More Interactions in ALeA

—

Towards New Added-Value Services based on Semantic Markup

Andrea Kohlhase[1][0000−0001−5384−6702] and Michael Kohlhase[2][0000−0002−9859−6337]

[1] Andrea.Kohlhase@hnu.de
Information Management
Neu-Ulm University of Applied Sciences
[2] Michael.Kohlhase@fau.de
Computer Science
FAU Erlangen-Nürnberg

Abstract. The ALeA system (Adaptive Learning Assistant) uses a fine-grained domain model, a learner model based on it, and semantically annotated learning objects to generate user-adaptive and interactive course materials for pre/postparation of lectures and self-study.
In this paper we propose new interactions and learning support services that enhance the learner experience without requiring new markup facilities – in essence enhancing the didactic capabilities of the system without further investment into marking up learning objects.

1 Introduction

The ALeA system (Adaptive Learning Assistant; see [Ber+23]) uses fine-grained, modular domain models and semantically marked up learning objects together with a learner competency model to generate learner-adaptive and interactive learning materials for self-study and pre/postparation of lectures. The learners' interactions with the system are in turn used to keep the learner model updated.

A first version of the system (developed in the **Voll-KI** project) has been in active use for six courses with over 1000 Math/CS/AI students at FAU Erlangen-Nürnberg; see [VKC]. The system has been evaluated positively by the students who used it. To make the system even more engaging we are looking for extensions in terms of added-value services it can offer based on the existing underlying semantic markup, this giving more return on the existing semantic investment.

In this paper we explore new added-values for ALeA in terms of learning support services and interactions for Math and Computer Science oriented lectures. Here, we only present "paper prototypes" of the services and interactions, i.e., simple mockups that showcase the functionality, where we can check that all of the information in the presented interactions actually comes from the semantic markup, but no actual implementation. This is in spirit with the Voll-KI project's paper prototypes (see [VKE]) that have already been shown very influential for advances in Voll-KI.

ALeA uses the STEX format – an extension of LATEX for semantic markup – as the main authoring format. STEX sources can either be formatted to PDF by `pdflatex`

or to semantically annotated HTML via `rusTeX` [RT], which in turn can be harvested for semantic content by the MMT system [MMT] that drives the learning support services and interactions in ALeA. Authoring STEX learning objects and building PDF or HTML/MMT is supported by a dedicated STEX plugin for the VScode IDE and an MMT-based build system that can automate the build process of large STEX corpora (currently $\sim$ 15.000 small files). Correspondingly, all examples and source listings in this paper are in STEX; we will explain the pertinent details as we go along. For details see [MK22; KM]. We are working on similar extensions for the MS Office suite and Markdown for subsets of the functionality to scale authorship beyond the core STEM community STEX restricts it to at the moment.

In Sections 2 to 4 we present several learning support services and in Section 5 additional novel interactions – all of them harvesting added-values from semantic markup. Section 6 concludes the paper and discusses future work.

Acknowledgments The work reported in this article was conducted as part of the VoLL-KI project (see `https://voll-ki.de`) funded by the German Research/Education Ministry under grant 16DHBKI089.

2 Working with Definienda

One of the signature interactions in ALeA is the one on term references – i.e., occurrences of technical terms that have been defined elsewhere: when hovering on a term reference t , a rendering of the corresponding definition from the domain model appears. When a learner clicks on the hover, this raises a popup, that gives access to metadata on t and a system-generated guided tour for t, i.e. a bottom-up explanation for t restricted to the concepts deemed unknown to the particular learner by the learner model.

The definienda – defining occurrences of a technical term – do not carry such an interaction. We propose to add an interaction for any definiendum d that gives a learner information on d including, but not limited to:

Relevance How often is the symbol of d referenced in the course C; this gives the learner a sense of the importance of d. Here, relative weights (compared to other concepts introduced in C) possibly tabulated by the context (e.g., definitions, theorems, problems, and explanatory text) might be helpful.

Range A list of occurrences o in C with an indicator for how far away o from d in C is (we could call that **usage radius** $u_C^d(o)$ [3] and rank the list accordingly), together with

Exploration a guided tour for any occurrence o providing a local context a learner is interested in as a look-ahead/exploration into the usage of d, which we call **guided exploration**.

Exploration Size Each guided exploration should indicate its length, as shorter ones promise more immediate satisfaction.

Difficulty Level From all learner models the information can be collected, which definiendum was worked on the most (we call it **difficulty level**, which in turn

[3] Both the distance between o and d in the course C as well as the size of the guided tour from d to o come to mind as possible measures for u_C^d. The latter is appealing as it is a learner-specific measure.

might be measured by the time it takes for the learner-model's assessment to reach a given threshold).

Global Exploration An analogous list of occurrences outside of C for learners that have resources for exploring beyond the course C.

The concept of topology in the course AdvMath-1

Direct Relevance			*Indirect Relevance*			*Concepts based on topology*	
context	#	weight	context	#	weight	concept	$u_C^d(o)$
definitions	18	1.7	definitions	183	3.6	topological space	1
theorems	7	.6	theorems	67	1.3	open set	1
exam problems	8	.7	exam problems	103	2.1	continuous	3
other problems	10	.9	other problems	27	.5	topological manifold	5
text	7	.6	text	53	1	homeomorphism	6
total	**52**	**5**	**total**	**433**	**8**		...more

Fig. 1. An interaction on a definiendum

Figure 1 shows a first design of definiendum hover. The two panels on the left show relevance measures in various contexts[4]. We distinguish between direct references to d and indirect ones where terms defined from d are reference. Both **direct relevance** and **indirect relevance** important to judge the importance of d.

This information can serve the learner as an **information scent**, i.e., *"a user's (imperfect) perception of the value, cost or access path of information sources obtained from proximal cues"* [WHA07], a term originally introduced in [PC99] in the context of human information foraging. Here, it is assumed that people try to maximize their rate of gaining valuable information and modify their user interactions accordingly.

The right panel in Figure 1 shows local explorations ranked by usage radii.

This pre-computed metadata summary can be complemented by a specialized semantic search facility that focuses on the definiendum d, in particular allowing to focus on any combination of *i*) definienda/recaps *ii*) exemplanda for *iii*) assertions using *iv*) questions/answers for *v*) study groups touching on *vi*) who – i.e., which other learners[5] – are currently working with d and might be willing to discuss or answer questions about it. Answers to the "who" questions are particularly interesting, since they offer a **social awareness scent**, a special kind information scent relevant to learning (see [Stä08]).

3 Playing with Dependencies

Existing STEX markup gives us the **terminological dependency order** $\preceq_t$, a relation on concepts defined by semantically marked-up definitions: In a nutshell – if a definiendum c appears in a definition together with term references $t_1, \ldots, t_n$, then $t_i \preceq_t^d c$ – the **definition-induced dependency relation** – for all $1 \leq i \leq n$ and $\preceq_t$ is the transitive closure of $\preceq_t^d$.

[4] We suspect that a significant portion of learners will mainly focus on the weight of a concept in the exam problems.

[5] ... who have allowed to be discovered this way in their preferences.

Intuitively, if $p \preceq_t c$ then understanding[6] p is a prerequisite for understanding a concept c. For instance, in the definition in Figure 2, the concept of topology terminologically depends on empty set, elementhood, the subset relation, big union, finite, and big intersection.

```
\begin{sdefinition}
  \vardef{bset}[bind]{X}\vardef{top}{\mathcal{O}}
  Let $\bset$ be a \sn{set}, then a \sn{set system} $\top$ is called a
  \definame{topology}, iff
  \begin{enumerate}
  \item \inlineass[name=eset.cond]{$\inset\eset\top$}.
  \item \inlineass[name=fullset.cond]{$\inset\bset\top$}.
  \item \inlineass[name=union.con]{$\inset{\munionCollection{s}Ss}\top$
      if $\sseteq{S}\top$}.
  \item \inlineass[name=fincap.cond]{$\inset{\mintersectCollection{s}Ss}\top$
      if $\sseteq{S}\top$ is \sn{finite—cardinality?finite}}.
  \end{enumerate}
\end{sdefinition}
```

Let X be a set, then a set system $\mathcal{O}$ is called a **topology**, iff

1. $\emptyset \in \mathcal{O}$
2. $X \in \mathcal{O}$
3. If $S \subseteq \mathcal{O}$, then $\bigcup_{s \in S} s \in \mathcal{O}$. (**closed under unions**)
4. If $S \subseteq \mathcal{O}$ is finite, then $\bigcap_{s \in S} s \in \mathcal{O}$. (**closed under finite intersections**)

Fig. 2. Definition of topology

So far, the ALeA system only uses $\preceq_t$ as an ordering principle (prerequisites before defined concepts) in the generation of guided tours. But arguably understanding $\preceq_t$ is a learning goal and/or competency metric in and of itself, as it seems correlated[7] with having an overview over the terminology of a domain and how the various definitions play together. This seems particularly valuable in settings like ALeA, since this is a "global" property, whereas existing learning support services are largely "local".

We propose that a **dependency patience** game might be a suitable interaction/gamification to assess and practice understanding $\preceq_t$. We envision a patience-like game, where learners are given a random sequence of "term cards", which must be placed on a board in terminological dependency order. As $\preceq_t$ is a partial ordering, the learner builds up a DAG corresponding to $\preceq_t$. Figure 3 shows an initial segment of such a patience game for a course of set-theoretic topology (with game steps from left to right and the card stack with a randomly shuffled set of the terms are introduced in the course which; here topological space, topological manifold, empty set, $\mathbb{N}$, ...). Concretely, in each step she has the choice to place the next card

– above one of the roots of the DAG (i.e. one of its initial nodes)

[6] This seems similar for other cognitive dimensions of ALeA's competence model [AK09] except – notably – remember: one can remember a definition without remembering the definitions of the prerequisites.

[7] So far this is only an intuition of the authors of this paper, we are not aware of any scientific studies that support this conjectured correlation; we plan to substantiate this in the future: the methods proposed here will enable us to do so.

– below any of the nodes,
– on any of the edges to indicate that the card goes between its end points, and
– on any of the nodes to indicate that the concept is $\preceq_t$-equivalent to it,

and to add arbitrary edges or mark any existing edge as "immediate" – i.e., as belonging to $\preceq_t^0$, the transitive reduction of $\preceq_t$.

If any placement contradicts $\preceq_t$, the move is canceled, the falsely placed term card is shuffled back into the term stack, points are deducted, and possibly feedback is given.

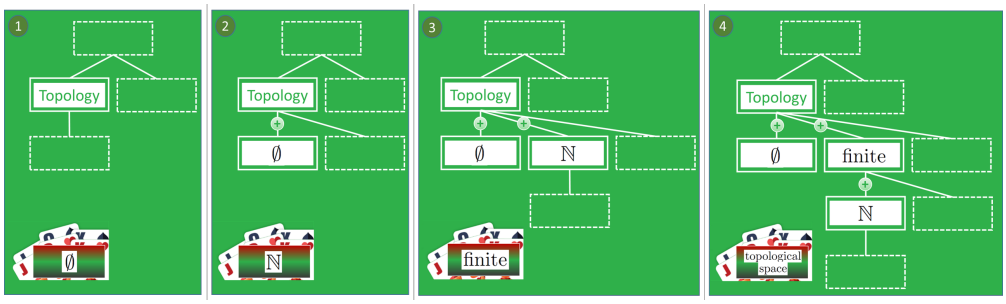

Fig. 3. The first four steps of a dependency patience game about the concept topology

The dependency patience service would only be offered in ALeA when

1. A sub-domain of suitable size and complexity has been covered and
2. the learner has – according to the ALeA learner model – mastered remembering the requisite definitions sufficiently.

Otherwise the game would be either 1) too easy or 2) too hard/frustrating.

To make the game more engaging – and using ALeA system more enticing to students, it could be run as a competition, announcing the principal availability of the game when the lecture has completed a section or chapter and publishing high-scores.

4 Generating Problems from Examples

One of the biggest concerns in ALeA is to come up with rich information about the learning process for each learner. One way to achieve this is to create sufficiently many problems that can be used to assess the competency of the learner interacting with the system. These should be simple enough and quick enough to solve so that they do not interrupt the learning flow but should test specific competencies to inform the learner model that in turn informs (competency-adaptive) learning support services.

In this section we will see how semantically marked up examples can be used to systematically generate problems that are open to self-assessment or auto-grading. The results of this process can either directly be used in ALeA or used by authors as a basis of explicitly represented learning objects to tweak the wording or feedback to the learner.

4.1 The Theory of Examples

Somewhat simplifying [Koh06, section 15.4], an **example** is a structure $E := \langle p, a, \pi \rangle$ consisting of

- a symbol p denoting a property,
- an expression a denoting a mathematical object, and
- (optionally) a proof π that $p(a)$ holds in the current context

We call p the **exemplandum**(*plural* **exemplanda**, the property to be exemplified), a the **exemplans** (*plural* **exemplantia**, the witness object for the property p), and π the **justification** of E (the reasoning why a indeed satisfies p). Correspondingly, in a **counterexample** (an example for the complement of p) π is a proof of $\neg p(a)$.

For instance, if we take the number 7 to be an example of a prime number, then 7 is the exemplans, the property of being prime the exemplandum, and the (trivial) proof that indeed 7 is prime the justification. Similarly, 7 can be a counterexample for the property of being even and 8 a counterexample for being prime.

Often, the situation is more complex like giving an example for a topology. Here, the exemplandum is the *property of being a topology*, which – given a base set X – is a condition on families $\mathcal{O} \subseteq \wp(X)$ of subsets of X, that naturally decomposes into four separate conditions (see Figure 2 below).

There are two canonical examples for a topology: the trivial topology, where $\mathcal{O} = \{\emptyset, X\}$ and the discrete topology, where $\mathcal{O} = \wp(X)$. In both cases, the justification π decomposes into four subproofs of the four conditions – all of them trivial in both examples.

Note that these two examples are **generic** in the sense that they are "example constructors" that – given a particular set – provide a topology. So given enough **concrete** – i.e. non-generic – examples for sets (like $\mathbb{N}$ or $\mathbb{Z}$) they systematically generate concrete exemplantia $\langle \mathbb{N}, \{\emptyset, \mathbb{N}\} \rangle$, $\langle \mathbb{Z}, \{\emptyset, \mathbb{Z}\} \rangle$, $\langle \mathbb{N}, \wp(\mathbb{N}) \rangle$, $\langle \mathbb{Z}, \wp(\mathbb{Z}) \rangle$, ... for topologies.

4.2 Semantic Markup for Examples and more in sTeX

The sTeX syntax to annotate $\mathbb{N}$ as an example for being a set –based on the already existing domain model for the natural numbers – is given in Figure 4. Note, that the

```
\begin{sexample}[name=natset.ex,for=set]
  \usemodule[smglom/arithmetics]{mod?naturalnumbers}
  $\NaturalNumbers$ is a \sn{set}.
\end{sexample}
```

Example: $\mathbb{N}$ is a set.

Fig. 4. Example: $\mathbb{N}$ is a set

exemplandum is stated in the for attribute of the sexample environment ($p = set$) and the exemplans a is the $\mathbb{N}$ object.

And we have a counterexample as well: $\{X\}$ alone is no topology over X, since $\emptyset$ is missing. The sTeX syntax for this looks like an example, but note the distinguishing value for the style attribute:

We assume that definition for the exemplandum $p = topology$ from Figure 2 and the following theorem/proof combo as a justification π:

```
\vardef{Xv}[bind]{X}
\begin{sexample}[name=topdiscrtop.ex,for=topology,args=set]
  \usemodule[smglom/toplology]{mod?open-set-topology}
  \usemodule[smglom/sets]{mod?powerset}
  For any \sn{set} $\Xv$, $\powerset\Xv$ is a \sn{toplogy} on $\Xv$.
\end{sexample}
```

Example: For any set X, $\wp(X)$ is a topology on X.

Fig. 5. Example: $\wp(X)$ is a topology

```
\begin{sexample}[style=counterexample,name=noesetnotop.ex,for=topology,args=set]
  \usemodule[smglom/toplology]{mod?open-set-topology}
  \usemodule[smglom/sets]{mod?powerset}
  For any \sn{set} $\Xv$, $\set\Xv$ is a not a \sn{toplogy} on $\Xv$.
\end{sexample}
\spfsketch[name=nosetnotop.just,for=noesetnotop.ex]
   {$\ninset\eset{\set\Xv}$, so the first condition is violated}
```

Example: For any set X, $\{X\}$ is a not a topology on X.
Proof Sketch: $\emptyset \notin \{X\}$, so the first condition is violated

Fig. 6. Counterexample: $\{X\}$ is no topology

```
\vardef{bset}[bind]{X}
\begin{sassertion}[style=lemma,name=powtop]
  Let $\bset$ be a \sn{set}, then $\powerset\bset$ is a \sn{topology}
\end{sassertion}
\begin{sproof}[for=powtop]{We prove the four conditions separately}
  \begin{subproof}[for=eset.cond]
    $\inset\eset{\powerset\bset}$ by definition as $\sseteq\eset\bset$
  \end{subproof}
  ...
\end{sproof}
```

Lemma: Let X be a set, then $\wp(X)$ is a topology
Proof: We prove the four conditions separately

1. $\emptyset \in \wp(X)$ by definition as $\emptyset \subseteq X$.
2. ...

Fig. 7. Example Assertion and Justification

4.3 Some Categories of Generated Quiz Problems based on Examples

For the application in the ALeA system, we are only interested in what we will for now call **quiz problems**, i.e., problems that are open to self-assessment or auto-grading. These can support the learning flow on-the-fly without having to wait for feedback by others – leading to more interactive engagement with the current content.

We start with the observation, that **naming tasks** like *"Name three examples of topologies over* $\mathbb{R}$*."* are not quiz problems, as they are open-end questions whose answers can neither be self-assessed nor automatically assessed.

Example Choosers We can generate quiz problems, which we call **example choosers**, in the form of multiple choice problems, in which the correct exemplantia for an exem-

plandum must be chosen. Here, we can simply draw on a set amount of semantically annotated examples and counterexamples in the learner objects or the domain model.

The generated problem would be formalized as can be seen in Figure 8. Here, we use

```
\begin{sproblem}
  \usemodule[smglom/topology]{mod?open-set-topology}
  \objective{apply}{topology}
  Which of the following are \sr{topology}{topologies} over $\NaturalNumbers$?
  \begin{mcb}
    \mcc[T,Ftext={by \sref{powtop}}]{$\powerset\NaturalNumbers$}
    \mcc[F,Ttext={by \sref{noesetnotop.just}}]{$\set{\bset}$}
    \mcc[T,Ftext={the topology conditions are trivially/vacuously met}]
    {$\set{\eset,\NaturalNumbers}$}$
  \end{mcb}
\end{sproblem}
```

Problem: Which of the following are topologies over $\mathbb{N}$?

- ☐ $\wp(\mathbb{N})$
- ☐ $\{\mathbb{N}\}$
- ☐ $\{\emptyset, \mathbb{N}\}$

Fig. 8. A Generated *Example Chooser*

the two examples trivial and discrete topology given in Figure 5, the counterexample in Figure 6 (as a distractor in the multiple-choice problem) and the assertion in Figure 7 giving rise to an example, which we assume to have been marked up explicitly as such. Note that all the text in the example chooser is either generic (e.g. the setup sentence) or copied from the markup of the involved objects. Note that the problem generation can also automate the objective annotations for problems and choices; a very tedious task when manually authoring problems.

Justification Spotters We can also generate problems that make use of the existence of examples or counterexamples with justifications for a certain property. These we call **justification spotters**. For instance, if we want to drill in on the conditions of the definition of a topology, we can generate such a quiz problem as can be seen in Figure 9.

Note, that $\mathbb{N}$ was introduced as a counterexample for a topology, because the $\emptyset$ is not included.

Frame Fitters The idea is that linked pair of example can be framed by the inner and the outer examples. For instance in Figure 10, the console.log function is an example for a JS function and the content of the sexample environment is an example for console.log. We call q these two examples **linked**: the exemplans of the one is the exemplandum of the other, which can thus be thought of as an "iterated" example for a JS function – indeed it has a JavaScript code fragment as a central feature. But they are "framed" – i.e. put into context – differently.

We can use the material from Figure 10 to generate a problem that ask for the best fit frame. Figure 11 has the result; we call this kind of quiz problems a **frame fitter**.

```
\begin{sproblem}
  \usemodule[smglom/topology]{mod?open-set-topology}
  \objective{remember}{topology}
  Is $\top=\set{\NaturalNumbers}$ a \sn{topology}? \yesFnoT
  If not, which conditions are violated?
  \begin{mcb}
    \mcc[T,obj=eset.cond@apply]{$\inset\eset\top$}
    \mcc[obj=fullset.cond@apply]{$\inset\bset\top$}
    \mcc[obj=union.cond@apply]{$\top$ is closed under unions}
    \mcc[obj=fincap.cond@apply]{$\top$ is closed under finite intersections}
  \end{mcb}
\end{sproblem}
```

Problem: Is $\mathcal{O} = \{\mathbb{N}\}$ a topology? ☐ Yes ☐ No
If not, which conditions are violated?

- ☐ $\emptyset \in \mathcal{O}$
- ☐ $X \in \mathcal{O}$
- ☐ $\mathcal{O}$ is closed under unions
- ☐ $\mathcal{O}$ is closed under finite intersections

Fig. 9. A Generated *Justification Spotter*

```
1  \begin{sexample}[for=javascript?function,name=clogJSEx.
2    title="console.log as a JavaScript function"]
3    \exemplans{console.log} is a \exemplandum{javascript?funccton}.
4  \end{sexample}
5
6  \begin{sexample}[for=console.log, title="Usage of console.log", name=clogUsage]
7    \usemodule[smglom/computing]{mod?bug}
8    \begin{scodefragment}[lang=JavaScript]
9      function square (n) {
10       console.log ("entered function square with argument " + n);
11       return (n * n);
12       console.log ("exited function square with result " + n * n);
13     }
14   \end{scodefragment}
15   In the \sn{console} we can check whether the content contains e.g. \scode{function
16     square} and moreover whether \sn{argument} and \sn{value} are as expected.
17 \end{sexample}
```

Fig. 10. Linked Examples

Note that the boxed text in (the presentation of) Figure 11 is directly copied from the example in Figure 10 (lines 7-17) in the problem generation.

Also note that we should be able to generate frame fitter quiz problems for linked definienda analogously, mixing techniques from frame fitters and

5 Local Interactions

The new learning support services in the last sections can be complemented and enhanced by additional interactions that support working with ALeA-presented learning objects and generally streamline the user/learner experience. We list them here without paper prototypes as they are relatively obvious.

```
\begin{sproblem}
  \usemodule[smglom/computing]{mod?bug}
  \objective{understand}{console.log}
  Consider the followign code example:
  \begin{scodefragment}[language=JavaScript]
    function square (n) {
      console.log ("entered function square with argument " + n);
      return (n * n);
      console.log ("exited function square with result " + n * n);
    }
  \end{scodefragment}
  In the \sn{console} we can check whether the content contains e.g. \scode{function
    square} and moreover whether \sn{argument} and \sn{value} are as expected.
  What is it an example for?
  \begin{mcb}
    \mcc[T,obj=eset.cond@apply]{\sref{clogUsage}}
    \mcc[obj=fullset.cond@apply]{\sref{clogJSEx}}
  \end{mcb}
\end{sproblem}
```

Problem: Consider the following code example:

```
function square (n) {
  console.log ("entered function square with argument " + n);
  return (n * n);
  console.log ("exited function square with result " + n * n);
}
```

In the console we can check whether the content contains e.g. entered function square and moreover whether argument and value are as expected.

What is it an example for?

- ☐ Usage of console.log
- ☐ console.log as a JavaScript function

Fig. 11. A Generated *Frame Fitter*

1. **Adaptive highlighting**: In the current default presentation UI of ALeA, all definienda are highlighted in boldface magenta, and all term references in cyan. While this makes the semantic annotations easily accessible – e.g., for interactions in ALeA or for PDF-based debugging while authoring learning objects – some readers object to the colorful text. Also, the ALeA choices may clash with the institutional styles. While the colors themselves are easy to customize (both in the STEX as well as in the ALeA CSS) we propose a different user-adaptive highlighter in ALeA, that could e.g.
   - restrict highlighting to the first occurrence of a term reference in a document context. Several context levels are possible: for instance, a reducing fashion like in a paragraph or a more maximized fashion like in the current screen view (of a browser), a page (of a PDF), or even the document itself.
   - adapt the highlighting to the reader's familiarity – as estimated by the learner model – with the term.

   Note that highlighting has two potential effects: it draws attention to the respective words or to their interaction affordances. The latter specifically supports learners in ALeA and the former gives rise to a good content overview to experts. Even

so, highlighting always draws attention *away* from the content or didactic intent. This is no problem for experts, who already know the content, but for the others it may be. We will need careful experimentation with the tradeoffs underlying these conflicting effects to see what highlighting strategy will be most effective for which readers.

2. **Delayed elaboration**: Learning objects like slides often employ didactic reductions – i.e., a concept is presented in a telegraphic style – only to elaborate this further in the course notes. In interactive media like ALeA courses, we might want to offer the elaborations – represented by **begin**{sparagraph}[style=elaboration] in STEX– only upon user interaction (e.g. on hover) and only if the user model deems the prerequisites fulfilled.
3. **Word cloud glossary**: For each learning object – e.g. a slide – we can compute – i.e. query MMT for – the set of prerequired symbols that are possibly unfamiliar to the reader. Their verbalizations can be visualized in a word cloud where the level of familiarity – according to the learner model – is used as a size measure. This word cloud can be associated with e.g. slides, sections, etc. and act as a glossary: clicking on a term will pop up a definition and/or guided tour.
4. **Live demo**: If the learning object can be demoed (e.g., the intro of a JS function like confirm), then the authors of the learning object could include code in an \scode macro or scodefragment environment. The code can be demoed in a sandbox programming environment. For instance, it could be either runnable in a Jupiter kernel or as a first stage simply (in case of JS) in the Browser. Concretely we would extend the scodefragment with a livedemo key that triggers – depending on context – different behaviors in ALeA:
   - present an image/HTML screenshot if the output format is PDF without the pdfcomment
   - use PDF comment functionality if it is
   - run an arbitrary program with local data if not connected to the Internet
   - use online functionality/servers if we are.

6 Conclusion & Future Work

We have proposed and explored additional added-value services for the ALeA system focusing on ones that can be realized based on existing STEX markup facilities. We have described larger learning support services making use of definitions, dependencies, or example markup and local interactions that can be realized with relatively little effort.

They all promise to add to the didactic capabilities of ALeA without having to extend the underlying semantic markup facilities in STEX. Note that it might still be necessary – and with the new services and interactions now directly profitable – to extend the corpus of semantically marked up learning objects. The examples and their explanations in this paper can serve as an informal style guide for this.

The next logical step will be implementation and experimentation with the learning support services and interactions in ALeA. While the latter can almost directly be realized in the ALeA frontend, the former also require extensions in the ALeA backend. For the definienda in Section 2 we need a new "definiendum metadata API" in the backend (MMT and the learner model server). For the patience game in Section 3 the game mechanics (where to place what when and what player feedback is appropriate/helpful where) and the learner model updates are needed – the terminological dependency

relation is already realized in MMT. Finally, for the generation of the three kinds of problems in Section 4 we need to slightly extend the markup facilities for examples, so that the generation of exemplantia and problems can work. We will have to experiment to determine the best generation strategies: as STEX sources that can be edited by authors as presented in this paper or in the backend so that they can directly be used (and automatically adapted to the learner) in the ALeA system.

References

[AK09] Lorin W. Anderson and David R. Krathwohl. *A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational objectives*. New York: Longman, 2009.

[Ber+23] Marc Berges et al. "Learning Support Systems based on Mathematical Knowledge Managment". In: *Intelligent Computer Mathematics (CICM) 2023*. LNAI. in press. Springer, 2023. URL: https://url.mathhub.info/CICM23ALEA.

[KM] Michael Kohlhase and Dennis Müller. *The sTeX3 Manual*. Tech. rep. URL: https://github.com/slatex/sTeX/blob/main/doc/stex-manual.pdf (visited on 06/06/2023).

[Koh06] Michael Kohlhase. *OMDoc – An open markup format for mathematical documents [Version 1.2]*. LNAI 4180. Springer Verlag, Aug. 2006. URL: http://omdoc.org/pubs/omdoc1.2.pdf.

[MK22] Dennis Müller and Michael Kohlhase. "sTeX3 – A LaTeX-based Ecosystem for Semantic/Active Mathematical Documents". In: 43.2 (2022). Ed. by Karl Berry, pp. 197–201. URL: https://kwarc.info/people/dmueller/pubs/tug22.pdf.

[MMT] *MMT – Language and System for the Uniform Representation of Knowledge*. Project web site. URL: https://uniformal.github.io/ (visited on 01/15/2019).

[PC99] Peter Pirolli and Stuart Card. "Information Foraging". In: *Psychological Review* 106.4 (1999), pp. 643–675. DOI: 10.1037/0033-295X.106.4.643. URL: http://dx.doi.org/10.1037/0033-295X.106.4.643.

[RT] *sLaTeX/RusTeX*. URL: https://github.com/sLaTeX/RusTeX (visited on 04/22/2022).

[Stä08] Helge Städtler. "Virtuelle Proxemik: Konzeption, Implementierung und Evaluation einer Komponente zur Bereitstellung proxemischer Information im e-Learning". PhD thesis. Computer Science, Universität Bremen, 2008. URL: https://media.suub.uni-bremen.de/handle/elib/2558?locale=de (visited on 08/07/2023).

[VKC] *VoLL-KI based Courses at FAU*. URL: https://courses.voll-ki.fau.de (visited on 07/21/2023).

[VKE] *VoLL-KI Experiments*. URL: https://courses.voll-ki.fau.de/en/exp (visited on 08/05/2023).

[WHA07] Wesley Willett, Jeffrey Heer, and Maneesh Agrawala. "Scented Widgets: Improving Navigation Cues with Embedded Visualizations". In: *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)* 13 (6 2007), pp. 1129–1136. URL: http://vis.stanford.edu/papers/scented-widgets.