A Framework for Semantic Publishing of Modular Content Objects

Michael Kohlhase, Deyan Ginev, Vlad Merticariu

Computer Science, Jacobs University Bremen, Germany
`<first-initial>.<last-name>@jacobs-university.de`

Abstract. We present the Active Documents approach to semantic publishing (semantically annotated documents associated with a content commons that holds the background ontologies) and the Planetary system (as an active document player).
In this paper we explore the interaction of content object reuse and context sensitivity in the presentation process that transforms content modules to active documents. We propose a "separate compilation and dynamic linking" regime that makes semantic publishing of highly structured content representations into active documents tractable and show how this is realized in the Planetary system.

1 Introduction

Semantic publication can range from merely equipping published documents with RDFa annotations, expressing metadata or inter-paper links, to frameworks that support the provisioning of user-adapted documents from content representations and instrumenting them with interactions based on the semantic information embedded in the content forms. We want to propose an entry to the latter category in this paper. Our framework is based on *semantically annotated documents* together with semantic background ontologies (which we call the **content commons**). This information can then be used by user-visible, semantic services like program (fragment) execution, computation, visualization, navigation, information aggregation and information retrieval (see Figure 6). Finally a document player application can embed these services to make documents executable. We call this framework the **Active Documents Paradigm** (ADP), since documents can also actively adapt to user preferences and environment rather than only executing services upon user request. In this paper we present the ADP with a focus on the Planetary system as the document player (see Figure 1)

The Planetary system (see [Koh+11; PDF] for an introduction) is a Web 3.0 system[1] for semantically annotated document collections in Science, Technology, Engineering and Mathematics (STEM). In our approach, *documents published in the Planetary system become flexible, adaptive interfaces to a content commons*

[1] We adopt the nomenclature where Web 3.0 stands for extension of the Social Web with Semantic Web/Linked Open Data technologies.

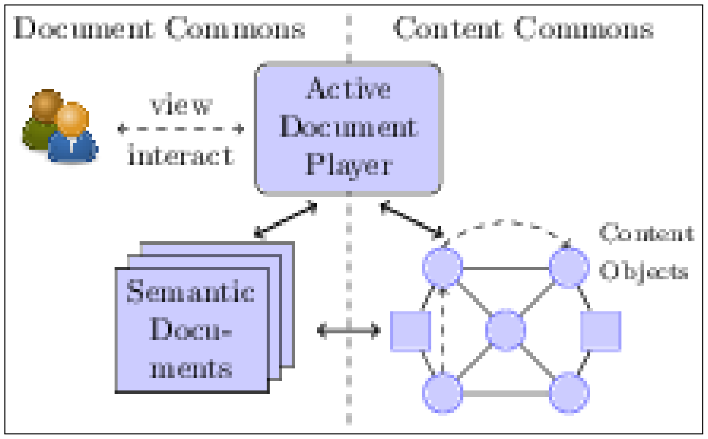


Fig. 1. The Active Documents Paradigm

of domain objects, context, and their relations. The system achieves this by providing embedded user assistance through an extended set of user interactions with documents based on an extensible set of client- and server side services that draw on explicit (and thus machine-understandable) representations in the content commons.

However, the flexibility and power designed into the active documents paradigm comes at a (distribution) cost: Every page that is shown to the user has to be assembled for the user in a non-trivial compilation process (which we call the **presentation** process) that takes user preferences and context into account. On the other hand, if the content is organized modularly, it can be reused across contexts. This presents a completely new set of trade-offs for publishing. One of them is that an investment in modular and semantic representational markup enhances reusability and thus may even lower the overall cost of authoring. We will explore another such trade-off in this paper: optimizing the distribution costs for modular content by "separate compilation and dynamic linking" (SCDL).

In the next section we will look at the organization of the content presented to the user. This will constitute the conceptual backdrop against which we can discuss the issues involved in SCDL and how we have solved them in the Planetary system.

2 Organization of Content/Narrative Structure

The Planetary system is intended as a *semantic publishing framework*, i.e. as a system providing the baseline capabilities needed for multiple specialized instantiations. We have shown the initial feasibility of the concept in a variety of publicly available case studies[2] ranging from pre-semantic archives of scientific

[2] Note that all of these are research systems under constant development, so your mileage may vary.

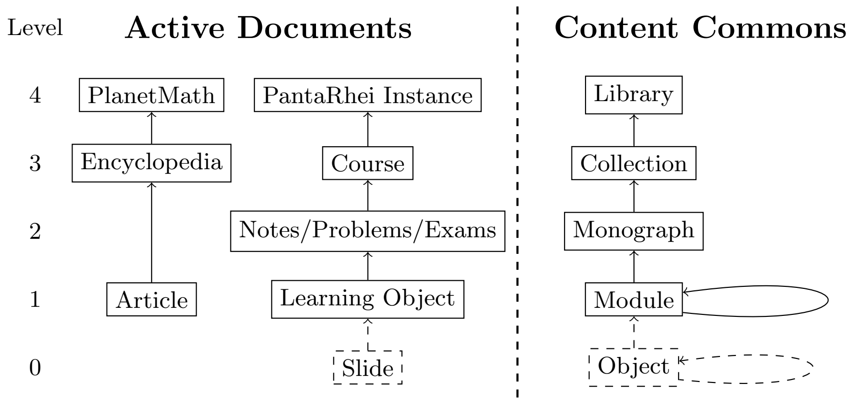


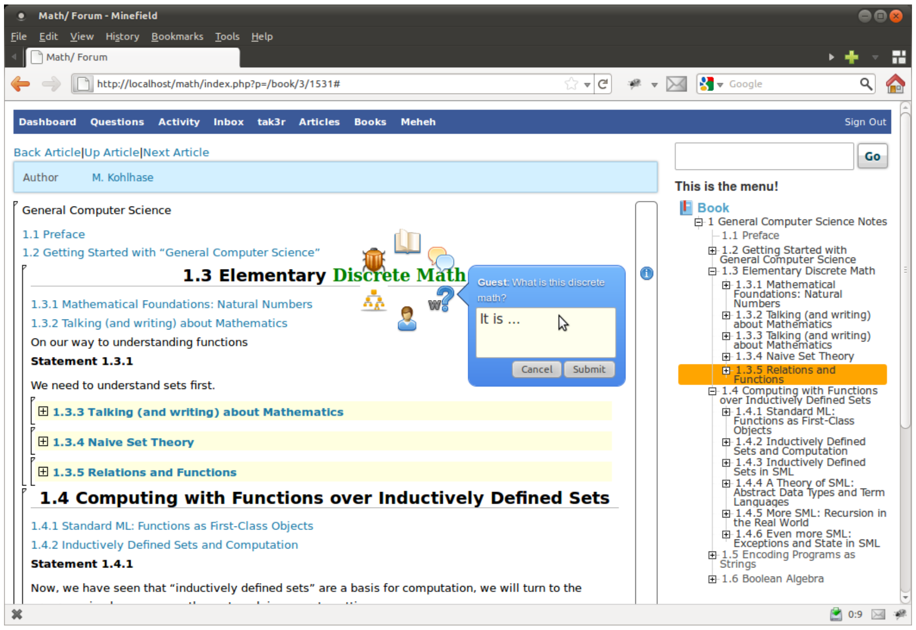
Fig. 2. Levels of Organisation of Content

literature [arX], over a community-driven mathematical encyclopedia [PM] and the course system PantaRhei [PR], to a community portal of formal logics [LA]. As a consequence of this, we employ the general, modular knowledge structure depicted in Figure 2.

2.1 Levels of Content/Documents

The lowest level consists of atomic "modules"[3], i.e. content objects that correspond to small (active) documents dedicated to a single topic. For a course management system these might be learning objects (either as single modules or module trees), for an encyclopedia these would be the individual articles introducing a topic. Note that technically, we allow modules to contain (denoted by the arrows) other modules, so that larger discourse structures could be formed. For example, sections can be realized as modules referencing other modules of subsections, etc. The next level up is the level of "monographs", written works on a single subject that have a complete, self-contained narrative structure, usually by a single author or group of authors who feel responsible for the whole monograph. As a content object, a monograph is usually built up from modules, e.g. as a "module tree" that corresponds to sectioning structure of traditional books, but often also includes front and backmatter such as a preface, acknowledgements (both special kinds of modules), table of contents, lists of tables and figures, an index and references (generated from content annotations). Figure 3 shows course notes in the PantaRhei system, while other documents at the monograph level are articles in a journal, or books in a certain topical section of a library.

[3] The level of objects below modules consists of individual statements (e.g. definitions, model assumptions, theorems, and proofs), semantic phrase-level markup, and formulae. Even though it carries much of the semantic relations, it does not play a great role for the document-level phenomena we want to discuss here in this paper.


Math/ Forum - Minefield
File Edit View History Bookmarks Tools Help
Math/ Forum
http://localhost/math/index.php?p=/book/3/1531#
Google
Dashboard Questions Activity Inbox tak3r Articles Books Meheh
Sign Out
Back Article|Up Article|Next Article
Go
Cancel
Submit
0:9

Author M. Kohlhase

General Computer Science

1.1 Preface

1.2 Getting Started with "General Computer Science"

1.3 Elementary Discrete Math

1.3.1 Mathematical Foundations: Natural Numbers

1.3.2 Talking (and writing) about Mathematics

On our way to understanding functions

Statement 1.3.1

We need to understand sets first.

⊞ 1.3.3 Talking (and writing) about Mathematics

⊞ 1.3.4 Naive Set Theory

⊞ 1.3.5 Relations and Functions

1.4 Computing with Functions over Inductively Defined Sets

1.4.1 Standard ML: Functions as First-Class Objects

1.4.2 Inductively Defined Sets and Computation

Statement 1.4.1

Now, we have seen that "inductively defined sets" are a basis for computation, we will turn to the

Guest: What is this discrete math?

It is ...

This is the menu!

Book
- ⊟ 1 General Computer Science Notes
  - 1.1 Preface
  - ⊞ 1.2 Getting Started with General Computer Science
  - ⊟ 1.3 Elementary Discrete Math
    - ⊞ 1.3.1 Mathematical Foundations: Natural Numbers
    - ⊞ 1.3.2 Talking (and writing) about Mathematics
    - ⊞ 1.3.3 Talking (and writing) about Mathematics
    - ⊞ 1.3.4 Naive Set Theory
    - ⊞ 1.3.5 Relations and Functions
  - ⊟ 1.4 Computing with Functions over Inductively Defined Sets
    - ⊞ 1.4.1 Standard ML: Functions as First-Class Objects
    - ⊞ 1.4.2 Inductively Defined Sets and Computation
    - ⊞ 1.4.3 Inductively Defined Sets in SML
    - ⊞ 1.4.4 A Theory of SML: Abstract Data Types and Term Languages
    - ⊞ 1.4.5 More SML: Recursion in the Real World
    - ⊞ 1.4.6 Even more SML: Exceptions and State in SML
  - ⊞ 1.5 Encoding Programs as Strings
  - ⊞ 1.6 Boolean Algebra

Fig. 3. A Monograph (Course Notes) in the Planetary system

Multiple monographs can be combined into collections, adding special modules for editorial comments, etc. Concrete collections in the document realm are encyclopedias, academic journals, conference proceedings, or courses in a course management system. Finally, the library level collects and grants access to collections, concrete, modern-day examples are digital libraries, installed course management systems, etc. In practice, a library provides a base URI that establishes the web existence of the particular installation. In the Semantic Web world, the library is the authority that makes its resources addressable by URLs.

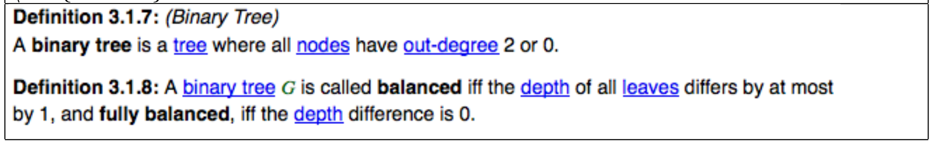
2.2 Content Objects and their Presentations in Active Documents

To understand the differences between content objects and the documents generated from them in the presentation process, let us consider the example in Figure 4. Even though internally the content objects in Planetary are represented in OMDoc [Koh06], we will use the surface language sTeX[4] for the example, since this is what the author will write and maintain. sTeX is a variant of LaTeX that allows to add semantic annotations in the source. It can be transformed

[4] We speak of an OMDoc surface language for any language that is optimized for human authoring, but that can be converted to OMDoc automatically. Note that all semantic algorithms referred to in this paper run on the generated OMDoc, since it is XML-based and syntactically much more regular and thus more machine-processable.

into OMDoc via the LaTeXML daemon [GSK11] for management in Planetary; see [Koh08] for details. We are using an example from a mathematical document[5] since content/presentation matters are most conspicuous there. In our experience, sTeX achieves a good balance (at least for authors experienced with LaTeX) between conciseness and readability for mathematical documents. In particular, since sTeX documents such as the one in Figure 4 can be transformed to PDF via the classical `pdflatex` for prototyping and proofreading. The semantic editing process can further be simplified by *semantic document development environments* like sTeXIDE [JK10], which provides edit-support services like semantic syntax highlighting, command completion/retrieval, and module graph management.

```
\begin{module}[id=binary−trees]
  \importmodule[\KWARCslides{graphs−trees/en/trees}]{trees}
  \importmodule[\KWARCslides{graphs−trees/en/graph−depth}]{graph−depth}
  ...
  \begin{definition}[id=binary−tree.def,title=Binary Tree]
    A \definiendum[binary−tree]{binary tree} is a \termref[cd=trees,name=tree]{tree}
    where all \termref[cd=graphs−intro,name=node]{nodes}
    have \termref[cd=graphs−intro,name=out−degree]{out−degree} 2 or 0.
  \end{definition}
  ...
  \begin{definition}[id=bbt.def]
    A \termref[name=binary−tree]{binary tree} $G$ is called
    \definiendumalt[bbt]{balanced}
    iff the \termref[cd=graph−depth,name=vertex−depth]{depth} of all
    \termref[cd=trees,name=leaf]{leaves} differs by at most by 1, and
    \definiendum[fullbbt]{fully balanced}, iff the
    \termref[cd=graph−depth,name=vertex−depth]{depth} difference is 0.
  \end{definition}
  ...
\end{module}
```


Definition 3.1.7: *(Binary Tree)*
A **binary tree** is a tree where all nodes have out-degree 2 or 0.

Definition 3.1.8: A binary tree G is called **balanced** iff the depth of all leaves differs by at most by 1, and **fully balanced**, iff the depth difference is 0.

Fig. 4. Content and Presentation of an Object in sTeX

The sectioning structure monographs is generated from special sTeX files that encode the narrative and hierarchical structure of the course. Listing 1.1 shows the narrative/hierarchical module that corresponds to Section 1.3 in Figure 3.

Listing 1.1. A Narrative/Hierarchical Module

```
\begin{omgroup}[id=sec.edmath]{Elementary Discrete Math}
```

[5] Actually from a second-semester course on Computer Science hosted in PantaRhei—an instance of the Planetary system that is optimized for active course notes and discussions.

```
\inputref{\KWARCslides{dmath/fragments/nat−induction}}
\inputref{\KWARCslides{dmath/fragments/mathtalk}}

\begin{frame}
  \frametitle{On our way to understanding functions}
  \begin{omtext}
    We need to understand sets first.
  \end{omtext}
\end{frame}

\inputref{\KWARCslides{dmath/fragments/naive−sets}}
\inputref{\KWARCslides{dmath/fragments/functions}}
\end{omgroup}
```

In these files, the omgroup environments are generic sectioning devices, and the frame/\frametitle combination specifies text fragments that can be used as slides in lectures and transitionary text in the course notes. The **\inputref** macros are variants of the LaTeX **\input** macro but with the added functionality that the specified modules will be displayed as folded in Planetary by default.

The upper half of Figure 4 shows the content representation of a module on binary trees, and its presentation in Planetary is in the lower box. The first aspect that meets the eye is that the presentation process[6] adds the textual marker "**Definition 3.1.7**" which is not present in the content representation

```
\begin{definition}[id=binary−tree.def,title=Binary Tree]
```

Note that there are (at least) four issues at hand here pertaining to the presentation of the text marker:

1. The marker "**Definition**" is context-sensitive: The presentation of a Spanish text would have generated "**Definición**".
2. The number "**3.1.7**" is content-sensitive in a totally different way: it is determined by the document structure, here it is a consequence of being the seventh definition in the first section in chapter 3.
3. The "house style" of a journal might use a different font family for the whole textual marker, for the text of the definition, or add an end marker for a distinctive layout. For instance in mathematical publications, theorems are usually set in italics and proofs use a little box on the right of the last line as an end marker.
4. Finally, the whole text marker may be left out altogether in some situations, where a less formal presentation is called for.

Note that all these considerations have to be taken into account when referencing objects like these definitions as well. More so, these dimensions combine into a unique multi-dimensional point, which identifies the exact presentation of a document fragment. A content reference **\sref**{binary−tree.def} might be presented as "**Def. 3.1.7**", in the same context as above (again subject to language, house style, etc). Note that here the style (e.g. the keyword) and generated contextual

[6] We disregard the presentation of formulae in content representation like OpenMath or content MathML into presentation MathML in this paper and refer the reader to [KMR08] for details.

locators (e.g. the number) of the referenced object determines the actual label of the reference[7]. We follow the context dimensions specified in [KK08, Chapter 3], but note that many of the phenomena involve a separate, publishing context dimension (e.g. "house style").

Another phenomenon related to referencing is induced by the term reference **termref**[cd=graphs,name=vertex]{node}, which identifies the phrase "node" as a technical term and links it to its defining occurrence by the symbol name (here vertex) and the module name (also called content dictionary; here graphs). The specified module must be accessible in the current module via the **importmodule** relation and must contain a definition that contains a definiendum with symbol name vertex. The content module in Figure 4 specifies a module/content dictionary with name balanced−binary−trees, whose first definition supplies a definiendum with name balanced−tree via the **definiendum** macro, which is referenced in the second definition. Note that in the presentation process where term references are displayed e.g as hyperlinks to the definition the name-based semantic links have to be converted into regular URI references. For this presentational conversion to hyperlinks one utilizes not only the module tree structure (i.e. visibility relationship) but also the library context that provides the base of the URI[8]

Finally, note that some content objects contribute to the context of other objects higher up in the content hierarchy in Figure 2. A good example for this are the definienda discussed above. In sTeX, they trigger index entries that populate the backmatter of monographs that include the respective module. Section titles populate the frontmatter in a similar way. Concretely, we have a top-level index stub in the backmatter, which "builds" itself from the context. In a sense, the index is an abstract concept with volatile presentation, generated from the module tree with the help of the content commons, which answers what objects should be indexed.

3 Separate Compilation & Dynamic Linking

We have seen above that the various contexts (conceptual/document/language) have a significant effect on the presentation. But observe that if all the context-dependent parts of the presentation can be generated (albeit laboriously), the content representations are context-independent and can be reused in different contexts. This makes the content representations very portable. Consider for instance the definitions in our example above. They have been reused not only in nine instances of the "General Computer Science II" course in the years 2004-2012 (each time with different numbers due to additions or deletions of preceding material), but also in different courses, e.g. as a recap in a more advanced CS

[7] a rather peculiar notion of context when viewed from a content-only perspective

[8] In the sTeX implementation this is realized by the practice/hack of abstracting from file paths with special macros like \KWARCslides which can be instantiated with local file paths by pdflatex and with library URIs by the LaTeXML conversion.

course (without definition marker); see Figure 5 for a typical situation. Clearly, we cannot reasonably pre-compute all necessary presentation variants.

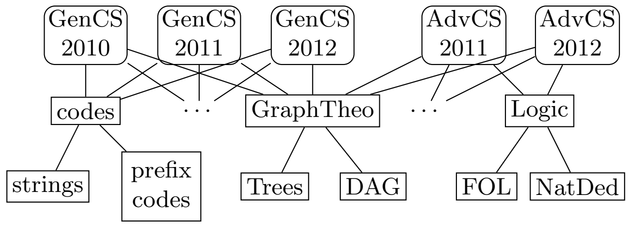


Fig. 5. Reuse of Course Fragments

Actually, the number of presentation variants can be unbounded: the Planetary system can generate "guided tours" (self-contained explanations adapted to the user's prerequisite knowledge) for any concept in a document.

Computationally, the described situation is analogous to (and in fact conceptually influenced by) the situation in software design, where large programs are broken up into reusable source modules. As source modules are re-used in many programs, it is important that compilers support a regime of "separate compilation and linking" to make software development tractable: if one of many software modules used in a program changes, only that one module has to be re-compiled and the whole program re-linked. The first factor that enables this is the observation that for compilation of a module only the (relatively stable) signatures[9] of modules it depends on are needed, not the (relatively change-prone) module implementations. The second factor is that source modules can be compiled into a form, where references to functions imported from other modules are left symbolic and can later be replaced by concrete static references by the linker. We will call such forms of modules **contextable**, since they are contextualized by the linker in the way described.

In the Planetary architecture semantic publishing consists of the transformation of content structures encoded in sTeX to active documents encoded in HTML5[10] (see Section 3.2 for details). To foster reuse, and make the process tractable, we want to assemble active documents from reusable content modules much in the same way as assembling an executable program from source modules. To make the separate compilation analogy fertile for semantic publishing it is useful to look at the role of context in the separate compilation regime: source modules are compiled into a context-independent form, which is then contextualized by linking compiled modules together into a consistent configuration for

[9] Signatures contain the names of functions/procedures, possibly their types, but not their implementations.

[10] We use HTML5 as it integrates HTML for document layout with MathML for formula presentation, SVG for diagrams, and RDFa for document-embedded metadata and is supported by the major browsers.

a concrete program. In the next two sections we examine how the two factors identified as crucial for the separate compilation regime can be obtained in the context of semantic publishing.

3.1 Contextable Presentations

Just as in programming, *separate* compilation of content modules into active documents is impossible without contextable structures in the presentation. It is an original contribution of our work to introduce them in the document setting. Concretely, we make use of the XML styling architecture and computes context-independent presentations that can be contextualized later. For instance, the HTML header for the first definition in Figure 4 has the following form.

```
<div id="binary−tree.def" class="omdoc−definition">
    <span class="omdoc−statement−header">
        <span class="omdoc−definition−number"/>7</span>
        <span class="omdoc−statement−title">
            Binary Tree
        </span>
    </span>
    ...
```

We can then add (house) style information via CSS:

```
span.omdoc−statement−header {font−weight:bold}
span.omdoc−statement−title:before {content:"("}
span.omdoc−statement−title:after {content:")"}
span.omdoc−definition−number:after {content: ":␣"}
span.omdoc−definition−number:before {content:"Definition␣"}
```

Note that the keywords are not represented explicitly in the HTML presentation, but added by content declarations in the CSS. This allows to overwrite the default ones via cascaded language-specific CSS bindings, e.g. using

```
span.omdoc−definition−number:before {content:"Definición "}
```

Note furthermore, that the presentation process only adds preliminary statement numbers in the HTML presentation (here the number 7, since the definition is the seventh statement in the module). In the Planetary system, these numbers are dynamically overwritten by the linker with values computed from the context; in our example “3.1.7”. The case for references is similar; for the table of contents shown in Figure 3 the presentation generates the contextable HTML5 fragment

```
 <div class="omdoc−expandableref">
      <span class="omdoc−ref−number">4</span>
      <span class="omdoc−reftitle">
          <a href="../computing−dmath.omdoc"
                class="expandable">
              Computing with Functions over Inductively
              Defined Sets
          </a>
    </span>
</div>
```

in the table of contents on the right and in the text. The CSS class omdoc−expandableref triggers the Planetary interaction that expands the references in place to get the expanding ToC and the main document that can be folded/unfolded via the Mathematica-style folding bars on the extreme left.

3.2 Supporting the Logistics of Separate Compilation: Dynamic Linking

The role of the module signatures (think C header files) is taken by sTeX module signatures, i.e. auxiliary files generated from sTeX content modules that excerpt the information about references, modules and their dependencies; see [Koh08] for details. This information is used to establish a mapping between the content commons and the document commons (see Figure 1) that can be queried for the semantic interaction services embedded into the active documents.

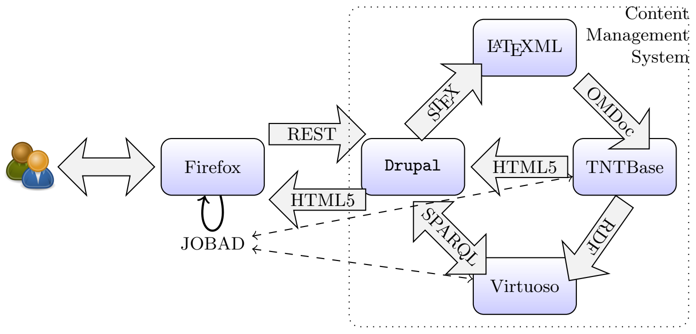


Fig. 6. The Planetary System Ecosystem

Actually, to understand the compilation and linking phases in Planetary, consider the system architecture in Figure 6. The document commons and content commons are layouted in a generic web browser and encapsulated versioned XML knowledge store TNTBase [ZK09] respectively. Planetary acts as a semantic content management system, where semantic web services are integrated into a central *Container* Management System (CMS) that mediates all user interaction. Note that in this architecture, we greatly extend the role of the *content management subsystem* (denoted by the dotted box in Figure 6). The CMS (initially Vanilla Forums, later Drupal) supplies management and interaction at the "container level", i.e., without ever looking into the documents it manages (hence the somewhat non-standard name). The management of *structured document content* is split between TNTBase and the RDF triple store in Planetary, since they can perform semantic services. In this situation, the separate compilation workflow proceeds in four steps.

1. sTeX is converted to OMDoc via the LaTeXML daemon [GSK11], this is then converted to HTML5+RDFa. Both transformations are highly dependent on

notation information in the content commons, so they are under the control of the TNTBase system, which stores the content commons.

2. Planetary caches the contextable presentations that are generated by the TNTBase system in special containers (**CC nodes**). New presentations are requested from TNTBase whenever *a)* the content module in TNTBase has changed, and *b)* a user requests a to view the module.
3. Planetary hosts a triple store (Virtuoso) of structural metadata from the content commons that can be used for semantic services and document-level features, such as different views based on various selection criteria for an encyclopedia.
4. Finally, the CMS calls the linker contextualize the contextable presentations generated by TNTBase and augmented by the semantic services, and delivers it as the payload of the CMS-generated web pages (next to the CMS-administrative parts like the blue dashboard on top of Figure 3)

To support the linker, Planetary hosts structural information about the knowledge items at the different levels in Figure 2. Recall that Planetary itself constitutes the library level, so it keeps a list of collections it hosts[11], the monograph/-module structure of each collection is represented as an ordered DAG whose vertices are CC nodes and whose edges are induced by the \inputref macros in the sTeX source; we call this graph structure the **CMM structure** of the respective Planetary instance. In this sense, Figure 5 depicts part of a CMM structure of a course library. Observe that the inner nodes of the CMM structure are narrative/hierarchical modules like the one in Listing 1.1 while the leaves are modules like the one in Figure 4.

Individual collections, monographs, and modules are represented by CC nodes as the reachable sub-DAGs rooted that node. Note that this DAG will usually be unrolled into a document tree for presentation to the user. Planetary instances usually give the user access to a distinguished subset of collections, monographs, and modules as dedicated entry points; they correspond to the **Planetary activedocs** in the library (active documents to distinguish them from "generic documents").

Finally, observe that the user-visible pages in Planetary activedocs can be represented as a pair $\langle n, \pi \rangle$, where n is a a CC node and π is a CMM path π out of n: the path π identifies the contextable page content in terms of the collection, monograph, or module that serves as narrative context. In the examples in Figure 3 and Section 2.2, the numbering is linked into the contextable modules whenever a page is viewed, based on this information. Recall we need this *dynamic (i.e. view-time) linking* as modules are re-used in different activedoc contexts.

A fortunate side effect to the recent switch from vanilla forums to drupal as the CMS in Planetary is the availability of the drupal books module [DruB], which already supports a part of the CCM structure and the induced user inter-

[11] In all cases we have looked at, a list is sufficient; have never found a narrative structure mediating this list.

actions. So we ported our implementation to it and have extended it by a linking component which we now sketch.

3.3 Contextualizing Contextable Presentations

There are two main aspects to linking in the document setting: generated content like tables of content/figures, indices, glossaries, or bibliographies and labels for references and reference targets. We will concentrate on the labeling aspects here, as the former can be reduced to rather standard SQL or XQuery or SPARQL (either will work, given a framework for extracting the respective metadata; in Planetary we use a combination).

For the computation of labels we first observe that the labels of reference targets (chapters, sections, definitions, or equations) in a user-visible page $\langle n, \pi\rangle$ is determined by n and another "house style". Once the labels are computed, for the **label-carrying elements** (LCE; XML elements that will become a possible reference target) in a user-visible page $\langle n, \pi\rangle$ we can register the LCEs by their id in the **label map** maintained by Planetary. The linker simply looks up the current labels by reference id for references. Note that in this lookup we key by both the reference id as well as the activedoc, because only together they determine a unique label. Note furthermore, that this holds for activedoc-internal references as well as for inter-activedoc references, even though the former do not explicitly specify the activedoc.

Let us now turn back to label computation. As in LaTeX[12], the Planetary labeling system is based on a set of **counters**, i.e. named variables that range over the natural numbers. Counters are are pre-ordered by a tree-formed relation $\prec$. Every LCE $\mathcal{E}$ has an associated counter $c_{\mathcal{E}}$. The label of a LCE $\mathcal{E}$ is computed from the ordered set $c_\prec(\mathcal{E}) := \{d \big| d \preceq c_{\mathcal{E}}\}$ via a **label function** $f_{\mathcal{E}}$ which is a function from lists of natural numbers to XML node sets[13].

The linker recursively goes over the contextable presentations scanning for LCEs. Whenever an LCE $\mathcal{E}$ is encountered, the counter $c_{\mathcal{E}}$ is incremented and all counters $d \succ c_{\mathcal{E}}$ are re-initialized to 0. The label of $\mathcal{E}$ is defined as $f_{\mathcal{E}}(c_\prec(\mathcal{E}))$. Note that the computed label of a LCE $\mathcal{E}$ is determined by the pair $\langle c_{\mathcal{E}}, f_{\mathcal{E}}\rangle$ which we call a **labeling specification**. We will write $\mathcal{E} : \langle c_{\mathcal{E}}, f_{\mathcal{E}}\rangle$ for associating a

[12] The situation in the active documents paradigm is different from the one for LaTeX is different in two respects: LaTeX mixes label computation, page layout, and styling in a single formatting process, where as web documents do layout and styling after the document transformation in the client system (usually a web browser). Having these two phases in semantic publishing allows (and forces) us to reconsider which parts of the computation to do where. On the other hand LaTeX only allows re-use of activedoc fragments at the file level (via the `\input` macro), since the TeX formatter can only sequentially read source files. In web documents, we have "random document fragment access" and can enable client-side computation via JavaScript, which gives us additional possibilities for flexible document reuse and transformation workflows.

[13] Without loss of generality, we assume that all our web documents are XML-based and conflate linearly ordered sets with lists (recall that $\prec$ is tree-formed)

LCE with its labeling specification. A **labeling regime** is a finite map from LCE to labeling specification. The Planetary system associates a labeling regime with each activedoc which is used for linking.

Let us consider the following labeling regime for a book as an example. It is divided into parts, which are referenced by capital Roman numbers. These are further subdivided into chapters which are labeled by arabic numbers, and contain figures, definitions and theorems. The latter three are numbered by chapter, but figures separately from definitions and theorems (which we call "statements" together). We can model this by the set $\{p, c, f, s\}$ of counters with $p \prec c \prec f$ and $c \prec s$. We have the LCEs $\mathcal{P} : \langle p, \rho \rangle$ (for parts), $\mathcal{C} : \langle c, \alpha \rangle$ (chapters), $\mathcal{F} : \langle f, \alpha \rangle$, (figures), $\mathcal{D} : \langle s, \alpha \rangle$ (definitions), and $\mathcal{T} : \langle s, \alpha \rangle$ (theorems), where ρ is the function that returns (capitalized) roman numbers, and α arabic numbers for a counter value.

4 Conclusion

In this paper we have explored the conceptual and practical decoupling and interaction of content and presentation in the active documents paradigm of semantic publishing. Our main focus rested on the interaction of content object reuse and context sensitivity of the presentation process. To make semantic publishing of highly structured content representations into active documents tractable we have developed a "separate compilation and dynamic linking" regime for transforming highly structured content representations into active documents. The concrete realization in the Planetary system hinges on the development of contextable pre-presentations that are contextualized at document load time.

While the basic architecture has been realized in the Planetary system, there is still a lot to explore in the active documents paradigm and its SCDL implementation. One crucial aspect is that while SCDL makes building active documents tractable, it also leads to the well-known "late binding problems" (aka "DLL Hell"), if modules change without adaptation of the dependent ones. We are currently working on an integration of an ontology-based management of change process [AM10] into the Planetary system (see [Aut+11]). This tries to alleviate late binding problems by analyzing the impacts of a change via the dependency relation induced by the semantic structure of the content commons and supports authors in adapting their work. To complement this, we are currently developing a notion of "versioned references" that support the practice of creating and cultivating "islands of consistency" in the presence of change (see [KK11]). We hope that together, these measures can lead to semantic content management workflows that alleviate the side-effects of the semantic publishing workflow described in this paper.

Acknowledgements A precursor of the current "books module" in PantaRhei has been realized by Catalin David, Bogdan Matican and Stefan Mirea. We are grateful to Constantin Jucovschi for extending the Planetary system with the prerequisites for the system.

References

[AM10] Serge Autexier and Normen Müller. "Semantics-based Change Impact Analysis for Heterogeneous Collections of Documents". In: *Proceedings of the 10th ACM symposium on Document engineering.* Ed. by Michael Gormish and Rolf Ingold. DocEng '10. Manchester, United Kingdom: ACM, 2010, pp. 97–106. DOI: `http://doi.acm.org/10.1145/1860559.1860580`.

[arX] *Planetary arXiv Demo.* URL: `http://arxivdemo.mathweb.org` (visited on 09/27/2010).

[Aut+11] Serge Autexier et al. "Workflows for the Management of Change in Science, Technologies, Engineering and Mathematics". In: *Intelligent Computer Mathematics.* Ed. by James Davenport et al. LNAI 6824. Springer Verlag, 2011, pp. 164–179. URL: `http://kwarc.info/kohlhase/papers/planetary-moc.pdf`.

[Dav+11] James Davenport et al., eds. *Intelligent Computer Mathematics.* LNAI 6824. Springer Verlag, 2011.

[DruB] *Book module: Creating Structured Documents — drupal.org.* URL: `https://drupal.org/node/284` (visited on 04/22/2012).

[GSK11] Deyan Ginev, Heinrich Stamerjohanns, and Michael Kohlhase. "The LaTeXML Daemon: Editable Math on the Collaborative Web". In: *Intelligent Computer Mathematics.* Ed. by James Davenport et al. LNAI 6824. Springer Verlag, 2011, pp. 292–294. URL: `https://svn.kwarc.info/repos/arXMLiv/doc/cicm-systems11/paper.pdf`.

[JK10] Constantin Jucovschi and Michael Kohlhase. "sTeXIDE: An Integrated Development Environment for sTeX Collections". In: *Intelligent Computer Mathematics.* Ed. by Serge Autexier et al. LNAI 6167. Springer Verlag, 2010, pp. 336–344. arXiv: `1005.5489v1 [cs.OH]`.

[KK08] Andrea Kohlhase and Michael Kohlhase. "Semantic Knowledge Management for Education". In: *Proceedings of the IEEE; Special Issue on Educational Technology* 96.6 (June 2008), pp. 970–989. URL: `http://kwarc.info/kohlhase/papers/semkm4ed.pdf`.

[KK11] Andrea Kohlhase and Michael Kohlhase. "Maintaining Islands of Consistency via Versioned Links". In: *Proceedings of the 29th annual ACM international conference on Design of communication (SIGDOC).* 2011, pp. 167–174. URL: `http://kwarc.info/kohlhase/papers/sigdoc2011-verslinks.pdf`.

[KMR08] Michael Kohlhase, Christine Müller, and Florian Rabe. "Notations for Living Mathematical Documents". In: *Intelligent Computer Mathematics.* Ed. by Serge Autexier et al. LNAI 5144. Springer Verlag, 2008, pp. 504–519. URL: `http://omdoc.org/pubs/mkm08-notations.pdf`.

[Koh+11] Michael Kohlhase et al. "The Planetary System: Web 3.0 & Active Documents for STEM". In: *Procedia Computer Science* 4 (2011): *Special issue: Proceedings of the International Conference on Computational Science (ICCS).* Ed. by Mitsuhisa Sato et al. Finalist at the Executable Paper Grand Challenge, pp. 598–607. DOI: `10.1016/j.procs.2011.04.063`.

[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`.

[Koh08] Michael Kohlhase. "Using LaTeX as a Semantic Markup Format". In: *Mathematics in Computer Science* 2.2 (2008), pp. 279–304. URL: `https://svn.kwarc.info/repos/stex/doc/mcs08/stex.pdf`.

[LA] *Logic Atlas and Integrator*. URL: `http://logicatlas.omdoc.org` (visited on 09/22/2010).

[PDF] *Planetary Developer Forum*. URL: `http://planetary.mathweb.org/` (visited on 09/15/2012).

[PM] *PlanetMath.org – Math for the people, by the people*. URL: `http://planetmath.org` (visited on 11/11/2012).

[PR] *Panta Rhei, the Active Course Site at Jacobs University*. URL: `http://panta.kwarc.info` (visited on 02/22/2013).

[ZK09] Vyacheslav Zholudev and Michael Kohlhase. "TNTBase: a Versioned Storage for XML". In: *Proceedings of Balisage: The Markup Conference*. Vol. 3. Balisage Series on Markup Technologies. Mulberry Technologies, Inc., 2009. DOI: `10.4242/BalisageVol3.Zholudev01`.