sTeX – A System for Flexible Formalization of Linked Data

Andrea Kohlhase
German Research Center for Artificial Intelligence (DFKI)
Enrique-Schmidt-Str. 5
28359 Bremen, Germany
Andrea.Kohlhase@dfki.de

Michael Kohlhase
Jacobs University Bremen
P. O. Box 750561
28725 Bremen, Germany
m.kohlhase@jacobs-university.de

Christoph Lange
Jacobs University Bremen
P. O. Box 750561
28725 Bremen, Germany
ch.lange@jacobs-university.de

ABSTRACT

We present the sTeX system, a semantic extension of LaTeX, that allows for producing high-quality PDF documents for (proof)reading and printing, as well as semantic XML/OMDoc documents for the Web or further processing. Originally created as an invasive, semantic frontend for XML documents, we use sTeX in a Software Engineering case study as a formalization tool and upgrade it to deal with modular presemantic vocabularies and relations and generating Linked Data based on *all* structural explications. We present a tool chain that starts with an sTeX editor and ultimately serves the generated documents as XHTML+RDFa Linked Data via an OMDoc-enabled, versioned XML database.

Categories and Subject Descriptors

I.2.4 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods—*Representation languages*; I.7.2 [**Document and Text Processing**]: Document Preparation

Keywords

formalization, LaTeX, linked data, software engineering, semantic authoring, annotation, metadata, RDFa, vocabularies, ontologies

1. INTRODUCTION

An important issue in the Semantic Web community was and still is the "Authoring Problem": How can we convince people not only to use semantic technologies, but prepare them for creating semantic documents (in a broad sense) too? Here, we were interested in formalizing a collection of LaTeX documents into a set of files in the OMDoc format, an XML vocabulary specialized for managing mathematical information, and further on to Linked Data for interactive browsing and querying on the Semantic Web.

Concretely, the object of the study was a safety component for autonomous mobile service robots developed and certified as SIL-3 standard compliant (see [10]) in the course of the 3-year project "Sicherungskomponente für Autonome Mobile Systeme (**SAMS**)" at the German Research Center for Artificial Intelligence (DFKI). Certification required the software development to follow the V-model (figure 1) and to be based on a verification of certain safety properties in the proof checker Isabelle [30]. The V-model mandates e. g. that relevant document fragments get justified and linked to corresponding fragments in other members of the document collection in an iterative refinement process (the arms of the 'V' from the upper left over the bottom to the upper right and in-between in figure 1).

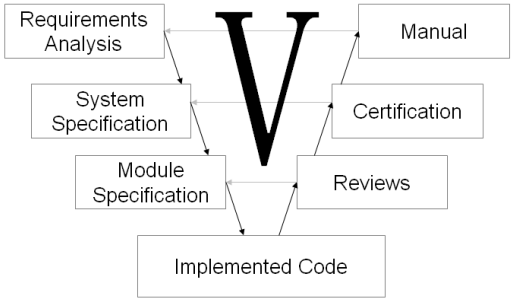

Figure 1: A Document View on the V-Model

System development with respect to this regime results in a highly interconnected collection of design documents, certification documents, code, formal specifications, and formal proofs. This collection of documents "**SAMSDocs**" [31] make up the basis of a case study in the context of the FormalSafe project [9] at DFKI Bremen, where they serve as a basis for research on machine-supported change management, information retrieval, and document interaction. In this paper, we report on the formalization project of the collection of LaTeX documents which were created in SAMS (that we will also abbreviate with SAMSDocs).

Not surprisingly, the interplay between the fields Semantic Web and Human-Computer Interaction played an important role as the "Authoring Problem" of the first is often tackled via methods of the second. One such approach is that of "*invasive technology*" [17] with the basic idea that from a user's perspective, semantic authoring and general editing are the same, so why not offer semantic functionalities as an extension of well-known editing systems, thereby 'invading' the existent ones. Therefore, we started out with an invasive

OMDoc frontend for LaTeX documents called **sTeX** [22]. Notably, it evolved in a participatory design process in our project into an invasive formalization tool. We started with LaTeX not only because a good portion of our case study was written in it, but also as LaTeX constitutes the state-of-the art authoring solution for many scientific/technical/mathematical document collections. Despite its text-based nature it is the most efficient tool for the task.

In section 2, we will present the sTeX system, especially its realization of Linked Data creation. Then we describe in section 3 the formalization process of `SAMSDocs` with sTeX, our challenges, and our solutions. In section 4 we report the enhancements of sTeX realized in and for the case study. Having sTeX documents with Linked Data and ontological markup, we describe (potential) services and their implementation design in section 5. Section 6 concludes the paper.

2. sTeX: OBJ.-ORIENTED LaTeX MARKUP

sTeX [22, 33] is a variant of LaTeX that is geared towards marking up the semantic structure underlying a document. The main concept in sTeX is that of a "**semantic macro**", i. e., a TeX command sequence $\mathcal{S}$ that represents a meaningful (mathematical) concept $\mathcal{C}$: the TeX formatter will expand $\mathcal{S}$ to the presentation of $\mathcal{C}$. For instance, the command sequence `\positiveReals` (from listing 1) is a semantic macro that represents a mathematical symbol — the set $\mathbb{R}^+$ of positive real numbers. While the use of semantic macros is generally considered a good markup practice for scientific documents (e. g., because they allow to adapt notation by macro redefinition and thus increase reusability), regular TeX/LaTeX does not offer any infrastructural support for this. sTeX does just this by adopting a semantic, 'object-oriented' approach to semantic macros by grouping them into "modules", which are linked by an "imports" relation. To get a better intuition, consider

Listing 1: An sTeX module for Real Numbers

```
\begin{module}[id=reals]
  \importmodule[../background/sets]{sets}
  \symdef{Reals}{\mathbb{R}}
  \symdef{greater}[2]{#1>#2}
  \symdef{positiveReals}{\Reals^+}
  \begin{definition}[id=posreals.def,
   title=Positive Real Numbers]
    $\defeq\positiveReals
          {\setst{\inset{x}\Reals}{\greater{x}0}}$
  \end{definition}
  ...
\end{module}
```

which would be formatted to

Definition 2.1 (Positive Real Numbers): $\mathbb{R}^+ := \{x \in \mathbb{R} \mid x > 0\}$

Here, sTeX's `\symdef` macro generates a respective semantic macro, for instance the `\positiveReals` with representation $\mathbb{R}^+$. Note that the markup in the module `reals` has access to semantic macros `\setst` ("set such that") and `\inset` (element-hood) from the module `sets` that was imported by the document `\importmodule` directive from the `../background/sets.tex`. Furthermore, it has access to the `\defeq` (definitional equality) that was in turn imported by the module `sets`.

From this example we can already see an organizational advantage of sTeX over LaTeX: we can define the (semantic) macros close to where the corresponding concepts are defined, and we can (recursively) import mathematical modules. But the main advantage of markup in sTeX is that it can be transformed to XML via the LaTeXML system [28]: Listing 2 shows the OMDoc [21] representation generated from the sTeX sources in listing 1. **OMDoc** is a content-oriented representation format for mathematical knowledge that extends the formula markup formats OpenMath [5] and MathML [2].

Listing 2: An XML Version of Listing 1

```
<theory xml:id="reals">
 <imports from="../background/sets.omdoc#sets"/>
 <symbol xml:id="Reals"/>
 <notation>
  <prototype><OMS cd="reals" name="Reals"/></prototype>
   <rendering><m:mo>ℝ</m:mo></rendering>
 </notation>
  <symbol xml:id="greater"/><notation>...</notation>
  <symbol xml:id="positiveReals"/><notation>...</notation>
 <definition xml:id="posreals.def" for="positiveReals">
   <meta property="dc:title">Positive Real Numbers</meta>
   <OMOBJ>
     <OMA>
       <OMS cd="mathtalk" name="defeq"/>
       <OMS cd="reals" name="positiveReals"/>
       <OMA>
         <OMS cd="sets" name="setst"/>
         <OMA>
          <OMS cd="sets" name="inset"/>
          <OMV name="x"/>
          <OMS cd="reals" name="reals"/>
         </OMA>
         <OMA>
           <OMS cd="reals" name="greater"/>
           <OMV name="x"/>
           <OMI>0</OMI>
         </OMA>
       </OMA>
     </OMA>
   </OMOBJ>
 </definition>
  ...
</theory>
```

One thing that jumps out from the XML in this listing is that it incorporates all the information from the sTeX markup that was invisible in the PDF produced by formatting it with TeX.

OMDoc itself has been used as a storage and exchange format for automated theorem provers, software verification systems, e-learning software, and other applications [21, chapter 26], but due to its focus on semantic structures, it is not intended to be consumed by human readers. The Java-based JOMDoc [15] library uses the `notation` elements to generate human-readable XHTML+MathML from OMDoc. Figure 2 shows the result of rendering the document from listing 2 in a MathML-aware browser. In contrast to the PDF output we can directly create from sTeX, XHTML+MathML allows for interactivity. In particular, our JOBAD JavaScript framework enables modular services, which utilize the semantic structure of the mathematical formulae [11]. In our rendered documents, each formula in human-readable Presentation MathML carries the original semantic OpenMath representation of the formula, as shown in listing 2, as a hidden annotation.

We have implemented client-side JOBAD services for folding and unfolding subterms of formulae and for control-

ling the display of redundant brackets in complex formulae, which exclusively rely on the annotations inside the document. The symbol definition lookup service, shown in figure 2, interacts with a server backend: It traverses the links to `symbol` and their corresponding `definition` elements that are established by the `OMS` elements in OpenMath – for example, `<OMS cd="sets" name="inset"/>` encodes the URI `../background/sets.omdoc#inset` – and retrieves the document at that URI as XHTML+MathML.[1] JOBAD's ability to integrate an arbitrary number of services, which can talk to different server backends and which are enabled depending on the context, i. e., the semantic structure of the part of a mathematical formula that the user has selected, turns our rendered mathematical documents into powerful mashups [24]. On any symbol, for example, definition lookup is enabled. On any expression where a number is multiplied with a special symbol representing a physical unit, a unit conversion client that talks to a remote unit conversion web service is enabled. The JOBAD architecture has been designed without depending on a particular backend; for most of our services we are using the extensible XML-aware database TNTBase [35, 36, 8], which has special support for OMDoc and integrates the JOMDoc rendering library.

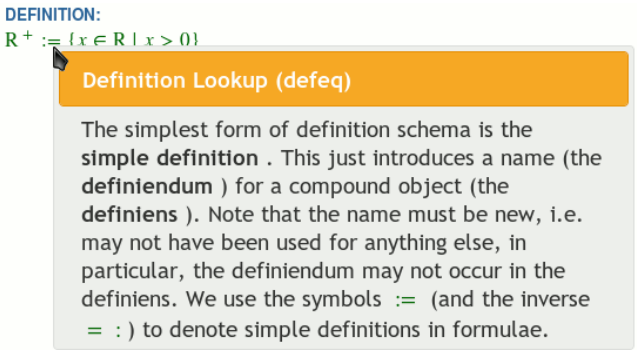

Figure 2: Listing 1 as Dynamic XHTML+MathML

3. FORMALIZATION WITH STEX

In this section we describe the process of formalizing the `SAMSDocs` collection of LaTeX documents created in the course of the `SAMS` project with the STEX system. We use the user's perspective to point to the requirements for STEX that evolved in this process.

As we all know all too well: Formalizing is never easily done. In our project we had the additional challenge of doing it without corruption of the PDF layout that was produced with LaTeX. Here, STEX fits well as it generates PDF and transforms to XML. In figure 3 we can see the general course of action:

i) we identified document fragments ("**objects**") that constitute a coherent, meaningful unit like the state of a document "vg." or its description "zur Prüfung vorgelegt [submitted for certification]", then

ii) we translated it into the STEX format, realizing for example that "vg." is a recurring symbol and "zur Prüfung vorgelegt" its definition (therefore designing the `SAMSDocs` macro "SDdef"), and finally

iii) we polished these macros in the STEX specific sty-files so that the PDF layout remained as before and the XML represented the intended logical structure, for instance the use of the XML nodes "symbol" and "definition".

Note that definitions are common objects in mathematical documents, therefore STEX naturally provides a `definition` environment. So why didn't we use that? Because the OMDoc document model underlying the STEX→LaTeXML transformation does not allow definitions in tables as these are stand-alone objects from an ontological perspective. If one *authors* a formal document, this view is taken, so no problem arises, but if one *formalizes* an existing document, layout and cognitive side-conditions have to be taken into account. We therefore realized that we could not simply add basic STEX markup to the LaTeX source yielding formal objects, we rather needed to add pre-formal markup in the formalization process (we speak of **(semantic) preloading**).

Whenever project-wide layout schemes were discovered, that were frequently used, we extended the macro set of STEX suitably (enabling preloading the *project structure*). The table layout for example was often used for lists of symbol definitions. So we created the `SDTab-def` environment which can host as many `SDdef` commands as wanted. This helped the efficiency of the formalizing process tremendously.

Another difference between authoring and semantic preloading consisted in the *order of the formalization steps*. While the order of the first typically consists of "**chunking**" (i. e., building up structure e. g. by setting up theories), "**spotting**" (i. e., coining objects), and "**relating**" (i. e., explicating relationships between objects or structures), the order of the second is made up of spotting, then relating *or* chunking. The last two were done simultaneously, because STEX offers a very handy inheritance scheme for symbol macros — as long as the chunks are in order, which could be sensibly done for some but not for all at this stage in the formalization process. Generally, many 'guiding' services of STEX, that STEX considered to be features, turned out to be too rigid.

As a consequence we heavily used very light annotations at the beginning: It was sufficient to identify a certain document fragment and to mark it with a referencable ID like "zustand-doc-vg". Shortly afterwards, we realized that some more basic markup was necessary, since we wanted to explicate our knowledge of types/categories of these objects and their conceptual belonging. For this we developed a set of "**ad-hoc semantification macros**" with named attributes like `SDobject[id]`, `SDmore[id, cat, for]`, `SDisa[id,cat,for,follows,theory,imports,tab]`, or `SDreferences[id, file,refid]`[2]. The 'more' functionality provided by `SDmore` was required due to logically contiguous objects that were interspersed in a document. With this set we preloaded the *document structure* (which is

[1]This is the MathML way of representing Linked Data. In section 5, we will see how this feature has been extended to RDFa Linked Data.

[2]We use subsets of a general attributes set for all of our STEX extensions to lower the learning curve for the use of the markup macros.

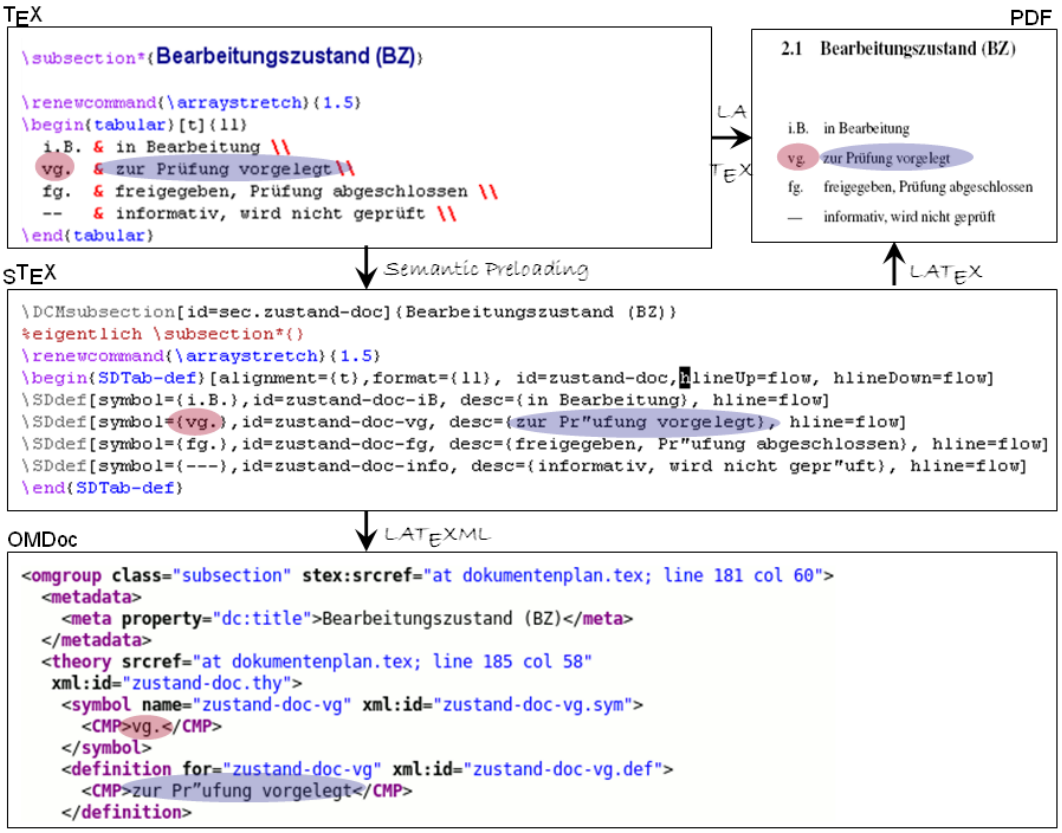

Figure 3: The Formalization Workflow via sTEX: Definition Table of "document state"

quite different from the layout structure, e. g. by subsections that is supported by sTEX core features, see `DCMsubsection` in figure 3). Note that the ad-hoc semantification macros enabled the formalizer to develop her own metadata vocabulary.

As soon as the document boundaries went down, we realized that an object had many occurrences in several of the documents in the `SAMSDocs` collection. For example, first an object was introduced as a high-level concept in the contract, then it was specified in another document, refined in a detailed specification, implemented in the code, reviewed at some stage, and so on until it was finally described in the manual. Thus, we had to preload the *collection structure* as well, which consisted in the development process model, the V-model as seen in figure 1. Here, we built our personal **V-model macros**, e. g. `SemVMrefines`, `SemVMimplements`, or `SemVMdescribesUse`.

Additionally, we created an sTEX extension especially suited for preloading the *organizational structure*. This is considered different from the project structure as organizational markup is very probable to be reusable for other projects with the same organizational structure. For example, `SAMS` used a document version management as well as a document review history, so that environments `VMchangelist`, `VMcertification` with corresponding list entry macros `VMchange`, `VMcertified` were built. Another example is the processing state of a document, which can be marked up

Figure 4: Referencing a "document state"

easily by using the macro `VMdocstate` as seen in figure 4.

We noted that the necessary formalization depth of some documents was naturally deeper than others. For example, it didn't seem sensible to formalize the contract too much as it was created as a high-level communication document, whereas the detailed specification needed a lot of formalization. The manual had an interesting mixed state of formality and informality, as it was again geared towards communication, but it needs to be very precise.

In conclusion we note that the mathematical content of the documents (i. e., the mathematical objects and their relations) was only one of the knowledge sources that needed to

be formalized and marked up. All the arrows in figure 1 are examples of relations between document fragments in the SAMSDocs corpus that needed to be made explicit. For situations like these, we had added RDFa [1] as a flexible metadata framework to the OMDoc format [27]. In the course of the case study, the RDFa integration was revised and extended and will be part of the upcoming OMDoc version 1.3 [23]. The main idea for this integration is to realize that any concrete document markup format can only treat a certain set of objects and their relations via its respective native markup infrastructure. All other objects and relations can be added via RDFa annotations to the host language, if the latter is XML-based. It is crucial to realize that for machine support, the metadata objects and relations are given a machine-processable meaning via suitable ontologies. Moreover, ontologies are just special cases of (mathematical) theories, which import appropriate theories for the logical background, e. g. description logic, and whose symbols are the entities (class, properties, individuals) of ontologies. Thus, sTeX and OMDoc can play a dual role for linked data in documents with mathematical content. They can be used as markup formats for the documents and at the same time as the markup formats for the ontologies. We have explored this correspondence for OMDoc in previous work and implemented a translation between OMDoc and OWL [27, 26]; the design and extension of the sTeX frontend is one of the contributions of this paper.

4. A METADATA-EXTENSION OF sTeX

To understand our contribution note that we can view LaTeX and sTeX as frameworks for defining domain-specific vocabularies in classes and packages; LaTeX is used for layout aspects, and sTeX can additionally handle the semantic aspects of the vocabularies. sTeX uses this approach to define special markup e. g. for definitions (see lines 10 and 31 in listing 2). Note that to define sTeX markup functionality like the `definition` environment, we have to provide a LaTeX environment definition (so that the formatting via LaTeX works) and a LaTeXML binding (to specify the XML transformation for the `definition` environment). As the OMDoc vocabulary is finite and fixed, sTeX can (and does) supply special LaTeX macros and environments and their LaTeXML bindings. But the situation is different for the flexible, RDFa-based metadata extension in OMDoc 1.3 we mentioned above. At the start of the SAMSDocs preloading effort, sTeX already supported a common subset of metadata vocabularies. For instance the Dublin Core `title` element in line 11 of listing 2 is the transformation result of using the KeyVal [7] pair `title=...` in the optional argument of the `definition` environment.

For the SAMSDocs case study we started in the same way by adding a package with LaTeXML binding to sTeX. The `\VMdocstate` macro shown in the "sTeX" box of figure 4 allowed us to annotate a document with its processing state. This is transformed to an RDFa-annotated `omdoc` root element, as shown in the "OMDoc" box underneath and in the black, solid parts of the RDF graph in figure 5. We can already see that the sTeX extension for SAMSDocs exactly consists in a domain-specific metadata vocabulary extension, and that using the custom vocabulary hides markup complexity from the author. Again, SAMSDocs only needed a finite vocabulary extension, so this approach was feasible, but of restricted applicability, since developing the SAMSDocs package for sTeX involves insights into sTeX internals and LaTeXML bindings. Thus this extension approach lacks flexible user-extensibility that is needed to scale up.

To enable user-extensibility, we add a new declaration form `\keydef` to the core sTeX functionality — like `\symdef` in that it is inherited via the "imports" relation, only that it defines a KeyVal key instead of a semantic macro. To understand its application, we rationally reconstruct the `v:hasState` relation from the example in the OMDoc box of figure 4. To do this, we use sTeX to create a metadata vocabulary for document states: we create a module `certification` which defines the `hasState` metadata relation and adds it to the KeyVal keys of the `document` environment. The `metalanguage` macro is a variant of `importmodule` that imports the meta language, i. e., the language in which the meaning of the new symbols is expressed; here we use OWL.

Listing 3: A Metadata Ontology for Certification

```
\begin{module}[id=certification]
 \metalanguage[../background/owl]{owl}
 \keydef{document}{hasState}
 \symdef{zustand-doc-vg}[1]{vg. #1}
\begin{definition}[for=hasState]
 A document {\definiendum[hasState]{has state}} x, iff
 the project manager decrees it so.
 \end{definition}
 \begin{definition}[for=zustand-doc-vg]
A document has state \definiendum[zustand-doc-vg]{vg. x},
 iff it has been submitted to x for certification.
 \end{definition}
\end{module}
```

So, if we import the `certification` metadata module, we can write

```
\importmodule[../ontologies/cert]{certification}
\begin{document}[hasState=zustand-doc-vg]
...
\end{document}
```

to generate RDFa annotations that correspond to the red dotted arrow in figure 5. Note that in the state of formalization shown in figure 4, the SAMSDocs-specific RDF vocabulary still has a pre-semantic structure. With the sTeX extension we can express that the processing state is actually intended to be a symbol in a metadata theory, not just some semantic object in some file. In listing 3 we use the `\symdef` directive to generate the symbol `zustand-doc-vg` and `\keydef` to generate a metadata relation `hasState` that is expressed by a key of the same name, which is added to the `document` environment. When processed by LaTeXML, `\keydef` takes care of generating correct URIs for the metadata relations and their target resources, resulting in an RDFa output syntactically similar to figure 4. In conclusion we note that the sTeX extension allows us to rationally recreate the effect we previously achieved with the custom `\VMdocstate` and `\SDreferences NoObj` macros. Note that we did not have to extend the LaTeXML bindings at all for this extension.

5. sTeX DOCUMENTS AS LINKED DATA

The translation of classical sTeX to OMDoc and further to XHTML+MathML (see section 2) enables interactive services for mathematical structures. Now, that sTeX supports formalization with arbitrary metadata (cf. section 4),

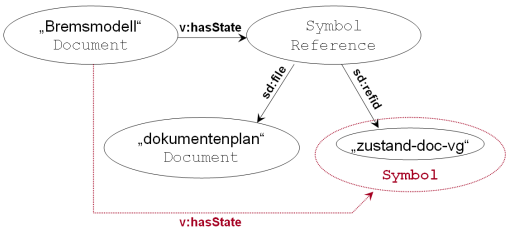

Figure 5: RDF View on a “doc. state” Assignment

it should also be possible to utilize *these* metadata for services. The JOBAD service architecture (see section 2) gives uniform access to common queries in the document browsing user interface. In the SAMSDocs scenario this might be a query for all persons who have worked on the current document. This can directly be answered from the metadata of the revision log. Another typical query would consist in asking for all parts of a specification that have to be re-certified. Answering this query involves revision logs (for finding documents that have changed since the last certification), the collection structure (V-model dependencies of changed parts), and the mathematical structure (logical dependencies). In [18] we have elaborated on such SAMSDocs queries from the point of view of their stakeholders (like engineers, project managers, certifiers), particularly exploring the multi-dimensionality of the formal structures. Here, we will summarize the extensions made to our system architecture to enable these services.

As a first step, we made the JOMDoc renderer preserve the RDFa metadata from the OMDoc documents, now generating XHTML+MathML+RDFa. Additionally, the mathematical structures had to be preserved in the rendered output. Even though OMDoc uses native non-RDFa markup for these structures, exploiting the OMDoc ontology we can transform it into RDF form (see [25, 8] for more information). Existing JOBAD services recognized mathematical formulae in XHTML presentations of OMDoc documents by their semantic structure (e. g. whether they use previously defined symbols or physical units). Similarly, new services can recognize whether a chunk of an XHTML document is, e. g., an implementation of a specification fragment, and by which user requirement that is induced. Compared to the previously existing definition lookup service, the principle of retrieving content from a target URI and displaying it in a popup remained the same, the URIs are just provided by different annotations.

Secondly, we have extended the folding of subterms of mathematical formulae to higher-level structures, such as requirements, code fragments, or steps of structured proofs. We have implemented this using the rdfQuery JavaScript library [34], which parses all RDFa annotations of a document into a local triple store that can be queried using SPARQL-like JavaScript functions. On the server side, we have extended TNTBase [35], our versioned database backend and web server/application framework to accept commits of sTEX documents, automatically convert them to OMDoc, and then serve OMDoc, XHTML+MathML+RDFa, and, optionally, RDF/XML, according to the Linked Data best practices [14].

Even the pre-semantic annotations like the ones shown in figure 4 afford interactive services: A generic reference can already be utilized for lookup and navigation. Providing additional information in the instance document or in the ontology (e. g. the knowledge about the target of a reference being a symbol or a processing state) allows for making the service user interface more specific and enables the display of more relevant related information. For the generic pre-semantic “references” relation, it does not make sense to display a list of all semantic objects that it relates to each other, as that list would be large and there would be no obvious way of ranking or filtering it. But once more specific link types are used, such as the “has state” link, that information can be used to display a list of documents grouped by state.

Queries across documents cannot be answered using the above-mentioned rdfQuery: client side queries require a combination of querying a local triple store and crawling links. In our setup, we have experimented with SQUIN [13], a frontend to the Semantic Web Client library [4], which gives access to Linked Data via a simple HTTP frontend at very low integration costs: If the server provides standard-compliant Linked Data, then the client simply has to access the URL of the SQUIN server, providing a SPARQL query as a parameter. An alternative would have been AJAR library, a part of the Tabulator Linked Data browser [3], which implements the same functionality in JavaScript. In our setup, SQUIN acts as a proxy between the client-side JavaScript code and our Linked Data. While a Linked Data crawler is most flexible when data are distributed across many servers (e. g. when an OMDoc document links to DBpedia), its query answering capabilities are only as good as the Linked Data being served. For example, if the RDF(a) does not contain backlinks (like links from a mathematical theory to the theories it imports *and* to the theories by which it is imported), then an AJAR- or SQUIN-powered client cannot query links in both directions. Moreover, the performance of such a solution is limited, as it requires memory for the local triple store as well processor time for query answering on the client side. Therefore, in the SAMSDocs setting, where the queries are currently limited to a document collection on a single server, the best solution is storing the triples on that same server, and making them accessible via a standard query interface. Concretely, we make a SPARQL endpoint available as an extension to TNTBase [8]. In a larger Software Engineering scenario (like a document collection of a company with multiple departments) a combination with a Linked Data crawler may have advantages: if all these departments publish their document collections as Linked Data in the company intranet (see for instance [32] for the actuality of this example), crawling these may reveal previously unknown connections, e. g. colleagues dealing with structurally similar problems who could lend advice. Note that local vocabularies resulting from ad-hoc semantification need not be a barrier to knowledge exchange: Linked Data practices recommend connecting occurrences of semantically equivalent resources in different data sets by *owl:sameAs*. Alternatively, if it turns out that one department uses a “better” vocabulary for their data, the sTEX metadata extensions make it easy to adopt it: all we have to do is to change the sTEX bindings or \keydefs.

6. CONCLUSION AND FUTURE WORK

We reported on a formalization case study, where we use the sTeX format, a document formatting system and specification platform for semantic, mathematical vocabularies, on a document corpus from Software Engineering. To cope with the the multi-dimensional semantic structure implicit in the document collection, we extended sTeX into a markup platform for semi-formal ontologies and Linked Data (in our case semi-formal documents with RDFa-based metadata annotations).

The work reported here is related to, but different from the SALT format [12], which also allows to embed Linked Data relations into LaTeX. SALT embeds a fixed set of rhetorical and bibliographical relations as annotations in the generated PDF, whereas sTeX allows a flexible set of relations, which it only embeds as RDFa into the generated OMDoc and XHTML+MathML. We have concentrated on the XHTML+MathML target, since it supports dynamic interactions via our JOBAD system. An export of the metadata relations to PDF should be possible with the technology employed in SALT, we leave this to future work.

The key observation from our case study is that if we use sTeX as a human- and document-oriented frontend for Linked Data documents, we can approach the formalization of semi-formal document collections as a process of "*document and ontology co-development*", where (in our case pre-existing) documents are semantically preloaded with inter- and intra-document relations, whose meaning is given by (project-specific or general, reusable) metadata ontologies. As we have seen in section 3, preloading documents and developing metadata ontologies in a joint frontend format reduces formalization barriers; for instance, we often have to elaborate informal document fragments into the metadata ontologies; see the discussion about "vg.".

For practical applicability of the sTeX-based approach, machine support for authoring and managing sTeX document collections is crucial. As a client-side counterpart to the integrated repository and Linked Data publishing solution provided by TNTBase [8], we are currently developing an integrated collection authoring environment sTeXIDE for sTeX on the basis of the Eclipse framework [16]. We expect that extending sTeXIDE to operationalize the new sTeX functionality presented in this paper will turn it into an IDE for document collection and ontology co-development that will enable authors to cope with the complexities of dealing with large collections of semi-formalized documents. On the other hand, we expect the modular sTeXIDE system to be a good basis for deploying supportive services in a flexible document collection environment.

We conjecture that the sTeX based workflow for document and ontology co-development can be extended to arbitrary Linked Data applications. We are currently working on two sTeX extensions:

Flexible Metadata for LaTeX Documents Our experimental `rdfameta` package [20] redefines common LaTeX commands (e. g. the sectioning macros) so that they include optional KeyVal arguments that can be extended by `\keydef` commands. With this extension, we can add RDFa metadata to any existing LaTeX document and generate linked data (XHTML+RDFa documents) via the LaTeXML translator. Currently the coverage of the `rdfmeta` package is minimal; we will extend this in the future.

Generating OWL-XML from sTeX In the original sTeX workflow, we use the OMDoc format for representing ontologies. This allows a deeper mathematical modeling and documentation than OWL, which is widely used on the Semantic Web. Even though we have a working translation of OMDoc ontologies to OWL (encoded as RDF/XML)[27], a direct sTeX to OWL transformation would be nice. Simply using our experimental `owl2onto` class [19] instead of the `omdoc` class from sTeX in the LaTeX preamble will cause LaTeXML to generate OWL – here in the direct OWL XML serialization – instead of OMDoc for documents with restricted markup.

Acknowledgments. The authors gratefully acknowledge the careful work of Christoph Lüth, Holger Täubig, and Dennis Walter that went into preparing the SAMS document collection, which is the basis of this paper. Moreover, we like to thank the members of the FormalSafe project for valuable discussions.

7. REFERENCES

[1] B. Adida and M. Birbeck. RDFa Primer. W3C Working Group Note, World Wide Web Consortium (W3C), Oct. 2008.

[2] R. Ausbrooks, S. Buswell, D. Carlisle, G. Chavchanidze, S. Dalmas, S. Devitt, A. Diaz, S. Dooley, R. Hunter, P. Ion, M. Kohlhase, A. Lazrek, P. Libbrecht, B. Miller, R. Miner, M. Sargent, B. Smith, N. Soiffer, R. Sutor, and S. Watt. Mathematical Markup Language (MathML) version 3.0. W3C Candidate Recommendation of 15 December 2009, World Wide Web Consortium, 2009.

[3] T. Berners-Lee, Y. Chen, L. Chilton, D. Connolly, R. Dhanaraj, J. Hollenbach, A. Lerer, and D. Sheets. Tabulator: Exploring and analyzing linked data on the semantic web. In *Proceedings of the The 3rd International Semantic Web User Interaction Workshop (SWUI06)*, Nov. 2006.

[4] C. Bizer, T. Gauß, R. Cyganiak, and O. Hartig. Semantic web client library. `http://www4.wiwiss.fu-berlin.de/bizer/ng4j/semwebclient/`, seen Feb. 2010.

[5] S. Buswell, O. Caprotti, D. P. Carlisle, M. C. Dewar, M. Gaetano, and M. Kohlhase. The Open Math standard, version 2.0. Technical report, The Open Math Society, 2004.

[6] J. Carette, L. Dixon, C. Sacerdoti Coen, and S. M. Watt, editors. *MKM/Calculemus 2009 Proceedings*, number 5625 in LNAI. Springer Verlag, July 2009.

[7] D. Carlisle. *The* keyval *package*. The Comprehensive TeX Archive Network, 1999. Part of the TeX distribution.

[8] C. David, M. Kohlhase, C. Lange, F. Rabe, N. Zhiltsov, and V. Zholudev. Publishing math lecture notes as linked data. In L. Aroyo, G. Antoniou, and E. Hyvönen, editors, *ESWC*, Lecture Notes in Computer Science. Springer, June 2010. In press.

[9] FormalSafe. http://www.dfki.de/sks/formalsafe/, seen Dec. 2008.

[10] U. Frese, D. Hausmann, C. Lüth, H. Täubig, and D. Walter. The importance of being formal. In H. Hungar, editor, *International Workshop on the Certification of Safety-Critical Software Controlled Systems SafeCert'08*, volume 238 of *Electronic Notes in Theoretical Computer Science*, pages 57–70, Sept. 2008.

[11] J. Giceva, C. Lange, and F. Rabe. Integrating web services into active mathematical documents. In Carette et al. [6], pages 279–293.

[12] T. Groza, S. Handschuh, K. Möller, and S. Decker. SALT – semantically annotated LaTeX for scientific publications. In E. Franconi, M. Kifer, and W. May, editors, *ESWC*, number 4519 in Lecture Notes in Computer Science, pages 518–532. Springer, 2007.

[13] O. Hartig and J. Sequeda. SQUIN – query the web of linked data. http://squin.sourceforge.net, seen Feb. 2010.

[14] T. Heath et al. Linked data – connect distributed data across the web – guides and tutorials. http://linkeddata.org/guides-and-tutorials, seen Feb. 2010.

[15] JOMDoc project — Java library for OMDoc documents. http://jomdoc.omdoc.org, 2010. seen Feb.

[16] C. Jucovschi and M. Kohlhase. sTeXIDE: An integrated development environment for sTeX collections. submitted to MKM (Mathematical Knowledge Management) 2010, 2010.

[17] A. Kohlhase. Overcoming Proprietary Hurdles: CPoint as Invasive Editor. In F. de Vries, G. Attwell, R. Elferink, and A. Tödt, editors, *Open Source for Education in Europe: Research and Practise*, pages 51–56, Heerlen, The Netherlands, Nov. 2005. Open Universiteit Nederland, Open Universiteit Nederland. Proceedings at http://hdl.handle.net/1820/483.

[18] A. Kohlhase, M. Kohlhase, and C. Lange. Dimensions of formality: A case study for MKM in software engineering. submitted to MKM (Mathematical Knowledge Management) 2010, 2010.

[19] M. Kohlhase. owl2onto.cls: Marking up OWL2 Ontologies in sTeX. https://svn.kwarc.info/repos/stex/trunk/sty/owl2onto/owl2onto.pdf.

[20] M. Kohlhase. RDFa metadata in LaTeX. https://svn.kwarc.info/repos/stex/trunk/sty/rdfmeta/rdfmeta.pdf.

[21] M. Kohlhase. OMDoc – *An open markup format for mathematical documents [Version 1.2]*. Number 4180 in LNAI. Springer Verlag, Aug. 2006.

[22] M. Kohlhase. Using LaTeX as a semantic markup format. *Mathematics in Computer Science*, 2(2):279–304, 2008.

[23] M. Kohlhase. An open markup format for mathematical documents OMDoc [version 1.3]. Draft Specification, 2010.

[24] M. Kohlhase, J. Giceva, C. Lange, and V. Zholudev. JOBAD – interactive mathematical documents. In B. Endres-Niggemeyer, V. Zacharias, and P. Hitzler, editors, *AI Mashup Challenge 2009, KI Conference*, Sept. 2009.

[25] C. Lange. The OMDoc document ontology. web page at http://kwarc.info/projects/docOnto/omdoc.html, seen August 2008.

[26] C. Lange. *Semantic Web Collaboration on Semiformal Mathematical Knowledge*. PhD thesis, Jacobs University Bremen, 2010. submission expected in spring 2010.

[27] C. Lange and M. Kohlhase. A mathematical approach to ontology authoring and documentation. In Carette et al. [6], pages 389–404.

[28] B. Miller. LaTeXML: A LaTeX to XML converter. Web Manual at http://dlmf.nist.gov/LaTeXML/, seen March 2010.

[29] *MKM 2010*, 2010. submitted to MKM (Mathematical Knowledge Management) 2010.

[30] T. Nipkow, L. C. Paulson, and M. Wenzel. *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*. Number 2283 in LNCS. Springer, 2002.

[31] SAMS. SAMSDocs: The document collection of the SAMS project, 2009. http://www.sams-projekt.de.

[32] F.-P. Servant. Linking enterprise data. In C. Bizer, T. Heath, K. Idehen, and T. Berners-Lee, editors, *Linked Data on the Web (LDOW 2008)*, number 369 in CEUR Workshop Proceedings, Apr. 2008.

[33] Semantic Markup for LaTeX, seen July 2009. available at http://kwarc.info/projects/stex/.

[34] J. Tennison et al. rdfQuery – RDF processing in your browser. http://code.google.com/p/rdfquery/, seen Feb. 2010.

[35] V. Zholudev and M. Kohlhase. TNTBase: a versioned storage for XML. In *Proceedings of Balisage: The Markup Conference 2009*, Balisage Series on Markup Technologies. Mulberry Technologies, Inc., 2009. available at http://kwarc.info/vzholudev/pubs/balisage.pdf.

[36] V. Zholudev, M. Kohlhase, and F. Rabe. A [insert xml format] database for [insert cool application]. In *Proceedings of XML Prague 2010*, 2010.