Flexible Search Function for Online Courses in the Sense of Attribute Grammars

Carsten Lecon  
Professor  
Aalen University  
Germany

Marc Hermann  
Lecturer  
Aalen University  
Germany
An Introduction to
ATINER’s Conference Paper Series

Conference papers are research/policy papers written and presented by academics at one of ATINER’s academic events. ATINER’s association started to publish this conference paper series in 2012. All published conference papers go through an initial peer review aiming at disseminating and improving the ideas expressed in each work. Authors welcome comments.

Dr. Gregory T. Papanikos
President
Athens Institute for Education and Research

This paper should be cited as follows:

Flexible Search Function for Online Courses in the Sense of Attribute Grammars

Carsten Lecon
Marc Hermann

Abstract

Attribute Grammars are mostly used when specifying a compiler or a software program: A context-free grammar is enriched by variables (attributes) rules and conditions. This approach can also be adapted for a flexible search function – in our case in the context of online courses for e-learning: In general, a course consists of chapters – each composed of subsections (and sub-subsections, etc.). Furthermore, chapter as well as subsections have some metadata, for example title, learning matter (subject), points (of exercises), level (beginner, well advanced, expert), prerequisites, etc. Our data model allows defining inferred attributes alongside the hierarchical connections between objects (parent/children relationship). By doing so, it is also possible to apply aggregate functions like sum or average. For example, if a chapter has the title ‘SQL’, this value can inherit to the subsections, so that a search for ‘SQL’ will not only results in the chapter but also in the subsections – with less weighting. One the other hand, the points of exercise subsections can be transferred to the superior chapter using a sum function, so that a search for all exercises with a specific total point is possible. Furthermore, another way of structure oriented searching is possible: The (recursive) structure of the learning object can be described by object-valued attributes. For example, the titles of all objects of a hierarchy can be described as a character string o6.titlepath='Databases/Languages/SQL'. In this manner, also regular expressions can be used, for example in order to look for all subsections of chapters with the title 'Data Models' or 'SQL':*/('Data Models'|'SQL')/*. When generating online courses, we use an XML document, so an easy access to structure information as well as to the metadata is possible.

Keywords: e-learning, Attribute Grammar, XML, Search Function.

Acknowledgments: Parts of this project was funded by the federal state of Baden Wuerttemberg (Germany) (‘Programm hochschuldidaktische Projekte’), 2017. The authors would like to thank the students Angelika Svoboda, Michael Probst, Marcel Hudy, and Ufuk Karaali for the implementation of parts of the implementation of our tool, as well as Marius Glaess for preparing a course.
Motivation/Background

The heterogeneity of the students is increasing. In Germany, one of the reasons is that the qualification rules and conditions have changed over the last decades. In addition, Germans are starting to study earlier. Formal learning can easily lead to a heavy burden on students who do not have the same previous knowledge as others. Lecturers, on the other hand, cannot respond to these different knowledge bases. However, general conditions cannot be further minimized. Informal learning could be the solution to meet all students. For example, additional learning material can be offered by lecturers or students themselves. Letting the students create the additional learning material is also referred to as learning through teaching, which is beneficial for all involved. Students in the role of a teacher are more aware of the topic. Students that get the additional learning material get an alternative explanation and a different perspective. And last but not least, the lecturers do not have to respond to the heterogeneity of the students too much in the lessons.

Learning by teaching is usually done face-to-face or in tutorial. A more sustainable approach is to provide the learning matter electronically in a way that is easily accessible and extensible. This makes it possible to add additional learning material at any time, if some students need more information or knowledge in the topics. In order to give the students a “red thread”, the learning matter should be well structured. This can result in a ‘mini-online-course’ with exercises for self-monitoring.

The learning matter can be divided in several ways to support different preferences of learning and different approaches. For example, alternative trails can be made available which support learning approaches for students that prefer pictorial representation of information, or for students, that only want to recapitulate the learning content with exercises and summaries.

In some cases, the students only want information to a very specific or relevant topic. In this case, it is advantageous to have a search function at hand. Beyond standard search, where a full-text search or a search in titles is provided, there is more interesting data in an online learning-course that can be made available for searching: the level of difficulty, prerequisites, type of media, etc. This will be the main focus in this work.

In Learning Management Systems (LMS), it is often possible to generate own content. However, the resulting learning material often can’t be exported to other systems and so is bound to the LMS itself; besides, the use of the appropriate tools for constructing good learning content can be rather complex.

To simplify this process, we have developed a tool for rapid content generation (Lecon and Hermann, 2018). In our approach, the content consists of arbitrary assets (media object), which then are put together to a mini-course. The assets could also be reused in other courses as well. An XML schema was specified for describing the structure of the mini-course. The composition of an XML file can be done by a graphical user interface. A special XML parser then generates a set of HTML pages from the XML file and the asset files. The
resulting HTML course can then be used online or offline on almost any
desktop or mobile device, only restricted by the media used.

Our approach allows a very flexible search functionality – going beyond
the classical full text search: Besides, of many (also pedagogical) metadata, we
we can also profit from structure information given by the XML document.

The rest of this paper is organized as follows: First, we show the
background, in which this work is integrated and we give a short overview of
comments about the general problem of the heterogeneity of the students at
universities (next section). In section “Search Functionality” we describe our
search functionality and how it will be implemented. The paper ends with a
short summary and an outlook with further work.

Classification of this Project and Related Work

The aim of this project is to install a kind of Blended Learning at
universities. However, we see this project as an addition to the existing
curriculum; we do not intend to reorganize the study organization like – for
example – described in Chamberlain and Reynolds (2007). We address two
aspects:

- Our intention is to make students to be more self-dependent – in order
to react to the ‘comfort’ of sitting in the lecture hall. One possibility to
address this problem is to bring students to be active learners. This can
be done by an alternative access to the learning subject as an addition to
the lectures by our ‘mini courses’. In doing so, the students are forced
to learn self-independent. We hope, this also will lead to more self-
discipline, which – in turn – results to an increasing motivation (Gong
et al., 2009).

- Another aspect of our work is to react to the increasing heterogeneous
previous knowledge of the students at the beginning of the study. In
online courses the learning behavior can be tracked. Via Learning
Analytics (e.g. Lockyer and Dawson, 2011) it is possible to react to the
specific needs of the individual learner. In a presence learning scenario
(as in the most universities), the teacher can observe the students.
However the possibilities to adapt the lecture to individual needs are
very highly limited. And, the data for a learning analytics analysis come
from a learning management system (Santos et al., 2012). However,
often the students are learning outside the learning management system,
for example by web searching. Therefore, we propose an approach, in
which various learning material can be constructed very fast.

Nowadays, the students are used to get information from online resources;
also commenting and even the generation of own content (Web 2.0) is usual.
Hence, our ‘mini courses’ are a conventional learning tool.

In the following we describe flexible search functionality, where we
combine good practices from database technology and compiler construction. In doing so, structure information can be integrated in queries. This approach is not new: Among others, we combine scientific findings on queries in semi structured data – for example from (Abiteboul, 1999) –, in using attribute grammars for modeling of (semi) structured data – for example from (Neven, 1999) –, and in combining structure and content information – for example (Berlanga et al., 1999); in Zhang et al. (2010) a model (Structure and Content Model) is used for comparing (semi) structured documents regarding the structure and the content.

The specification of our data model is inspired by Attribute Grammars, which are used at compiler construction. In particular, we use the concept of inherited and synthesized attributes (Slonneger and Kurtz, 1995).

An evaluation is not part of this paper, because the search functionality for the tool is still in progress as are other parts like the graphical user interface and HTML templates.

Search Functionality

The main use of mini courses are in the sense of microteaching: In contrast to a complete curriculum, the mini course serves as an additional learning material. Thus, it should be easy to decide, if one mini course is suitable and it should be easy to find the actual relevant topics (as part of the mini course). This requires a powerful query language.

Mostly, when using a search formula only one input field is given. Internally, a search engine typically uses words that appear in the document (full text search). The ranking of the search results often is determined by the frequency of the words and the position inside the document (headline, capture of a figure …). The weighting of the search result also depends on how often a website is linked by other pages. This is kind of structure information. However, we argue, that the structure inside a website (hierarchy of the web pages and the hyperlinks between the pages) can also be used to enrich the power of a query. If a web for example page for example has an HTML heading like ‘Java Course’, this title is also applied to the subpages implicitly.

In our approach, all information about a course is described by an XML document. Thereby, structure information is available in form of the hierarchy of the XML elements and by the linking between elements. Also, several metadata are represented as attributes. The structure (hierarchically order, sequences) exists implicitly (XML structure). There is semi a structured data view to the data, which already was described in (Abiteboul, 1999).

Attributes, which can be determined automatically, are for example: file type, size of the file, date. Other attributes can be specified by the author during the creating phase of the course.

Based on this metadata, flexible search functionality is possible. For example one can seek for
More comprehensive search functionality can be inferred not only from the content but also from the structure (see above), which is described by the hierarchy and sequences, as well as by learning trails and hyperlinks. Hence, we extend the standard search functionality to so-called structure-describing attributes, reference attributes and inherited attributes.

**Structure-Describing Attributes**

The (recursive) structure of the learning object can be described by object-valued attributes. For example, the titles of all objects of a hierarchy can be described as a character string (referred to Figure 1):

```
o6.title path='Databases/Languages/SQL'
```

In this manner, also regular expressions can be used, for example in order to look for all subsections with the title 'Data Models' or 'SQL':

```
*/('Data Models'|'SQL')/*
```

In our case (Figure 1), the objects 2 and 6 would result.

**Figure 1. Short Example of a Course**
Reference Attributes

The data model can define attributes, which are object-valued. That is, they are references to one or more objects. For example:

- o1.children={o2,o3,o4}
- o6.parent=o3
- o4.link=o7

The search expression o1.children results in the objects 2, 3 and 4. o1 itself can be the result of another search. This means that a search result can be a component of a complete query expression (see below ‘component based query language’). In other words, every query component is like an individual ‘view’ to the data.

Derived Attributes

In addition, our data model allows defining inferred attributes: Attributes can ‘inherit’ the values alongside the hierarchical connections between objects (parent-/ children relationship, links, trails). By doing so, it is also possible to apply aggregate function like sum or average.

As an example, consider exercise sheets (see Figure 2), which are organized in a hierarchical structure (see Figure 2, left side): At the top, the title of the exercise is described (‘Exercise SQL’), this exercise consists of three sheets, each with a subtitle and points. The points can be inherited upwards, whereby the sum of all points is calculated, so the top element ‘Exercise 1’ offers the total number of points, which can be got at best. On the other hand, the title of the top element can be inherited to the subjacent sheets downwards.

In Figure 2, the inherited attributes are written in italic.

Figure 2. Inheriting of Attributes and Attribute Values

This means, searching for objects with the title ‘SQL’ results not only in
the top element (‘Exercise 1’), but also in the dependent objects ‘Exercise 1 a’, ‘Exercise 1 b’ and ‘Exercise 1c’.

In order to distinct origin and inherited attributes, the inherited values can be weighted. This will be used to rank the search results. If the sub objects also has an attribute ‘title’, the title of the super object (‘Exercise 1’) can inherit via an union operator; so that both attribute values are considered; generally with different weighting of each value.

Also, it is possible to pass attribute values along a (learning) trail. For example, the topic of one page can be transferred to the other pages of the same trail – with a less weighting of the search result as well. This means, that a specific topic exists anywhere in the actual learning path.

Attributed context free Grammar Approach

The underlying data model can be described by an attributed context free Grammar. Referring the above example, we extend the objects as follows:

- A lecture consists of slides and exercises and is described by a title
- A slide is described by a theme
- An exercise consists of sheets
- A sheet is described by a theme and points

An attributed context free Grammar can look like this (objects are terminal symbols):

\[
\begin{align*}
\text{lecture} & \rightarrow \text{slide}^* \text{ exercise}^* \\
\text{theme} & := \text{slide.theme} \cup (\cup \text{exercise.theme}) \\
\text{exercise.title} & := \text{lecture.title} \\
\text{slide} & \rightarrow \text{object} \\
\text{exercise} & \rightarrow \text{sheet}^* \\
\text{exercise.point} & := \Sigma(\text{sheet.point}) \\
\text{exercise.theme} & := \cup(\text{sheet.theme}) \\
\text{sheet} & \rightarrow \text{object}
\end{align*}
\]

Regarding the implementation of the data structure, there are some specification rules:

- Kind of derivation of the attributes: above (inheriting), below (synthesized)
- Modification of the derived attribute: Attribute value may be modified/ may not be modified; origin attribute value will be overwritten or will be expanded by the derived attribute value(s)
- Calculation: Sum, union, etc.
- Integrity constraints
Component Based Query Language

The above-depicted features go into a data model based on an attributed context free grammars approach – based on Lecon and Seehusen (2002). A query consists of components, where every component represents a single search predicate, for example full text search or metadata search. The components can be joined, for example via and/or connection. A block of a query component can build a subquery.

For every query component the user interface uses an own input field. It might be reasonably assumed that a graphical query language can be used, by which every component is represented as a visual element (for example a box) and these elements are combined by association links (for example by a line).

However, this is a rather challenging task because of the specification of the semantics of each graphical element; the graphical representation of the query can quickly be confusing. For example, the graphical query language XML-GDM (Ceri et al., 1999; see Figure 3) for XML documents was not very successful. Further research is found in Erwig (1998).

Figure 3. XML-GL: Graphical Query Language for XML Documents

Our attempts are not less complicated. A query using a graphical notation is depicted in Figure 4. Apparently, the semantic is hard to understand: In this case, we look for pages, which are located in the learning trail ‘Exercise’, whereby none of these pages is part of the learning trail ‘B’ simultaneously.
Furthermore, the implementation of a graphical user interface is very complex as well as the handling by the users. Hence, we have decided to use a text form for querying (see below).

**XML Document**

The XML definition (see Figure 5) allows specific queries, whereby main attribute as well as inherited attributes can be used, for example the title of a chapter and/or a single page, author, time, level, belonging to a learning trail, etc. Also pages, which are followed by a quiz, can be selected. Of course, full text search is possible. The input for full text results from the content of text (written inside the XML file) and text files; integrated PDF files are converted into text before (a conversion of Word and PowerPoint files is in progress). There are some special features:

1. The *title* attribute of a page is transmitted alongside the learning trails (with minor weighting): The title of the first element of the trail will get the highest weighting.
2. This also applies for (internal) hyperlinks: The title of the source element (page) will be transmitted to the destination element (with minor weighting).
3. When calculating the ranking of the query results, the pages, which are referred by one or more quizzes (attribute *reference* of the *quiz* element), will be ranked higher.
Some aspects regarding the XML DTD are of interest:

- In the generated course, the attribute `contentType` of the element `content` will be shown as an icon in the margin of the HTML page in order to symbolize the kind of content. In the search form, a pull down list of all possible content types will be provided.
- If the (optional) attribute `title` of the element `chapter` is given, this text will serve as a prefix of the title of every subsequent page. For example ‘2.1 Database: Data Models’ and ‘2.2 Database: SQL’ instead of ‘2.1 Data Models’ and ‘2.2 SQL’. This means, the search for `title='Database'` will deliver all pages in chapter 2, because the title attribute of the chapter attribute is transmitted to all subsequent elements.
- The `file` attribute of the `quiz` element refers to a file, in which a quiz is stored (in this way, the same quiz can be used in different courses). The optional attribute ‘references’ refers to pages, in which the topics of the quiz are treated. When generating the course, the (also generated)
quizzes are equipped with hyperlinks to appropriate pages of the course.

- The *hidden* attribute of the element *page* serves for contents, which are firstly invisible. However, when the user presses on the text represented by the *onClick* attribute, the hidden text (attribute *hidden*) will appear. In this way, simple self-tests can be realized: The text shows a question, and after clicking the question text, the answer will be blended in. The search function takes into account these hidden texts.

- Specifying learning trails can be done in two ways (in both cases, the belonging to a learning trail of a page will be taken into account when searching):
  - Attribute *trail* of the element *content* (see Figure 5)
  - Automatic trail detection by finding key words (in the plain text of a page), which are specified in the configuration file of our tool

**Implementation**

The search form is flexible: It is possible to adapt the form individually, for example by adding new search fields (see Figure 6). By adding a new search field, the input field will be generated accordingly to the kind of the field: When choosing attributes with predefined values (for example level, content type), a list of all possible items will be generated. In figure 7 one can see the step just before inserting the new search field ‘Content Type’. The input field for full text search ever is the last field. The single search expressions can be connected by ‘and’, ‘or’ or ‘and not’ conditions. In addition, each search expression can be weighted – in the range of 1 to 10 (this value will internally convert to the interval up to 1).

Figure 6 represent a search for pages with the title ‘kinetosis’ (title of page itself or of the enclosing chapter) [must-meet the criterion] or pages with beginner level [nice-to-have criterion], whereas no summary appear [must-meet criterion], furthermore, the page should contain the word ‘car’.

Empty entries are ignored.

**Figure 6. Search Form (Mockup)**

<table>
<thead>
<tr>
<th>Search Form</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Title:</strong> kinetosis</td>
</tr>
<tr>
<td><strong>Level:</strong> Beginner</td>
</tr>
<tr>
<td><strong>Content type:</strong> Summary</td>
</tr>
<tr>
<td><strong>Full text search:</strong> car</td>
</tr>
<tr>
<td><strong>Weight:</strong> 10</td>
</tr>
</tbody>
</table>

Add search field: ❯
Because the course should be used offline, no database system is used. Instead, the structure information as well as metadata of the pages (as specified in the XML document) is stored in a ‘database’ file. The implementation of the query processing is realized as a Java Applet (see Figure 8).

In order to generate the (HTML) course, the XML DOM tree is traversed. Therefore, not only the metadata (XML attributes) but also structure information (hierarchy of chapter/subchapter) is available. Parallel to the generation process, the ‘database’ is built. Because (HTML) pages are the smallest unit (for the user), the internal database consists of a collection of relevant information for each page (extract), (see Table 1).
Table 1. Database Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Multiple</th>
<th>Automated Weighting</th>
</tr>
</thead>
<tbody>
<tr>
<td>title</td>
<td>yes (title of page and chapter)</td>
<td>1 (page title)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0..1 (chapter title)</td>
</tr>
<tr>
<td>level</td>
<td>yes</td>
<td>1</td>
</tr>
<tr>
<td>content type</td>
<td>yes</td>
<td>1 (per each)</td>
</tr>
<tr>
<td>author</td>
<td>yes</td>
<td>0..1 (per each)</td>
</tr>
<tr>
<td>time</td>
<td>no</td>
<td>1</td>
</tr>
<tr>
<td>type</td>
<td>yes</td>
<td>1 (per each)</td>
</tr>
<tr>
<td>trail</td>
<td>yes</td>
<td>1 (per each)</td>
</tr>
</tbody>
</table>

The above described search form offers some opportunities of adaption already (adding/ removing search fields, weighting of the particular search expressions). Additionally, it is planned, that the user can adapt the search functionality by adjusting some further parameters, for example:

- Specifying the value of the weights for inherited attributes (title attribute of page and chapter elements) – and if this even should take place.
- Specifying the value of the weights for inherited attributes (title attribute of trails) – and if this even should take place.
- Specification of trails (on the basis of given key words, which are in the text of the generated pages exist)
- Reusing the search result in order to create a new course – consisting of the found pages.

Conclusion and Further Work

We have presented a tool for generating so called offline ‘mini courses’. The motivation for this project is:

- Reacting to the increasing heterogeneity of previous knowledge of the student: Some learning matters can be repeated and presented in another way for self-learning (as addition to the lectures and tutorials).
- The choice of the learning content of a lecture at a university (or a school) is restricted due to the curriculum. The consequence is that especially some new and innovative topics cannot be dealt with. Additional learning material – as mini courses (for self-learning) – can complete the lecture. The realization of these courses can be done by the teacher and/ or the students itself (we have tried this approach successfully) – in the sense of ‘learning by teaching’ (Stollhans, 2016; Biswas et al., 2005).

Search functionality for e-learning course is natural. In this specific context, this is even truer, because often the learning is situational and time
sensitive, so that a quick access to the desired learning material is of advantage.

Our search functionality bases on content information (full text) and a rich set of metadata; furthermore, structure information is considered. For this, internally we use a data model inspired by attributed context free grammars (known in the area of compiler construction).

Our next steps are (among others): We will complete the implementation of the search functionality. Also, subqueries (query components, see above) will be possible, like subqueries in SQL. These components represent a data set, which can be combined with the other queries – that also means, components appear in the list of ‘search fields’ (without parameter).

Performing a full text search (momentarily) all matches are treated equal. However, studies have shown, that there is an order when reading a text (with pictures): First (all) pictures, then picture descriptions (there exists an own attribute in our XML document), then headlines, then plain text. Therefore, it makes sense, to weight the matches accordingly.

So far, the transmitting (inheriting) of attribute values is restricted to text. However, also numeric attributes should be treated – as in the above example (Figure 2). An automatic detection and interpretation of numeric values in the texts is difficult and – probably – not clear. One possibility to implement this is to mark the numeric values in the text by special characters. Then the user could also use aggregate functions (sum, average …), which are specified in the XML document (eventually with additional attributes; in the present state of our graphical user interface, this is possible – actually, our XML parser validate, if the XML document is well formed only (not if this document is valid).

We will optimize the query processing (indexing process, caching …). In this context, we will test and evaluate alternative possibilities to store the data locally, for example Web Storage, whereas the data could be represented as serialized JSON document.

In order to integrate any type of media (especially PDF documents, Microsoft Word and PowerPoint files), we plan to implement converting programs, which convert these files into text files (which then be seen as plain text – for full text search); also we intend to extract relevant information from media files (sound, video): modification date, format, size, resolution, sampling rate, … - apart from the normally available information: Subtitle and description.

It could be useful to store and load a complete query; for example as a new search component, which can be adapted for future queries.

Furthermore, we will examine, if a hybrid version (online/ offline) of the search functionality is possible, since nowadays the students are mostly online (via mobile devices). In this case, we could use a standard database, in which the individual course of the students is stored; a server could process the queries faster.

We will examine, if our approach can be adapted to external courses

---

(online course) as well; in this case, the material will be transformed into our data model (by extracting metadata and structure information).

References

164. 2016.