New automatic interpreter for complex UDC numbers

Attila Piros
Software Developer, Debrecen, Hungary

Abstract: The paper describes a new solution for parsing, i.e. interpreting, complex UDC numbers which is based on a set of algorithms and does not require access to the UDC Master Reference File. The novel approach of this application is that it stores components of the analysed notation in an intermediate format (in this case, XML) by automatic means with no loss of data or information. The output file, due to its richness can be converted into different formats, such as standard markup and data exchange formats or simple lists of the recommended entry points of the UDC number. The program can also be utilized to create authority records containing complex UDC numbers which can be comprehensively analyzed in order to be retrieved effectively. The program, as well as the corresponding schema definition it employs, is under continuous development. The current version of the interpreter software is now available online for testing purposes on the following web site: http://piros.udc-interpreter.hu. The future plan is to implement conversion methods for standard formats and to create standard online interfaces in order to make it possible to utilize the features of the software as a service. This would result in the algorithm being able to be employed both in existing and future library systems to analyze UDC numbers without any significant programming effort.

1. Introduction

The synthetic principle of Universal Decimal Classification (UDC) is an advantageous feature as it allows the combination of concepts in the process of indexing and thus provides a great power in describing complex subjects in order to support information retrieval. However, in order to take advantage of this kind of indexing - an information retrieval system has to support searching of the individual semantic components within a complex classification notation.

In addition to this, the pre-combination of concepts may create a new concept, which not only inherits the semantics of its parts but acquires its own, new meaning. The same UDC number when used in different relationships may represent different subjects.

This leads to two important requirements with respect to the management of complex UDC numbers within an information retrieval system:

a) the integrity of the semantic components of the UDC data elements have to be preserved, i.e. it has to be possible to access and search for each individual component of a complex UDC number and to trace its original verbal description in the process of information retrieval;

b) complex UDC numbers have to be presented and treated as a whole together with the verbal expression containing additional meaning that the number as a whole represents and which cannot be implicated from the individual parts.

In the past the following solutions were devised and implemented:

1. Dealing with complex notation "as given"

a) simpler approach - complex UDC notations were entered in the bibliographic databases as simple text but in the process of information retrieval a parsing program was utilized to split UDC into individual components to allow for notation searching;

b) advanced approach - parsing programs were linked to the UDC Master Reference File to have access to verbal representation of the UDC notation and to allow for searching using words.

2. Representing and storing a complex notation as structured and coded data;

a) simpler solution - components of the complex notations were coded and entered in the specific field of the bibliographic database in a way that allows the notation access and searching;

b) advanced solution - authority control, i.e. complex notations, are stored in classification authorities which would give provision for each individual notation from the complex notation statement to be identified, accessed and linked to its verbal representation.

Without authority control, handling complex notations is difficult. Tasks such as the comparison of different numbers or matching complex numbers to each other automatically taking the relationships into account are hard to implement. This can also restrict subject browsing, which is one of the most important reasons for the use of classification in an information retrieval system (Slavic, 2006). Displaying either the artificial or the natural language terms in different forms may also be problematic without using an authority file just as is joining the concepts to other knowledge organization systems.

An authority record may contain not only the classification number and its metadata, but also additional semantic information. This may include broader, narrower and related classes, mappings to thesauri, subject heading lists or other classifications and access points using either the codes of the classification or keywords in natural languages (Slavic, Cordeiro & Riesthuis, 2007).

Aiding the construction of authority records in different formats or converting them between formats automatically can also be useful in reducing the effects of the above problem.

However, creation and maintenance of an authority control is very costly time and effort consuming and requires expertise that many libraries do not have. As a consequence, the practice of authority control in managing UDC numbers can only be found in well managed national and university libraries or library networks. The majority of bibliographic data that can be found in library networks is not maintained in authorities and encouraging the authorities to be created requires some kind of robust automated solution.

In this article we introduce an approach that may provide a solution not only for processing, parsing and interpretation of big sets of legacy UDC data that exist in the bibliographic domain but also offer a way of representing UDC data in a manner that enables the creation and population of authority records. If understood in its simplest way this approach allows for the interpretation of complex UDC notations and for storing the result of this analysis ready for further processing and use, be it linking UDC to words, creating indexes for searching or populating authority records.

The solutions proposed in this paper are outcome of a doctoral research conducted at the University of Debrecen (Hungary).

2. Proposal for the more powerful representation of UDC notation

In the course of this research into improving the ways of storing UDC notations in order to enhance precision during information retrieval, it became clear that almost every UDC number can be represented with the same hierarchical structure. This is determined by the conceptual precedence of the operators and the definitions of common and special auxiliaries (the exceptions are mainly caused by breaking the rules concerning combining numbers) which describes their syntactic structure

keeping all relevant information regarding its parts, the way they join to each other, their role in the compound, or perhaps their order if it is relevant.

Hierarchical and other relationships can be extracted automatically from this type of representation making it useful in order to determine (and develop) methods to aid authority control, as was explained in the previous section.

A great advantage to the representation mentioned above is its independence from citation order. If two numbers differs only in the citation order of their parts, they have the same logical structure, which will result in the same representation. However, order can be added as an attribute of the parts of the numbers in order to make a difference if the citation order has a semantic significance[1].

2.1. The syntactic structure of complex UDC numbers

The next step in this research was to examine the UDC notation syntax and connecting symbols used to form complex notational expressions. First to acknowledge is that the precedence order of the operations is as follows:

+ Coordination
: Simple relation
:: Order-fixing
[ ] Subgrouping
` Synthesis (within special auxiliaries)

The operators above can join numbers or ranges of numbers in the schedules. The top three can also join subgroupings inside which the same hierarchy operates (although it cannot contain nested subgroupings). In addition to the connecting symbols above, there is also synthesis within special auxiliaries which make it possible to join numbers of a given part of a schedule (such as numbers for chemical compounds or political parties). It is also possible to use operators inside common auxiliaries; the rules depend on the type of the auxiliary.

The concepts joined by operators can contain zero or one main table number (or a range of main table numbers) with zero or more (or a range of) special auxiliaries joined to this and zero or more common auxiliaries (which can also contain numbers, ranges, operands and special auxiliaries).

The special auxiliaries in UDC contain very different solutions in order to describe the most specific subject facets (such as properties, processes, operations, techniques, instruments) which have to be handled in different ways[2].

2.2. Describing the structure of the UDC numbers using XML

A series of UDC numbers (simple and complex) can be represented and possibly stored in a database as a tree. As with every tree, this structure can be described with XML which has the following advantages:

- XML is a standard, widely used format to represent trees with improved and documented support in every considerable programming language;

1 The question of citation order in UDC is discussed in more detail by Robinson (2003).

2 A detailed introduction and overview of facets presented in UDC special auxiliaries are discussed by Gnoli (2011)

- it is possible to define a schema definition to determine the exact format to describe objects;
- using an intermediate format makes conversion possible to other formats;
- the schema definition can also be utilized to validate UDC numbers in order to find syntactical errors in them.

This paper will discuss feasible ways of building a UDC-specific XML schema for describing the most detailed and complex UDC numbers (containing not only the common auxiliary signs and numbers, but also the different types of special auxiliaries).

The XML file generated based on the schema definition can be used to describe every possible form of the UDC numbers which observe the rules of UDC, and due to its comprehensiveness can be converted into different formats automatically[3].

2.2.1. XML Schema Definition (XSD)

The intermediate XML uses its own namespace instead of standard namespaces such as Dublin Core, SKOS or UDC's own Master Reference File (UDC MRF) format. Its elements should be replaced with standard types while converting it to standard formats.

The complex types of the XSD describe the possible elements of UDC numbers: the numbers of the schedules, the auxiliary signs and the UDC number itself.

For example, the following complex type describes the common auxiliaries of human ancestry, ethnic groups and nationality:

```
<xsd:complexType name="common_auxiliary_of_ethnics_number">
<xsd:complexContent>
<xsd:restriction base="udc:common_auxiliary_number_abstract">
<xsd:sequence>
<xsd:element name="special_auxiliary" type="udc:special_auxiliary" minOccurs="0" maxOccurs="unbounded"/>
<xsd:element name="nonudc_notation" type="udc:nonudc_notation" minOccurs="0"/>
<xsd:element name="alphabetical_specification" type="udc:alphabetical_specification" minOccurs="0"/>
</xsd:sequence>
<xsd:attribute name="number1" type="udc:common_auxiliary_of_ethnics_number_string" use="required"/>
<xsd:attribute name="number2" type="udc:common_auxiliary_of_ethnics_number_string" use="optional"/>
</xsd:restriction>
</xsd:complexContent>
</xsd:complexType>
```

An operator can be described also by a complex type containing its possible operands.

The simple types of the XSD were introduced for validation purposes; the constraints defined for the validation contain regular expressions to describe the possible formats of the numbers in the different schedules.

Each of the UDC concepts is described by a complex element. In accordance with the schema, the XML must contain at least one element to describe a UDC concept. This element contains the other elements as can be seen in the hierarchy above. It also contains the notation of the number as an attribute and the descriptions of the concept in different languages as elements too.

3 The XML schema for UDC number descriptions is available online under Creative Commons license at: http://piros.udc-interpreter.hu#xsd.

3. The automatic interpretation of complex UDC numbers

Converting UDC numbers manually to a complex format such as the one mentioned above is an unrealistic expectation; supporting building these representations, as far as possible automatically, is a well-founded requirement. An additional advantage of this approach is that the existing records could also be processed and converted.

Processing the UDC numbers directly has the following advantages:

- Although the databases and the formats used in information retrieval systems vary, the UDC number itself is a stable point in all of them.
- In this way the system would not require the indexers and the maintainers of the authority files to utilize special, system related knowledge, consequently they can concentrate on building the numbers taking maximum advantage of their skills and competencies related to the classification,
- Future research can be conducted based on the processing of UDC numbers so as to enhance the efficiency of searching and browsing for example by discovering the connections between the complex concepts.

The parsing of complex UDC numbers is a topic which has been subject to thorough research and investigation since the 1960s.

The most comprehensive research of this issue was conducted by Gerhard Riesthuis (1998). He explained all questions regarding parsing complex UDC numbers, in order to create algorithms to recognize their parts and make them retrievable by using even the definitions of these parts in natural languages (using the descriptions contained by UDC MRF). He suggested and explained a set of groups of algorithms in order to parse the numbers in more steps and also provided sample programs.[4]

The combination of the algorithms published by Riesthuis can be extended in order to generate the XML representations of UDC numbers as well. In the course of this research, however, a new algorithm was created which is better suited for the XML schema discussed earlier in Section 1.

3.1. The underlying principles of the interpreter

The proposed algorithm for UDC notation interpretation has to follow the syntax rules for UDC subject synthesis, as far as possible, with no loss of information. This specifically means the following:

- The algorithm must recognize those numbers and only those numbers which keep to the rules for synthesizing UDC numbers even if they appear very rare in the common practice of indexing or in a particular collection.
- The algorithm must retain all of the information stored by the number. Only recognizing some parts of the number is unsatisfactory, as is recognizing all of its parts but losing the information pertaining to their context and role.
- The parsing method must be fully syntactic as far as is possible: processing the numbers must be based on the rules for the composition of concepts and can employ information regarding the meaning of the numbers only if it is unavoidable.
- The process must be fully automated: the program must be able to analyze numbers without requiring user interaction during the parsing process.

The basic requirement for this task is access to the UDC schedules and online availability is an important

4 The summary of the methods mentioned above was also reported in *Extensions and Corrections to the UDC* (Riesthuis, 1997; 1999).

consideration[5]. The decomposition of UDC notations for the purpose of information retrieval requires a decomposition program, UDC MRF data in a format compatible with this program and a middleware between the above and the information retrieval system (Slavic-Overfield, 2005). Any development in this area will require some kind of Web services to access and represent classification (Slavic, 2006).

With the evolution of the Internet and related infrastructure, the Software-as-a-Service (SaaS) model has been on the increase significantly. Using the SaaS delivery model would allow UDC numbers to be analysed by a remote service and only a thin layer of middleware to join its output to the client software would be required. In this way the results would be able to be utilized for the purposes of managing authority files as well as by systems employing other methods to access UDC numbers.

In order to exploit this possibility to the full, one should implement different interfaces to access the output of the algorithm in as many different formats as are necessary.

3.2. The basics of how the interpreter works

The rules governing the usage of UDC determine a formal language over an alphabet of decimal digits, dots and the characters of UDC metalanguage symbols. The interpreter is an automata which recognizes this formal language.

The input words are UDC numbers and the states of the automata describe the type and role of that part of the number to which the letter being processed belongs. While processing the number, the automata generates a tree which contains the parts of a number, based on the rules of precedence regarding auxiliary signs and joining auxiliary numbers as explained in the previous sections. It also contains some additional information which may be necessary during a search.

If the automata does not recognize the input word, then the given number is invalid or cannot be interpreted in the given UDC version. This means that in principle the language recognized by the automata specified by a year is just the UDC edition for that year.

The inputs of the algorithm are the UDC number and the year of the UDC edition which was used to build it; the output will be the XML representation of the number, or an error message which describes the problem if it cannot be interpreted in the given version.

Using the year is enough for the purposes of syntactical analysis, in so far as it takes the year of introduction of the changes in the rules into account. However, to identify the parts of the numbers and then join them to the MRF using a database ID is a more precise solution (Slavic & Isaac, 2009). Future plans should include the possibility to define these data as well.

The proposition is to construct an algorithm which is able to parse UDC numbers in a syntactic way as far as possible. This would mean that parsing two different numbers having the same structure would result in similar representations, differing only in the numbers themselves. It would also require every UDC number to be parsed without the need for identifying the numbers, based on the punctuation signs and rules only.

However, there are special auxiliaries and various indexing rules which result in complex numbers in which information regarding the exact location of their parts in the schedules determine the structures of the numbers and the access points to them. For example, apostrophes can be used to synthesize numbers in some places in the schedules (e.g. in classes 329 or 547). Recognizing the access points in

5 The UDC editions used in this research are Hungarian UDC editions published in 1990 (Egyetemes Tizedes Osztályozás, 1990) and 2005 (Egyetemes Tizedes Osztályozás, 2005), the BSI UDC printed standard edition published in 2005 (UDC, 2005), English UDC Online (2013) and the UDC Summary (Multilingual Universal Decimal Classification Summary, 2011).

the parallel division instruction "subdivide as" require information about source and target notation (Riesthuis, 1999).

The interpreter is under continuous development; the algorithm has been improved depending on the current status of the research; standard interfaces and conversion methods have started to be worked out as well. A test version of the software is available on the following web page: http://piros.udc-interpreter.hu#process.

Under the 'parse/validate' menu item one can type in a UDC number, the year of publishing the UDC edition was used to build that and its descriptions in different languages. After having clicked on 'Process' button the interpreter will process the given number.

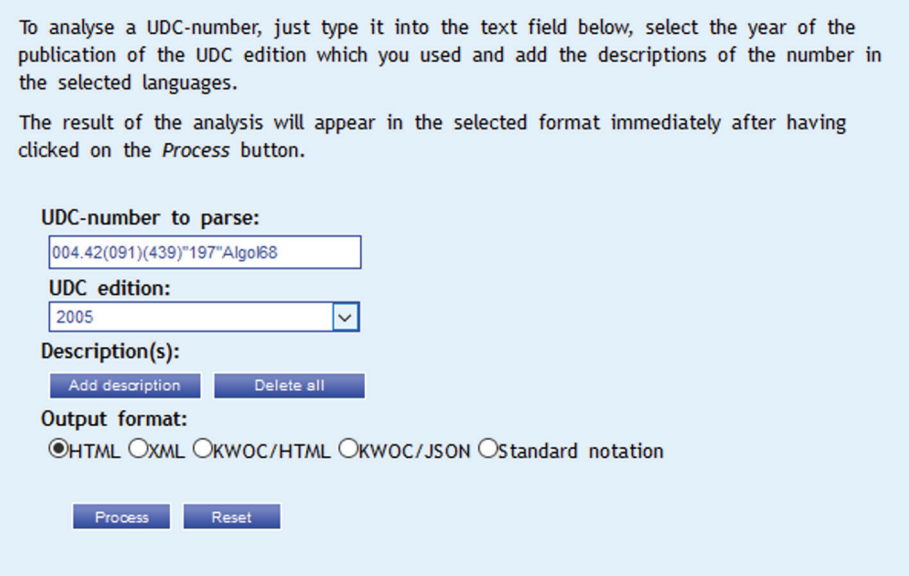
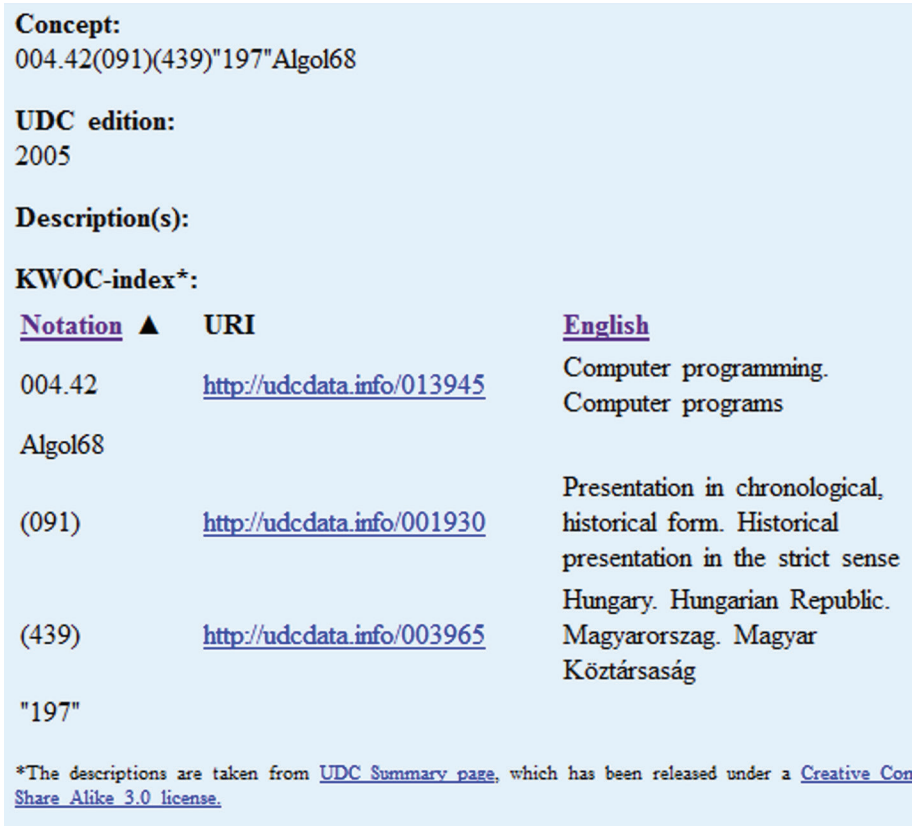


Figure 1: *The input form of the interpreter*

The result is available in XML format, however, depending on the output format selected, the software is also able to generate the HTML representation or a standardized form of the number, just as the KWOC index of its elements (in HTML or as a JSON string) from the generated XML.



Concept:
004.42(091)(439)"197"Algol68

UDC edition:
2005

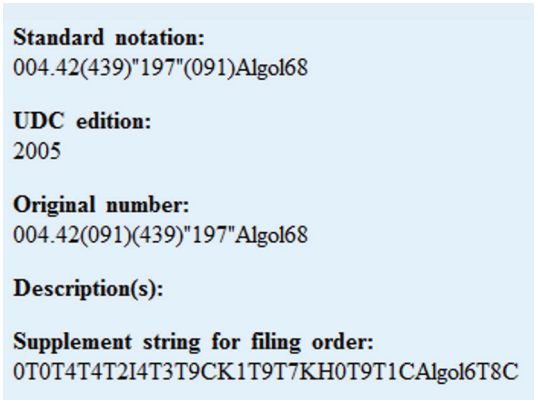
Description(s):

KWOC-index*:

Notation ▲	URI	English
004.42	http://udcdata.info/013945	Computer programming. Computer programs
Algol68		
(091)	http://udcdata.info/001930	Presentation in chronological, historical form. Historical presentation in the strict sense
(439)	http://udcdata.info/003965	Hungary. Hungarian Republic. Magyarorszag. Magyar Köztársaság
"197"		

*The descriptions are taken from UDC Summary page, which has been released under a Creative Con
Share Alike 3.0 license.

Figure 2: *KWOC index as output*



Standard notation:
004.42(439)"197"(091)Algol68

UDC edition:
2005

Original number:
004.42(091)(439)"197"Algol68

Description(s):

Supplement string for filing order:
0T0T4T4T2I4T3T9CK1T9T7KH0T9T1CAlgol6T8C

Figure 3: *The standardized form of a number*

Future plans also involve implementing further output formats and make the service available for software through standard REST endpoints.

3.3. Examples of parsing UDC numbers

The first example demonstrates the special case of describing literary works:

```
<ns:udc_concept
xmlns:ns="http://piros.udc-interpreter.hu/#xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
udc_edition="2005"
notation="821.111SHAK7ROM.03=112.2">
<ns:description xml:lang="EN">
Shakespeare: Romeo and Juliet (translated to German)
</ns:description>
<ns:main_table_number number1="821.111">
<ns:special_auxiliary xsi:type="ns:special_auxiliary_number_finaldigits"
number1="7"/>
<ns:special_auxiliary xsi:type="ns:special_auxiliary_pointnought_number"
number1=".03"/>
<ns:alphabetical_specification order="1" text="SHAK" standard=""/>
<ns:alphabetical_specification order="2" text="ROM" standard=""/>
<ns:common_auxiliary_independent
xsi:type="ns:common_auxiliary_of_language" order="1">
<ns:common_auxiliary_of_language_number number1="=112.2"/>
</ns:common_auxiliary_independent>
</ns:main_table_number>
</ns:udc_concept>
```

The XML above was generated by the software introduced in the previous section.

The concept contains the notation itself, its descriptions (in different languages) and the year of the UDC edition used to build it. Its elements are a main table number and the auxiliaries joining it.

The interpreter is able to recognize not only the alphabetical additions for the name of the author and the title of the work, but also the special auxiliaries for “individual work” and “translation”.

The following, more complex example[6] demonstrates the interpretation of extensions i.e. UDC number range (two numbers connected by /):

```
<ns:udc_concept
xmlns:ns="http://piros.udc-interpreter.hu/#xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
udc_edition="2005"
notation="323(47+57)1/.2:81(47+57)'01">
<ns:description xml:lang="EN">
National and ethnic minorities. Internal political activities. Origins (of languages). Soviet Union.
</ns:description>
<ns:main_table_relation>
<ns:main_table_number number1="323.1" number2="323.2" order="1">
<ns:common_auxiliary_independent xsi:type="ns:common_auxiliary_of_place" order="1">
<ns:common_auxiliary_of_geography_addition>
<ns:common_auxiliary_of_place_number number1="(47)" order="1"/>
```

6 The example is taken from Riesthuis (1997).

```
<ns:common_auxiliary_of_place_number number1="(57)" order="2"/>
</ns:common_auxiliary_of_geography_addition>
</ns:common_auxiliary_independent>
</ns:main_table_number>
<ns:main_table_number number1="81" order="2">
<ns:special_auxiliary xsi:type="ns:special_auxiliary_apostrophe_number" number1="'01" order="1"/>
<ns:common_auxiliary_independent xsi:type="ns:common_auxiliary_of_place" order="1">
<ns:common_auxiliary_of_geography_addition>
<ns:common_auxiliary_of_place_number number1="(47)" order="1"/>
<ns:common_auxiliary_of_place_number number1="(57)" order="2"/>
</ns:common_auxiliary_of_geography_addition>
</ns:common_auxiliary_independent>
</ns:main_table_number>
</ns:main_table_relation>
</ns:udc_concept>
```

The example contains a subject expressed by the number range: 323.1/.2 - which is represented by its ends. We can use the same method for the ranges either contained by the schedules or combined by the indexer; in this way a range can be retrieved by itself, or its elements. It is also possible to break it down into its elements using MRF[7]. The addition inside the geographical auxiliary must be kept in this case, because this combination is contained in the schedules and should be rendered retrievable by both the addition itself and its operands. This example also demonstrates that the interpreter is able to handle intercalation just like special auxiliaries following common auxiliaries.

The last example demonstrates the precedence order of the auxiliary signs:

```
<ns:udc_concept
xmlns:ns="http://piros.udc-interpreter.hu/#xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
udc_edition="2005"
notation="061.1(100)::[54+66]">
<ns:description xml:lang="EN">
IUPAC - International Union of Pure and Applied Chemistry
</ns:description>
<ns:main_table_orderfixing>
<ns:main_table_number number1="061.1" order="1">
<ns:common_auxiliary_independent xsi:type="ns:common_auxiliary_of_place" order="1">
<ns:common_auxiliary_of_place_number number1="(100)"/>
</ns:common_auxiliary_independent>
<ns:main_table_subgrouping order="2">
<ns:main_table_addition>
<ns:main_table_number number1="54" order="1"/>
<ns:main_table_number number1="66" order="2"/>
</ns:main_table_addition>
</ns:main_table_subgrouping>
</ns:main_table_orderfixing>
</ns:udc_concept>
```

7 The method of breaking down a range into its constituent numbers is discussed by Buxton (1992) and explained in detail by Riesthuis (1999).

3.4. Recognizing mistakes by the indexer

The analysis of UDC numbers makes it possible to recognize mistakes made during indexing as well. Figure 4 illustrates a few examples of recognized typographical mistakes.

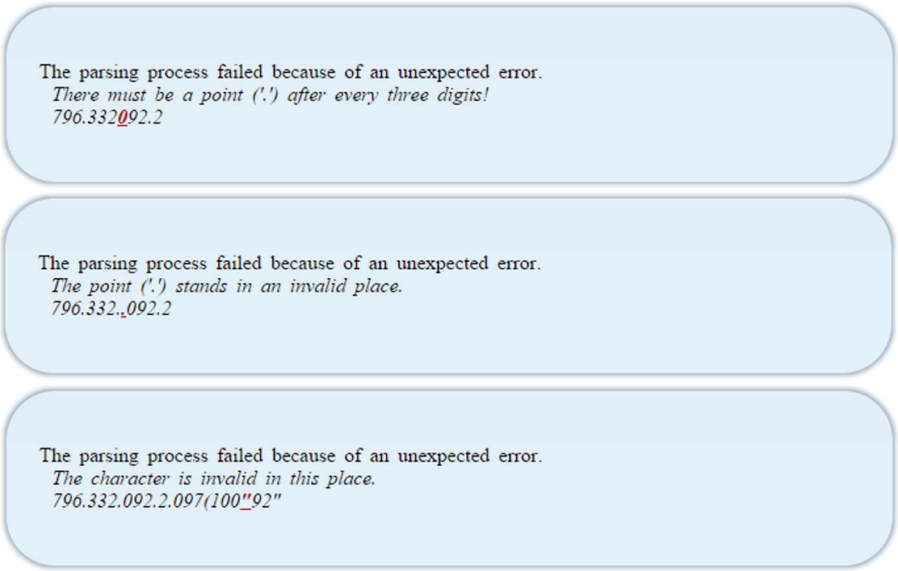


Figure 4: *Recognizing typographical mistakes*

Assigning an incorrect UDC edition may also result in an invalid number as illustrated in Figure 5.

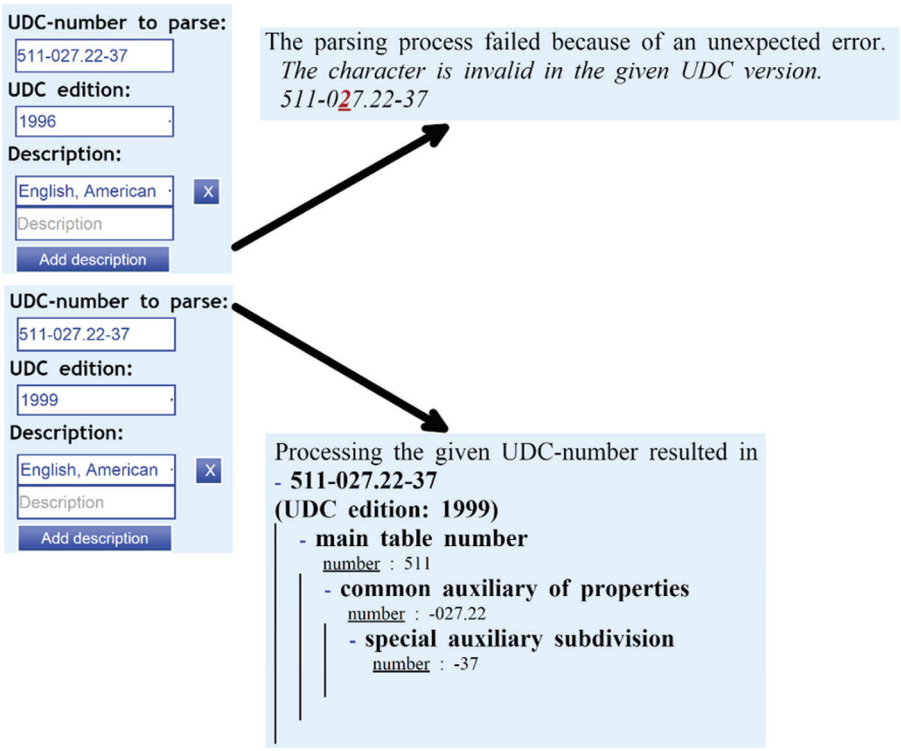


Figure 5: *Recognizing incorrect versions*

4. The conversion of the results

4.1. Conversion to MARC

MARC 21 Format for Authority Data (MARC 21 Format for Authority Data, 2000) defines field 080 for storing the UDC classification data and field 150 for topical term headings. For example:

```
150##$aHistory$vSchoolbook$y19th century$zBaltic states
080##$a94$x(474)$x"19"$x(075)$2BIP 0017-1 : 2005
```

The content of field 080 can be generated automatically from the XML representation of the number:

```
<ns:main_table_number number1="94">
<ns:common_auxiliary_independent xsi:type="ns:common_auxiliary_of_place">
<ns:common_auxiliary_of_place_number number1="(474)"/>
</ns:common_auxiliary_independent>
<ns:common_auxiliary_independent xsi:type="ns:common_auxiliary_of_time">
<ns:common_auxiliary_of_time_number number1=""19""/>
</ns:common_auxiliary_independent>
<ns:common_auxiliary_independent xsi:type="ns:common_auxiliary_of_form">
<ns:common_auxiliary_of_form_number number1="(075)"/>
</ns:common_auxiliary_independent>
</ns:main_table_number>
```

The determination of captions, hierarchical and see also relationships (in this case to the hidden element of the range) requires using MRF data and human interaction as well.

The generation of the MARC records in UNIMARC Classification Format (Concise UNIMARC Classification Format, 2000) can be similar to that of MARC 21.

4.2. Conversion to SKOS/XML

The following example is taken from the presentation by Slavic & Isaac (2009). The UDC number combination used in the SKOS example was 37:004 (“application of computers in education”). A SKOS/XML form of this number (using UDC Summary Linked Data) may be:

```
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:skos="http://www.w3.org/2004/02/skos/core#"
<skos:Concept rdf:about="37:004">
<skos:broader rdf:resource="http://udcdata.info/024974"/>
<skos:broader rdf:resource="http://udcdata.info/013566"/>
</skos:Concept>
</rdf:RDF>
```

Broader (or, optionally, semantic) relationships can be determined from the XML representation automatically. However, phase relationships which were also mentioned in the presentation cited above cannot be recognized automatically.

5. The test set

Online availability and the generation of XML output make it possible to compile an integrity test set, in order to ensure the software always produces the expected output. The easiest way to achieve the above is to create a small application, which is able to send HTTP requests to the application containing the UDC numbers and compare the responses with previously saved XML files. Test cases usually need to contain valid UDC numbers only in cases when not only the punctuation marks alone determine the result of the process (as was mentioned in section 2.2), otherwise they should abide by the rules regarding syntax.

Compiling a test set as mentioned above helps not only to maintain the integrity of the existing parts of the software during further implementation, but to review the rules of UDC regarding building complex numbers and facilitate a better comprehension of their meanings. The test set employed during the implementation is also available on the home page of the interpreter.

6. Conclusion

Handling complex concepts is a significant issue which arises during the automation of a faceted, analytico-synthetic classification system. Authority control provides a satisfactory solution to the issues and questions that have been raised. However, its effectiveness may be enhanced by employing automated methods to analyze numbers, discover their possible entry points, or represent them in different formats.

This paper provides explanations of the principles of representing complex UDC numbers in XML format and introduces a software interpreter made available online which is capable of interpreting UDC numbers in a given format. I hope that that the research described here will advance the application of UDC and make better use of its synthetic features.

Acknowledgment

I would like to express my appreciation to Dr. István Boda for his valuable and constructive suggestions during the planning and development of this research work and to extend my great thanks for the support provided by my wife and my daughter. I would also like to acknowledge the help of Dr. Aida Slavic in writing and publishing this paper.

References

Buxton, A. Computer Searching of UDC Numbers. *Encyclopedia of Library and Information Science*, 51, 14 (1992), pp. 132-151.

Concise UNIMARC Classification Format. Concise Edition, 2000. International Federation of Library Associations. Available at: http://www.ifla.org/archive/ubcim/p1996-1/concise.htm.

Egyetemes tizedes osztályozás. Rövidített kiadás. 1. kötet Táblázatok. FID Publ. No. 691. Budapest: OSZK-KMK, 1990.

Egyetemes tizedes osztályozás. 1. kötet Táblázatok 1-2. rész. UDC Publ. No. P057. Budapest: OSZK KI, 2005.

MARC 21 Format for authority data. Library of Congress, 1999. Available at: http://www.loc.gov/marc/authority/.

MARC 21 Format for classification data. Library of Congress, 2000. Available at: http://www.loc.gov/marc/classification/.

Multilingual Universal Decimal Classification Summary. The Hague: UDC Consortium, 2011. (UDCC Publication No. 088). Available at: http://www.udcc.org/udcsummary/php/index.php.

Riesthuis, G. J. A. Decomposition of complex UDC notations. *Extensions & Corrections to the UDC*, 19 (1997), pp. 13-19.

Riesthuis, G. J. A. Zoeken met woorden: hergebruik van onderwerpsontsluiting. [doctoral thesis] Amsterdam: Leerstoelgroep Boek-, Archief- en Informatiewetenschap, 1998.

Riesthuis, G. J. A. Searching with words: re-use of subject indexing. *Extensions & Corrections to the UDC*, 21 (1999), pp. 24-32.

Robinson, G. Citation order in UDC. *Extensions & Corrections to the UDC*, 25 (2003), pp. 19-27.

Slavic, A. Classification management and use in a networked environment: the case of the Universal Decimal Classification. Doctoral thesis. London: University of London, 2005. Available at: http://discovery.ucl.ac.uk/1334914/.

Slavic, A. Interface to classification some objectives and options. *Extensions & Corrections to the UDC*, 28 (2006), pp. 24-45. Available at: http://hdl.handle.net/10150/105459.

Slavic, A.; Cordeiro, M. I.; Riesthuis, G. Enhancement of UDC data for use and sharing in a networked environment. [Talk presented at The 31st Annual Conference of the German Classification Society on Data Analysis, Machine Learning, and Applications, March 7-9, 2007, Freiburg i. Br., Germany]. Available at: http://arizona.openrepository.com/arizona/handle/10150/106330.

Slavic, A.; Isaac, A. Identifying management issues in networked KOS: examples from classification schemes. NKOS 2009 workshop, ECDL conference. Corfu, Greece. Available at:https://at-web1.comp.glam.ac.uk/pages/research/hypermedia/nkos/nkos2009/presentations/slavic_isaac_NKOS2009_06.pdf.

UDC. Universal Decimal Classification: standard edition: volume 1: systematic tables. London: British Standards Institution, 2005.

UDC. English UDC Online. The Hague: UDC Consortium, 2013. Available at: http://www.udc-hub.com/en/login.php.