Research Proposal for a Ph. D. thesis

Towards an Ontology-Driven Management of Change

Normen Müller
June 26, 2006

School of Engineering and Science,
International University Bremen,

n.mueller@iu-bremen.de

Abstract. I want to work on formal management of change built on top of informal document engineering processes. The key features of the proposed system are ontological relations between (informal) document formats, extending document states by a concept of variants, classification of change relations, and a calculus for reasoning on changes.

The main purpose is to facilitate and enhance collaboration on large document collections and therefore to improve information consistency, reuse and distribution.

If I had eight hours to chop down a tree,
I'd spend six sharping my ax.

— *Abraham Lincoln*

Contents

1 Introduction

We live in the information age: Huge amounts of information are available at our fingertips and computers influence every aspect in life. In particular we have to deal with e-documents everywhere. *Document engineering*,

> is the computer science discipline that investigates systems for documents in any form and in all media. As with the relationship between software engineering and software, document engineering is concerned with principles, tools and processes that improve our ability to create, manage, and maintain documents [Doc06].

Of this broad field only small parts have found their way into practice, e.g. *document management systems* (DMS). Current DMS are designed to coordinate the collaborative creation and maintenance process of documents through the provision of a centralized repository. The focus is primarily on the documents themselves. Relations between and within as well as effect of changes on these relations are largely neglected, although information reuse and distribution could seriously benefit by such relation management. Therefore human reviewers are needed for *management of change* (MoC), i.e., to maintain consistency after modifications. A costly, tedious, and error-prone factor in document life-cycle that is often neglected to cut cost leading to sub-optimal and often disastrous results.

1.1 A Running Example

To sharpen our intuition about the issues involved let us consider the following situation (Figure 1): Immanuel — a coauthor of a technical report $\mathcal{R}$ — is responsible for some sections therein. He starts writing with some fundamentals $\boxed{1}$ and then builds on that: $\boxed{2} \rightarrow \boxed{1} \leftarrow \boxed{3}$. To enable other authors and interested parties to review and reuse his work he commits $\mathcal{R}$ to a shared DMS. Andrea — a division leader, reporting the work of her group to a client — accesses the DMS and obtains a working-copy of $\mathcal{R}$. She decides to set up some slides $\mathcal{S}$ based on Immanuel's parts of $\mathcal{R}$ in a different order. After a while Immanuel's coauthor Michael checks out the current version of $\mathcal{R}$. He notices some discrepancies within $\boxed{1}$, modifies it to his satisfaction yielding $\boxed{1}$, and commits his revision back to the DMS.

In current DMS this is were the story ends and the problems start:

P1 Do the modifications of $\boxed{1}$ conflict with the unchanged $\boxed{2}$ and $\boxed{3}$? So do Michael or Immanuel also have to modify $\boxed{2}$ and $\boxed{3}$?

P2 What sort of modifications did Michael perform, i.e., did he modify the meaning, the layout or did he just correct some typos?

P3 How will Andrea be get informed so that she does not miss-represent the state of affairs?

P4 Does Andrea actually need the modified version of $\boxed{1}$?

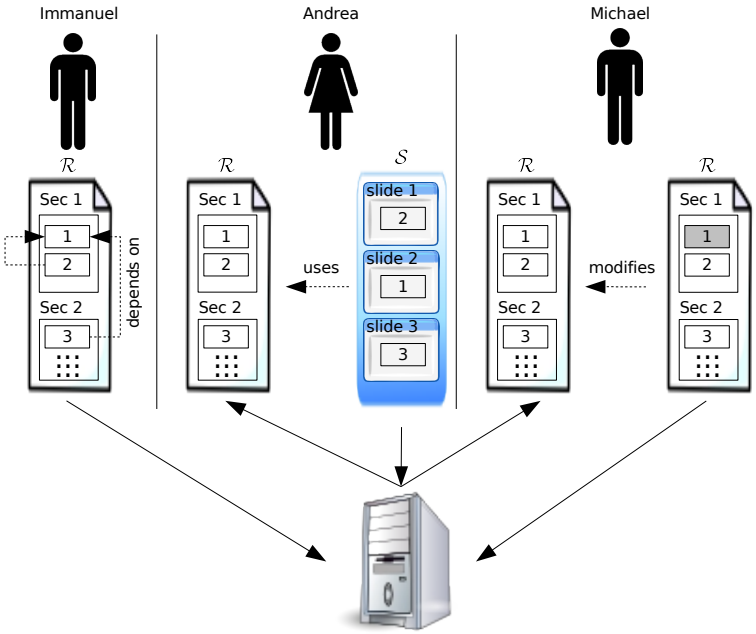

Figure 1: Actual state of DMS

Recapitulating the problem:

Relations between and within the documents are not represented in current DMS*s* (†)

i.e., copies of $\mathcal{R}$ do not display the fact that $\boxed{2}$ and $\boxed{3}$ depend on $\boxed{1}$ and copies of $\mathcal{S}$ do not display the fact that $\mathcal{S}$ uses $\mathcal{R}$ and $\boxed{1}$, $\boxed{2}$, and $\boxed{3}$ in particular.

Thus current DMSs do not solve (**P1**) – (**P4**)! Immanuel would have to contact Michael to get detailed information of the applied modifications or he would have to completely re-read $\boxed{1}$ and verify on his own if the modifications are in conflict with $\boxed{2}$ or $\boxed{3}$. So this work-flow becomes tedious and error-prone. In particular there is still the open question: Who informs Andrea? Neither Immanuel nor Michael are aware of the fact Andrea is setting up some slides partially based on their technical report. Thus, Andrea has to inform herself, i.e., continuously check the state of $\mathcal{R}$ and verify by herself if, regarding her slides, the applied modifications are significant.

To avoid these inefficiencies, conflicts, and delays, and to emphasize the importance of common information spaces in decentralized working environments the integration of a system support into DMSs to manage modifications as well as relations is indispensable.

1.2 The Approach in this Thesis

In my thesis I will develop an ontology-driven management of change integrated into *informal* document engineering processes. Before I go into technicalities I will give a short survey of my proposal:

A Structured View of Documents I propose to base MoC, information reuse, and consistency on a *structured view of documents* (cf. section 4). In this context I regard documents as structured compositions of *information units*. To identify information units as well as to define *non-grammatical relations* (cf. subsection 4.1) between them I base my approach on knowledge representation methods, in particular on the notion of a *system's ontology*[1] [KBM06]. To foster collaboration and reuse I separate documents into two layers (cf. section 4.2) keeping both under version control but extending the well-known concept of versions and revisions by a concept of *variants* (cf. section 4.3).

Reasoning on Changes I will identify changes between two *document states* (cf. section 4.3) based on a document-sensitive, structural `diff`-algorithm (cf. section 5.1). In order to compute the *long-range effect of changes* (cf. section 5.2) my proposed system will enable authors to classify the evaluated differences. Therefore I propose a *taxonomy of change relations* (cf. section 5.2.1). To systematically *reason on classified structural differences* (cf. section 5.2.2) I will develop inference rules consolidated in a *change relation calculus*.

Prototype System I will implement the MoC approach in a prototype system *`locutor`*. This implementation will progress in parallel with theory development and serves as a continual reality check to evaluate the concepts (cf. section 6).

1.3 Objectives

The objectives of my thesis work are:

- O1 Modeling system ontologies to be open to any (specific) application area.
- O2 Capturing of non-grammatical relations between information units to enable management of change "information_unit-by-information_unit" rather than "line-by-line".
- O3 Computation of effects of changes subject to classified change relations, i.e., identification of semantic long-range conflicts.
- O4 Identifying exactly *when*, *where*, *why*, and *by what* updates corrupt documents w.r.t. grammatical and non-grammatical relations.

[1] This is an ontology describing the data model of a representation format independently of its respective syntactical realization.

O5 Extension of document states by a second dimension, i.e., to consider not only different versions and revisions of information units — the first dimension — but also different variants.

O6 Integration of management of change into arbitrary DMS without requiring adaptations to document engineering processes, i.e., authors are not required to adapt their editing practices.

Summary, I hope to seriously facilitate information consistency, reuse, and thus information distribution by implementing a management of change regarding the complex relations between document versions, revisions, and variants.

2 Related Work

SUBVERSION [SVN06] is a free/open-source version control system for collaborative development. It maintains a history of file and directory versions. The files and directories are checked out of the repository into a local project work area. This called the "working directory". Changes are made to files in the "working directory". After changes are made to create the next working version, the files are checked back in to the repository. In order to achieve convergence for all working directories the system propagates corresponding `diff`-scripts during the next update-cycle.

SUBVERSION like functionality constitutes the minimal requirement I expect of any repository to integrate formal management of change into. However, instead of this file- and line-based approach I regard documents as structured compositions of information units and consider the dependencies between (fragments of) documents. Thus instead of merely revealing local conflicts, i.e., to identify conflicting lines within files, my approach will be able to reveal long-range conflicts, i.e., to identify conflicting information units cause of existing dependencies between them.

The CDET [SBRS03] system is concerned with consistent document engineering. The user can stipulate external consistency rules such that the system is able to capture informal consistency requirements. In case of rule violation the system generates consistency reports as suggestion DAGs (short: S-DAGs). These S-DAGs provide a convenient way to visualize inconsistencies and repair actions.

This approach does neither consider a system of variants nor any management of change. The only document states are versions and revisions. Inconsistencies are pointed out, but the source of the underlying modification is neglected. The system is not aware of information units for a fine-grained information management. Furthermore, the formal consistency rules have to be explicitly specified rather than implicitly inferred based on ontologies and change relations.

[KA03] and [EK04] propose a collaborative content management system for distributed mathematical knowledge base systems. This system is based on the version control model of the CVS system [CVS], but substitutes structural semantic versions for `diff`, `patch`

and `merge` used in CVS. Primarily this approach expands a grammar by an equality theory to compute less intrusive edit scripts[2], but lacks any management of change.

Thus, as to the fact that equality theory is one main aspect in management of change, I will base the computation of structural differences on this approach.

Some initial research has been conducted on methods and tools managing the consistency and change of documents: For formal documents like programs or specifications I refer to the HETS [Het] and MAYA [May] systems. For informal documents like mathematical textbooks I refer to the MMiSS [MMi] project.

However, all these systems ([MS05, Mos05]) base their MoC on the inherent underlying (formal) mathematical structure of the documents. Thus they are restricted to specific problem domains, where we have mathematical formalizations. Thus, in order to be able to also handle (purely) formal documents I will use the insights of the these systems and discuss them in greater detail in the next section.

3 Methods

In my thesis, I will use the technique known as *development graphs* [Hut00] to manage formal documents and formal parts of documents, respectively. The notion of development graphs allows for a logical encoding of structured specifications incorporating a management of change to minimize proof work in case of changing specifications. It is already successfully implemented in the MAYA system [AHMS02] which is specialized to formal software engineering and verification.

Following the MMiSS [MMi] project, a general-purpose approach for maintaining structured documents that are semantically annotated, I will regard the concept of *variants*. This expands the application area not only "in-the-breadth" but also "in-the-depth".

For the document format, as already mentioned, I favor, w.l.o.g., XML-applications [XMLa]. I will use XML-applications not only to provide the foundation for managing structural documents but rather to manage information units of composing structural documents. The grammatical relations between information units within a document will be specified in the expressive RELAXNG [Rel] format. To compute structural differences between two documents I will on the one hand evaluate XML-`diff` tools, like the HARMONY project [Har] and the XMLDIFF project [XMLb]. On the other hand I will pursue the unification-based techniques [EK04] to approach the problem of XML difference computing.

The specification format for ontologies and change relations is a central part of the research I want to undertake.

Given that documents evolve over time, I plan to study the *theory of temporal logic* [HWZ00] and *temporal databases* [AHdB96] to find out how the applicability could be appropriated within this context, i.e., for storing, managing, and maintaining information units as well as the relations between them.

[2] A representation of the document differences.

4 A Structured View of Documents

I define w.l.o.g. a *document* as *a self-contained* XML*-based composition of information units*.

Probst et al. [PRR97] posits that to obtain meaning from a single *data* element, e.g. a formula or a quantity, we need another component: We need some *context* for its interpretation (see [KK05] for a deeper explanation). That is why "self-contained" is part of my definition.

The reason I base my definition on XML formats is twofold: On the one hand I want to foster open, structural document formats and on the other hand I want to leverage context indication in the form of content markup.

This combination of content markup and information units makes it a document by my definition.

The following sections describe how I propose to identify data elements in the notion of information units and how to define non-grammatical relations between them. Based on that I present a two-layered view of documents which I will finally expand to a *two-layered two-dimensional view*.

4.1 Informations Units and Ontological Relations

The reason why I base my approach on ontologies is to be not bound to any specific application area or specific XML application. Tom Gruber defines ontologies as "*an explicit specification of a conceptualisation*" [Gru93], borrowing from the Artificial Intelligence literature on *Declarative Knowledge*, which is concerned with the formal symbolic representation of knowledge [GN87].

Fundamental to my approach — as well as to the Declarative Knowledge approach — is the notion of *conceptualization*. That is, an abstract and simplified view of the *domain of interest*, which is being represented. This domain could be a part of reality or an entirely fictitious environment. Such a conceptualization consists of concepts that are assumed to exist in the domain of interest as well as the relationships that hold between them.

So to identify information units of composing documents, I propose the concept of *information unit*s to be part of any (user-defined) system's ontology. The elaboration of a concretion of the term "information unit" is a further part of the research I want to undertake. For the purpose of this proposal one can pragmatically think of information units as "*tangible/visual text fragments potentially adequate for reuse*" constituting the content of documents. To distinguish the term "information unit" between common speech and the ontological concept, I will call from now on the ontological concept Infom[3]. In this regard I consider any concepts "is_a"-related to the concept Infom to be an information unit.

To distinguish between *grammatical* and *non-grammatical* relations, I call the later *ontological* relations and subsume both by the term *structural* relations. The reason

[3] A little word-play to "atom". I use the word "atom" in terms of not being further divisible.

why I choose the term "ontological" is twofold: (1) In contrary to grammatical relations these relations are places within a system's ontology (2) The term "semantical" usually used at this place is for my taste too "overloaded".

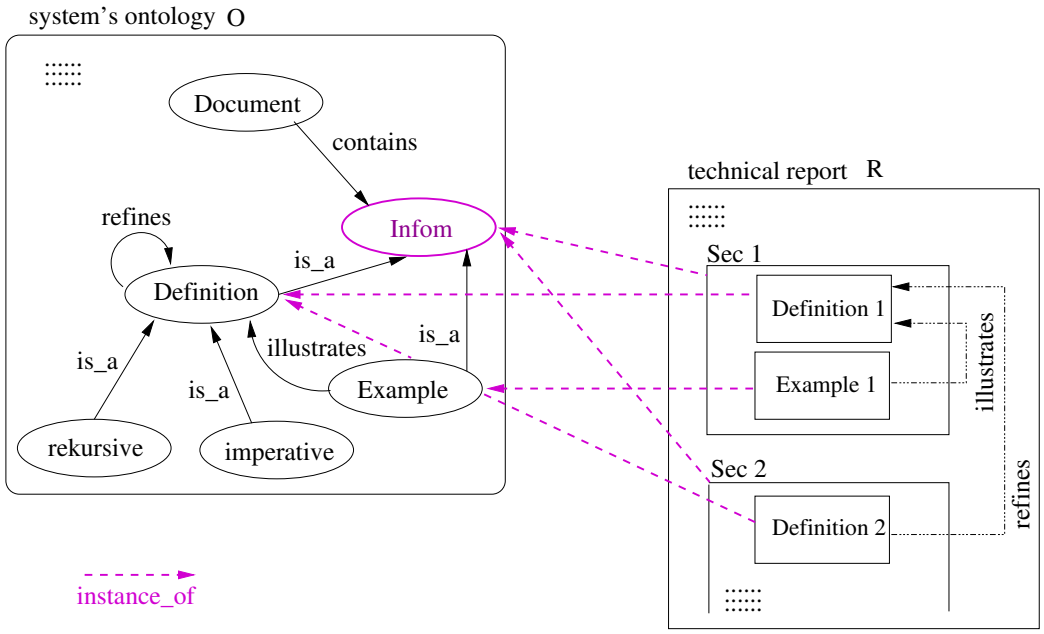

Figure 2: Information Units and Ontological Relations

To clarify the terms INFOM and *ontological relations* let us recall our running example. We presume one of the authors of the technical article $\mathcal{R}$ has established a system's ontology $\mathcal{O}$ declaring all concepts and relations of the domain of interest $\mathcal{R}$ is related to, e.g. an ontology describing the concepts of a customer requirement specification (Figure 2). Now, Immanuel does have the ability to "tag" his fragments of $\mathcal{R}$ by concepts of $\mathcal{O}$. Thus, he is able to explicitly identify information units: 1 is an individual of the concept "definition" Def, 2 is an individual of the concept "example" Ex *illustrating* the first Def, and 3 is also an individual of the concept "definition" Def but *refining* the first one. Note, regarding the pragmatic definition of information units, Immanuel is also able to "tag" grouping elements within $\mathcal{R}$, e.g. sections and paragraphs, by concepts of $\mathcal{O}$.

Thus, by making information units and relations between them explicit, we solved the former problem (†).

In the next sections I will use INFOMs and *structural relations* to establish a *two-layered, two-dimensional view* of documents.

4.2 Narrative and Content Layer

Following [VD04] and [Koh06] I separate documents into two layers: A *narrative* and a *content* layer both of which consist of INFOMs and are composed via relations. The pictorial representation of the two layers is given in Figure 3.

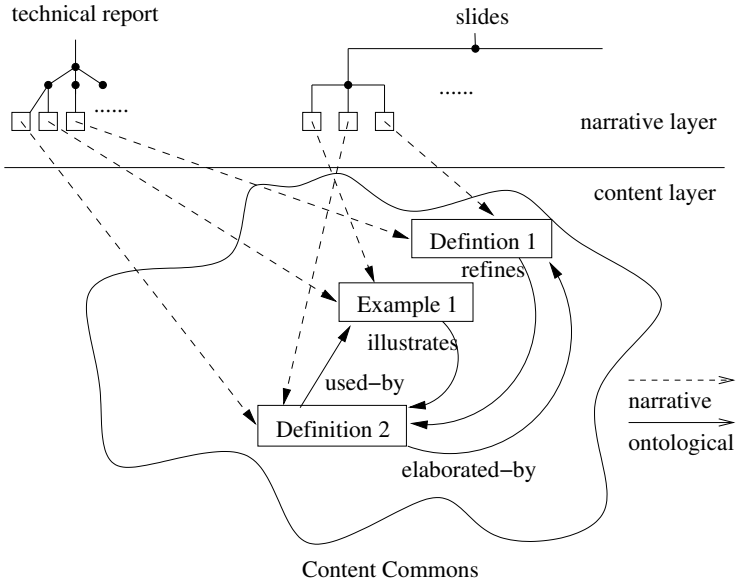

Figure 3: Narrative and Content Layer

The presentational order of information units in documents is represented on the narrative layer whereas the information units themselves and the ontological relations between them are placed in the content layer[4]. The connection between the narrative and the content layer is represented via *narrative relations* (analogous to symbolic links in UNIX). The information units and the ontological relations build up the "content commons" [CNX07]. Thus we clearly separate conceptual level from discourse presentation level.

Figure 4 consolidates the classes of relations we defined so far. Structural relations $\mathcal{SR}$ subsume grammatical $\mathcal{GR}$ and ontological relations $\mathcal{OR}$. As to the fact system's ontologies describe the data model behind the representation format the grammatical relations have to be a subset of the ontological relations. Narrative relations $\mathcal{NR}$ are controlled by structural relations, i.e., the order of referenced INFOMs is verified. For example, without a previous definition the usage of a technical term within technical report "does not make sense".

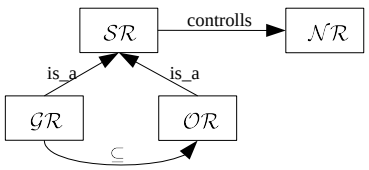

Figure 4: Taxonomy of Relations

[4]How far information units could also emerge on the narrative layer is a further research I want to undertake.

To clarify the significance of such a layered view of documents, let us go back to our running example. For simplicity we assume the initial identified information units are derived from the technical report $\mathcal{R}$. Thus Andrea — author of the slides — does not have to copy these information units but rather just "links"[5] to them. Only the new order of the old information units within the new information product is stored on the narrative layer and narrative relations refer to the respective information units already stored on the content layer.

Note, by assembling information units and respective structural relations we build up the foundations for a interdisciplinary information pool, i.e., pooling of information units related to various domains of interest. Therefore in my further research I will also investigate how to compose documents of heterogeneous[6] INFOMs to provide information harvesting at a highest level.

So up to now we have reached a *two-layered view of documents* but have neglected the *ontological relations* between the identified information units so far! Only by using this additional information we will be able to establish a consistent and expressive management of change, i.e., we will be able to handle dependencies between information units and compute effects of changes (cf. section 5). Therefore look back on the situation in our scenario Michael modifying information unit 1, say the first Def. Now he is aware of the interrelations between the different parts of $\mathcal{R}$, in particular *locutor* will notify him about the fact that 2 and 3 depend on 1. Furthermore, by recognizing the narrative relations, *locutor* can also notify Andrea about the modifications (**P3**). We will discuss how to solve (**P1**), (**P2**) and (**P4**) in section 5.

To further advance information harvesting and reuse, I will in the next section extend the well-known temporal document dimension comprising versions and revisions by a second dimension expanding the two-layered view of documents to a *two-layered two-dimensional* view.

4.3 The Concept of Variants

Following initial work in the MMiSS [MMi] project, in my approach I am also aware of the concept of *variants*. This expands the application area not only "in-the-breadth" but also "in-the-depth". Thus, by extending the well-known concept of *versions* and *revisions* by the concept of variants, the life-cycle of documents will no longer be only along a horizontal time line but also along a vertical line of variants. On the document level I call the concept of versions, revisions, and variants *document states*. I will model the concept of variants by expanding the (default) set of ontological relations by a further one called *variant-of*.

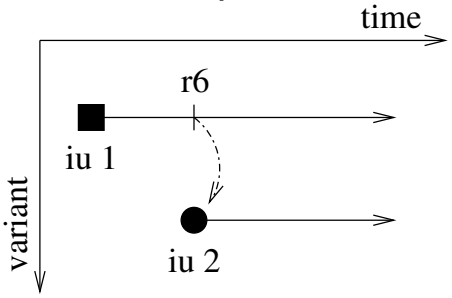

Figure 5: The Variant Dimension

[5] Concretion of "links" between entire documents is a further part of the research I want to undertake.
[6] INFOMs declared in different system's ontologies.

To demonstrate the dimension of variants in a more "dimensional" way Figure 5 depicts another possible scenario: After modifying any information unit iu_1 several times (up to revision number r_6[7]) another user or the initial user herself decides to develop a variant of iu_1. To keep it simple one can imagine iu_2 to be an "language-variant" of iu_1, e.g. iu_1 is written in English and iu_2 in German. By a user annotating information unit iu_2 to be a variant of information unit iu_1 we will be able to build up a complete management of variants, i.e the states and changes of the original information unit, the variants, and all relations between any of them will be managed as well.

To sharpen the notion of the term *variant* in our running example let us go back to Andrea. Remember she wanted to set up some slides $\mathcal{S}$ regarding Def, Ex, and Def from the technical report $\mathcal{R}$. However, in general slides represent a different, say more compact presentation of information. So Andrea will not use the InfoMs one-to-one, but rather modify them to "fit" her presentation. Figure 6 demonstrates the described situation[8]. Andrea is now able to characterize her new information units and the relations between $\mathcal{S}$ and $\mathcal{R}$ still hold. It appears that the two-layered, two-dimensional view of documents is represented by a graph consisting of a narrative layer and a content layer (NarCon).

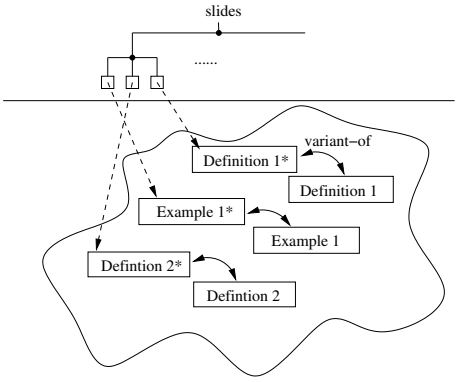

Figure 6: Variants of Infoms

Based on the arising complex network between documents and information units, respectively, I also propose to integrate value-added services into *locutor*. E.g. one of them identifies most referenced InfoMs to capture "useful" and "valuable" information units. Thus I recognize a further open research question: How to enable authors to search the content commons, i.e., how to handle the following scenario: Let there be an article $\mathcal{A}_1$ consisting of InfoMs Λ and Ω. Now another author wants to write an article $\mathcal{A}_2$ also using Λ. How do we assist the second author? Does he have to check out $\mathcal{A}_1$, copy-and-paste Λ into $\mathcal{A}_2$ and *locutor* will take care to identify that Λ is already inside the content commons? And, in particular, how does the author get to know that Λ exists, anyway? Therefore I hope the case study (cf. section 6) will uncover authors request.

Up to know we have elaborated a structured view of (informal) documents represented by NarCons and thereby already facilitated information reuse. Now, in the next section, I will describe how to develop a management of change on NarCons to achieve *consistent* information reuse, i.e., I will develop a MoC on NarCons to maintain consistency during the development of various document states.

[7] Think of the well-known Subversion work-flow.
[8] I omit further ontological links for a better readability.

5 Management of Change on NarCons

In this section I describe my first ideas towards a management of change. Thus this section is less a report on solutions, than an attempt to publicize my first suggestions towards a consistent management of change. Figure 7 depicts a survey of my proposed MoC system.

Documents will have to identify the underlying language $\mathcal{L} := \langle \mathcal{M}, \mathcal{O} \rangle$ they are an instance of: I will regard a language to be a pair consisting of a model $\mathcal{M} = (\mathcal{G}, \mathcal{E})$ and a system's ontology $\mathcal{O}$. A model $\mathcal{M}$ consists of a grammar $\mathcal{G}$ together with a equality theory $\mathcal{E}$. $\mathcal{G}$ defines the syntactical rules to build up valid document and following the initial work in [EK04] $\mathcal{E}$ defines when two NarCons are considered to be equal. Thus the system will use the model $\mathcal{M}$ to compute structural differences Δ between two document states (cf. section 5.1). Following the MMiSS project [MMi] I will use a system's ontology $\mathcal{O}$ to describe further semantic dependencies (cf. section 4.1). The *locutor* system will use this additional information to compute long-range effects of changes (cf. section 5.2). However, to operate on representatives rather than on singletons I propose a taxonomy of change relations $\mathcal{CR}$ (cf. section 5.2.1) to enable authors to classify Δ. So to systematically reason on such classified Δ (cf. section 5.2.2), i.e., to compute the structural semantic closure[9] ($\mathcal{SSC}$) ∇ I will develop inference rules consolidated in a change relation calculus $\mathcal{CRC}$.

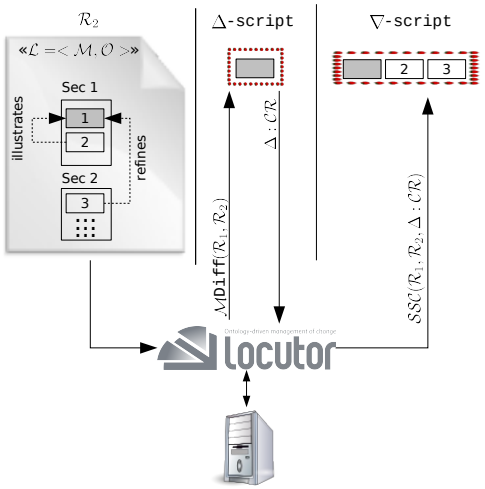

Figure 7: Management of Change

Particular I want to bring into light that annotating is rewarded by getting even more automatic assistance in the future:

> "*The flatter a document*
> *the less the assistance!*"

Figure 8, called the "The Shifting Wave", depicts this slogan. In my approach I want to lead authors to annotate informal documents step-by-step, i.e., to provide informal documents more and more with structural semantics. As a consequence of each single step the wave shifts a little bit more towards the formal world and thus can be better kept under control by formal systems, i.e., the computation of long-range

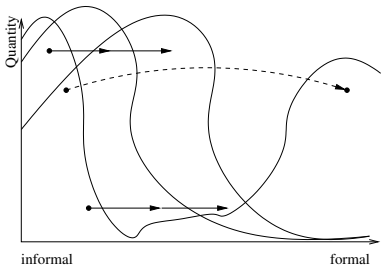

Figure 8: The Shifting Wave

[9] I use the term "structural semantics" in sense of marking-up the meaning by structure. I do not need any entailment relation to model semantics but rather concentrate just on the structure.

effects is improved. But note, I do not want to ask too much of authors all at once! It is up to an author to which level she will annotate her changes.

5.1 Computation of Structural Differences

I propose to base my computation of structural differences on the insights of XML-`diff` tools (cf. section 3) and the initial work of [EK04]. According to this I will transform `diff`–algorithms and unification-based techniques, proposed there, to operate on NarCon-graphs.

My first suggestion for such a computation of structural differences is to define a function $\mathcal{M}\texttt{Diff}$ with following signature:

$$\mathcal{M}\texttt{Diff} : \mathcal{D} \times \mathcal{D} \rightarrow \Delta$$

$\mathcal{D}$ denotes NarCon-graphs and Δ a `diff`-script comprising structural differences between NarCon-Graphs.

With "$\mathcal{M}$" in the function name I want to stress to model a stronger notion of equality to generate more compact and less intrusive edit scripts. For instance, if we know that whitespace carries no meaning in a document format, two documents are considered equal, even if they differ (with respect to the distribution of whitespace characters) in every single line; as a consequence, Δ would be empty. This motivates the following general statement of the problem at hand [EK04]:

> **The General Difference Computation Problem (DCP)**: Let $\mathcal{K}$ be a class of NarCons and an equality theory $\mathcal{E}$ on $\mathcal{K}$. Given two NarCons $\mathcal{S}$ and $\mathcal{T}$, find an optimal edit-script that transforms $\mathcal{S}$ to $\mathcal{T}$.

Particular I will engage the general DCP modulo an equality theory ($\mathcal{E}$-DCP) left unaddressed in [EK04].

To exemplify the functionality of $\mathcal{M}\texttt{Diff}$ let us go back to our running example. If we apply $\mathcal{M}\texttt{Diff}$ on $\mathcal{R}$ after the modifications initiated by Michael the output of $\mathcal{M}\texttt{Diff}(\mathcal{R}_1, \mathcal{R}_2)$ would be $\Delta = \{$  $\}$.

Up to this stage I want to point out that I did not use any ontology-based information[10], but only operate on properties defined in $\mathcal{M}$. Furthermore I want stress, that I will not handle information units in terms of a "black box", but consider changes within the inner structure as well as in the content, e.g. modifications on the actual text of Def. So one could say, that we have achieved a NarCon-based variant of Subversion so far.

But let us now consider a situation where Michael modified the meaning of Def. The output of $\mathcal{M}\texttt{Diff}$ would be same, omitting Ex and the second Def, which is incorrect.

In the next section I will explain how I propose to extend Δ to also capture the structural semantic closure of structural differences.

[10] If one wants to involve ontologies at this stage this would correspond to the creation of an ontology $\mathcal{O}$ with just a concept "document" "is_a"-related to the concept *infom*.

5.2 Computation of Long-Range Effects of Changes

By regarding *all* relations in general and the ontological relations in particular the system will be able to compute long-range effects of changes and give authors significant feedback of the impact of their modifications.

5.2.1 A Taxonomy of Change Relations

In order to be able to reason on changes, say to reason on Δ, I will develop a taxonomy of *change relations* $\mathcal{CR}$ to classify structural changes. As to the matter of fact the implementation of a automatism to classify structural changes is "AI-hard" I will enable authors to annotate Δ with $\mathcal{CR}$ (short: $\Delta : \mathcal{CR}$). Note, by this additional information about structural changes we solve (**P2**)! So we extend the *two-valued states* of changes, i.e., modified and non-modified, to *annotated two-valued states* of changes. To clarify the notion of a $\mathcal{CR}$-taxonomy I demonstrate a "first-try-example" in Figure 9.

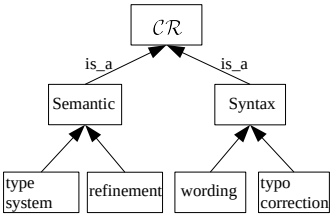

Figure 9: A $\mathcal{CR}$-taxonomy

To demonstrate the emphasis of classified change relations within my approach, let us recall our running example, again especially regarding Michael: He modified the first Def without knowing other information units depending on this one. We already solved this problem with the new view of documents and NarCon-graphs, respectively. However, so far we are only able to notify Michael and Immanuel about the fact that there are some dependencies, rather than to notify them about the effects of Michael's modifications on these dependencies. So if Michael now classifies his modifications to be syntactical, e.g. typo corrections, the system will compute and fix these changes with respect to the structural relations defined in $\mathcal{L}$, i.e., the system will merge, the typo corrections into the next document state of existing working-copies just like in the Subversion approach. If, however, Michael classifies his changes to be semantical, e.g. he changed the entirely type system of the first Def, the situation to "fix" such a modification changed! In order to compute the long-range effects of changes, say the $\mathcal{SSC}$ I will elaborate a system for *reasoning on classified structural changes*.

5.2.2 Reasoning on Classified Structural Differences

To systematically reason on annotated changes, say to reason on $\Delta : \mathcal{CR}$, I will develop inference rules consolidated in a $\mathcal{CR}$-Calculus operating on NarCons. Regarding my proposed calculus I will have a closer look at the $\mathcal{DG}$–calculus operating on development graphs (cf. section 3) to evaluate what properties and rules can be adopted for NarCon-graphs. A main aspect in this analysis will be the structural properties of development graphs and the calculus itself. Then, based on the $\mathcal{CRC}$, I propose to deduce the effects of changes on structural relations, i.e., with these "rules of re-action to changes" at hand I will define an algorithm to compute for each $\Delta : \mathcal{CR}$ (short: $\ddot{\Delta}$) the structural semantic

closure ∇. Therefore I propose another function with following signature:

$$\mathcal{SSC} : \mathcal{D} \times \mathcal{D} \times \ddot{\Delta} \to \nabla$$

Here ∇ extends Δ in sense of $\nabla := \Delta \cup \{(iu, \texttt{trace}(iu)) | iu \in \mathcal{IU}_{\mathcal{O}}\}$, where $\mathcal{IU}_{\mathcal{O}}$ denotes the set of semantically affected INFOMs and $\texttt{trace}(iu)$ represents the path of involved ontological relations.

To clarify the functionality of my suggested $\mathcal{SSC}$ function, let us again take our running example into account but now let us assume Michael changed the meaning of the first Def, e.g. he classifies his changes to be a modification to the type system of Def denoted by the $\mathcal{CR}$ concept $\mathcal{TS}$. So $\mathcal{SSC}$ would compute

$$\mathcal{SSC}(\mathcal{R}_1, \mathcal{R}_2, \blacksquare : \mathcal{TS}) = \{\blacksquare, (\text{Ex}, \textit{illustrates}), (\text{Def}, \textit{refines})\}$$

So we finally solved (**P1**) and (**P4**) and are able to give answers to the until now outstanding question "How does one Δ affect existing relations and how do existing relations affect the computation of ∇, respectively?"

As can seen from the illustrative running example the "great challenge" of my thesis is

- to define *ontological relations* for MOC, e.g. a possible additional relation might be *adapted-analogously*, to facilitate authors to augment their informal documents by more *structural semantics*
- to define proper *change relations* to "characterize" modifications
- to define a calculus parameterized by *classified change relations* operating on NARCONS

in order to compute how changes will be reflected onto the pool of information units of composing documents. I hope the result will improve consistent information reuse and distribution.

6 Case Study

I will undertake three case studies to evaluate applicability of my proposed system:

The Lecture Study A "NARCON-like" approach has already been successfully used within the sTEX project [Koh05b] to enable authors to add semantic information to documents without changing the visual appearance. A large corpus of slides for the lecture General Computer Science I & II at International University Bremen have been marked up by my supervisor MICHAEL KOHLHASE using sTEX. But the project currently lacks any management of change! So this gives me a great ability to test my suggestions on a large amount of data.

The e-Learning Study The `Connexions` e-Learning system is a rapidly growing collection of free scholarly materials and a powerful set of free software tools to help *authors* publish and collaborate, *instructors* rapidly build and share custom courses,

and *learners* explore the links among concepts, courses, and disciplines [CNX07]. As a matter of fact that during my thesis I am sponsored by the EU-project ONCE-CS [ONC05] to integrate OMDoc [Koh06] into the Connexions projects. Besides integrating my MoC into the system, I will add more structural semantics to the corpus of this projects via the OMDoc system's ontology to improve the links among concepts, courses, and disciplines.

The Wiki Study SWiM [LK07] is a semantic wiki for collaboratively building, editing and browsing a mathematical knowledge base. Its pages, containing mathematical theories, are stored in OMDoc format. This project is currently developed by Christoph Lange for his master thesis. Christoph Lange is a upcoming Ph. D. student in the KWARC group[11] and so I hope to benefit from his collaborations and the SWiM user interface on the one hand and to assist his work with my MoC on the other hand.

7 Preliminary Work Plan and Schedule

Figure 10 depicts the schedule of my preliminary work plane for my thesis. After 12 months I am well in schedule. I decided to separate my time schedule into five major elements. The acronym for these elements creates an acrostic known as RADDD, or RAD[3], a high-level process model formally identified in [Sch02]:

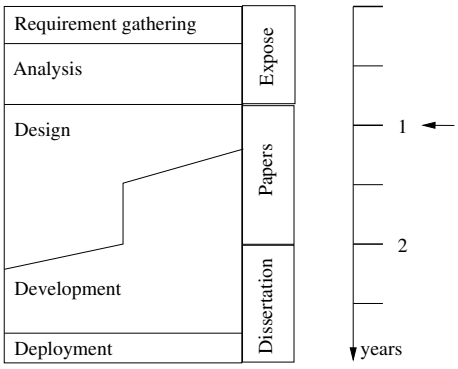

Figure 10: Preliminary schedule

Requirement gathering In the first 3 months I gathered information about requirements of a sophisticated management of change. I interviewed authors and read papers relating to information reuse and distribution. Additionally I examined existing tools, e.g. Maya, Hets, and CDET, implementing management of change for formal documents in order to acquire some insights which might be valuable for my management of change regarding informal documents.

Analysis In next phase I started to consolidate the theoretical methods for my management of change, i.e., I began to set up OMDoc system's ontology, define change relations, and to think about the $\mathcal{CR}$ calculus. In further steps I will set up an analysis in the notion of Unified Modeling Language (UML) diagrams, e.g. use case diagrams and class diagrams, to define a conceptual model based on the requirements.

Design The key to successful software development is to start with a good design. A good design captures the functional requirements of a program and describes, at a high level, the plan for achieving those requirements. So on this specification level I will do both: (1) Define formal methods (system's ontology, change relations and change relation

[11] http://kwarc.eecs.iu-bremen.de/

calculus) (2) Design a solution, say the a software plan, for my proposed system. I will publicize the formal and technical achievements out of this phase in conference papers.

Development On the implementation level I will develop and test the designed solution within the already mentioned case studies (cf. section 6). This phase interferes with the design phase, so that I am able to do straightforward testing of my proposed (theoretical) approaches. These two phases together will last to the end of my period except for 3 months.

Deployment As depicted in Figure 10 in last three months I will exclusively conclude the composition of my dissertation based on the results and papers of previous phases.

8 References

[AHdB96] Serge Abiteboul, Laurent Herr, and Jan Van den Bussche. Temporal versus First-Order Logic to Query Temporal Databases. In *ACM Symposium on Principles of Database Systems*, pages 49–57. ACM Press, 1996.

[AHMS02] Serge Autexier, Dieter Hutter, Till Mossakowski, and Axel Schairer. The Development Graph Manager MAYA (system description). In Helene Kirchner, editor, *Proceedings of 9th International Conference on Algebraic Methodology And Software Technology (AMAST'02)*, number 2422 in LNCS. Springer Verlag, 2002.

[CNX07] Connexions. Project homepage at http://www.cnx.org, seen February 2007.

[CVS] Concurrent versions system: The open standard for version control. Web site at http://www.cvshome.org.

[Doc06] The ACM Symposium on Document Engineering. Web site at http://www.documentengineering.org, seen April 2006.

[EK04] Frederick Eberhardt and Michael Kohlhase. A Document-Sensitive XML-CVS Client. unpublished KWARC blue notes, 2004.

[GN87] Michael R. Genesereth and Nils J. Nilsson. *Logical foundations of artificial intelligence*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1987.

[Gru93] Thomas R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, June 1993.

[Har] The Harmony Project. Web site at http://www.seas.upenn.edu/~harmony/.

[Het] Hets: The Heterogeneous Tool Set. Web site at http://www.informatik.uni-bremen.de/agbkb/forschung/formal_methods/CoFI/hets/.

[Hut00] Dieter Hutter. Management of Change in Structured Verification. In *Proceedings 15th IEEE International Conference on Automated Software Engineering*, number 2000 in ASE, pages 23–34. IEEE Computer Society, 2000.

[HWZ00] Ian Hodkinson, Frank Wolter, and Michale Zakharyaschev. Decidable Fragments of First-Order Temporal Logics. *Annals of Pure and Applies Logic*, 106:85–134, 2000.

[KA03] Michael Kohlhase and Romeo Anghelache. Towards collaborative content management and version control for structured mathematical knowledge. In Andrea Asperti, Bruno Buchberber, and James Harold Davenport, editors, *Mathematical Knowledge Management, MKM'03*, number 2594 in LNCS, pages 147–161. Springer Verlag, 2003.

[KBM06] Bernd Krieg-Brückner and Achim Mahnke. Semantic Interrelation and Change Management. In OMDoc – *An open markup format for mathematical documents [Version 1.2]* [Koh06], chapter 26.6, pages 274–277.

[KK05] Andrea Kohlhase and Michael Kohlhase. An Exploration in the Space of Mathematical Knowledge. In Kohlhase [Koh05a].

[Koh05a] Michael Kohlhase, editor. *Mathematical Knowledge Management, MKM'05*, number 3863 in LNAI. Springer Verlag, 2005.

[Koh05b] Michael Kohlhase. Semantic markup for TEX/LATEX. Manuscript, available at http://kwarc.info/software/stex, 2005.

[Koh06] Michael Kohlhase. OMDoc – *An open markup format for mathematical documents [Version 1.2]*. Number 4180 in LNAI. Springer Verlag, 2006.

[LK07] Christoph Lange and Michael Kohlhase. A Semantic Wiki for Mathematical Knowledge Management. In *Emerging Technologies for Semantic Work Environments: Techniques, Methods, and Applications*. Idea Group, 2007. To appear; chapters under review.

[May] MAYA: The Development Graph Manager. Web site at http://www.dfki.de/~inka/maya.html.

[MMi] MMISS: Multimedia in Safe and Secure Systems. Web site at www.mmiss.de.

[Mos05] Till Mossakowski. *Heterogeneous Specification and the Heterogeneous Tool Set*. Habilitation, Universität Bremen, 2005.

[MS05] Achim Mahnke and Jan Scheffczyk. Engineering Mathematical Knowledge. In Kohlhase [Koh05a].

[ONC05] Open Network of Centres of Excellence in Complex Systems. Web site at http://complexsystems.lri.fr/Portal/tiki-index.php, 2005.

[PRR97] G. Probst, St. Raub, and Kai Romhardt. *Wissen managen*. Gabler Verlag, 4 (2003) edition, 1997.

[Rel] A Schema Language for XML. Web site at http://www.relaxng.org/.

[SBRS03] Jan Scheffczyk, Uwe M. Borghoff, Peter Rödig, and Lothar Schmitz. A Comprehensive Description of Consistent Document Engineering. Report, University of the Federal Armed Forces Munich, November 2003.

[Sch02] Joseph Schmuller. *SAMS Teach Yourself UML (Second Edition)*. SAMS, 2002.

[SVN06] The Subversion Project. Web site at http://subversion.tigris.org/, seen August 2006.

[VD04] Katrien Verbert and Erik Duval. Towards a Global Component Architecture for Learning Objects: A Comparative Analysis of Learning Object Content Models. In *Proceedings of the EDMEDIA 2004 World Conference on Educational Multimedia, Hypermedia and Telecommunications*, pages 202–208, 2004.

[XMLa] Extensible Markup Language (XML) 1.0 (Third Edition). Web site at http://www.w3.org/TR/REC-xml/.

[XMLb] The XMLDiff Project. Web site at https://xmldiff.dev.java.net/.