REPORT ON OpenDreamKit DELIVERABLE D4.3
Distributed, Collaborative, Versioned Editing of Active Documents in MathHub.info

MICHAEL KOHLHASE

Due on	02/31/2017 (Month 18)
Delivered on	27/06/2017
Lead	Friedrich-Alexander Universität Erlangen/Nürnberg (FAU)
Progress on and finalization of this deliverable has been tracked publicly at: `https://github.com/OpenDreamKit/OpenDreamKit/issues/92`	

DELIVERABLE DESCRIPTION, AS TAKEN FROM GITHUB ISSUE #92 ON 2017-01-11

- **WP4:** User Interfaces
- **Lead Institution:** Jacobs University Bremen
- **Due:** 2016-08-31 (month 12) → delay to be negotiated to month 18
- **Nature:** Report
- **Task:** T4.7(#75), (#75) T4.8(#76) (#76)
- **Proposal:** p. 47
- **Final report**

CONTENTS

1. Introduction

MathHub is a portal for active mathematical documents ranging from formal libraries of theorem provers to informal – but rigorous – mathematical documents lightly marked up by preserving LaTeX markup. In the OpenDreamKit project MathHub acts as

- a portal and management system for theory-graph structured active documents, i.e. documents that use the semantic structure of the document and the knowledge context it links to to render semantic services embedded in the document – it becomes *active*, i.e. interactive, reactive – see [**ODK-D4.3**], self-adapting – see [**ODK-D4.2**].
- the repository for the Math-in-the-Middle (MitM) ontology (see [**DehKohKon:iop16**; **ODK-D6.2**]). This ontology is used as a basis for interoperability of the mathematical software systems that make up the OpenDreamKit VRE toolkit, which is a crucial concern for work package **WP6**.

As the authoring, maintenance and curation of theory-structured mathematical ontologies and the transfer of mathematical knowledge via active documents are an important part of the OpenDreamKit VRE toolkit, the editing facilities in MathHub play a great role for the project. This report discusses the main design decisions of the editing facilities in MathHub; they can be assessed at `http://mathhub.info`.

In Section 2 we briefly present the MathHub system as a basis and in Section 3 we drill in on the editing-specific system features. Section 4 concludes the report.

2. The MathHub System/Portal

2.1. System Architecture and Realization

MathHub is realized as an instance of the Planetary System [**Kohlhase:ppte12**], which we have substantially extended in the course of the work reported here. 1

The system architecture has three main components:

i) a versioned *data store* holding the source documents
ii) a *semantic service provider* that imports the source documents and provides services for them
iii) and a *frontend* that makes the sources and the semantic services available to users.

Specifically, we use the GitLab repository manager [**GitLab:on**] as the data store, the MMT API [**Rabe:MAGMS13**; **uniformal:on**] as the semantic service provider and build system, and Drupal [**drupal:on**] as the user-pointing frontend. Figure 1 shows the overall architecture of the MathHub system.

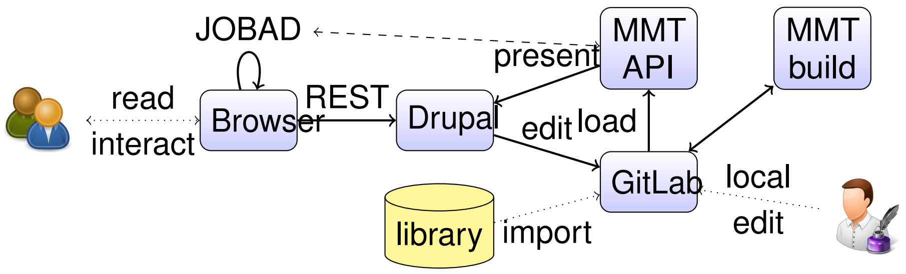


FIGURE 1. The MathHub Architecture

In this setup, Drupal serves as a container management system[1] that supplies uniform theming, user management, discussion forums, etc. GitLab on the other hand, provides versioned storage of the content documents, and organizes them into repositories owned by users and groups (math archives).

The MMT API provides semantic services (notation-based, presentation, definition lookup, relational navigation, dependency management, etc.) to the user, primarily by dedicated JOBAD [**GLR:WebSvcActMathDoc09**] modules. Note that even though the active document functionalities and semantic editing support in MathHub are based on OMDoc/MMT representation of the content, the authors interact with the content in the source format. Both of these representations are versioned in GitLab and are converted into OMDoc/MMT by an MMT-based build system that 2

In order to deal with flexiformal mathematical content in OMDoc, we have also extended the MMT API, which was previously restricted to fully formal content. In the extended MMT API, each MMT service works whenever it is theoretically applicable (e.g. type checking when there exists type information, change management when there is dependency information, etc.).

2.2. Deployment and Content

MathHub is deployed at `http://mathhub.info` and has reached a state where it can be used for the OpenDreamKit project, but has not been scaled yet much beyond 10 000 documents and a couple dozens of users and repositories.

[1]To Do: maybe a brief reminder about "MMT", "flexiformal", ...?

[1]Drupal and similar systems self-describe as content management systems, but they actually only manage the documents without changing their internal structure.

[2]To Do: missing end of sentence

Specifically, we are currently hosting a test set of formal and informal mathematical content to develop and evaluate system functionality; concretely:

i) the SMGloM termbase with ca. 1500 small sTeX files containing definitions of mathematical terminology and notation definitions.

ii) ca. 6500 files with sTeX-encoded teaching materials (slides, course notes, problems, and solutions) in Computer Science,

iii) the LATIN logic atlas with ca. 1000 meta-theories and logic morphisms,

iv) the Mizar Mathematical Library of ca. 1000 articles with ca. 50.000 theorems, definitions, and proofs, and

v) a part of the HOL Light Library with 22 theories and over 2800 declarations.

Already now, it is unique in its class in that it gives a unified interface to multiple theorem prover libraries together with linguistic and educational resources. Now that the ground work has been laid, we anticipate the rapid integration of new semantic services, editing support and new content.

3. Distributed, Collaborative, Versioned Editing

For the OpenDreamKit project, we have developed and are hosting on MathHub

(1) the "Math-in-the-Middle ontology" [**MitM:on**], which hosts a flexiformal development of the mathematical knowledge for system interoperability in OpenDreamKit (see [**DehKohKon:iop16**; **ODK-D6.2**]),

(2) the "ODK System ontologies" [**ODKsysonto:on**], a collection of OMDoc/MMT theories and codecs that describe the mathematical content of (parts of) the LMFDB [**lmfdb:on**], OEIS [**oeis**] and other data sources, bridging the internal (i.e. system/database oriented) and external (mathematical) views of the content.

These two MathHub libraries are in an early state currently and curating them is a major task in OpenDreamKit's WP6: "Data/Knowledge/Software-Bases". Therefore, the editing workflows we report on here are crucial to the OpenDreamKit project.

Given the distributed and collaborative nature of the OpenDreamKit project, the editing facilities need to be as well. As experience in software engineering shows – and flexiformal active documents are like software in many respects – this is only possible using (concepts of) revision control systems. In MathHub we took the major design decision to use the distributed revision control system GIT [**GIT:on**] as the basis for all general storage, authentification, distribution, and collaboration functionalities and build domain-specific functionality on top of that. We took the minor design decision to use the GitLab [**GitLab:on**] system as the repository management system – rather than e.g. GitHub [**GitHub:on**], since it is open source and allows us to run our own server and configure it more by patching the code.

We organize the content into **libraries** by area or intended application – e.g. the two libraries discussed at the top of this section and further divide them up into **math archives** [**HorIacJuc:cscpnrr11**], which standardize the file system layout of all dimensions of mathematical representations: source, content, presentation, narration, ... At the GitLab level, libraries are modeled as "groups" and individual math archives as repositories. As GIT – and thus repository managers like GitLab – only allow authentification at the repository level, math archives are mostly used for authentification and access control in MathHub.

The advantage of the GIT-based setup is that we can combine two methods for accessing the contents of MathHub:

i) an online, web-based editing/interaction workflow for the casual user, in the spirit of the Planetary system and

ii) an offline editing/authoring workflow based on a GIT working copy.

We will describe both separately in sections 3.2 and 3.5, after clarifying the setting in MathHub a bit more.

3.1. Building a Math Knowledge Base by Editing Surface Language Documents

The unifying representation format of the MathHub system is flexiformal OMDoc/MMT, which is structured as a theory graph. As OMDoc/MMT is optimized for machine processing, actual content is authored and edited as documents in a **surface format**: a human-oriented syntax that can be compiled into OMDoc/MMT. MathHub currently supports five surface formats

(1) HTML5 and TEX/LATEX (as a minimally flexiformal document formats)

(2) sTEX (semantic TEX/LATEX), an extension of LATEX that allows to annotate LATEX documents with semantic properties and relations.

(3) MMT surface syntax.

(4) TWELF (Edinburgh Logical Framework in TWELF syntax)

Additionally, the native formats of the theorem prover libraries imported into MathHub are handled by special importers on a system-by-system basis. Only sTeX and MMT syntax are relevant for the OpenDreamKit project, so we will ignore the others in this report. In all cases, the conversion to OMDoc/MMT is a multi-step, multi-document, dependency-driven process, which is handled by the MMT build system [**mmt:buildsys:on**] (see the upper right hand corner in Figure 1). The MathHub system uses a separate build system process that pulls changes from the MathHub GitLab server, re-builds any affected files, and pushes the results to the MathHub GitLab server again. From that the MathHub system can pull the new state of the libraries and update the local MMT API.

3.2. Local MathHub Editing

The local editing workflow is important for power authors who want to edit more than one file simultaneously or have customized modes for the surface languages in their preferred text editors. A user can fork or pull the relevant repositories from the MathHub GitLab, edit them and submit them back to MathHub either via a pull request to the repository masters or a direct commit/push. As the content is usually highly networked and distributed across multiple math archives (and thus GIT repositories), we have developed a command line tool lmh (local MathHub; see [**lmh:on**] and [**lmh-docker:on**] for a docker container that bundles up all system requirements for lmh) that manages working copies across repository borders taking into account e.g. cross-archive dependencies. lmh also supports running the build system locally and previewing HTML5 renderings of the generated OMDoc/MMT.

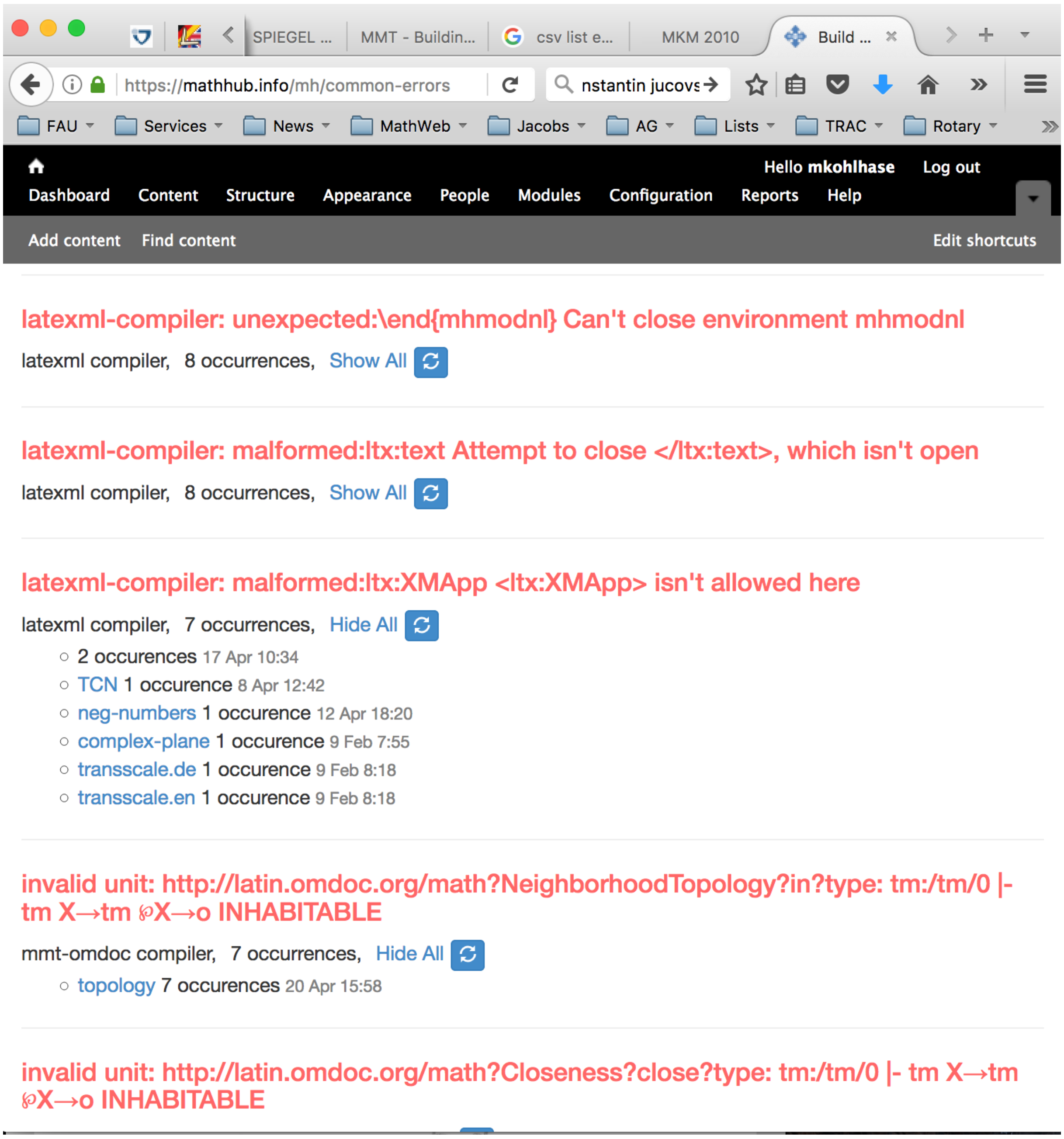
The concrete editing support for flexiformal, active mathematical documents crucially depends of the depth of formalization. Completely informal mathematical documents are traditionally written in LaTeX and the traditional authoring tools are well-suited for this task. Any cross-repository effects can be handled by lmh, so we will discuss more semantic document formats next.

3.3. Local Editing Support for Lightly Formalized Mathematical Content

sTeX [**Kohlhase:ulsmf08**; **sTeX:github:on**] is an extension of the TeX/LaTeX that allows the annotation of semantic properties into the document source. These annotation are largely invisible in the PDF output, but can be transferred into OMDoc/MMT when formatted with the sTeX plugin to LaTeXML [**GinStaKoh:latexmldaemon11**]. We call the process of adding such source annotations **semantic preloading**.

Both formatting pipelines highlight different features of the lifecycle of flexiformal documents. Usually, the content is initially developed with casual preloading – whatever is convenient while authoring – when first writing the original material. This is then formatted to PDF, which is checked and iterated for mathematical and visual correctness. Even though we have a long history of supporting surface language editing in editor-based IDEs [**JucKoh:sidesc10**; **JucEth12:redsys**], the most important practical aspects of knowledge curation in MathHub are well-handled by the lmh tool, which provides archive curation functionalities such as basic link checking, pre-translation, and build system support. An advantage of this approach is that it is largely editor-independent and thus does not constrain the user in her choice of tools.

In a second phase, the authors fully preload the documents semantically. In this phase, the content is built into OMDoc/MMT, loaded into the MMT API and checked for semantic link-consistency. In our experience the most important "semantic editing service" is to track the errors encountered during these semantic checks. While it would be helpful to have editor/IDE integration for the ensuing semantic debugging



latexml-compiler: unexpected:\end{mhmodnl} Can't close environment mhmodnl

latexml compiler, 8 occurrences, Show All

latexml-compiler: malformed:ltx:text Attempt to close </ltx:text>, which isn't open

latexml compiler, 8 occurrences, Show All

latexml-compiler: malformed:ltx:XMApp <ltx:XMApp> isn't allowed here

latexml compiler, 7 occurrences, Hide All

- 2 occurences 17 Apr 10:34
- TCN 1 occurence 8 Apr 12:42
- neg-numbers 1 occurence 12 Apr 18:20
- complex-plane 1 occurence 9 Feb 7:55
- transscale.de 1 occurence 9 Feb 8:18
- transscale.en 1 occurence 9 Feb 8:18

invalid unit: http://latin.omdoc.org/math?NeighborhoodTopology?in?type: tm:/tm/0 |- tm X→tm ℘X→o INHABITABLE

mmt-omdoc compiler, 7 occurrences, Hide All

- topology 7 occurences 20 Apr 15:58

invalid unit: http://latin.omdoc.org/math?Closeness?close?type: tm:/tm/0 |- tm X→tm ℘X→o INHABITABLE

FIGURE 2. The MathHub Error Viewer

process, the main advantage[2] can already be reaped by a special, error aggregating viewer. Figure 2 shows a fragment, where we see source errors (the first two) and error messages probably caused by compiler problems. In both cases, the user can click on a particular occurrence and enter a web-based editing cycle which gives access to the logs and results of the respective transformation steps. Any source errors can directly be fixed using the web editing facilities (see Section 3.5). 3

4

[2]This effect may however be partially caused by the fact that we currently still also have errors in the transformation pipeline, which necessitates full rebuilding of the MathHub libraries interleaved with source debugging.

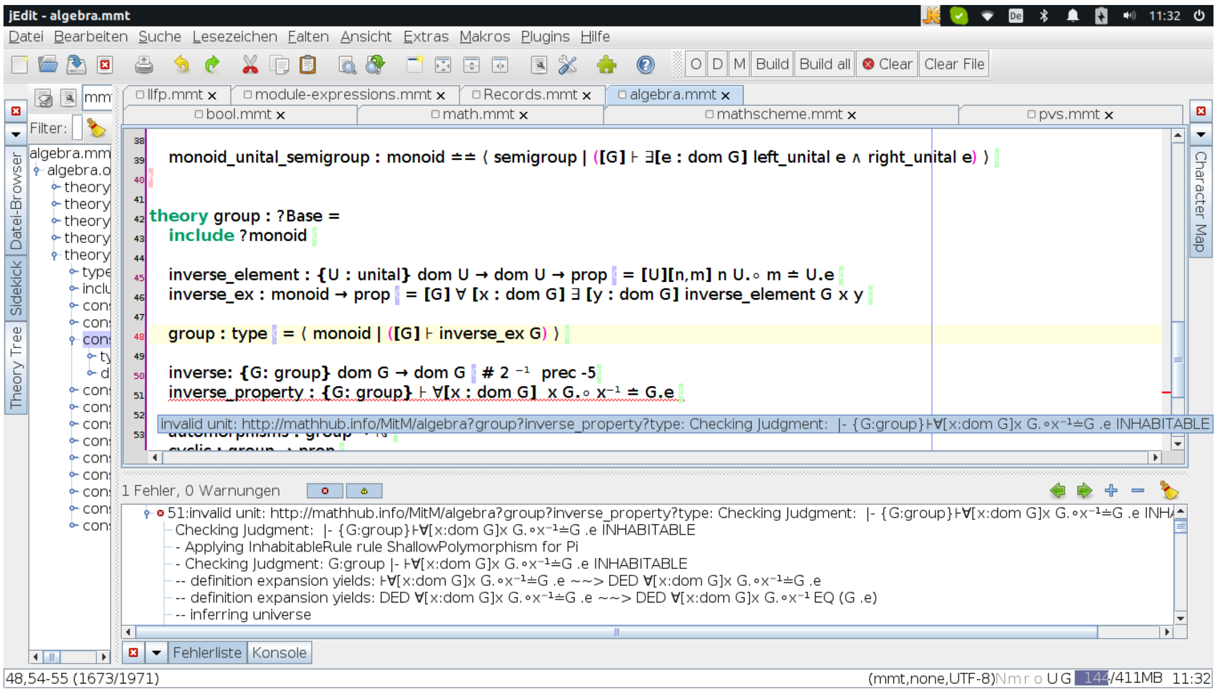
[3]TO DO: There is some repeat in the last two sentences

[4]TO DO: this web-based edition cycle is not quite about local editing support, right? Should it be moved to some other section?

3.4. Local Editing Support for Formal Mathematics

Editing formal mathematical content poses a different set of challenges. As the content is formal (i.e expressed in a formal language with well-defined semantics), its syntax and semantics are fully machine-checkable, and this defines the standard of mathematical quality. Consequently, an editing process that integrates syntax/semantics-checking – which takes the form of type-checking in MMT– is needed for editing formal mathematics.

Figure 3 shows a JEdit-based IDE for OMDoc/MMT content in MMT surface syntax that is tightly coupled with the MMT API for semantic services; see [**Rabe:LII14**] for details. In a nutshell, we can see the structural view of the files in the panel on the left and the source file in the central pane. Both are cross-linked for better structural and source navigation. In this example we also have a type checking error in line 51 of the file `algebra.mmt`. It is indicated by the red underline and the hovering the mouse over it gives the local error message, and the error console at the bottom give additional details.


jEdit - algebra.mmt
Datei Bearbeiten Suche Lesezeichen Falten Ansicht Extras Makros Plugins Hilfe
O D M Build Build all Clear Clear File
llfp.mmt module-expressions.mmt Records.mmt algebra.mmt
bool.mmt math.mmt mathscheme.mmt pvs.mmt
Filter:
Datei-Browser Sidekick Theory Tree
Character Map
1 Fehler, 0 Warnungen
Fehlerliste Konsole

```
39 monoid_unital_semigroup : monoid ≐≐ ⟨ semigroup | ([G] ⊦ ∃[e : dom G] left_unital e ∧ right_unital e) ⟩
40
41
42 theory group : ?Base =
43   include ?monoid
44
45   inverse_element : {U : unital} dom U → dom U → prop = [U][n,m] n U.∘ m ≐ U.e
46   inverse_ex : monoid → prop = [G] ∀ [x : dom G] ∃ [y : dom G] inverse_element G x y
47
48   group : type = ⟨ monoid | ([G] ⊦ inverse_ex G) ⟩
49
50   inverse: {G: group} dom G → dom G # 2 ⁻¹ prec -5
51   inverse_property : {G: group} ⊦ ∀[x : dom G] x G.∘ x⁻¹ ≐ G.e
52
invalid unit: http://mathhub.info/MitM/algebra?group?inverse_property?type: Checking Judgment:  |- {G:group}⊦∀[x:dom G]x G.∘x⁻¹≐G .e INHABITABLE
```

```
51:invalid unit: http://mathhub.info/MitM/algebra?group?inverse_property?type: Checking Judgment:  |- {G:group}⊦∀[x:dom G]x G.∘x⁻¹≐G .e INH
  Checking Judgment:  |- {G:group}⊦∀[x:dom G]x G.∘x⁻¹≐G .e INHABITABLE
  - Applying InhabitableRule rule ShallowPolymorphism for Pi
  - Checking Judgment: G:group |- ⊦∀[x:dom G]x G.∘x⁻¹≐G .e INHABITABLE
  -- definition expansion yields: ⊦∀[x:dom G]x G.∘x⁻¹≐G .e ~~> DED ∀[x:dom G]x G.∘x⁻¹≐G .e
  -- definition expansion yields: DED ∀[x:dom G]x G.∘x⁻¹≐G .e ~~> DED ∀[x:dom G]x G.∘x⁻¹ EQ (G .e)
  -- inferring universe
```

```
48,54-55 (1673/1971)                    (mmt,none,UTF-8)Nmr o UG 144/411MB 11:32
```

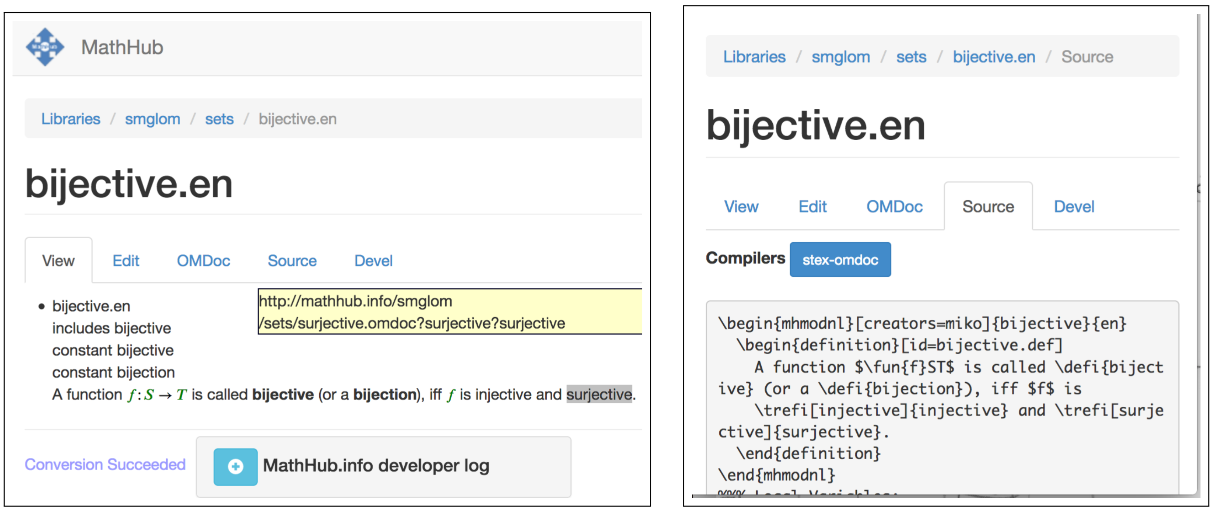
FIGURE 3. Local Editing of MMT Surface Syntax in JEdit

In our experience, a dedicated, tightly integrated IDE is indispensable for the development of formal mathematical content. This is analogous to the development of software – another form of formal content – but for formal mathematical documents type-checking – which includes proof checking via the Curry-Howard isomorphism – puts even more semantic constraints on the edited material and thus an even heavier strain on the author/maintainer. ToDo:5

3.5. Web-based MathHub Editing

In the web-based workflows, we have the same requirements for integrating editing and semantic correctness management as in the local workflows. In the current state of development of MathHub, we aim for a courser granularity of integration: we integrate ToDo:6

[5]TO DO: specify what part of this section, if anything, has been achieved as part of this deliverable
[6]TO DO: ? coarser?


MathHub

Libraries / smglom / sets / bijective.en

bijective.en

View Edit OMDoc Source Devel

- bijective.en
  includes bijective
  constant bijective
  constant bijection
  A function $f : S \to T$ is called **bijective** (or a **bijection**), iff f is injective and surjective.

http://mathhub.info/smglom/sets/surjective.omdoc?surjective?surjective

Conversion Succeeded

MathHub.info developer log

Libraries / smglom / sets / bijective.en / Source

bijective.en

View Edit OMDoc Source Devel

Compilers stex-omdoc

```
\begin{mhmodnl}[creators=miko]{bijective}{en}
  \begin{definition}[id=bijective.def]
    A function $\fun{f}ST$ is called \defi{bijective} (or a \defi{bijection}), iff $f$ is
    \trefi[injective]{injective} and \trefi[surjective]{surjective}.
  \end{definition}
\end{mhmodnl}
```

FIGURE 4. Presentation and Source Views of A Glossary Entry

at the document level, which is commensurate with the fact that the quest for reusability in active documents leads to small documents overall. Figure 4 shows the presentation of (the english language binding) of a glossary module for bijectivity. The breadcrumbs at the top localize it in the sets[7] archive in the SMGloM library. The content presentation shows its dependencies and a human-oriented presentation of the definition of bijectivity. There are five tabs with different forms of the same content, the source view (expanded on the right of Figure 4), the generated OMDoc, and an edit tab. The latter is just a link to the the corresponding GitLab edit page; see Figure 5.

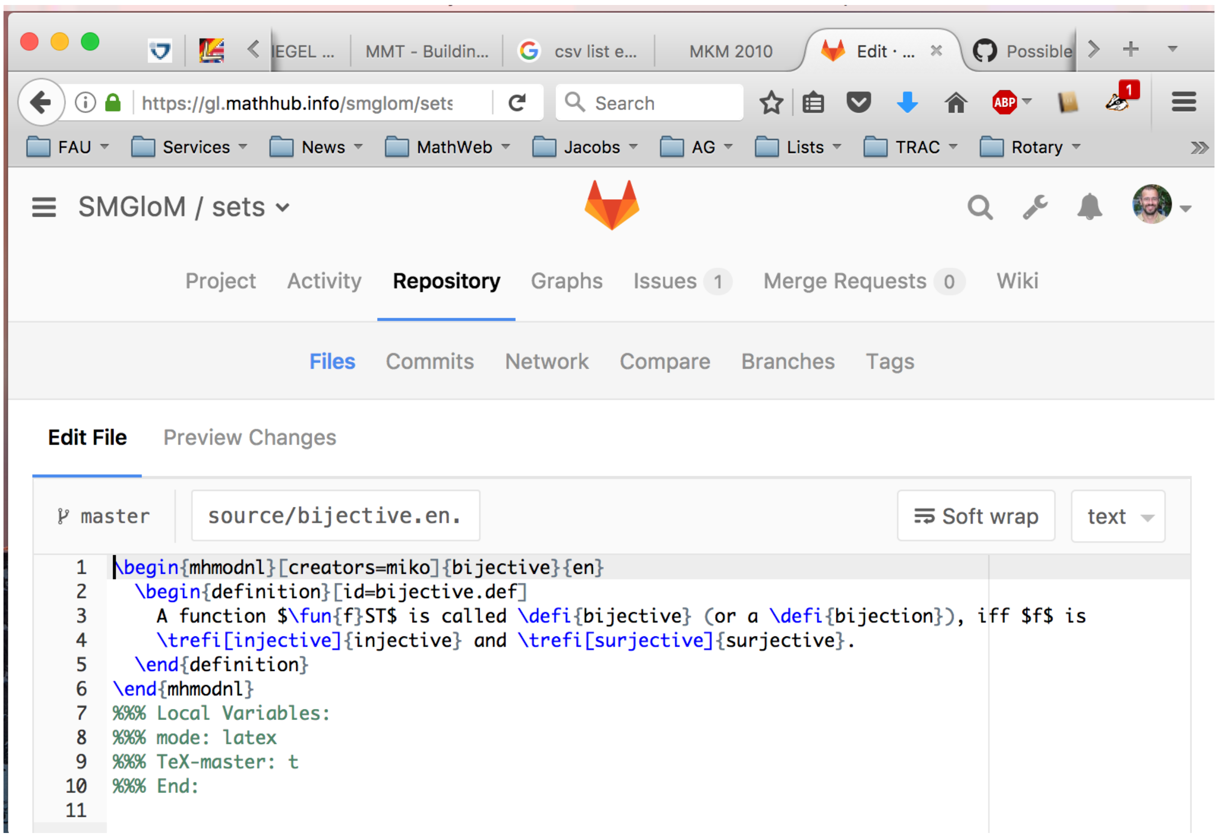
This delegation is commensurate with the division of labor in the MathHub system; GitLab suports the full range of revision control interactions. In this case, users with sufficient access rights can directly commit to the repository (and thus change the math archive directly) users with less rights can directly initiate a pull request and submit their changes to the archive maintainers in this way.

But this delegation is not without technical challenges. An important concern is that the user identities of the Drupal and GitLab instances have to be synchronized. Fortunately, users can log into both systems with their GitHub account, which solves the problem at least for the OpenDreamKit project, where GitHub is universally employed and thus all partners have GitHub accounts. Another problem is that GitLab only offers editing facilities for files it considers text files, and it considers MMT syntax files as binary, since they contain lower ASCII symbols for component, declaration, and module separators. Unfortunately, the media types are not configurable in GitLab, so we have to patch the code. Building on an open-source system like GitLab and not the closed-source GitHub allows to do that, but this still poses administration hassles with the fast-moving release cycle of GitLab.

4. CONCLUSION

We have presented the facilities for distributed, collaborative, and versioned editing in the MathHub system. MathHub borrows much of the functionality from the underlying GIT and GitHub systems, but customizes them to the particular problem of authoring, maintaining, and curating highly structured knowledge bases of flexiformal mathematical knowledge.

[7] TO DO: it's hard to tell that sets is in a different font and should be interpreted as a name and not a word



```
\begin{mhmodnl}[creators=miko]{bijective}{en}
  \begin{definition}[id=bijective.def]
    A function $\fun{f}ST$ is called \defi{bijective} (or a \defi{bijection}), iff $f$ is
    \trefi[injective]{injective} and \trefi[surjective]{surjective}.
  \end{definition}
\end{mhmodnl}
%%% Local Variables:
%%% mode: latex
%%% TeX-master: t
%%% End:
```

FIGURE 5. Editing a Glossary Entry in GitLab

Acknowledgements

The implementations reported on in this report have been carried out by Mihnea Iancu, Constantin Jucovshi, Tom Wiesing, and a host of students at Jacobs University.