Workflows for the Management of Change in Science, Technologies, Engineering and Mathematics

Serge Autexier[1], Catalin David[1,2], Dominik Dietrich[1], Michael Kohlhase[2], and Vyacheslav Zholudev[1,2]

[1] Safe and Secure Cognitive Systems, German Research Centre for Artificial Intelligence (DFKI), Bremen, Germany
[2] Computer Science, Jacobs University Bremen, Germany

Abstract. Mathematical knowledge is a central component in science, engineering, and technology (documentation). Most of it is represented informally, and—in contrast to published research mathematics—subject to continual change. Unfortunately, machine support for change management has either been very coarse grained and thus barely useful, or restricted to formal languages, where automation is possible. In this paper, we report on an effort to extend change management to semi-formal document collections and to integrate it into a semantic publishing system for mathematical knowledge. We validate the long-standing assumption that the semantic annotations in flexiformal documents that drive the machine-supported interaction with documents can support semantic impact analyses at the same time. But in contrast to the fully formal setting, where adaptations of impacted documents can be automated to some degree, the flexiformal setting requires much more user interaction and thus a much tighter integration into document management workflows.

1 Introduction

As the Web 2.0 age is dawning for mathematics, more and more *mathematical development* is moving online; not just *publications*. An example of this is the PolyMath site, where upon the recent announcement of a proof of $P \neq NP$, the mathematics community has organized itself in a WiKi and found a significant gap in the proof within two weeks; see [4]. The PlanetMath community which has collaborated on 8500 graduate-level encyclopedia articles over 10 years [20] is another, and also the Mizar community, who have formalized more than 60000 definitions, assertions, and proofs and have machine-checked them over the last 40 years. Finally, the Cornell EPrint Archive [21] has amassed over 660 000 scientific articles over 20 years. The hallmark of all these efforts is that they are massive collaborations by many individuals, distributed widely both geographically and temporally. The first three examples have another characteristic that is becoming more and more important: the knowledge items are interdependent and mutable (subject to change). The sheer size of the knowledge collections together with the fact that many authors do not even know (of) each other induces consistency and coherence problems. In this situation, the need to integrate the mechanisms for "change management" (CM) into the flexiformal digital libraries seems obvious. CM makes use of the fact that MKM formats explicitly represent the relations between objects to compute related objects and predict the way changes affect them; see [1, 8, 18] for recent progress in this field.

This paper reports on the experiment of integrating CM into the Planetary system, a new flexiformal Digital library system, which we will present in the next section. In Section 3, we describe the information present in the sources by way of an extended example and show how these can be used for change management. In Section 4, we present the *DocTIP* system and the CM procedure it implements, so that we can show the integration from an architectural point of view in Section 5. Section 6 revisits the example from Section 3 to show how the information travels through the systems involved. In Section 7, we discuss related work and Section 8 concludes the paper.

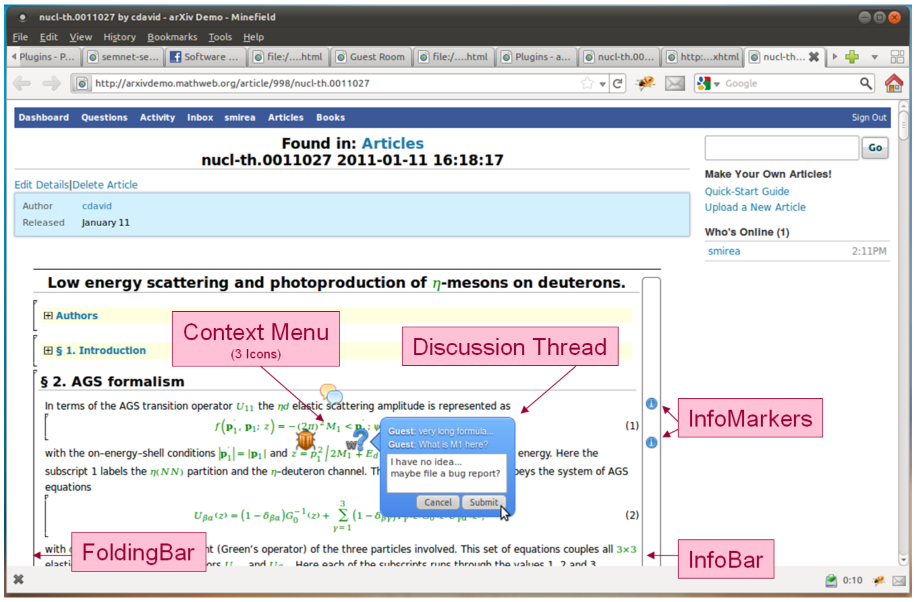
2 The Planetary System

The Planetary system (see [3, 14, 19] for an introduction) is a Web 3.0 system[3] for semantically annotated document collections in Science, Technology, Engineering and Mathematics (STEM). The system 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. 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 our approach, *documents published in the* Planetary *system become flexible, adaptive interfaces to a content commons* 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 (see Fig. 1).

The Planetary system has been used on the course notes of a two-semester introductory course in Computer Science [6] held at Jacobs University by one of the authors in the last eight years. While the basic concept of the course stayed the same over the years, whole topics have been added/moved/deleted, examples and results have been added, and formulations have been sharpened. All of these changes had consequences that were sometimes difficult to foresee, and sometimes led to problematic teaching situations (when the consequences

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


Context Menu
(3 Icons)
Discussion Thread
InfoMarkers
FoldingBar
InfoBar
File Edit View History Bookmarks Tools Help
Dashboard Questions Activity Inbox smirea Articles Books Sign Out
Go
Cancel Submit

nucl-th.0011027 by cdavid - arXiv Demo - Minefield

http://arxivdemo.mathweb.org/article/998/nucl-th.0011027

Found in: Articles
nucl-th.0011027 2011-01-11 16:18:17

Edit Details|Delete Article

Author cdavid
Released January 11

Make Your Own Articles!
Quick-Start Guide
Upload a New Article
Who's Online (1)
smirea 2:11PM

Low energy scattering and photoproduction of η-mesons on deuterons.

⊞ Authors

⊞ § 1. Introduction

§ 2. AGS formalism

In terms of the AGS transition operator U_{11} the ηd elastic scattering amplitude is represented as

$$f(\mathbf{p}_1', \mathbf{p}_1; z) = -(2\pi)^2 M_1 < \mathbf{p}_1'; \psi \quad (1)$$

with the on-energy-shell conditions $|\mathbf{p}_1'| = |\mathbf{p}_1|$ and $z = p_1^2/2M_1 + E_d$ … energy. Here the subscript 1 labels the $\eta(NN)$ partition and the η-deuteron channel. Th… beys the system of AGS equations

$$U_{\beta\alpha}(z) = (1 - \delta_{\beta\alpha})G_0^{-1}(z) + \sum_{\gamma=1}^{3}(1 - \delta_{\beta\gamma}) \quad (2)$$

with … nt (Green's operator) of the three particles involved. This set of equations couples all 3×3

Guest: very long formula...
Guest: What is M1 here?
I have no idea...
maybe file a bug report?

0:10

Fig. 1. Course Notes in the PLANETARY system

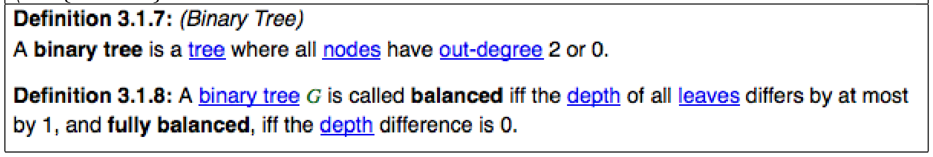
had not been anticipated). The course notes currently comprise 300 pages with over 500 slides organized in over 800 files. This is at the limits of what is manually manageable for the instructor who has authored all of the material; it would be impossible for a new instructor to take over the material (and change it to her liking). It becomes increasingly difficult to manage the over 1000 homework, quiz, and exam problems that have largely been provided by the more than 30 teaching assistants that have accompanied the course over the years.

3 A Planetary Workflow

To get a better intuition for the problems involved in managing changes in flexiformal document collections, consider the situation in Fig. 2 and Fig.3, which we will use as a running example. The lower part of Fig. 2 shows two well-known definitions from the theory of binary trees and Fig. 3 a lemma that depends on them, as they are referenced in its proof. Clearly, if one of the definitions is changed, then we have to revisit the proof and possibly adapt it or even the lemma to the changed situation.

For humans, it is simple to detect the underlying dependency in principle, but there is a strong possibility that it will be overlooked in practice; especially, if the conceptional distance between a proof and the definitions is large (e.g., because it involves many intervening definitions and assertions). Therefore, authors need system support to keep large mutable knowledge collections in a consistent state. In the situation of our running example, we can make use of the fact

```
\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
    \definiendum[bbt]{balanced binary tree} 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. 2. Two definitions and their sTEX sources

that the two text fragments were originally written as semantically annotated sTEX course notes [6] for PLANETARY. As such, they contain a lot of semantic annotations that are originally added to drive services like definition lookup, notation adaptation, and just-in-time prerequisites delivery, which also induce a good approximation of the semantic dependency relation that is needed for impact analysis.

Let us consider these annotations in the sTEX sources in Fig. 2 and Fig.3. In the first proof step (the sTEX **spfstep** environment) in Fig. 3, the “definition of a binary tree” is referenced, and this reference is marked up by a URI reference encoded in the optional argument of the **premise** macro inside the **justification** element. In the second proof step, the property of being “balanced” is exploited. The fact that the word “balanced” is used as a technical term is marked up with the **termref** macro, whose optional first argument points to the **definiendum** with name bbt in the module binary−trees in Fig. 2.

Intuitively, the relations encoded in these annotations induce the dependency that signals a possible semantic impact of a change to one of the definitions in Fig. 2. There are at least three possible ways an author can benefit from an automated impact analysis based on the semantic annotations in the sTEX sources.

C1 An author who wants to change something in one (or both) of the definitions in Fig. 2 can request an estimation of the total impacts costs of a change.

```
\begin{module}[id=bbt−size]
\importmodule[binary−trees]{binary−trees}
. . .
```

Lemma 3.1.9 *Let $G = \langle V, E\rangle$ be a* $\underline{\textit{balanced binary tree}}$ *of* $\underline{\textit{depth}}$ *$n > i$, then the set $V_i := \{v \in V : dp(v = i)\}$ of* $\underline{\textit{vertexes}}$ *at* $\underline{\textit{depth}}$ *i has* $\underline{\textit{cardinality}}$ *2^i.*
Proof: by $\underline{\text{induction}}$ over the $\underline{\text{depth}}$ i

```
  \begin{spfstep}
    By the \begin{justification}[method=byDef]
      \premise[uri=binary−trees,ref=binary−tree.def]{definition of a binary tree}
    \end{justification}, each $\inset{v}{V_{i−1}}$ is a leaf or has
    two children that are at depth $i$.
   \end{spfstep}
   \begin{spfstep}
     As $G$ is \termref[cd=binary−trees,name=bbt]{balanced}
     and $\gdepth{G}=n>i$, $V_{i−1}$ cannot contain leaves.
    \end{spfstep}
    ...
 \end{sproof}
\end{module}
```

Fig. 3. A lemma and proof that depend on the definitions in Fig. 2

C2 An author who actually changes (one of) the definitions can request an immediate impact analysis, which gives a list of potentially affected knowledge items. This list should be cross-linked to the (presentations of) the affected items, so to simplify navigation. For every item the author will have to decide whether it is really affected and how to adapt it (possibly creating new impacts in the process).

C3 Authors or maintainers of a given knowledge item can be notified of an impact to "their" knowledge item upon changes to elements it depends on.

Note that **C1** and **C2** together constitute what one could call a "push workflow of change management" whereas **C3** corresponds to a "pull workflow". The abundance of semantic references — 12 in this little example — already shows that machine support is indispensable in larger collections. Note furthermore that both of these workflows should be completely independent of the "commit policies" of the knowledge collection. The change management subsystem should support committing partially worked off impact lists — e.g., for the weekend or to pass them on to other authors.

4 *DocTIP*

The *DocTIP* system [5] provides a generic framework that combines sophisticated structuring mechanisms for heterogenous formal and semi-formal documents with an appropriate change management to maintain structured relations between different documents. It is based on abstract document models and abstract document ontologies that need to be instantiated for specific document kinds, such as OMDoc. The heart of the system is the *document broker*, which maintains all documents and provides a generic update and patch-based synchronisation protocol between the maintained documents and the connected *components* working on these documents. Components can be authoring (and display) systems, or analysis and reasoning systems offering automatic background processing support, or simply a connection to a repository allowing to commit and update the documents.

If the document broker obtains a change for some of its documents, the changes are propagated to all connected components for that document. A configurable impact analysis policy allows the system designer to define if impact analysis is required after obtaining a change from some component. To perform the impact analysis the document broker uses the *GMoC* tool ([1] see below) to compute the effect of the change on all documents maintained by the document broker. The *GMoC* tool returns that information as impact annotations to each individual document, which are subsequently distributed to all connected components by the document broker.

4.1 Change Impact Analysis

The key idea to design change impact analysis for informal documents is the *explicit semantics method* which represents both the syntax parts (i.e., the documents) and the intentional semantics contained in the documents in a single, typed hyper-graph. Document type specific graph rewriting rules are used to extract the intentional semantics of documents and the extracted semantic entities are linked to their syntax source, i.e. their *origin*. That way, any change in the document results in semantic objects for which origins have been deleted or changed as well as syntax objects for which there does not exist corresponding semantic entities yet. The semantic objects are marked with this status information ("deleted", "added", "preserved"). This information is then exploited by analysis rules to compute the ripple effects of the changes on the semantics entities, which in a final stage are used to annotate the syntax parts, that is the documents. The *GMoC* tool is build on top of the graph rewriting tool *GrGen.NET* [10] and is parameterized over document type specific document meta-models and graph rewriting rule systems to extract the semantics and to analyze the impact of changes.

Document Meta-Models. To provide change impact analysis for PLANETARY, we developed a document meta model and graph impact analysis rules for OMDoc. The document meta model consists of a lightweight ontology of the relevant semantic concepts in OMDoc documents, — e.g., theories, symbol declarations and their occurrences, axioms, definitions, assertions, and their use in proofs and proof steps — together with semantic relations between concepts — e.g., import relations between theories, symbols and their definitions, assertions and their proofs. Note that the OMDoc meta-model abstracts over the OMDoc surface syntax. For instance, a definition can either be a definition-element

```
<symbol name=''unit''>
<definition xml:id="mon−d1" for="unit" type="informal">
 <CMP>
  A structure [(M,*,e)], in which [(M,*)] is a semi−group with unit [e] is called monoid.
 </CMP>
</definition>
```

where the symbol defined by the definition is given by the for attribute of the definition (boxes abbreviate OpenMath content here). The symbol itself is declared in a different element. This kind of definition typically occurs when OMDoc documents are created manually or obtained from formal representations. Alternatively, a definition can come as a "typed" omtext such as

```
<omtext type="definition" xml:id="binary−tree.def" about="#binary−tree.def">
  <CMP xml:id="binary−tree.def.CMP1" about="#binary−tree.def.CMP1">
    <p xml:id="binary−tree.def.CMP1.p1" about="#binary−tree.def.CMP1.p1">
      A <term cd="balanced−binary−trees" name="binary−tree"
                     role="definiendum">binary tree</term> </p> </CMP>
</omtext>
```

which typically happens, for instance, when generating the OMDoc files from an sTeX source file. Note that in this case the defined symbol is declared by the term element with role="definiendum". The fact that this definition defines that symbol comes from the structural nesting of the term inside the definition. Similar examples are theories which can either be imported into each other by using the explicit imports elements or simply by nesting theory-elements.

Conceptually, it does and should not matter in which form symbols and definitions are given in, and a mixture of both forms is also desirable to support the linking of mathematical content in OMDoc from different authoring sources. The document meta model declares these pure concepts and relations like an ontology. The intentional semantics of a given OMDoc document is a set of instances of these concepts and relations. The graph rewriting tool *GrGen.NET* is used as a reasoning framework and supports hypergraphs with typed nodes and edges. The types are simple types with sub-typing relations. This is exploited to subdivide the whole graph in a syntax and a semantic subgraph by introducing top-level types for either part. The OMDoc syntax elements are declared as subtypes of the syntax type and the OMDoc document being an XML tree can then naturally be represented as (syntax) nodes and relations. Analogously, the semantic concepts and relations from the OMDoc document meta-model are simply declared as subtypes of the semantic types.

Abstraction Phase of CIA. The abstraction phase of the impact analysis for OMDoc documents consists of extracting the intentional semantics from the given OMDoc documents. This is realized by a set of graph rewriting rules which analyse the OMDoc document to extract the semantic concepts and relations, and mark them as being added. Examples of such rules are the two left-most rules in Fig. 4 to extract definitions from "typed" omtext: The graph rewriting rules are named (e.g., `FindNewDefinition`) and have a left-hand side (the box labelled by `L`) indicating the pattern to match in a subgraph and a right-hand side (the box labelled `R`) by what the instantiated subgraph pattern is replaced. Identical

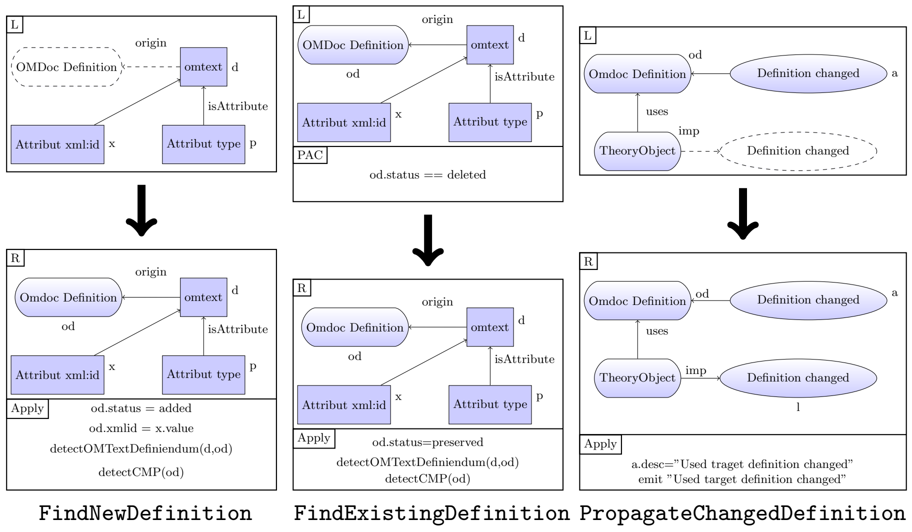


Fig. 4. Two Abstraction Rules and one Propagation Rule written top-down; we use rectangles for syntax nodes, rounded rectangles for semantic nodes and ellipses for impact nodes.

graph nodes and edges are additionally labelled by names, such as $x, d, p, od, \ldots$. Further conditions that must be satisfied to enable the graph rewriting step are positive application conditions (PAC), which must hold on the graph before rule application and negative application conditions (the dashed nodes and edges in the left-hand sides `L` or in extra NAC boxes—not used here) which must be false on the graph before rule application. These conditions can be graph patterns as well as boolean tests on attribute values. The application of the graph rewriting rule replaces the subgraph in `L` with the subgraph in `R` and additional adaptations can be triggered in the `Apply` part, such as adapting the value of attributes but also invoking further graph rewriting rules using their name (e.g., detectCMP).

The rules for the abstraction phase always come in two variants: one variant is for syntactic omtexts, for which there does not exist yet a semantic object in the semantic graph—for these, new semantic instances are introduced, marked as added and the origin of the semantic concept is represented explicitly by an Origin edge from the semantic node to the syntax node. The second variant is for syntactic omtexts, for which there exist already a semantic object in the semantic graph from a previous version of the document—for these, the semantic instances are maintained and marked as preserved. Both rules invoke further analysis rules to analyse the "body" of a definition, in order to find out whether the definition has changed (e.g., detectCMP). All semantic objects that are neither added nor preserved are marked as deleted by a generic rule operating over all semantic nodes and edges. Overall we have designed 91 rules for the abstraction phase that synchronizes OMDoc documents with their intentional semantics.

```
<impacts>
  <impact for="binary−tree.def" name="Definition␣changed"
          select="⟨xpath-to-definition-binary-tree⟩"/>
  <impact for="balanced−binary−tree.def" name="Definition␣Binary␣Tree␣changed"
          select="⟨path-to-definition-balanced-binary-tree⟩"/>
  <impact for="size−lemma−pf.derive2.method2.proof2.derive4.CMP1.p1.term2"
          name="Definition␣Binary␣Tree␣changed"
          select="⟨path-to-inproof-reference-to-balanced-binary-tree⟩"/>
</impacts>
```

Fig. 5. Example Impact Annotation File

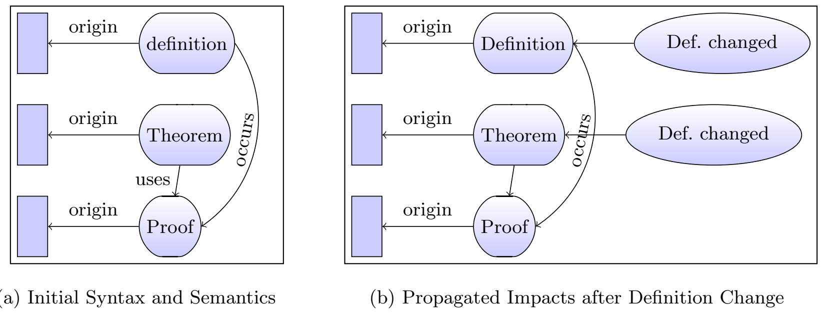


Fig. 6. Change Impact Analysis Phases

Propagation Phase of CIA. The second, so-called *propagation* phase, analyses the semantic graph and exploits the information about semantics objects and relations being marked as added, deleted or preserved to propagate the impact of changes through the semantic graph. Impacts are a third type of nodes, different from the syntax and semantic nodes. They contain a human-oriented description of the impact and can only be connected to semantic nodes. For instance, we have one marking a definition for some symbol, say f, as being changed, when its body has changed. Furthermore, we have rules that propagate that information further to definitions, that build upon f or proofs using that definition (see right-most rule of Fig. 4 for an example). Overall, we have 15 rules to analyse and propagate the impacts.

Projection Phase of CIA. Finally, we have the *projection* phase which essentially consists of one generic rule that projects the impact information of the semantic nodes backwards along the origin links to the syntactic node and creates a corresponding impact annotation for the syntactic part of the documents. The impact annotations are output in a specific XML format, where an impact annotation refers to the xml:id of the OMDoc content element in its for attribute and the name attribute contains the human-oriented description of the impact. For our running example we obtain the impact shown in Fig. 5.

4.2 Change Impact Analysis Workflow

The workflow inside *DocTIP* for the change impact analysis is to initially build up a semantics graph for all documents that shall be watched by *DocTIP*. For our

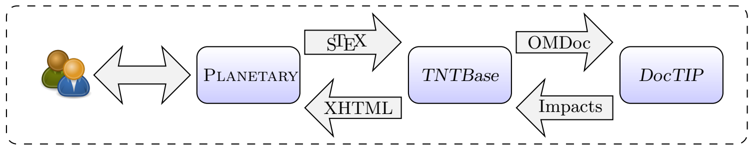


Fig. 7. Tool Chain

running example, the relevant parts of the initial graph is given in Fig. 6(a). Upon a change in some document, a semantic difference analysis between the old and the new OMDoc documents is performed, which results in a minimal change description on an appropriate level of granularity. This is also provided by *GMoC*, which includes a generic semantic tree difference analysis algorithm parameterized over document-type specific similarity models. The computed changes are applied to the syntactic document graph and subsequently the impact analysis rule systems of the three phases *abstraction*, *propagation*, and *projection* are applied exhaustively in that order.[4] For our running example we obtain a graph of the form in Fig. 6(b). As a result, the *DocTIP* system returns the computed impacts in the XML format described before for *all* documents it is maintaining (not only for the document that caused the change).

5 System Architecture

In order to add change management support for the workflows, we consider the architecture of the Planetary system (see Fig. 7). The user interacts with the Planetary system via a web browser, which presents the mathematical knowledge items based on their XHTML+MathML presentation in a WiKi-like form. The XHTML+MathML documents are rendered from content oriented mathematical knowledge items in OMDoc format. Along with the XHTML+MathML document versions, the Planetary system maintains the original sTeX document snippets, which the author can edit in the web browser. The OMDoc documents are maintained in the *TNTBase* repository together with their original sTeX source snippets. Any change in the OMDoc documents in *TNTBase* results in an update of the corresponding knowledge items in the Planetary system after rendering the OMDoc in XHTML+MathML. Upon edit of the sTeX snippets in the Planetary system, a new OMDoc is created from the sTeX sources [7] and pushed into *TNTBase*, which returns the XHTML+MathML presentation.

The *TNTBase* [24] is a Subversion based repository for normal files as well as XML files. It behaves likes a normal Subversion repository, but offers special support for XML documents by storing the revisions in a XML database. By

[4] Termination must be ensured by the designer of the rules systems. However, the *GrGen.NET*-framework comes with a strategy language, that allows for a fine-grained control over the rule executions, which helps a lot for designing the strategies of the different phases. It is also used to sequentialize the three phases.

this it allows efficient access via XQueries to XML objects. Moreover, it provides the concept of storing answers to queries as virtual files, which can be used like normal files, but any change in such a virtual file is automatically propagated to the original XML documents the affected parts came from. Finally, it allows the definition of document specific presentation routines, such as the XHTML+MathML rendering of OMDoc documents. For its role as repository for the PLANETARY system, it is important to note that the sTEX snippets and the corresponding OMDoc documents are stored together in the same directory in *TNTBase*, such as, for instance,

- the file balanced−binary−trees.tex that contains the source of Fig. 2, and
- balanced−binary−trees.omdoc that contains the OMDoc transformation.

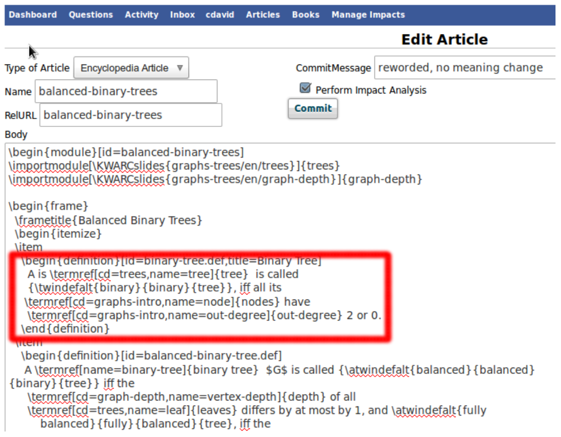
To add change management support, we connected the *DocTIP*-system to the *TNTBase* which returns impact information in form of annotations to the OMDoc documents, which are stored in the *TNTBase* as an extra file together with the OMDoc and the sTEX files, but with the extension ".imp", such as

- balanced−binary−trees.imp (in the XML format shown in Fig. 5).

Like the change in the OMDoc file, any change in the impacts file is forwarded as is by *TNTBase* to the PLANETARY system. The rendering of OMDoc in XHTML+MathML preserves the xml:id. Therefore, the PLANETARY assigns the impacts to the XHTML+MathML snippets using the for-attributes and presents on the WiKi-page.

6 Example Revisited

To see how the parts of the system interact, let us revisit the example from Section 3. Say the user found a typo in the binary trees module in Fig. 2. She opens the web editor and corrects it, and submits the changed module (see Fig. 8; note that the user requested a change impact analysis). The system communicates the changes to *DocTIP* (via *TNTBase* as described in the previous section), which determines the list of impacts based on the semantic relations described in Section 3. *DocTIP* in turn communicates them to *TNTBase*, which stores them for further reference and passes them on to the PLANETARY system. Moreover, it notifies the user about impacts by updating the superscript number on the "Manage impacts" field in the top menu bar (see Fig. 9). If the user decides to act on the impacts, she gets the impact resolution dialog in Fig. 9, which has a tab for every module that is impacted by the change. Note that the module is given to the user in its presented form as this is the most readable view, and, furthermore,

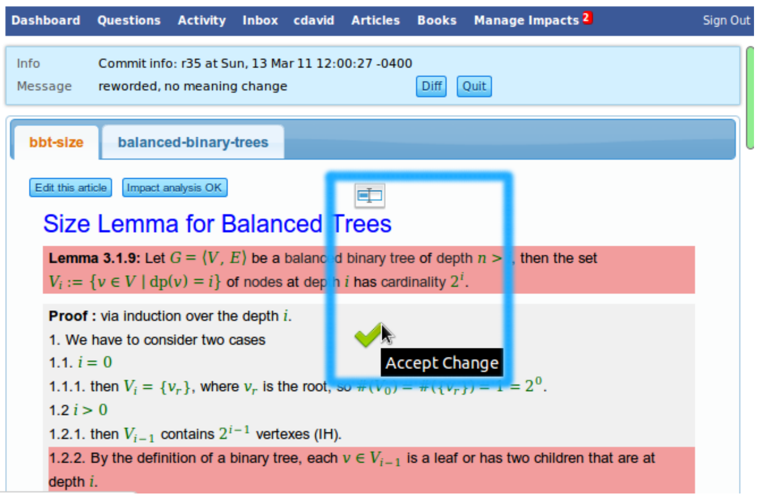


Type of Article: Encyclopedia Article
Name: balanced-binary-trees
RelURL: balanced-binary-trees
CommitMessage: reworded, no meaning change
[✓] Perform Impact Analysis
Body

```
\begin{module}[id=balanced-binary-trees]
\importmodule[\KWARCslides{graphs-trees/en/trees}]{trees}
\importmodule[\KWARCslides{graphs-trees/en/graph-depth}]{graph-depth}

\begin{frame}
 \frametitle{Balanced Binary Trees}
 \begin{itemize}
 \item
  \begin{definition}[id=binary-tree.def,title=Binary Tree]
   A is \termref[cd=trees,name=tree]{tree}  is called
   {\twindefalt{binary}{binary}{tree}}, iff all its
  \termref[cd=graphs-intro,name=node]{nodes} have
   \termref[cd=graphs-intro,name=out-degree]{out-degree} 2 or 0.
  \end{definition}
 \item
  \begin{definition}[id=balanced-binary-tree.def]
   A \termref[name=binary-tree]{binary tree}  $G$ is called {\atwindefalt{balanced}{balanced}
{binary}{tree}} 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 \atwindefalt{fully
      balanced}{fully}{balanced}{tree}, iff the
```

Fig. 8. Committing Changes in PLANETARY



Info Commit info: r35 at Sun, 13 Mar 11 12:00:27 -0400
Message reworded, no meaning change

Size Lemma for Balanced Trees

Lemma 3.1.9: Let $G = \langle V, E \rangle$ be a balanc d binary tree of depth $n >$, then the set $V_i := \{v \in V \mid \mathrm{dp}(v) = i\}$ of nodes at dep h i has cardinality 2^i.

Proof : via induction over the depth i.

1. We have to consider two cases

1.1. $i = 0$

1.1.1. then $V_i = \{v_r\}$, where v_r is the root, so $\#(V_0) = \#(\{v_r\}) = 1 = 2^0$.

1.2 $i > 0$

1.2.1. then V_{i-1} contains 2^{i-1} vertexes (IH).

1.2.2. By the definition of a binary tree, each $v \in V_{i-1}$ is a leaf or has two children that are at depth i.

Fig. 9. The Impact Resolution Dialog

we can use the identifiers in the impacts (see Fig. 5) to highlight the objects affected. For each of the affected objects, the user can then either discard the impact information if it was a spurious impact (via the checkmark icon in the "Accept Change" box) or edit the source of the impacted object (via the "edit" icon in the box) and mark it as resolved afterwards. Alternatively, she can use the action links above to make changes at the level of the whole module. Note that a conventional conflict resolution dialog via three-way merge as we know it from revision control systems does not apply to this situation, since we only have to deal with "long-range conflicts", i.e., impacts between different objects, not conflicting changes to a single object. When the user quits the impact resolution dialog, then all discarded and resolved impacts will be communicated to *TNTBase* together with the changes, which updates the set of tabled impacts and communicates the changes to *DocTIP* for a further round of CIA. Note that the storage of tabled impacts in *TNTBase* (the additional ".imp" files) makes the change management workflow more flexible over time. The need for this was not anticipated before the integration and triggered a re-design of the system functionality.

7 Related Work

In the context of software development, there exists several methods that estimate the scope and complexity of a change of a piece of software with respect to other modules and documentation, known as *software change impact analysis*. The methods are usually based on modeling data, control, and component dependency relationships within the set of source code. Such relationships can be automatically extracted using well-known techniques such as *data-flow anal-*

ysis [23], *data dependency analysis* [12], *control flow analysis* [16] , *program slicing* [15], *cross referencing* and *browsing* [2], and logic-based *defects detection* and *reverse engineering algorithms* [9]. From an abstract point of view, we have a similar set-up as we extract and collect relevant information and their dependencies in the semantical extension of the document. For example, the process of extracting dependencies between definitions, axioms, and theorems and their uses in proofs can be seen to be similar to a data-flow or dependency analysis for software. However, on the concrete level, our approach differs because the flexiformal documents we deal with do not have a formal semantics as software artefacts. Indeed, we cannot directly interpret the textual parts of STEX documents, but have to rely on the STEX markup manually provided by the author. Thus, the impact analysis can always only be as accurate as the manual annotations are. Furthermore, not having a formal semantics at hand, we cannot automatically check if a certain change really has an impact on other parts. In order to be "complete", we have to follow a possibilistic approach to propagate impacts and thus may get false positives, i.e., spurious impacts. Since impact information for some parts may trigger further impact propagations (due to the possibilistic approach), this may result in many spurious impacts in principle. For this a dependency management on impacts nodes themselves can be used (by adding dependency links between impacts in the rules) in order to propagate the deletion of spurious impact information by the user.

Requirement tracing [11] is the process of recording individual requirements, linking them to system elements, such as source code, and tracing them over different levels of refinement. Several tools have been developed to support requirement tracing, such as the *Doors* system [17]. Within our setting the change of an object, e.g., a definition, gives rise to an impact, such as to revise the proof of a theorem. Similar to requirements, these impacts are linked to concrete objects and may depend on each other. Also similar is that requirements are formulated in natural language and the requirements tracing system has no access to the semantics, hence also has to follow a possibilistic approach. Of course the type of relationships between requirements are tailored to that domain in requirement tracing as they are in our case. The main difference is that with our approach the relationships are not built into the tool, but can be defined externally in separate rule files. This allows the addition of new relationships, types of impacts and propagation rules, for instance in order to accommodate the various extensions of OMDoc like for exercises, but also for didactic knowledge. This will enable to add change impact analysis for E-learning systems like ActiveMath [22], that are based on OMDoc with their own didactic extensions and that lack change impact analysis support for the authors of course materials.

8 Conclusion

We have presented an integration of a management of change functionality into an active document management system. The combined system uses the semantic relations that were originally added to make documents interactive to propagate impacts of changes and ultimately help authors keep the collections of source modules consistent. The approach is based on impact analysis via graph rewriting rule systems for a core of the OMDoc format. CIA support for extensions of that core OMDoc can easily be added on demand and, due to the generic nature of impact descriptions and their handling in PLANETARY, the presentation module does not need to be adapted.

One limitation of the current integration that we want to alleviate in the near future is that our integration currently assumes a single-user mode of operation, as we have no means yet to consistently merge the three kinds of documents (sTEX, OMDoc and impacts file). Moreover, multiple users working on different branches that are partly merged on demand are also not supported yet. One of the main conceptual problems to be solved here is how to deal with propagating changes by "other authors". For that we plan to build in the notion of versioned links proposed in [13]

References

[1] Serge Autexier and Normen Müller. "Semantics-Based Change Impact Analysis for Heterogeneous Collections of Documents". In: *Proceedings of 10th ACM Symposium on Document Engineering (DocEng2010).* Ed. by Michael Gormish and Rolf Ingold. Manchester, UK, 2010.

[2] Shawn Anthony Bohner. "A graph traceability approach for software change impact analysis". PhD thesis. Fairfax, VA, USA: George Mason University, 1995.

[3] Catalin David et al. "eMath 3.0: Building Blocks for a social and semantic Web for online mathematics & ELearning". In: *1^{st} International Workshop on Mathematics and ICT: Education, Research and Applications.* (Bucharest, Romania, Nov. 3, 2010). Ed. by Ion Mierlus-Mazilu. 2010.

[4] *Deolalikar P vs NP paper.* URL: `http://michaelnielsen.org/polymath1/index.php?title=Deolalikar_P_vs_NP_paper&oldid=3654` (visited on 11/03/2010).

[5] *DocTIP: Document and Tool Integration Platform.* URL: `http://www.informatik.uni-bremen.de/agbkb/forschung/formal_methods/DocTIP/` (visited on 03/13/2011).

[6] *General Computer Science: GenCS I/II Lecture Notes.* `http://gencs.kwarc.info/book/1`. Semantic Course Notes in Panta Rhei. 2011.

[7] Deyan Ginev, Heinrich Stamerjohanns, and Michael Kohlhase. "The LaTeXML Daemon: A LaTeX Entrance to the Semantic Web". 2011.

[8] Dieter Hutter. "Semantic Management of Heterogeneous Documents (Invited Talk)". In: *Proceedings Mexican International Conference on Artificial Intelligence (MICAI-2009).* LNAI 5845. Springer, 2009, pp. 1–14.

[9] Yih-Feng Hwang. "Detecting faults in chained-inference rules in information distribution systems". PhD thesis. Fairfax, VA, USA: George Mason University, 1998.

[10] Edgar Jakumeit, Sebastian Buchwald, and Moritz Kroll. “GrGen.NET”. In: *International Journal on Software Tools for Technology Transfer (STTT)* 12.3 (2010), pp. 263–271.

[11] M. Jarke. “Requirements Tracing”. In: *Communication of the ACM* 41.12 (1998).

[12] J. Keables, K. Roberson, and A. von Mayrhauser. “Data Flow Analysis and its Application to Software Maintenance”. In: *Proceedings of the Conference on Software Maintenance*. Los Alamitos, CA.: IEEE CS Press, 1988, pp. 335–347.

[13] Andrea Kohlhase and Michael Kohlhase. “Maintaining Islands of Consistency via Versioned Links”. submitted. 2011.

[14] Michael Kohlhase et al. “The Planetary System: Web 3.0 & Active Documents for STEM”. In: accepted for publication at ICCS 2011 (Finalist at the Executable Papers Challenge). 2011.

[15] Bogdan Korel and Janusz Laski. “Dynamic slicing of computer programs”. In: *The Journal of Systems and Software* 13.3 (1990), pp. 187–195. ISSN: 0164-1212. DOI: `http://dx.doi.org/10.1016/0164-1212(90)90094-3`.

[16] Joseph P. Loyall and Susan A. Mathisen. “Using Dependence Analysis to Support the Software Maintenance Process”. In: *ICSM ’93: Proceedings of the Conference on Software Maintenance*. Washington, DC, USA: IEEE Computer Society, 1993, pp. 282–291. ISBN: 0-8186-4600-4.

[17] rcm2 Ltd. *DOORS - Dynamic Object-Oriented Requirements System*. `http://www.rcm2.co.uk`.

[18] Normen Müller. “Change Management on Semi-Structured Documents”. PhD thesis. Jacobs University Bremen, 2010.

[19] *Planetary Developer Forum*. URL: `http://trac.mathweb.org/planetary/` (visited on 01/20/2011).

[20] *PlanetMath.org – Math for the people, by the people*. URL: `http://planetmath.org` (visited on 01/06/2011).

[21] *`arxiv.org` e-Print archive*. URL: `http://www.arxiv.org` (visited on 01/08/2010).

[22] *The ActiveMath System*. URL: `http://www.activemath.org/` (visited on 03/11/2011).

[23] Lee J. White. “A Firewall Concept for both Control- Flow and Data-Flow in Regression Integration Testing”. In: *IEEE Trans. on Software Engineering* (1992), pp. 171–262.

[24] Vyacheslav Zholudev and Michael Kohlhase. “TNTBase: a Versioned Storage for XML”. In: *Proceedings of Balisage: The Markup Conference 2009*. Balisage Series on Markup Technologies. available at `http://kwarc.info/vzholudev/pubs/balisage.pdf`. Mulberry Technologies, Inc., 2009.