Dimensions of Formality: A Case Study for MKM in Software Engineering

Andrea Kohlhase[1] and Michael Kohlhase[2] and Christoph Lange[2]

[1] German Research Center for Artificial Intelligence (DFKI)
Andrea.Kohlhase@dfki.de
[2] Computer Science, Jacobs University Bremen
{m.kohlhase,ch.lange}@jacobs-university.de

Abstract. We study the formalization process of a collection of documents created for a Software Engineering project from an MKM perspective. We analyze how document and collection markup formats can cope with an open-ended, multi-dimensional space of first and secondary classifications and relations. We show that RDFa-based extensions of MKM formats by flexible "metadata" relations referencing specific vocabularies are well-suited to encode and operationalize this. This explicated knowledge can be used for enriching interactive document browsing, for enabling multi-dimensional metadata queries over documents and collections, and for exporting Linked Data to the Semantic Web and thus enabling further reuse.

1 Introduction

The field of Mathematical Knowledge Management (MKM) tries to model mathematical objects and their relations, their creation and publication processes, and their management requirements. In [CF09, 237 ff.] Carette and Farmer analyzed "*six major lenses through which researchers view MKM*": the document, library, formal, digital, interactive, and the process lens. Quite obviously, there is a tension between the formal aspects "library", "formal", "digital" – related to machine use of mathematical knowledge – and the informal ones "document", "interactive", "process" – related to human use.

We encountered and dealt with all of these aspects in an extended case study in Software Engineering. The goal of this study was to create a document-oriented formalized process for Software Engineering, where MKM techniques are used to bridge the gap between informally stated user requirements and formal verification. The object of the study was a safety component for autonomous mobile service robots developed and certified as SIL-3 standard compliant (see [FHL+08]) in the course of the 3-year project "Sicherungskomponente für Autonome Mobile Systeme (SAMS)" at the German Research Center for Artificial Intelligence (DFKI). Certification required the software development to follow the V-Model (figure 1) and to be based on a verification of certain safety properties in the proof checker Isabelle [NPW02]. The V-Model mandates e.g. that relevant document fragments ("**objects**") get justified and linked to the

corresponding objects in other members of the document collection in a successive refinement process (the arms of the 'V' from the upper left over the bottom to the upper right and between arms in figure 1).

System development with respect to this regime results in a highly interconnected collection of design documents, certification documents, code, formal specifications, and formal proofs. This collection of documents (we call it "**SAMSDocs**" [SAM09]) make up the basis of a case study in the context of the FormalSafe project [For08] at DFKI Bremen, where they serve as a basis for research on machine-supported change management, information retrieval, and document interaction. The idea for using SAMSDocs as a FormalSafe case study was based on the assumption that a collection created with a strong formalization pressure would be easier to semantify than a regular collection.

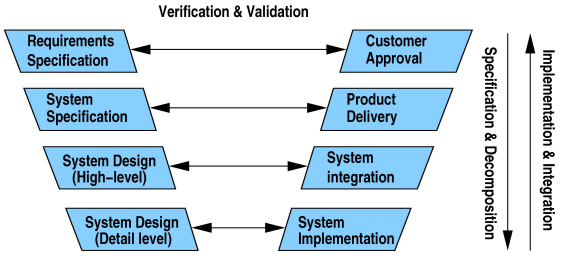

Fig. 1. The V-Model of Software Engineering

In this paper we report on — and draw conclusions from — the formalization of the LaTeX documents of SAMSDocs, particularly the inherent multi-dimensionality of the explicated structures (see section 2). The consequences for possible markup approaches we explore in section 3. Section 4 discusses *added-value services* enabled by multi-dimensional structured representations and section 5 concludes the paper.

2 Dimensions of Formality in SAMSDocs

We use the SAMSDocs corpus as a case study for what kind of implicit knowledge can be formalized. The structure of a collection – induced e.g. by the V-Model process – has a strong influence on the formality of and within the documents. At the same time, the structure itself is an object of formalization which makes the structural cues inscribed in the documents explicit.

We used the sTeX system [Koh08], a semantic extension of LaTeX, in order to both publish the documents as high-quality human-readable PDF and formal machine-processable OMDoc [Koh10], an XML format, via LaTeXML [SKG+10]. Mathematical, structural relations have a privileged state in sTeX, and its command sequence/environment syntax is analogous to the native element and attribute names in OMDoc. sTeX supports the collection view, since it allows to mark up a `theory/imports` structure as a collection-level structure. Unfortunately, it soon became apparent that this logic-inspired structure was too rigid for the intended stepwise knowledge explication. For example, sTeX's generated, convenient symbol macros could not be used without chunking document fragments into theories first. Fortunately, sTeX not only offers OMDoc 1.2 core features, it also allows for the OMDoc 1.3 scheme of metadata via RDFa [ABMP08]

annotations (see [LK09,Koh10]). This enables formalization on a much broader basis, since we can add *pre*-formal markup in the formalization process, we speak of **(semantic) preloading**. Such extensional SᴛEX packages can thus serve as a development platform for metadata vocabularies for specific structured representations, mirroring the metadata extensibility of OMDoc. This is additionally and particularly useful as complex RDFa annotations, which involve long and unintuitive URIs, can be hidden under simple command sequences for preloading[3].

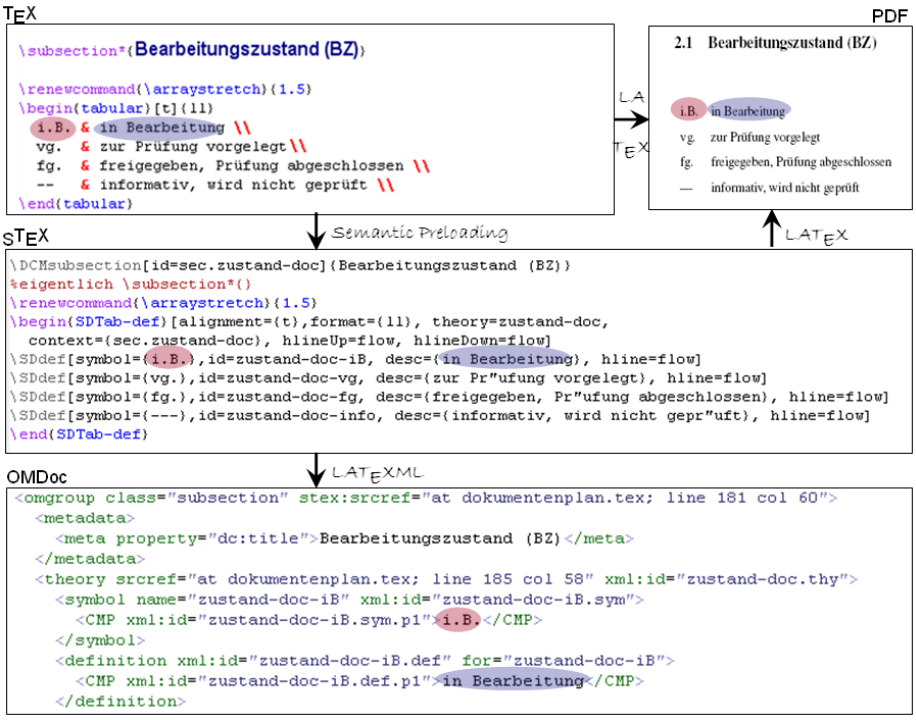

Fig. 2. The Formalization Workflow with SᴛEX-SD

In figure 2 we can see an example of a concrete formalization workflow where the original `tabular` TEX environment contains a list of symbols for document states with its definitions, e.g. "i.B." for "in Bearbeitung [in progress]". SᴛEX enables the person doing the structural explication to create collection-specific independent layout commands for the PDF and the OMDoc output; in our case we call this extension package **SᴛEX-SD** . For instance, we replaced the use of the `tabular` environment in figure 2 by employing a project-specific `SDTab-def` environment together with a list of `SDdef`-commands.

[3] We are currently developing an SᴛEX-based markup infrastructure for defining metadata vocabularies together with a module-based inheritance mechanism of custom metadata macros. This would alleviate the need for package extensions and allow metadata extensibility inside the SᴛEX format. The inspiration for this extension was a direct consequence of [KKL10].

As we were interested in the kind of explicated, previously implicit knowledge for future (re)use, we analyzed the semantic preloading and found distinct dimensions of formality, which we present in the following:

Preloading the Project Structure: A specific problem of the SAMS project constrained that the PDF form of the preloaded SᴛEX documents was not supposed to differ from the originals. Here, we extended SᴛEX-SD by syntactic sugar for project-wide layout structures, such as the definition table in figure 2 as this construct was used throughout SAMSDocs. In more detail, we introduced the `SDdef` command, which translates to an OMDoc element `symbol` named "*zustand-doc-iB*" and a corresponding `definition` element. Moreover, the new environment `SDTab-def` groups all respective `SDdef` entries into a PDF table *and* into an OMDoc theory. Such macro-support alleviated the project-specific markup progress and made it much more efficient.

Preloading the Document Structure: Spotting objects, i.e., identifying document fragments as autonomous objects, was a major part in the formalization process. Once spotted, they were first preloaded with a referencable ID, turning them into an object, and then classified and interrelated. For example, in the PDF screenshot in figure 2 we can easily spot the symbol "i.B." and its definition "in Bearbeitung" in the row of the definition table. The row as a document fragment turns into an object in the SᴛEX screenshot as it is attributed the ID "zustand-doc-iB", categorized as "definition" and implicitly specified via SᴛEX-SD to be a symbol and its definition. Once all relevant objects for a concept were gathered, their relating and chunking process started. Relating was attempted with light markup using commands like `\SDreferences` created for SᴛEX-SD or with SᴛEX's feature symbol macros. Latter could only be used if the respective module/theory structure for the collection was in place, that is if chunking had partially been done.

$$q(t) = s_G(v(t))$$
$$\frac{dq}{dt} = \frac{ds}{dv}\frac{dv}{dt}$$

Fig. 3. s is Braking Distance?

Explicating the document structure is not always obvious, since many of the documents contain recaps or previews or material that is normatively introduced in other documents/parts to make documents self-contained or enhance the narrative structure. Consider for example figures 3 and 4, which are actually clippings from the detailed specification "`Konzept-Bremsmodell.pdf`". Note the use of s resp. s_G, *both* pointing in fig. 3 to the braking distance function for straight-ahead driving (which is obvious from the local context), whereas in fig. 4 s represents the general arc length function of a circle, which is principally different from the braking distance, but coincides here.

Preloading the Collection Structure: Many concepts have occurrences in several of the documents in the collection SAMSDocs. Such occurrences are related but they are not occurrences of the same object. For example, an object was introduced as a high-level concept in the contract, then it was specified in another

$$a = s\frac{\sin\frac{\phi}{2}}{\frac{\phi}{2}} = s\,\mathrm{sinc}\frac{\phi}{2}$$

Fig. 4. Yet another Braking Distance s?

document, refined in a detailed specification, implemented in the code, reviewed at some stage, and so on until it was finally described in the manual. To markup the connectivity of these occurrences of one concept, we preloaded the collection structure, which consisted in the development process model, the V-Model as seen in figure 1. Here, we extended sTEX-SD by our personal semantic V-Model macros, e.g. `SemVMrefines`, `SemVMimplements` (to be used in the C-code documents), and `SemVMdescribesUse`.

Preloading the Organizational Structure: Besides the project structure we also found some organizational structure, that is probably not only used for a specific project (e.g. in a department) but for all. In the SAMS project such organizational structure consisted for example in a document version management and a document review history. Therefore we built another LaTeX package for preloading support of versioning (with semantic macros like `VMchangelist` and `VMchange`) and for reviewing (with `VMcertification` and `VMcertified`).

Preloading the Document Layout Structure:A typical document layout is structured into established parts like sections or modules. If we want to keep this grouping information in the formal XML document, we might use sTEX's `DCM` package. In the sTEX box in figure 2 we find for example the command `DCMsubsection` with attributes containing the title of the subsection and an ID, that can be used in the usual LaTeX referencing scheme.

Finally, we like to remark that the sTEX-SD preloading process was executed as "*in-place formalization*" [SIM99] and frequently considered several of the above structures for the object at hand at the same time. Therefore, the often applied metaphor of "formalization steps" does not mirror the formalization process in our case study. We found that, the important aspect of the formalization was not its sequence, which we consider particular to the SAMSDocs collection, but the fact that the distinct 'steps' addressed distinct kinds of knowledge. Note that these knowledge kinds only interact relatively lightly, so that we can consider them as independent **dimensions of a *multi*-dimensional space of knowledge** that is explicated in the formalization process of the document collection.

3 Multi-Dimensional Markup

Structured representations are usually realized as files marked up in formats that reflect the primary communicative intent and markup preferences of the author. The range of markup formats currently used for structured representation documents varies from PDF over office document formats like the Open Document Format or Office Open XML to scientific publishing formats like LaTeX on the informal side and includes a plethora of system-specific logics on the formal side. Collections of informal documents are usually structured by application-specific metadata like the Math Subject Classification [Soc09] or bibliographic metadata schemes [Dub08]. Formal systems increasingly contain custom modularization infrastructures, ranging from simple facilities for inputting external files to elaborate multi-logic theory graphs [MML07].

In the evaluation of document formats it is important to realize that every representation language concentrates on a subset of possible relations, which it treats with specific language constructs. No given format can natively capture *all* aspects of the domain to be represented with special-purpose markup primitives but has to relegate some of them to secondary mechanisms. In our case-study the V-Model structure was a good example, that was quite natural to use in hindsight but not from the beginning. In representation formats that support fragment identifiers — e.g. XML-based ones — these relations can be expressed as standoff-markup in RDF (Resource Description Framework [RDF04]), i.e., as subject-predicate-object triples, where subject and object are URI references to a fragment and the predicate is a reference to a relation specified in an external vocabulary or ontology[4]. As we have XML-based formats for informal documents (e.g. XHTML+MathML+SVG) and formal specifications (OpenMath or Content MathML), we can in principle already encode structured representations, if we only supply "metadata vocabularies" for their structural relations. Indeed this is the basic architecture of the "Semantic Web approach" to eScience, and much of the work of the MKM can be seen as attempts to come up with good "metadata vocabularies" for the mathematical/scientific domain. But this view disregards many practical workflow issues in the MKM domain:

W1) XML/RDF-based markup becomes huge and difficult to edit, unless supported by customized editing facilities.

W2) RDF stand-off markup is notoriously difficult to keep up to date.

W3) Authors, publishers, and readers have invested heavily in certain workflows and technology stacks and find it difficult to change.

In response to **W2)**, RDFa [ABMP08] has been developed: a set of attributes for embedding RDF annotations into XHTML. We see RDFa as a markup technology for making arbitrary XML-based languages extensible by inter- and intra-document relations. Similarly, RDFa serves as a vehicle for document format interoperability: all relations from a format D that cannot be natively represented in a format D' can be represented as RDFa triples, where the predicate is from an appropriately designed "metadata vocabulary" that describes the format D. For instance an OMDoc `<theory>` element can be represented as `<div typeof="http://omdoc.org/ontology#Theory">` in XHTML. This interoperability allows to accept all XML-based formats as structured representation formats that **F1)** support RDFa **F2)** and fine-grained text structuring with `div`/`span`-like elements everywhere. We have detailed the necessary extensions for our OMDoc format in [LK09]; a mature integration is part of OMDoc 1.3 [Koh10]; see also section 4 for a discussion of the services this affords. Analogous extensions for any of the XML-based formats used in the MKM community

[4] The difference between "vocabulary" and "ontology" is not sharply defined. Vocabularies are often developed in a bottom-up community effort and tend to have a low degree of formality, whereas ontologies are often designed by a central group of experts and have a higher degree of formality. Here, we use "vocabulary" in its general sense of a set of terms from a particular domain of interest. This subsumes the term "ontology", which we will reserve for cases that require a more formal domain model.

should be rather simple following this example. Note that the pragmatic restriction to XML-based representation formats is not a loss of generality. The three classes of non-XML languages in the MKM sphere are: *i)* computational logics *ii)* TEX/LATEX, and *iii)* PostScript/PDF. We see the computational logics as compact front-end formats that are optimized for manual input of formal structured representations; it is our experience that these can be transformed into the XML-based OpenMath, MathML, or OMDoc without loss of information (but with a severe loss of notational conciseness). We consider TEX/LATEX as analogous for informal structured representations; they can be transformed to XHTML+MathML by the LATEXML system [SKG+10]; see a LATEX extension for more formal structured representations in the next section. The last category of formats are presentation/print-oriented output and archival formats where the situation is more problematic: PostScript (PS) is largely superseded by PDF which allows standard document-level RDF annotations via XMP and the finer-granular annotations we need for structured representations via extensions as in [GMH+07] or [Eri07]. But PS/PDF are usually generated from other formats (mostly office formats or LATEX), so that alternative generation into XML-based formats like XHTML or OMDoc can be used.

Given this discussion we will use OMDoc as the representation for structured representations, as it has be largest coverage of MKM formats and we are most familiar with it. Given the necessary extensions and translators, any other XML-based MKM format (e.g. XHTML+MathML+SVG) would work as well.

4 Multi-Dimensional Added-Value Services

We have shown that the explication of knowledge results in an open-ended, multi-dimensional space of first and secondary classifications and relations and how MKM formats can cope with this. In this section, we carry this forward to the application domain: Are these document formalizations — formal *and* informal ones — beneficial for services supporting real users? Concretely, we envision three use scenarios in the SAMS context and present added-value services that retrieve and display *multi-dimensional* information.

Engineers, having the task to implement a certain piece of software, start with gathering information about the specification for the code to be written. In particular, they look up the specific detailed specification document and will (probably) do it again and again until the task is accomplished. Concrete questions that arise might be:

(i) How much of this specification has already been implemented?
(ii) What is the definition for a certain (mathematical) symbol?[5]
(iii) In what state is the proof of a specific equation, has it already been formally verified so that it is safe to ground my implementation on it?
(iv) Whom can I ask for further details?

[5] See figures 3 and 4 for two symbols having the same appearance but different meanings.

Assuming multi-dimensional markup like the one in SAMSDocs, an RDF-based information retrieval system can supply useful answers. For example, it can answer (ii) when technical terms in natural language are linked to the respective formal mathematical symbols they represent. For replies to (i) and (iii) we note that if all collection links are merged into a common RDF graph, then their original placement and direction no longer makes a difference. So if we have links from the Isabelle formalization to the respective C-code and links from this C-code to a specification fragment as realized in the V-Model structure of SAMSDocs, then we can follow the graph from the specification through to the state of the according proof. Drawing on the V-Model links combined with the semantic version management or the review logs, the system can deduce the answer for (iv): The code in question connects to a specification document, that has authors and reviewers. This service can be as fine-grained as one is willing to formalize the granularity of the version and review management. Now, if we admit further dimensions of Linked Data into the picture, then the system might find persons with similar interests in terms of the FOAF vocabulary (Friend of a Friend [BM07]), as has been elaborated in the use case descriptions of the ExpertFinder initiative [Exp07].

Project Managers need to have an overview of many aspects of the project and might therefore be interested in the following issues:

(i) *Software Engineering Process* How much code has been implemented to satisfy a particular requirement from the contract? Has the formal code structure passed a certain static analysis and verification? A manager would not want to inspect that manually by running Isabelle, but needs high-level figures of, e. g., the number of mathematical statements without a formally verified proof.

(ii) *Certification* What parts of the specification, e. g. requirements, have changed since the last certification? What other parts does that affect, and thus what subset of the whole specification has to be re-certified?,

(iii) *Human Capital* Who is in charge of a document? How could she be replaced if necessary, taking into account colleagues working on the same or on related documents – such as previous revisions of the same document, or its predecessor in terms of the V-Model, i. e. the document that is refined by the current one?

Exploiting the multi-dimensionality of formalized knowledge, how the issues can be tackled becomes obvious.

Certifiers have to understand a system in order to approve it. They need similar lookup services as the engineers above, but on a higher level. For inspection, a certifier might first be interested in an overview, such as a list of all relevant concepts in the contract document, then she would like to follow the links to the detailed specification and further on to the actual implementation. For more information, she might contact the project investigator instead of the particular author of a code snippet. Moreover, the certifier might not be familiar with the mathematical or physical background theories the specification is grounded on and would thus follow dependency links to external

resources. The certifier also needs to understand what parts of the whole specification are subject to a requested re-certification. Finally, a certifier's rejection of a certain part of a document affects all elements in the collection that depend on it. Again, a system can support a certifier's efficiency by combining the explicated information of distinct knowledge dimensions.

We will now present in more detail how concrete services utilize the structural dimensions found in SAMSDocs. The **relations** of interest, which have been identified and made explicit in the preloading process described in section 2, using language constructs reviewed in section 3, now have to be *operationalized*. Our services draw on representations of the semantic structures as RDF Linked Data – for a set of best practices for publishing RDF; see [BCH07] – using different vocabularies for different dimensions, so they can be uniformly queried and browsed. Here, we will focus on how queries against the multi-dimensional knowledge structures enable services for Software Engineering.

We have developed an integrated work environment that enables these services. The technical foundation, which is independent of the number of dimensions of the knowledge space, has been described in detail in [DKL+10]. The central component is the versioned `TNTBase` [ZK09] database with special support for XML that supports fine-grained access to document fragments and on-the-fly generation of dynamic XHTML+MathML+RDFa renderings. We envision that the engineers, managers, and certifiers in our scenarios interact with these renderings. Our setup supports embedded information lookup using the JOBAD architecture [GLR09], turning the rendered documents into command centers for executing queries and displaying their results without forcing the user to switch to a different context: While reading about a concept $\mathcal{C}$, such as the braking distance s in figures 3 and 4, the user may want to look up information about another concept $\mathcal{C}'$, such as the velocity of the robot, which is related to $\mathcal{C}$ via some relation R (here: a generic reference in a pre-formal stage of formalization), or she performs a query for all available $\mathcal{C}'$ that are related to $\mathcal{C}$ via R.

As the OMDoc transformation of an STEX document preserves all explicated structures, we can equally speak of the formalized SAMSDocs collection as STEX or OMDoc documents. Semantic annotations in such OMDoc files consist of native XML markup for mathematical structures and now additionally of (syntactically different) RDFa for the other dimensions of knowledge. The semantic annotations will be traversed (for lookup) and they will be queried – usually in the direction of one or more specific dimensions. When speaking about relations from the point of view of their operationalization by services, we will now use the term **link**. Note that we have two possible ways of answering the respective queries: in the client and on the server. For the latter, the OMDoc markup and the all RDFa annotations are translated into uniform RDF (see [Lan09,Lan08,Lan10] for more information) and stored in an RDF database ("triple store") integrated into the TNTBase server for efficient collection-wide querying of multi-dimensional structures (see examples below). For client-side query answering we make use of the RDFa annotations of the renderings the user interacts with. Note that now *all* RDF data available in that triple store it can

be combined with external information about people and organizational structures expressed using FOAF and stored in RDF/XML files by existing FOAF tools, or think of legacy data in a relational database, which is exposed as RDF Linked Data in the "Engineers" use case (iv).

Listing 1.1. Finding a Substitute for an Employee via the V-Model

```
SELECT ?potentialSubstituteName WHERE {
  # for each document Alice is responsible for, get all of its parts
  # i.e. _any_ kind of semantic (sub)object in the document
  ?document        vm:responsible <../employees#Alice> ;
                   omdoc:hasPart ?partOfDocument .

  # for those parts (= concepts) that are linked to their
  # V-Model predecessors that they refine, ...
  ?partOfDocument  semVM:refines ?vModelPredecessor .

  # ... the document containing that predecessor, the person
  # responsible for that document, ...
  ?otherDocument   omdoc:hasPart ?vModelPredecessor ;
                   vm:responsible ?potentialSubstitute .

  # ... and her name
  ?potentialSubstitute foaf:name ?potentialSubstituteName .
}
```

Concretely, we showcase a potential SPARQL RDF query [PS08] in listing 1.1 for finding a substitute for employee Alice via the V-Model relations as envisioned in the "Project Managers" use case above. Let us assume that Alice can be identified by a URI and first retrieve all documents in the collection for which Alice is known to be the responsible person. Then we are interested in the objects of these documents that refine 'previous' objects, and for the latter we finally determine the responsible persons. For example, if Alice was responsible for the detailed specification of the braking distance function for straight-ahead driving s_G and if s_G were a refinement of the general braking distance s introduced in a concept specification with Pierre as assigned responsible person, then Pierre might be considered as substitute for Alice.

Listing 1.2. Finding Objects that are Subject to Recertification

```
SELECT ?subjectToReCertification WHERE {
 # We assume that the "last certification" is available as a semantic object
  :lastCertification dc:date ?lastCertDate .

  # Find all requirements in the collection, and the last change
  ?req rdf:type semVM:Requirement
      vm:lastChange ?lastReqChange .
  FILTER (?lastReqChange > ?lastCertDate)

  # Three sets of objects are subject to re-certification:
  # 1. dependent objects (transitive) in terms of the V-model
 { ?subjectToReCertification semVM:dependsOn ?req }
  UNION
  # 2. dependent objects (transitive) in terms of OMDoc-formalized math. structures
  { ?subjectToReCertification omdoc:dependsOn ?req }
  UNION
 # 3. the requirements themselves
  # this construct binds the requirements found previously to the
  # ?subjectToReCertification variable
  { ?subjectToReCertification rdf:type semVM:Requirement .
    FILTER (?subjectToReCertification = ?req) }
```

As an example that draws on the multi-dimensionality of the formalization, consider a certifier who has to re-certify a system because of an unavoidable change in an object $\mathcal{O}$ (listing 1.2). As subsequent changes of this object's change are documented in the document *version management*, we can use this information as a first filter. But we are only interested in those documents, that contain links from or to $\mathcal{O}$ via the *V-Model* structure (simplified in the shown query to `SemVMdependsOn`). It is easy to imagine that the *mathematical structure* adds another filter e.g. since the certifier might only be interested in changes in content and not form.

Our JOBAD architecture wraps such queries into service modules and gives access to them from a user interface embedded into the rendered documents. Services are available wherever there are suitable RDFa or MathML annotations. The query in listing 1.1 depends on a particular employee and is therefore made available wherever [links to] employees occur in a document – for example when there is a `<link rel="vm:responsible" resource="..."/>` RDFa annotation. Another common service is the lookup service, which works on mathematical symbols (looking up their definition) and on RDFa links (looking up the linked object). As future work, we also envision equipping the context menu of certification documents with menu entries for committing an approval or rejection to the server, which would only be displayed to the certifier. The server could then trigger further actions, such as marking the document that contains a rejected object and all dependencies of that object as rejected, too.

In conclusion, we have shown that multi-dimensional queries are very natural in Software Engineering scenarios and that multi-dimensional markup affords multi-dimensional services. Note that if we interpret our dimensions as distinct contexts, then our services become context-sensitive as dimensions can be filtered in and out. The more dimensions are explicated in a document, the more context-sensitive services become available.

Note as well that each of our dimensions corresponds to a vocabulary. In the course of the SAMSDocs case study, most vocabularies have initially been implemented from scratch in a project-specific ad hoc way during the preloading phase described in section 2. But SᴛEX allows for elaborating vocabularies towards ontologies, which can be translated to RDF-based formats that reasoners understand [KKL10]. An alternative is reusing existing ontologies. FOAF for basic properties of persons and organizations has already been mentioned; the widely known Dublin Core element set that occurred in listing 1.2 is also available as an ontology. DOAP (Description of a Project [Dub10]) describes software projects – focusing on the top-level structure of public open source projects. DCMI Terms [DCM], a modernized and extended version of the Dublin Core element set, offers a basic vocabulary for revision histories – but not for reviewing and certification. Lin et al. have developed an ontology for the requirements-related parts of the V-Model (cf. [LFB96]). Happel and Seedorf briefly review further ontologies about Software Engineering [HS06]. Besides often being more formal than vocabularies developed ad hoc, the advantages of such existing on-

tologies are that reusable services may already have been implemented for them, and that they are more widely used, or more likely to be adopted by other projects, which facilitates the creation of Linked Data sets. As the SAMSDocs vocabularies can be integrated with existing ontologies by declaring appropriate subclass or equivalence relationships, services can make use of the best of both worlds.

5 Conclusion and Further Work

In this paper we have studied the applicability of MKM technologies in Software Engineering beyond "Formal Methods" (based on the concrete SAMSDocs document collection and its formalization). The initial hypothesis here is that contract documents, design specifications, user manuals, and integration reports can be partially formalized and integrated into a computer-supported software development process. To test this hypothesis we have studied a collection of documents created for the development of a safety zone computation, the formal verification that the braking trajectory always lies in the safety zone, and the SIL3 certification of this fact by a public certification agency. As the project documents contain a wealth of (informal) mathematical content, MKM formats (in this case our OMDoc format) are well-suited for this task. During the formalization of the LaTeX part of the collection, we realized that the documents contain an open-ended, multi-dimensional space of implicit knowledge that can be used for supporting projects — if explicated.

We have shown that RDFa-based extensions of OMDoc by flexible "metadata" relations referencing specific vocabularies can be used to encode and operationalize this knowledge space. We have pointed out that the "dimensions" of this space can be seen to correspond to different "metadata vocabularies". Note that the distinction between data and metadata blurs here as the OMDoc data model realized by native markup in the OMDoc format can also be seen as OMDoc metadata and could be realized by RDFa annotations to some text markup format, where the meaning of the annotations is given by the OMDoc ontology [Lan08,Lan10]. This "metadata view" is applicable to all MKM formats that mark up informal mathematical texts (e.g. MathDox [CCB06] and MathLang [KWZ08]) as long as they explicate their data model in an ontology. This observation makes decisions about which parts of the knowledge space to support with native markup a purely pragmatic choice and opens up new possibilities in the design of representation formats. It seems plausible that all MKM formats use native markup for mathematical knowledge structures (we think of them as primary knowledge structures for MKM) and differ mostly in the secondary knowledge structures they internalize. XHTML+MathML+RDFa might even serve as a baseline interchange format for MKM applications[6], since it is minimally committed. Note that if the metadata ontologies are represented in modular formats that admit theory morphisms, then these can be used as

[6] Indeed a similar proposal has been made for Semantic Wikis [VO06] which have related concerns but do not involve mathematics.

crosswalks between secondary metadata for higher levels of interoperability. We leave its development to future work.

The explicated secondary knowledge structures can be used for enriching interactive document browsing and for enabling multi-dimensional metadata queries over documents and collections. We have shown a set of exemplary added-value services based on the RDFa-encoded metadata, mostly centered around Linked Data approaches based on RDF-based queries. More services can be obtained by exporting Linked Data to the Semantic Web and thus enabling further reuse. In particular, the multi-dimensionality observed in this paper and its realization via flexible metadata regimes in representation formats allows the knowledge engineers to tailor the level of formality to the intended applications.

In our case study, the metadata vocabularies ranged from project-specific ones that had to be developed (e.g. definition tables) to general ones like the V-Model vocabulary for which external ontologies could be reused later on. We expect that such a range is generally the case for Software Engineering projects, and that the project-specific vocabularies may stabilize and be standardized in communities and companies, lowering the formalization effort entailed by each individual project. In fact we anticipate that such metadata vocabularies and the software development support services will become part of the strategic knowledge of technical organizations.

In [CF09, 241] Carette and Farmer challenge MKM researchers by assessing some of their technologies: "*A lack of requirements analysis very often leads to interesting solutions to problems which did not need solving.*". With this paper we hope to have shown that MKM technologies can be extended to cope with "real world concerns" (in Software Engineering). Indeed industry are becoming more and more aware of and interested in Linked Data (see e.g. [Ser08] and [LDF, Question 14]), so that the methods reported on in this paper can be integrated.

References

[ABMP08] Ben Adida, Mark Birbeck, Shane McCarron, and Steven Pemberton. RDFa in XHTML: Syntax and processing. W3C Recommendation, World Wide Web Consortium (W3C), October 2008.

[BCH07] Chris Bizer, Richard Cyganiak, and Tom Heath. How to publish linked data on the web, July 2007.

[BM07] Dan Brickley and Libby Miller. FOAF vocabulary specification 0.91. Technical report, ILRT Bristol, November 2007.

[CCB06] A. M. Cohen, H. Cuypers, and E. Reinaldo Barreiro. Mathdox: Mathematical documents on the web. In OMDoc – *An open markup format for mathematical documents [Version 1.2]*, number 4180 in LNAI, chapter 26.7, pages 278–282. Springer Verlag, August 2006.

[CDSCW09] Jacques Carette, Lucas Dixon, Claudio Sacerdoti Coen, and Stephen M. Watt, editors. *MKM/Calculemus 2009 Proceedings*, number 5625 in LNAI. Springer Verlag, July 2009.

[CF09] Jacques Carette and William Farmer. A review of mathematical knowledge management. In Carette et al. [CDSCW09], pages 233–246.

[DCM] DCMI Usage Board. DCMI metadata terms. DCMI recommendation, Dublin Core Metadata Initiative.

[DKL+10] Catalin David, Michael Kohlhase, Christoph Lange, Florian Rabe, Nikita Zhiltsov, and Vyacheslav Zholudev. Publishing math lecture notes as linked data. In Lora Aroyo, Grigoris Antoniou, and Eero Hyvönen, editors, *ESWC*, Lecture Notes in Computer Science. Springer, June 2010. In press.

[Dub08] Dublin Core metadata element set. DCMI recommendation, Dublin Core Metadata Initiative, 2008.

[Dub10] Edd Dubmill. DOAP – description of a project. `http://trac.usefulinc.com/doap`, seen Mar. 2010.

[Eri07] Henrik Eriksson. The semantic-document approach to combining documents and ontologies. *International Journal of Human-Computer Studies*, 65(7):624–639, 2007.

[Exp07] The ExpertFinder initiative, 2007.

[FHL+08] Udo Frese, Daniel Hausmann, Christoph Lüth, Holger Täubig, and Dennis Walter. The importance of being formal. In Hardi Hungar, editor, *International Workshop on the Certification of Safety-Critical Software Controlled Systems SafeCert'08*, volume 238 of *Electronic Notes in Theoretical Computer Science*, pages 57–70, September 2008.

[For08] FormalSafe. `http://www.dfki.de/sks/formalsafe/`, seen Dec. 2008.

[GLR09] Jana Giceva, Christoph Lange, and Florian Rabe. Integrating web services into active mathematical documents. In Carette et al. [CDSCW09], pages 279–293.

[GMH+07] Tudor Groza, Knud Möller, Siegfried Handschuh, Diana Trif, and Stefan Decker. SALT: Weaving the claim web. In Karl Aberer, Key-Sun Choi, Natasha Fridman Noy, Dean Allemang, Kyung-Il Lee, Lyndon J. B. Nixon, Jennifer Golbeck, Peter Mika, Diana Maynard, Riichiro Mizoguchi, Guus Schreiber, and Philippe Cudré-Mauroux, editors, *ISWC/ASWC*, number 4825 in Lecture Notes in Computer Science, pages 197–210. Springer, 2007.

[HS06] Hans-Jörg Happel and Stefan Seedorf. Applications of ontologies in software engineering. In *Proc. 2nd International Workshop on Semantic Web Enabled Software Engineering (SWESE '06)*, 2006.

[KKL10] Andrea Kohlhase, Michael Kohlhase, and Christoph Lange. sTeX – a system for flexible formalization of linked data. submitted to I-SEMANTICS 2010, 2010.

[Koh08] Michael Kohlhase. Using LaTeX as a semantic markup format. *Mathematics in Computer Science*, 2(2):279–304, 2008.

[Koh10] Michael Kohlhase. An open markup format for mathematical documents OMDoc [version 1.3]. Draft Specification, 2010.

[KWZ08] Fairouz Kamareddine, J. B. Wells, and Christoph Zengler. Computerising mathematical text with mathlang. *Electron. Notes Theor. Comput. Sci.*, 205:5–30, 2008.

[Lan08] Christoph Lange. The OMDoc document ontology. web page at `http://kwarc.info/projects/docOnto/omdoc.html`, seen August 2008.

[Lan09] Christoph Lange. Krextor – an extensible XML→RDF extraction framework. In Chris Bizer, Sören Auer, and Gunnar Aastrand Grimnes, editors, *Scripting and Development for the Semantic Web (SFSW2009)*, May 2009.

[Lan10] Christoph Lange. *Semantic Web Collaboration on Semiformal Mathematical Knowledge*. PhD thesis, Jacobs University Bremen, 2010. submission expected in spring 2010.

[LDF] Linked data FAQ. http://structureddynamics.com/linked_data.html.

[LFB96] Jinxin Lin, Mark S. Fox, and Taner Bilgic. A requirement ontology for engineering design. In *Proceedings of 3rd International Conference on Concurrent Engineering*, pages 343–351. Technomic Publishing Company, Inc., August 1996.

[LK09] Christoph Lange and Michael Kohlhase. A mathematical approach to ontology authoring and documentation. In Carette et al. [CDSCW09], pages 389–404.

[MML07] Till Mossakowski, Christian Maeder, and Klaus Lüttich. The heterogeneous tool set. In Orna Grumberg and Michael Huth, editors, *Proceedings of the 13th International Conference on Tools and Algorithms for the Construction and Analysis of Systems TACAS-2007*, number 4424 in LNCS, pages 519–522, Berlin, Germany, 2007. Springer Verlag.

[NPW02] Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*. Number 2283 in LNCS. Springer, 2002.

[PS08] Eric Prud'hommeaux and Andy Seaborne. SPARQL query language for RDF. W3C Recommendation, World Wide Web Consortium (W3C), January 2008.

[RDF04] Resource description framework (RDF). http://www.w3.org/RDF/, 2004.

[SAM09] SAMS. SAMSDocs: The document collection of the SAMS project, 2009. http://www.sams-projekt.de.

[Ser08] François-Paul Servant. Linking enterprise data. In Christian Bizer, Tom Heath, Kingsley Idehen, and Tim Berners-Lee, editors, *Linked Data on the Web (LDOW 2008)*, number 369 in CEUR Workshop Proceedings, April 2008.

[SIM99] Frank M. Shipman III and Raymond J. McCall. Incremental formalization with the hyper-object substrate. *ACM Trans. Inf. Syst.*, 17(2):199–227, 1999.

[SKG+10] Heinrich Stamerjohanns, Michael Kohlhase, Deyan Ginev, Catalin David, and Bruce Miller. Transforming large collections of scientific publications to XML. *Mathematics in Computer Science*, 2010. in press.

[Soc09] American Mathematical Society. Mathematics Subject Classification MSC2010. http://www.ams.org/mathscinet/msc/, 2009.

[VO06] Max Völkel and Eyal Oren. Towards a Wiki Interchange Format (WIF). In Max Völkel, Sebastian Schaffert, and Stefan Decker, editors, *Proceedings of the 1st Workshop on Semantic Wikis, European Semantic Web Conference 2006*, number 206 in CEUR Workshop Proceedings, Budva, Montenegro, June 2006.

[ZK09] Vyacheslav Zholudev and Michael Kohlhase. TNTBase: a versioned storage for XML. In *Proceedings of Balisage: The Markup Conference 2009*, Balisage Series on Markup Technologies. Mulberry Technologies, Inc., 2009. available at http://kwarc.info/vzholudev/pubs/balisage.pdf.