-Relationship Model – Toward a Unified View of Data. ACM Transactions on Database Systems 1(1), 9–36 (1976)
6. Codd, E.F.: A Relational Model of Data for Large Shared Data Banks. CACM 13(6), 377–387 (June 1970)
7. Coyle, K.: Understanding the Semantic Web: Bibliographic Data and Metadata. Library Technology Reports 46(1) (2010)
8. Dattolo, A., Iorio, A.D., Duca, S., Feliziani, A.A., Vitali, F.: Structural Patterns for Descriptive Documents. In: Baresi, L., Fraternali, P., Houben, G.J. (eds.) ICWE. Lecture Notes in Computer Science, vol. 4607, pp. 421–426. Springer (2007)

9. Dodds, L., Davis, I.: Linked Data Patterns (2010), http://patterns.dataincubator.org/book/
10. Eco, U.: The search for the perfect language. Blackwell (1995)
11. Gamma, E., Helm, R., Johnson, R., Vlissides, J.M.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, 1 edn. (1994)
12. Hay, D.C.: Data model patterns: a metadata map. Morgan Kaufmann (2006)
13. Kent, W.: Data and Reality. Basic assumptions in data processing reconsidered. North-Holland (1978)
14. Kent, W.: The many forms of a single fact. In: Proceeedings of the IEEE COMPCON (1988)
15. Kent, W.: The unsolvable identity problem. In: Extreme Markup Languages (2003)
16. Kuhn, T.S.: Second thoughts on paradigms. In: Suppe, F. (ed.) The Structure of Scientific Theories, pp. 459–82. University of Illinois Press (1974)
17. Loukides, M.: What is data science? O'Reilly radar (June 2010), http://radar.oreilly.com/2010/06/what-is-data-science.html
18. Mealy, G.H.: Another look at data. In: Proceedings of the 1967, fall joint computer conference. pp. 525–534. ACM (1967)
19. Rosch, E.: Prototype classification and logical classification: The two systems, pp. 73–86. Erlbaum (1983)
20. Rosenthal, D.S.H.: Bit preservation: A solved problem? The International Journal of Digital Curation 5(1), 134–148 (2010)
21. Silverston, L., Agnew, P.: The Data Model Resource Book, Vol. 3: Universal Patterns for Data Modeling. Wiley (2009)
22. Smith, D.W.: Phenomenology. In: Zalta, E.N. (ed.) The Stanford Encyclopedia of Philosophy. Summer 2009 edn. (June 2009), http://plato.stanford.edu/archives/sum2009/entries/phenomenology/